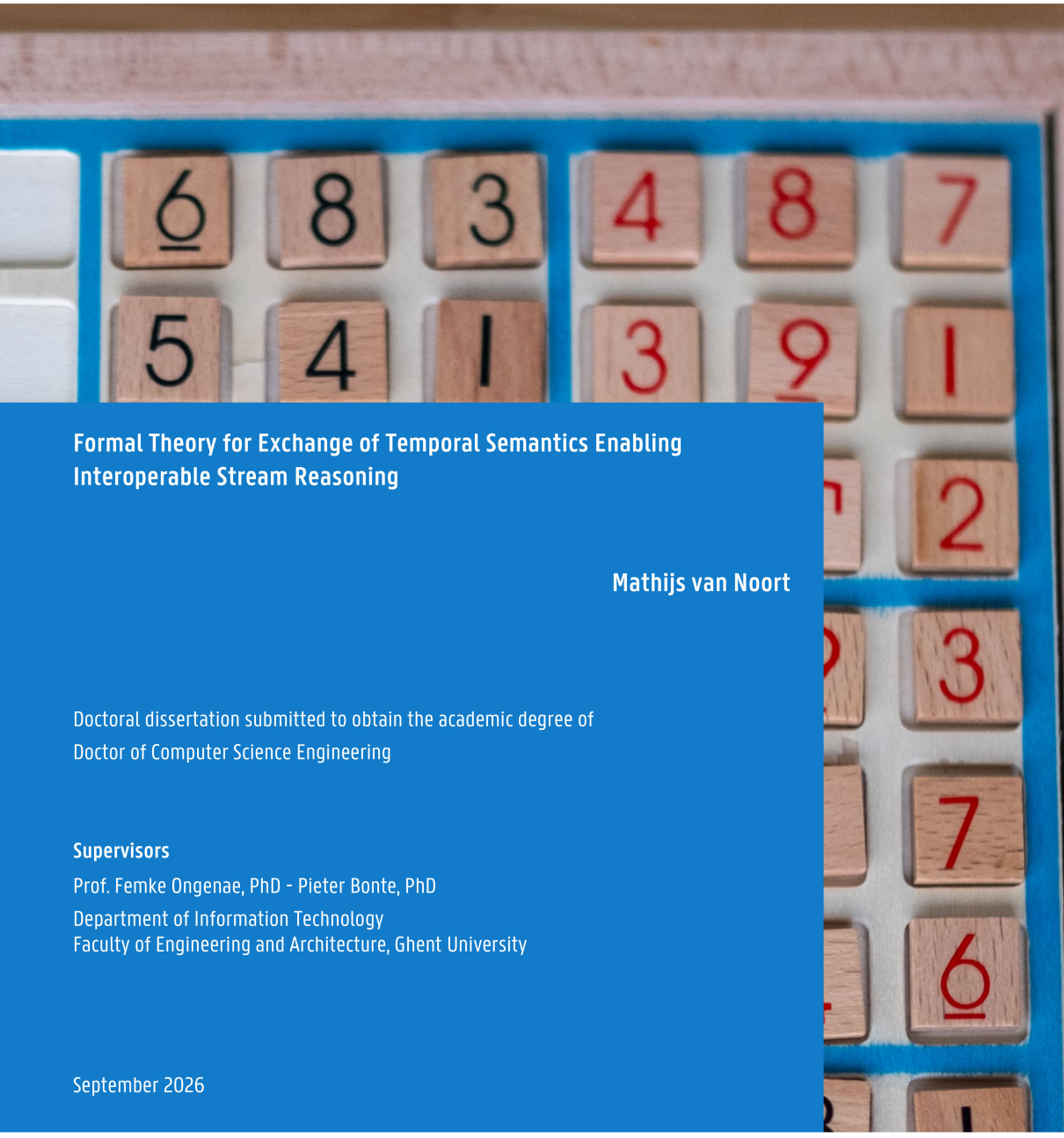
FACULTY OF ENGINEERING AND ARCHITECTURE



Formal Theory for Exchange of Temporal Semantics Enabling Interoperable Stream Reasoning

Mathijs van Noort

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Femke Ongenae, PhD - Pieter Bonte, PhD
Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

September 2026

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Formal Theory for Exchange of Temporal Semantics Enabling Interoperable Stream Reasoning

Mathijs van Noort

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Femke Ongenae, PhD - Pieter Bonte, PhD
Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

September 2026

GHENT UNIVERSITY

ISBN 978-94-93570-06-1
NUR 993
Wettelijk depot: D/2026/10.500/114

Members of the Examination Board

Chair

Prof. Luc Dupré, PhD, Ghent University

Other members entitled to vote

Prof. Bart Bogaerts, PhD, KU Leuven
Prof. Daniel de Leng, PhD, Linköping University, Sweden
Prof. Eric Laermans, PhD, Ghent University
Prof. Ruben Verborgh, PhD, Ghent University

Supervisors

Prof. Femke Ongenae, PhD, Ghent University
Pieter Bonte, PhD, Ghent University

Dankwoord

"The important thing isn't can you read music, it's can you hear it. Can you hear the music, Robert?"

– Niels Bohr in *Oppenheimer* (2023), by Christopher Nolan

Er zal door mijn voormalige wiskunde-leerkrachten stevig gefronst worden wanneer ze vernemen dat ene Mathijs van Noort een doctoraat behaald. Een vervelende en vooral luie leerling met een levensbedreigende allergie voor tussenstappen, die het nauwelijks kon opbrengen om op te letten in de les en tevreden knikt bij een score van 60%. Bijna even frappant is het dat het gaat om een doctoraat als *ingenieur*. 10 jaar geleden werd er immers – met een quasi-rancuneuze aversie voor alles wat toegepast is – gestart aan de bachelor Wiskunde, een keuze ingegeven door nieuwsgierigheid, koppigheid en vooral niet door logica. Hetgeen nu voorligt is het resultaat van die 10 jaar lange reis, met vele bochten, omwegen en vooral veel personages. Hieronder mijn beste, doch incomplete poging om de waaier aan personen te benoemen – en vooral, te bedanken – die aan dat traject hebben meegewerkt.

Allereerst Femke en Pieter, het olijke duo dat het aandurfde om in volle COVID-periode de gok te wagen die ene wiskundige onder hun vleugels te nemen. Hun geduld met mijn voorliefde voor formules zonder tekst en uitleg, bescheiden programeerskills en side quests in (onder meer) OAP-vertegenwoordiging is niet te onderschatten. Hun toewijding gedurende ettelijke iteraties aan follow-ups, nalezingen en brainstorms wordt ten zeerste geapprecieerd.

Niet enkel Femke en Pieter, maar ook de ruimere waaier aan collega's van KNoWS en IDLab zijn onontbeerlijk geweest in het tot stand brengen van dit doctoraat. De ettelijke gedachtewisselingen rond whiteboards en koffiemachines zijn onlosmakelijk verbonden met het tot stand gebrachte werk. De lunchpauzes en impromptu kantoorconversaties waren bovendien de noodzakelijke verankering om de mentale gezondheid – in de mate van het mogelijke – te kunnen bewaren.

Daarnaast wil ook de juryleden van dit doctoraat bedanken. Dankzij het rigorouze naleeswerk van Luc Dupré, Bart Bogaerts, Daniel de Leng, Eric Laermans en Ruben Verborgh kon de kwaliteit van dit doctoraat naar een hoger niveau worden getild. Hun expertise en tijdsinvestering mag dan ook niet onopgemerkt voorbijgaan.

Een doctoraat vraagt meer dan enkel een goede werkomgeving van collega's, mentoren en experten die bijstaan in het werk zelf. De mentale rollercoaster die een doctoraat kenmerkt heeft nood aan een grondige veiligheidsgordel, eentje die beschermt tegen de alle bochten, kronkels en wendingen die men tegenkomt. Vele vrienden hebben de verschillende ups en downs van dichtbij mogen meemaken, klaagliederen mogen aanhoren, stress-geïnduceerde rants moeten tolereren. Ik had echter niet zonder de vele opstekers gekund, de cafeetjes, terrasjes en activiteiten allerlei die me uit mijn whiteboard konden trekken en me herinnerden aan mijn menselijke kant.

In het bijzonder Cath, Tim, Nathan, Jarne, Jeffrey, Laura en Nicolas, een bende samengebracht door nachtelijke AV's, roddelpraat, verkeerde gedachtenkronkels en humor. Bij wijze van statistische anomalie kent menig van ons de geneugten van het doctoraatsleven. Onder de vaandel van Schotse leuzen mocht er dan ook vaak genoeg de draak worden gestoken met frivoliteiten en frustraties van het leven als (U)Gentenaar en de queeste richting een boek zoals deze. Dit boek is dan ook slechts een kleine mijlpaal op de weg die we samen afleggen, dewelke ongetwijfeld nog ellenlang zal doorlopen.

Van thuis uit heb ik altijd de onvoorwaardelijke steun gekregen om de richting na te streven die mij nauw aan het hart ligt. De vroege ochtenden, lange dagen en vertraagde treinen werden een stuk draaglijker dankzij een gevuld huis dat zelden een saai moment kende. Met Jeanine, Reinier, Bram en Ralf was er altijd genoeg animo in huis om mentaal te resetten, al dan niet in de vorm van het uitbreken van een muur of het verzeulen van een hoop stalen balken. Dankzij Bart geen tekort aan tafeldiscussies, gaande van economie en politiek tot technologie en fysica. Oma en opa, alhoewel fysiek wat verder op afstand, hoorden met plezier en bewondering de progressie aan van wat hun kleinzoon uitspookt. Wat verder terug in de tijd hoort daar ook nog opa Tetteroo bij, die maar al te graag hamerde op het belang van hoofdrekenen. Mogelijks is het daar zelfs allemaal begonnen. Ongeacht het precieze startpunt, heb ik altijd kunnen rekenen op Dasja – of zoals ik ze beter ken: mam. Ondanks het weinig spectaculaire schoolparcours stond je altijd achter mij en mijn keuzes. Ook al is de inhoud van dit boek niet per se *your cup of tea*, toch hield dat je niet tegen om het Kanaal over te steken en een cup of tea te komen delen. Weet dat ik net zo trots op jou ben als jij op mij.

Als sluitstuk rest er mij enkel nog de persoon te bedanken die de reden is dat ik tout court aan een doctoraat ben begonnen: Ruxandra. Om enigszins representatief te zijn voor het aandeel dat ze heeft gehad in dit doctoraat, zou volgende paragraaf op z'n eentje zo'n vijftien pagina's moeten beslaan. Het capteren van de rollercoaster van onze synchroondoctoraten, met onze viervoetige ghostwriters op sleeptouw, is onbegonnen werk. Toch een verhoede poging om neer te schrijven wat je allemaal hebt betekent.

'Geduldig' is niet een woord dat de meeste mensen snel met Rux zullen associëren, en tegelijkertijd schiet het tekort voor wat er de voorbije vier jaar van je gevraagd werd. Menig avonden en weekenddagen bleef ik koppig doorwerken, boven op mijn bureau

of zelfs midden in de living, in plaats van samen quality time te spenderen. Minstens even vaak maakte ik je veel te vroeg wakker om naar werk te vertrekken, waarmee je rustige ochtend aan diggelen lag. Pogingen om ons bureau tot een stijlvol geheel te maken werden steevast gefnuikt door het absurde 1.5m^2 whiteboard dat ik per se wilde hebben. Naderende deadlines gingen gepaard met een Mathijs die even geïrriteerd als irritant was en die te weinig omkeek naar de vloer, de vaatwas en al het andere huishouden. Al die kleine(?) dagelijkse frustraties werden getolereerd, hooguit met een zucht of een eyeroll erbij. Daarbovenop kwam dan nog eens een vier maand durende excursie weg van huis, waarbij je alleen achterbleef in een huis vol kattengejank. Toegegeven, ook dit avontuur was mede door jou ingegeven. Het is slechts één illustratie van de duizenden manieren waarop je mij pusht om de beste versie van mezelf te zijn. Weet echter dat ik zonder jou hier nooit zou staan. Ik ~~kan~~ wil me geen leven zonder jou inbeelden.

Gent, September 2026
Mathijs van Noort

Table of Contents

Dankwoord i

Table of Contents v

List of Figures xi

List of Tables xiii

List of Acronyms xv

List of Symbols xix

Samenvatting xxi

Summary xxv

1 Introduction 1

1.1 Semantic Interoperability in a Streaming, Decentralised World 1

1.2 Background on Declarative Programming 6

1.3 Challenges 10

1.3.1 Challenge C1: Semantic Fragmentation of Stream Reasoning (SR) Formalisms 11

1.3.2 Challenge C2: Foundational Choices in Temporal Semantics, Negation and Their Consequences 12

1.3.3 Challenge C3: Bridging Theoretical Expressivity and Practical Reasoning Infrastructure 13

1.4 Research Questions and Hypotheses 14

1.4.1 RQ1: Characterising Semantic Fragmentation in SR 15

1.4.2 RQ2: Relating Metric-Based and Window-Based Temporal Semantics . 16

1.4.3 RQ3: Temporal Reasoning under Open-World Assumptions 16

1.4.4 RQ4: Realising SR on Non-Temporal Reasoning Infrastructure 17

1.5 Outline 18

1.6 Publications 19

1.6.1 Publications in international journals (A1, listed in the Science Citation Index) 20

1.6.2 Publications in international conferences (P1, listed in the Science Citation Index) . . . 20
1.6.3 Publications in other peer-reviewed conferences (C1) . . . 21
References . . . 21

2 Navigating Streams: A Survey on 15 Years of Advances in Stream Reasoning Formalisms 25
2.1 Abstract . . . 26
2.2 Introduction . . . 26
2.3 Research Questions and Protocol . . . 29
2.3.1 Research Questions . . . 29
2.3.2 Protocol . . . 29
2.4 Preliminaries . . . 32
2.4.1 Streams and time . . . 32
2.4.1.1 Time Representation . . . 32
2.4.1.2 Representation of a stream . . . 32
2.4.1.3 Windowing . . . 33
2.4.2 Static relations . . . 34
2.4.2.1 First-Order Logic, Description Logics and Public Announcement Logic . . . 34
2.4.2.2 Set theory . . . 35
2.4.2.3 Answer Set Programming (ASP) . . . 35
2.4.3 Expressing temporal relations . . . 37
2.4.3.1 Signal Temporal Logic (STL) . . . 39
2.4.4 Datalog . . . 39
2.4.5 Resource Description Framework (RDF) . . . 40
2.4.6 Querying . . . 41
2.5 Related Surveys . . . 42
2.6 Dimensions of a SR formalism . . . 43
2.6.1 Structure of the Discussion . . . 45
2.6.2 Theoretical Dimensions . . . 45
2.6.3 Usage Dimensions . . . 51
2.7 Survey Results: Theory Dimension . . . 52
2.7.1 DL-based Formalisms . . . 52
2.7.1.1 DL-Lite TQL and its fragments . . . 52
2.7.1.2 $\text{LTL}_{\mathcal{EL}}^{\mathbf{X},\mathbf{G}}$ . . . 61
2.7.1.3 $\mathcal{EL}^{++}$ ontology streams . . . 61
2.7.2 Temporal Logic Formalisms . . . 62
2.7.2.1 DatalogMTL and its fragments . . . 62
2.7.2.2 Metric Spatio-Temporal Logic (MSTL) . . . 64
2.7.2.3 Predictive MTL (P-MTL) . . . 64
2.7.2.4 Predictive Mission-time Linear Temporal Logic (PMLTL) . . . 65
2.7.2.5 Temporal Equilibrium Logic (TEL) . . . 65
2.7.2.6 Metric Equilibrium Logic (MEL) . . . 66
2.7.2.7 Linear Temporal Public Announcement Logic (LTPAL) . . . 66

2.7.2.8 Accumulative Signal Temporal Logic (ASTL) 67
2.7.2.9 Probabilistic Signal Temporal Logic (ProbSTL) 67
2.7.2.10 Evolution Temporal Logic (EvTL) 68
2.7.2.11 Robustness Temporal Logic (RobTL) 69
2.7.2.12 Signal Diagnostic Processing Language (SDRL) 69
2.7.2.13 TRIO-based Event Specification Language (TESLA) 70
2.7.2.14 Temporal Action Logic (TAL) 70
2.7.3 SR Query Languages 71
2.7.3.1 Continuous SPARQL (C-SPARQL) 71
2.7.3.2 Streaming and Temporal ontology Access with a Reasoning-based Query Language (STARQL) 72
2.7.3.3 $\text{SPARQL}_{\text{stream}}$ 73
2.7.3.4 Continuous Temporal SPARQL (CT-SPARQL) 73
2.7.3.5 ETALIS Language for Events (ELE) 74
2.7.3.6 EP-SPARQL 74
2.7.3.7 Continuous Query Evaluation over Linked Streams (CQELS) 75
2.7.3.8 RSP-QL and RSP-QL* 75
2.7.4 ASP-based Formalisms 76
2.7.4.1 Continious ASP (C-ASP) 76
2.7.4.2 Logic-based framework for Analytic Reasoning over Streams (LARS) and its extensions 77
2.7.4.3 Probabilistic ASP (PrASP) 78
2.7.5 Miscellaneous Formalisms 78
2.7.5.1 Temporal Datalog 79
2.7.5.2 Event Calculus for Run-Time reasoning (RTEC) 79
2.8 Survey Results: Usage Dimension 80
2.8.1 DL-based Formalisms 80
2.8.2 Temporal Logic Formalisms 83
2.8.3 SR Query Languages 85
2.8.4 ASP-based Formalisms 87
2.8.5 Miscellaneous Formalisms 87
2.9 Discussion 88
2.10 Conclusion & Future Perspectives 94
References 96
Appendix A: Key Papers 107

3 Bridging Metric and Window-Based Stream Reasoning: Translation Between DatalogMTL and Logic-based framework for Analytic Reasoning over Streams 109
3.1 Abstract 110
3.2 Introduction 110
3.3 Background 113
3.3.1 DatalogMTL 113
3.3.2 LARS 117
3.4 Comparative prelude 122
3.5 Translating DatalogMTL into LARS 122

3.5.1 Preliminary Lemmas on DatalogMTL Semantics . . . 126
3.5.2 Correctness of the Translation by Structural Induction . . . 127
3.5.2.1 Since & Until Operators . . . 127
3.5.2.2 Existential Metric Operators . . . 131
3.5.2.3 Universal Metric Operators . . . 131
3.5.2.4 Assembling the Inductive Argument . . . 134
3.5.3 Lifting the Translation from Atoms to Rules and Programs . . . 134
3.5.4 Preservation of Program Semantics and Minimal Models . . . 137
3.5.5 Translating DatalogMTL Domain Knowledge – a Working Example . . . 138
3.5.6 Syntactic Restrictions and Translation Size Guarantees . . . 139
3.6 Translating LARS into DatalogMTL . . . 140
3.6.1 Representing LARS Streams and Structures in DatalogMTL . . . 140
3.6.2 A Translatable Fragment of LARS . . . 142
3.6.3 Translating LARS Formulae to DatalogMTL Metric Atoms . . . 144
3.6.4 From LARS Rules to DatalogMTL Programs . . . 146
3.6.5 Preservation of Models and Minimality . . . 147
3.6.6 Translating LARS Domain Knowledge – a Working Example . . . 149
3.6.7 Expressivity Limitations and Asymmetry of the Correspondence . . . 150
3.7 Discussion and Future Work . . . 151
3.8 Conclusion . . . 152
References . . . 153
Appendix A: Proof for Existential Metric Operators . . . 155

4 Efficient Temporal Reasoning with Non-Temporal Engines: Embedding DatalogMTL into Datalog 159
4.1 Abstract . . . 160
4.2 Introduction . . . 160
4.3 Related Work . . . 162
4.4 Background . . . 164
4.5 Translating DatalogMTL to Datalog$_{\mathbb{Q}}$ . . . 166
4.6 Implementation and Experiments . . . 173
4.7 Conclusion and Future Work . . . 176
References . . . 177

5 On Syntactical Simplification of Temporal Operators in Negation-free Metric Temporal Logic 183
5.1 Abstract . . . 184
5.2 Introduction . . . 184
5.3 Materials and Methods . . . 187
5.4 Results . . . 190
5.5 Discussion . . . 195
5.6 Conclusion . . . 197
References . . . 198

6 Conclusions 201

6.1 Consolidating the Contributions of this Dissertation 201
6.1.1 Research challenge C1: Semantic Fragmentation in Stream Reasoning formalisms 202
6.1.2 Research challenge C2: Foundational Choices in Temporal Semantics, Negation and Their Consequences 203
6.1.3 Research challenge C3: Bridging Theoretical Expressivity and Practical Reasoning Infrastructure 204
6.1.4 Overarching insights 205
6.2 Implications for Stream Reasoning Research and Practice 206
6.2.1 Implications for the Design of Stream Reasoning Formalisms 206
6.2.2 Implications for SR Engines and Implementations 207
6.2.3 Implications for Interoperability and Standardisation 208
6.3 Limitations, Open Questions, and Future Directions 209
6.3.1 Scope Limitations and Design Choices 209
6.3.2 Open Theoretical Questions 210
6.3.3 Directions for Future Research 210
6.4 Final Outlook 211
References 212

List of Figures

1.1 Overview of the IDC FutureScape: Worldwide IT Industry 2024 Top 10 Predictions, illustrating anticipated timing and relative cost/complexity of major IT trends for 2024–2029. Reproduced from [29]. 2
1.2 Illustration of a distributed patient monitoring scenario where streaming data from local sources is processed across edge, hospital, and cloud systems, and temporal reasoning results are exchanged between them. 11

2.1 Amount of papers found by Google Scholar on search term 'Stream Reasoning' between 2000 and 2024, grouped by year of publication. 28
2.2 Number of papers that remains after each step of the defined protocol 32
2.3 Overview of evaluation dimensions 44

3.1 Illustration of DatalogMTL dataset $\mathcal{D}_{\text{eg}}$ 115
3.2 Illustration of time-based windows $\tau^{0,30(1)}$ and $\tau^{7,7(7)}$ applied to stream S_{eg} at time 181. Each window application results in a substream of stream S_{eg}. 119

4.1 Facts derived by Rules (4.1) and (4.2) 167
4.2 Graphic representation of temporalisation of RDF triples 173
4.3 Average execution time on iTemporal benchmark 175

5.1 Illustration of Proof 5.2. The accolades of parts 1 and 2 have a set length of $i_2 - i_1$, but shift according to the exact location of t^- and t^+. Regardless of the exact location of t^- and t^+, the two accolades will always produce an area spanning at least $[t + i_1, t + i_2]$ in which A holds everywhere. 192
5.2 Expressibility dependencies between temporal operators under closed-world (left) and open-world (right) assumptions. Blue edges denote previously known dependencies, while green edges denote newly established dependencies. 196

List of Tables

1.1 Correspondence between Challenges, Research Questions and Chapters 18

2.1 Approximate amount of results per keyword between 2009 and 2024, as retrieved from Google Scholar 30
2.2 Linear Temporal Logic (LTL) future temporal operators 37
2.3 LTL past temporal operators 38
2.4 Summary of the T1 (theoretical basis) and T3 (expressivity of static relations) dimensions for SR formalisms 53
2.5 Summary of the T4 dimension (temporal expressivity) for the different surveyed formalisms. — Part A 54
2.6 Summary of the T4 dimension (temporal expressivity) for the different surveyed formalisms. — Part B 55
2.7 Summary of the T2 dimension (time and stream representation) for SR formalisms. — Part A 56
2.8 Summary of the T2 dimension (time and stream representation) for SR formalisms. — Part B 57
2.9 Summary of the T5 dimension (uncertainty in the data and in the stream) for SR formalisms. — Part A 58
2.10 Summary of the T5 dimension (uncertainty in the data and in the stream) for SR formalisms. — Part B 59
2.11 Summary of citations (U1) and software prototypes (U2) of SR formalisms. — Part A 81
2.12 Summary of citations (U1) and software prototypes (U2) of SR formalisms. — Part B 82
2.13 Summary of dimension U3, listing the applications per formalism, grouped per application domain. Formalisms without application-based papers in the considered literature are omitted from the table. — Part A 83
2.14 Summary of dimension U3, listing the applications per formalism, grouped per application domain. Formalisms without application-based papers in the considered literature are omitted from the table. — Part B 84

3.1 **Comparative summary of design elements in DatalogMTL and LARS.** 123
3.2 Correspondence γ of DatalogMTL metric atoms to semantically equivalent LARS formulae. 124

3.3 Correspondence Δ of LARS rule heads to DatalogMTL rule heads. 143
3.4 **Comparative summary of expressibility in DatalogMTL, LARS, and the newly defined LARS metric fragment based on the established translations.** 153

5.1 Semantics of Metric Temporal Logic (MTL) formulae 189

List of Acronyms

SR Stream Reasoning

CEP Complex Event Processing

SLD Streaming Linked Data

SW Semantic Web

FOL First-Order Logic

DL Description Logics

CWA Closed World Assumption

OWA Open World Assumption

TBox Terminological Box

ABox Assertional Box

ASP Answer Set Programming

RDF Resource Description Framework

RSP RDF Stream Processing

OWL Web Ontology Language

QoS Quality-of-Service

UAV Unmanned Aerial Vehicle

IoT Internet-of-Things

LP Logic Programming

IRI Internationalized Resource Identifier

NAF Negation-as-Failure

SPARQL SPARQL Protocol and RDF Query Language

EHDS European Healthcare Data Space

HT Here-and-There Logic

EL Equilibrium Logic

LTL Linear Temporal Logic

MTL Metric Temporal Logic

BMTL Bounded MTL

TQL Temporal Query Language

PAL Public Announcement Logic

MITL Metric Interval Temporal Logic

MSTL Metric Spatio-Temporal Logic

RCC-8 Region Connection Calculus

P-MTL Predictive MTL

TEL Temporal Equilibrium Logic

THT Temporal Here-and-There Logic

MLTL Mission-time Linear Temporal Logic

PMLTL Predictive Mission-time Linear Temporal Logic

MEL Metric Equilibrium Logic

MHT Metric Here-and-There Logic

LTPAL Linear Temporal Public Announcement Logic

STL Signal Temporal Logic

ASTL Accumulative Signal Temporal Logic

ProbSTL Probabilistic Signal Temporal Logic

EvTL Evolution Temporal Logic

RobTL Robustness Temporal Logic

SDRL Signal Diagnostic Processing Language

TESLA TRIO-based Event Specification Language

TAL Temporal Action Logic

C-SPARQL Continuous SPARQL

STARQL Streaming and Temporal ontology Access with a Reasoning-based Query Language

CT-SPARQL Continuous Temporal SPARQL

EP-SPARQL Event Processing SPARQL

CQELS Continuous Query Evaluation over Linked Streams

ETALIS Event-Driven Transaction Logic Inference System

ELE ETALIS Language for Events

C-ASP Continious ASP

LARS Logic-based framework for Analytic Reasoning over Streams

wLARS Weighted LARS

PrASP Probabilistic ASP

RTEC Event Calculus for Run-Time reasoning

SLR Synchronized Logical Reasoning Language

List of Symbols

$\mathbb{N}$ Natural numbers.

$\mathbb{Z}$ Integers.

$\mathbb{Q}$ Rational numbers.

$\mathbb{R}$ Real numbers.

T Time domain.

t, t', t_i Time points.

P, Q, R Predicate symbols.

x, y, z Variables.

c Constant.

τ Term.

$\bar{\tau}$ Tuple of terms.

$\mathcal{I}$ Interpretation.

$\mathcal{M}$ Structure or model.

S Stream.

S^* Global stream.

$S' \subseteq S$ Substream of a stream.

w Window function.

$\boxplus^w$ Window operator with window function w.

Π Logic program.

D Dataset.

$H(r)$ Head of rule r.

$B(r)$ Body of rule r.

$\models$ Satisfaction relation.

$\not\models$ Non-satisfaction relation.

$\equiv$ Semantic equivalence.

$\subseteq$ Subset relation.

$\cup$ Set union.

$\cap$ Set intersection.

$\wedge$ Conjunction (AND-operator)

$\vee$ Disjunction (OR-operator)

$\neg$ Negation (NOT-operator)

$\rightarrow$ Implication

$\Box$ Universal temporal modality.

$\Diamond$ Existential temporal modality.

$\mathcal{U}$ Until operator.

$\mathcal{S}$ Since operator.

$\boxplus$ Future universal metric operator.

$\boxminus$ Past universal metric operator.

$\dotplus$ Future existential metric operator.

$\dotminus$ Past existential metric operator.

$@_t$ Holds-at operator at time point t.

$\bigcirc$ Next-time point operator.

$\bullet$ Previous-time point operator.

Samenvatting

Veel hedendaagse informatiesystemen werken niet langer met statische datasets die éénmalig geanalyseerd en opgeslagen worden. In plaats daarvan verwerken zij datastromen: continue reeksen van observaties die in de tijd arriveren en een voortdurend veranderende wereld weerspiegelen. Voorbeelden hiervan zijn sensormetingen van draagbare toestellen, updates afkomstig van geconnecteerde infrastructuur, activiteitslogs van softwaretoepassingen en monitoringsdata in domeinen zoals gezondheidszorg, slimme steden, financiën en industriële automatisering. In dergelijke contexten wordt van systemen verwacht dat zij binnenkomende gegevens onmiddellijk interpreteren en tijdig beslissingen ondersteunen die reële gevolgen kunnen hebben.

Technologieën voor stream processing zijn zeer succesvol gebleken in het aanpakken van de technische uitdagingen van dergelijke datastromen. Zij maken het mogelijk om gegevens efficiënt te filteren, aggregeren en transformeren in real time. Veel toepassingen vereisen echter meer dan louter dataverwerking. Zij vereisen redenering over evoluerende gegevens, waarbij observaties geïnterpreteerd worden in samenhang met contextuele en domeinspecifieke kennis, doorgaans vastgelegd in declaratieve regels. Deze regels worden geëvalueerd over datastromen om betekenisvolle situaties te herkennen en conclusies te trekken die verder gaan dan de ruwe data zelf, bijvoorbeeld om te bepalen of een toestand voldoende lang aanhoudt, of een tijdelijke afwijking genegeerd kan worden, of of de afwezigheid van bepaalde observaties informatief is.

Om aan deze noden te beantwoorden is een brede familie van benaderingen ontstaan onder de noemer Stream Reasoning. Stream Reasoning combineert ideeën uit regelgebaseerde systemen, temporele logica en kennisrepresentatie om inferentie over evoluerende data mogelijk te maken. Door redeneringsmechanismen te integreren met modellen voor datastromen beogen deze benaderingen een hogere interpretatielaag te bieden die de kloof overbrugt tussen ruwe datastromen en betekenisvolle conclusies.

Naarmate Stream Reasoning zich heeft ontwikkeld, is een grote variëteit aan formalisms voorgesteld, elk gedreven door specifieke toepassingsvereisten en ontwerpdoelen. Sommige benaderingen leggen de nadruk op responsiviteit en concentreren zich op recente gegevens, terwijl andere focussen op expressiviteit en redenering over expliciete temporele intervallen in combinatie met rijkere achtergrondkennis. Deze diversiteit heeft brede toepassing mogelijk gemaakt, maar heeft tegelijk geleid tot een fundamentele uitdaging: het ontbreken van een gedeelde semantische basis.

De kern van deze uitdaging ligt in semantische diversiteit. Stream Reasoning-systemen verschillen niet alleen in implementatie, maar ook in hoe zij betekenis vastleggen. Zij hanteren uiteenlopende aannames over de representatie van tijd, de relevantieduur van informatie, de interpretatie van ontbrekende of vertraagde observaties en de onderscheidingen die zij kunnen uitdrukken. Deze aannames zijn diep verweven met de onderliggende redeneringsformalisms. Daardoor kunnen twee systemen dezelfde datastromen analyseren en vergelijkbare doelen nastreven, maar toch moeilijk vergelijkbare conclusies produceren omdat die conclusies steunen op verschillende semantische kaders.

Verschillen in temporele interpretatie spelen hierin een bijzonder prominente rol. Sommige Stream Reasoning-benaderingen maken gebruik van schuivende vensters die redenering beperken tot recente delen van de datastroom, terwijl andere werken met expliciete temporele duur- of intervalrepresentaties en conclusies laten afhangen van hoe condities zich over langere perioden ontwikkelen. Beide strategieën zijn waardevol, maar vertolken verschillende noties van persistentie, recentheid en relevantie. Wanneer systemen met verschillende temporele abstracties interageren, is het vaak onduidelijk hoe conclusies uit het ene systeem geïnterpreteerd moeten worden door het andere.

Aanvullende complexiteit ontstaat door de wijze waarop systemen omgaan met de afwezigheid van informatie. In sommige contexten wordt het ontbreken van een observatie geïnterpreteerd als bewijs dat een gebeurtenis niet heeft plaatsgevonden, terwijl dit in andere contexten als inconclusief wordt beschouwd, bijvoorbeeld door vertraagde communicatie of gedeeltelijke observatie. Deze keuzes beïnvloeden of conclusies onmiddellijk getrokken worden, uitgesteld blijven of later herzien worden, en bepalen of redenering monotone of niet-monotone eigenschappen vertoont. In dynamische en gedistribueerde omgevingen hebben dergelijke beslissingen belangrijke praktische implicaties.

Naast semantische vraagstukken bestaat er ook een spanningsveld tussen theoretische expressiviteit en praktische inzetbaarheid. Veel Stream Reasoning-benaderingen met rijke temporele semantiek zijn gerealiseerd in gespecialiseerde of experimentele redeneringsengines. In de praktijk vertrouwen systemen echter vaak op volwassen infrastructuren die sterk inzetten op robuustheid, schaalbaarheid en langdurig onderhoud, maar geen native ondersteuning bieden voor expressieve temporele redenering. Dit bemoeilijkt de inzet van geavanceerde Stream Reasoning-technieken zonder operationele garanties op te offeren.

Deze dissertation benadert deze problematiek door een fundamenteel perspectief op Stream Reasoning te ontwikkelen dat de nadruk legt op semantiek in plaats van uitvoering. In plaats van een nieuw redeneringssysteem of -taal voor te stellen, worden bestaande benaderingen geanalyseerd met als doel hun aannames expliciet en vergelijkbaar te maken. Een centrale methodologische keuze is het benaderen van Stream

Reasoning vanuit declaratieve redenering, waarbij systemen specificeren welke conclusies geldig moeten zijn onder bepaalde voorwaarden, terwijl de concrete berekening daarvan wordt overgelaten aan onderliggende engines.

Door redenering te verankeren in formele semantiek, zoals modellen en logische gevolgtrekking, maken declaratieve benaderingen het mogelijk om betekenis los te koppelen van implementatiedetails. Dit vergemakkelijkt het vergelijken en relateren van redeneringssystemen, zelfs wanneer zij gebruikmaken van verschillende algoritmes of infrastructuren. Vanuit dit semantische uitgangspunt levert deze dissertation vier belangrijke bijdragen.

Ten eerste biedt zij een gestructureerde karakterisering van het Stream Reasoning-landschap, waarin terugkerende semantische dimensies worden blootgelegd over een sterk uiteenlopende verzameling formalisms. De studie toont aan dat de waargenomen fragmentatie in het veld systematisch is en voortkomt uit verschillen in aannames over tijd, stroomrepresentatie en entailment, wat verklaart waarom interoperabiliteitsproblemen vaak optreden ondanks het gebruik van gedeelde data.

Daarnaast behandelt de dissertation twee complementaire aspecten van temporele semantiek. Enerzijds wordt een formele correspondentie vastgesteld tussen metrische en venstergebaseerde temporele redenering door transleerbare fragmenten van DatalogMTL en LARS te identificeren waarvoor temporele entailment en minimalemodelsemantiek behouden blijven. Dit verduidelijkt onder welke voorwaarden redeneringsresultaten uit verschillende temporele abstracties betekenisvol aan elkaar gerelateerd kunnen worden. Anderzijds onderzoekt de dissertation temporele redenering onder open-wereld-aannames via de studie van negatievrije fragmenten van Metric Temporal Logic. Door aan te tonen dat temporele expressiviteit behouden kan blijven met een gereduceerde set operatoren wanneer negatie wordt uitgesloten, wordt een robuuste expressieve kern geïdentificeerd voor omgevingen waar afwezigheid van informatie niet als falsiteit kan worden geïnterpreteerd.

Ten slotte adresseert de dissertation de kloof tussen expressieve redenering en praktische infrastructuur door aan te tonen dat bepaalde vormen van temporele redenering realiseerbaar zijn op bestaande, niet-temporele redeneringsplatformen. Een semantiekbehoudende vertaling naar Datalog laat zien dat geavanceerde temporele redenering kan worden ingezet met volwassen engines zonder formele garanties te verliezen. Dit onderstreept dat verbeterde interoperabiliteit niet noodzakelijkerwijs vraagt om het verlaten van gevestigde systemen, maar kan worden bereikt via zorgvuldig ontworpen semantische mappings.

Over het geheel genomen beargumenteert deze dissertation dat vooruitgang in Stream Reasoning meer vereist dan het toevoegen van nieuwe functies of het optimaliseren van prestaties. Zij vereist helderheid over betekenis. Door de semantische aannames onderliggende Stream Reasoning-formalisms expliciet en analytisch toegankelijk te maken, wordt het mogelijk om systemen te bouwen die redeneringsresultaten op

een principiële wijze kunnen uitwisselen en interpreteren. De gepresenteerde resultaten dragen zo bij aan Stream Reasoning-systemen die niet alleen expressief en efficiënt zijn, maar ook interoperabel en geschikt voor inzet in complexe, reële omgevingen.

Summary

Many modern information systems no longer work with static datasets that can be analysed once and stored away. Instead, they operate on streams of data: continuous flows of observations that arrive over time and reflect an evolving world. Examples include sensor readings from wearable devices, updates from connected infrastructure, activity logs from software systems, or monitoring data in application domains, such as healthcare, smart cities, finance, and industrial automation. In such settings, systems are expected to interpret incoming data as it arrives and to support timely decisions that may have direct real-world consequences.

Technologies for stream processing have proven highly effective at handling the technical challenges of such data. They offer scalable and efficient mechanisms for filtering, aggregating, and transforming data streams in real time. However, many applications require more than the ability to process data values. They require reasoning over evolving data, in which incoming observations are interpreted in light of contextual and domain knowledge, typically formalised as declarative rules, and these rules are evaluated over streams to recognise meaningful situations and derive conclusions that go beyond the data values themselves. For example, it may be necessary to determine whether a condition has persisted long enough to warrant attention, whether a temporary anomaly can be ignored, or whether the absence of an expected observation is informative in itself.

To address such needs, a broad family of approaches known as Stream Reasoning has emerged. Stream Reasoning combines ideas from rule-based systems, temporal logic, and knowledge representation to enable inference over evolving data. By integrating reasoning capabilities with streaming data models, these approaches aim to support higher-level interpretation of continuous information and to bridge the gap between raw data streams and meaningful conclusions.

As Stream Reasoning has developed, a wide range of approaches has been proposed, each motivated by different application requirements and design goals. Some approaches prioritise responsiveness and focus on recent data, while others emphasise expressivity and allow reasoning over explicit temporal durations and richer background knowledge. This diversity has enabled Stream Reasoning to be applied in many contexts, but it has also introduced a fundamental challenge: the lack of a shared semantic foundation across approaches.

At the heart of this challenge lies semantic diversity. Stream Reasoning systems differ not only in how they are implemented, but in how they express meaning. They make different assumptions about how time is represented, how long information remains relevant, how missing or delayed observations are interpreted, and which distinctions can be expressed at all. These assumptions are deeply embedded in the reasoning formalisms themselves. As a result, two systems may analyse the same data streams and pursue similar objectives, yet arrive at conclusions that are difficult to compare or combine because those conclusions are grounded in different semantic frameworks.

Differences in how time is interpreted play a particularly prominent role. Some Stream Reasoning approaches rely on sliding windows, which restrict reasoning to a recent portion of the data stream, such as the last few seconds or minutes. Other approaches reason in terms of explicit temporal durations or intervals, allowing conclusions to depend on how conditions evolve over longer periods. Both strategies are useful, but they embody different notions of persistence, recency, and relevance. When systems employing different temporal abstractions interact, it is often unclear how conclusions obtained under one interpretation should be understood under another.

Further complexity arises in how systems handle absence of information. In some settings, the absence of a reported event may be interpreted as evidence that the event did not occur. In other settings, the same absence may be treated as inconclusive, reflecting delayed communication or partial observability. Such differences affect whether conclusions are drawn, deferred, or later revised as new information becomes available. They also influence whether reasoning behaves in a monotonic or non-monotonic manner, that is, whether conclusions can be invalidated when additional data arrives. In dynamic, distributed environments, these choices have significant practical consequences.

Beyond semantic considerations, there is a further tension between theoretical expressivity and practical deployment. Many Stream Reasoning approaches that support rich temporal semantics or sophisticated inference mechanisms are realised in specialised or experimental reasoning engines. In contrast, real-world systems often rely on mature, well-tested infrastructures that prioritise robustness, scalability, and long-term maintainability, but do not natively support expressive temporal reasoning. This mismatch makes it difficult to deploy advanced Stream Reasoning capabilities in practice without sacrificing operational guarantees.

This dissertation addresses these issues by developing a foundational perspective on Stream Reasoning that focuses on semantics rather than execution. Rather than proposing a new reasoning language or system, it analyses existing approaches with the aim of making their assumptions explicit and comparable. A key methodological choice is to view Stream Reasoning through the lens of declarative reasoning, in which systems specify what conclusions should hold under given conditions, while leaving the details of how those conclusions are computed to the underlying engines.

By grounding reasoning in formal semantics, such as models and logical entailment, declarative approaches provide a principled way to define meaning independently of implementation. This makes it possible to compare and relate different reasoning systems, even when they use different algorithms or infrastructures. Such a semantic perspective is essential for addressing interoperability, as it shifts attention from how systems compute results to what those results actually mean. Building on this foundation, this dissertation makes four main contributions.

First, it provides a structured characterisation of the Stream Reasoning landscape that reveals recurring semantic dimensions across a highly diverse set of formalisms. The survey shows that apparent fragmentation in the field follows systematic patterns rooted in differing assumptions about time, stream representation, and entailment, and explains why interoperability failures typically arise even when systems operate over shared data.

Furthermore, the dissertation analyses two complementary aspects of temporal semantics. For a second contribution, it establishes a formal correspondence between metric-based and window-based temporal reasoning by identifying translatable fragments of DatalogMTL and LARS for which temporal entailment and minimal-model semantics are preserved. This result clarifies under which conditions reasoning outcomes obtained under different temporal abstractions can be meaningfully related.

Third, it investigates temporal reasoning under open-world assumptions by studying negation-free bounded fragments of Metric Temporal Logic. The dissertation characterises expressivity-preserving relationships between temporal operators in these fragments and shows that several bounded temporal modalities can be expressed using a smaller operator basis centered around 'Until' and 'Since'. These results clarify the expressivity structure of negation-free temporal reasoning under open-world assumptions and identify a reduced operator core for bounded fragments of Metric Temporal Logic.

Fourth, the dissertation addresses the gap between expressive reasoning and practical infrastructure by showing how certain forms of temporal reasoning can be realised using existing, non-temporal reasoning platforms. A semantics-preserving translation to Datalog demonstrates that advanced temporal reasoning can be deployed using mature engines without sacrificing formal guarantees. This contribution demonstrates that improved interoperability does not necessarily require abandoning established systems, but can be achieved through carefully designed semantic mappings.

Overall, this dissertation argues that advancing Stream Reasoning requires more than adding new features or optimising performance. It requires clarity about meaning. In this work, the semantic assumptions underlying Stream Reasoning formalisms are made explicit and amenable to analysis. By studying how different reasoning approaches relate at the foundational level, it becomes possible to build systems that can exchange and interpret reasoning results in a principled

way. The results presented here contribute toward Stream Reasoning systems that are not only expressive and efficient, but also interoperable and suitable for deployment in complex, real-world environments.

1 Introduction

Modern information systems are increasingly asked to make sense of the world as it unfolds. Rather than analysing complete datasets in isolation, they must interpret continuous streams of observations that evolve over time, originate from many independent sources, and often require immediate action. In such settings, correctness no longer depends solely on how efficiently data is processed, but on how meaning is attributed to evolving information across systems. This chapter introduces the conceptual landscape in which Stream Reasoning (SR) operates, identifies the key challenges that arise from semantic diversity and decentralisation, and formulates the research questions that guide this dissertation.

1.1 Semantic Interoperability in a Streaming, Decentralised World

A streaming reality. Today's digital systems increasingly operate in a streaming world. Data no longer arrives as a static dataset that can be stored, queried, and analysed at leisure, but as a continuous flow of observations generated over time. Sensor readings from wearable devices, updates from connected infrastructure, interactions on the Web, and monitoring data from distributed services all arrive incrementally and must often be acted upon as they are produced [2, 16, 20]. In domains such as healthcare, smart cities, and industrial monitoring, systems must interpret and act on this data as it is produced, frequently under time pressure and in safety-critical settings [29, 35]. The IDC FutureScape: Worldwide IT Industry 2024 Predictions

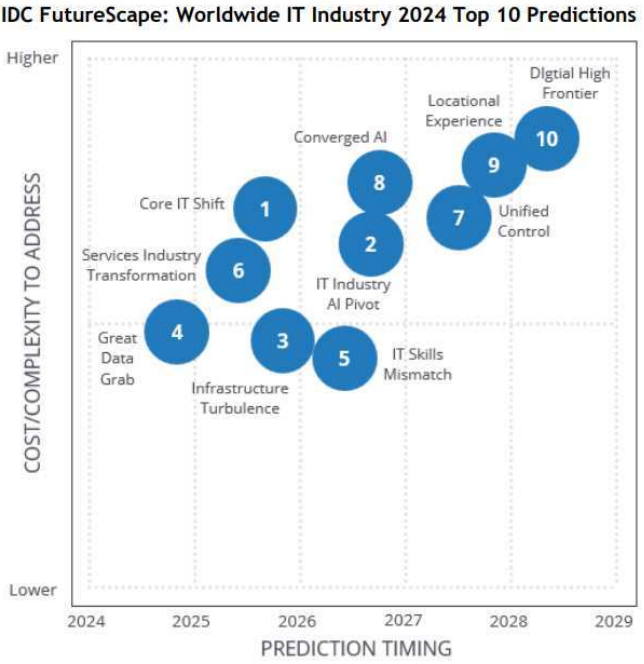


Figure 1.1: Overview of the IDC FutureScape: Worldwide IT Industry 2024 Top 10 Predictions, illustrating anticipated timing and relative cost/complexity of major IT trends for 2024–2029. Reproduced from [29].

report [29] identifies 10 global predictions for the period 2024-2029, depicted in Figure 1.1. These predictions, in particular 'Unified Control' (7), 'Converged AI' (8) and 'Digital High Frontier' (10), strongly emphasize the need for integration and coordination of AI-enabled, high-velocity data infrastructure.

What makes these Internet-of-Things (IoT) and healthcare environments particularly challenging is not only the volume or velocity of data, but the conditions under which the data is produced and consumed. Data streams rarely originate from a single, centrally managed source. Instead, they are generated by many autonomous components that differ in location, purpose, reliability, and temporal granularity, and that often reflect device-specific or vendor-specific design choices. In a modern healthcare setting, for example, patient data may be produced by wearable sensors at home, bedside monitors in hospitals, mobile applications used by patients and caregivers, and Web services that enrich observations with background knowledge.

Such components are introduced incrementally, evolve independently, and are rarely designed as part of a single, centrally controlled system [10, 16]. Processing is therefore increasingly distributed across edge devices, local infrastructure, and cloud services, driven by constraints such as latency, privacy, and robustness, with sensitive data often processed close to its source before being shared or aggregated [13].

Heterogeneity as a semantic problem. As a result, modern data-driven systems are inherently heterogeneous and decentralised. Individual components typically operate with only a partial and asynchronous view of the overall system state. Data may arrive at different rates, be delayed or reordered, be pre-processed locally, or be integrated with background knowledge at a later stage. When information or reasoning results are exchanged across components, implicit assumptions about relevance, validity, and interpretation can easily clash. Importantly, these mismatches are not merely engineering artefacts or implementation flaws [15]. They frequently arise from differences in how components interpret the meaning of the data they process [27].

This tension becomes particularly tangible once systems are expected to do more than store, forward, or aggregate data, and are instead required to reason about time. Many applications require the detection of temporal patterns, the recognition of situations that unfold gradually, or the assessment of whether a condition persists long enough to justify action. In healthcare, for instance, distinguishing a transient anomaly from a sustained deterioration can depend on minutes, and on how those minutes are interpreted by different parts of the system. Some components may reason explicitly about time points, durations, or intervals, while others may restrict attention to recent data using sliding windows [7, 11]. Both approaches are widely used and often appear interchangeable at an intuitive level, yet they embody different assumptions about temporal relevance, persistence, and absence of information.

The challenge is further amplified by the rise of autonomous and agent-based systems, as well as by emerging data-sharing initiatives such as data spaces [28]. Modern software agents operate continuously, maintain state over time, and coordinate with other agents based on partial and evolving information. In healthcare in particular, initiatives such as the European Health Data Space [19] aim to connect decentralised data holders, including hospitals, care providers, devices, and services, across organisational and national boundaries while allowing each participant to retain control over its data and systems. In such open and heterogeneous environments, components cannot assume complete knowledge, global synchronisation, or a shared temporal model [26]. Each system is therefore entitled to its own interpretation of time and information, shaped by local constraints, governance requirements, and objectives. When these components interact, interoperability failures may arise not because data cannot be exchanged, but because the conclusions drawn from that data rely on incompatible semantic assumptions.

(Stream) reasoning to enable semantic interoperability. Reasoning is concerned with drawing conclusions from given information by following explicitly defined rules. Rather than working with numbers or raw values, reasoning treats information as structured facts and relations and determines which conclusions must hold whenever certain conditions are satisfied [1]. A simple example is that if a system knows that every patient with condition A requires treatment B, and it

observes that a specific patient has condition A, then it can infer that treatment B applies to that patient, even if this was not stated explicitly. This form of inference relies on a clear notion of meaning [14]. Facts are not interpreted in isolation, but in terms of concepts, relations, and assumptions that define how they are meant to be understood. Within knowledge representation formalisms, this meaning is captured formally, making it possible to reason about information in a precise and predictable way rather than relying on informal interpretation [3]. In the context of the Semantic Web, this idea is realised through representation languages such as RDF, RDFS, and OWL, which attach explicit semantics to data by defining the meaning of classes, properties, and relationships [30, 31]. Reasoning engines can then use these semantics to derive information that is implicit in the data, to check whether stated facts are mutually consistent, and to support higher-level interpretations that go beyond what is directly asserted [23].

However, classical reasoning approaches were primarily developed for static or slowly changing datasets. They assume a relatively stable view of the world and typically operate on fixed knowledge bases. In modern streaming environments, these assumptions no longer hold. Data arrives continuously, observations may become outdated quickly, and conclusions may need to be revised as new information appears [16]. Traditional reasoning systems are not designed to interpret evolving situations, react in near real-time, and coordinate across decentralised components that each observe only part of the overall system state [15].

SR extends the principles of declarative reasoning to this dynamic setting. It provides formal mechanisms to reason over continuously evolving data while preserving explicit semantic foundations [10]. By combining logical inference with temporal models and stream semantics, SR extends the ideas and approaches of reasoning over static data to fast-paced streams of data [7, 33]. This furthermore adds an additional, temporal dimension to reasoning: evolving data enables one to reason about persistence, temporal patterns, and absence of observations [15].

As reasoning is extended to account for temporality, it also becomes applicable in settings where information is produced and interpreted by multiple, independently operating components, such as IoT and healthcare systems [27]. When data is generated continuously and processed at different locations, conclusions are no longer derived from a single, shared snapshot of the world, but from partial and evolving views of incoming streams. SR makes the semantic assumptions underlying such conclusions explicit, in particular with respect to time, relevance, and inference over changing information [7, 12]. By formalising these assumptions, SR allows conclusions drawn by one component to be understood, assessed, and related to those drawn by another, even when they rely on different temporal abstractions or operate under different local constraints. In this sense, SR not only enriches reasoning with a temporal dimension, but also provides the formal basis required to analyse and align reasoning outcomes across decentralised systems operating over shared data streams.

Metric-based and window-based temporal reasoning. SR formalisms differ fundamentally in how they model and exploit time. Two dominant paradigms can be distinguished: metric-based and window-based temporal reasoning. While both are used to reason over evolving data streams, they embody different assumptions about temporal relevance, persistence, and the scope of inference.

Metric-based approaches reason directly over an explicit time domain, typically using time points, intervals, or durations as first-class semantic objects. Temporal conditions are formulated in terms of metric constraints, such as how long a property has held or within which time interval an event must occur. This enables precise reasoning about persistence, overlap, and temporal distance, and is commonly used in temporal logics and rule-based formalisms that build on explicit models of time.

Window-based approaches, in contrast, delimit reasoning by selecting a finite portion of the stream relative to the current evaluation time, such as the most recent observations or a fixed number of past events. Reasoning is then performed over this dynamically changing substream, abstracting away from absolute time in favour of recency-based relevance. Window-based semantics are widely adopted in practice, as they align naturally with incremental processing, bounded memory, and low-latency requirements in streaming and edge environments.

Although both paradigms are often used to express intuitively similar temporal notions, for example detecting sustained conditions or recent anomalies, they differ in how temporal meaning is constructed and how conclusions depend on the surrounding context. These differences are not merely syntactic or operational, but reflect distinct semantic choices about what it means for information to remain relevant over time. Understanding how these paradigms relate is therefore essential for comparing, translating, and interoperating between SR systems.

Diversity without alignment. Over the past fifteen years, a wide range of SR formalisms has been proposed, combining ideas from temporal logic, rule-based reasoning, and Semantic Web technologies, each tailored to diverse application domains and design goals. While this diversity is a strength, it makes interoperability far from guaranteed. When systems rely on different semantics, identical sensor readings can lead to incompatible conclusions. For example, in a patient care network, an edge device, operating on window semantics, may infer improvement because it observes the heart rate within the last hour, i.e. a 1-hour window, and did not detect any anomaly. Meanwhile, a hospital system may reason using metric-based temporal semantics that explicitly account for how often and how long anomalies occurred over a longer period, such as multiple days, and may therefore infer the patient is at risk for heart problems. Although these approaches often appear interchangeable in informal descriptions, they rely on fundamentally different temporal abstractions, making it unclear how conclusions drawn by one component should be interpreted by another [7, 33].

Systems operating on streams must decide which information is relevant at a given moment, how long it should remain relevant, and how to treat delayed or missing observations. These decisions are shaped by underlying assumptions about time, relevance, and information completeness. When systems rely on different semantics, identical sensor readings can lead to incompatible conclusions. A hospital system may conclude that a patient is not receiving oxygen therapy because no corresponding record appears in its stream, whereas an edge component monitoring the same patient may refrain from drawing this conclusion, treating the absence of data as inconclusive due to possible communication delays or partial observability in the home setting. [10, 23].

Such interpretative challenges are central to many SR approaches. At the same time, they introduce significant sources of semantic variation. Reasoning over unbounded streams requires explicit choices about temporal reference, persistence of information, and the interpretation of absence. In open and distributed environments, assumptions, such as global synchronisation, complete knowledge of the stream, or closed-world interpretation, often do not hold. As a consequence, components deployed in different contexts may rely on different semantic conventions, even when they aim to reason about the same underlying phenomena [15, 24].

Fundamental challenge & goal of the dissertation. This semantic diversity exposes the fundamental challenge addressed in this dissertation:

> Systems may reason over the same data streams and pursue similar high-level objectives, yet produce results that are difficult to compare, combine, or reuse because they are grounded in different temporal semantics.

In heterogeneous and decentralised settings, interoperability therefore depends not only on shared data formats or protocols, but on a principled understanding of how semantic assumptions shape reasoning outcomes [12, 15, 16, 27]. This dissertation starts from that observation and investigates the following overarching goal:

> To establish the formal foundations needed to enable interoperable SR across systems that differ in their semantic assumptions about time, streams, and inference.

1.2 Background on Declarative Programming

This section provides background on declarative programming as a conceptual foundation for the remainder of this dissertation. Many of the SR approaches studied in this work are rooted in declarative formalisms, where reasoning behaviour is defined in terms of formal semantics rather than procedural execution. Consequently, the distinction between specifying what conclusions should follow from given information

and how those conclusions are computed plays a central role in analysing how different SR formalisms can be compared, related, or translated, even when they rely on different engines or infrastructures. The goal of this background section is not to provide an exhaustive introduction to logic or programming languages, but to establish the key ideas needed to support the semantic perspective adopted throughout the dissertation.

Declarative programming. At the core of many SR approaches lies the notion of declarative programming, which provides a principled way to make the semantic assumptions underlying reasoning over streams explicit. Interoperability challenges in SR often stem from differences in how systems interpret time, relevance, and absence of information, rather than from limitations of data processing techniques.

Declarative programming addresses this issue by separating the "what" of a problem specification from the "how" of its computation: instead of prescribing step-by-step procedures, declarative formalisms allow users to state properties, constraints, or rules that describe which conclusions should hold [1]. Given such a specification, the task of deriving conclusions is delegated to a reasoning engine, which is free to realise this process using different computational strategies as long as the intended semantics are preserved [22, 23]. This stands in contrast to imperative or procedural programming approaches, where control flow and execution order are the primary means of expressing system behaviour and where semantic assumptions are often intertwined with implementation details. Declarative programming allows domain experts to focus on describing how their problem space is governed by logic and know-how, instead of focusing on how this should be procedurally implemented to reach correct conclusions [5, 10].

Programs as logical descriptions. With its emphasis on formal description, declarative programming is rooted in formal (mathematical) logic. In many declarative settings, formal logic works with a set of rules, called a *program*, that specifies which facts or conclusions are allowed to hold. Rather than describing how conclusions should be computed step by step, such a program captures relationships between sets of conditions and states which conclusions are considered valid whenever those conditions are met [1]. For a patient, one can formulate a program that considers several conditions, say 'low fluid intake' and 'elevated body temperature', and ties these to a conclusion, say 'risk of dehydration'. The program does not need to prescribe how these observations are detected or monitored. The rule merely specifies the conditions under which the conclusion "risk of dehydration" holds.

In simple terms, a *model* is a concrete way of assigning meaning to the symbols used in a program such that all of its rules are satisfied. One can think of a model as a possible "reality" or configuration of facts that respects everything the program states. Even when the rules are clear, there may be an enormous number of such possible realities, i.e. not just a handful, but potentially 14.000.605 distinct versions

of reality that all satisfy the same set of rules, the program. Reasoning is concerned not with choosing one preferred reality, but with deriving conclusions that are guaranteed to hold in every valid one [3, 14].

Reasoning with models then amounts to asking the question: which facts hold in all such valid configurations? If a fact is true in every model of a program, it is considered a logical consequence of that program. Importantly, models are abstract objects used to define meaning: they are not executions or data structures, but conceptual tools that allow us to reason precisely about what a program expresses, independently of how conclusions are actually computed [14, 23].

Entailment, not execution. An important consequence of this approach is that the meaning of a program is defined independently of any particular execution strategy. Different reasoning engines may use different algorithms, data structures, or optimisation techniques, yet they are considered equivalent if they derive the same conclusions from the same program. This separation between specification and execution allows reasoning behaviour to be compared across systems and formalisms at the level of meaning rather than implementation. In the context of this dissertation, this property is essential, as it enables the study of how different reasoning languages relate semantically, even when they are realised using different algorithms or engines.

The semantic mismatches observed in SR stem not only from system architecture, but from the declarative foundations that govern how observations, absence of information, and temporal meaning are interpreted. Several families of declarative formalisms are relevant in SR, including First-Order Logic (FOL), Description Logics (DL), and rule-based languages such as Datalog. While these formalisms differ in expressive power and computational properties, they share a common characteristic: reasoning outcomes are determined by *logical entailment* rather than by operational execution traces. This makes declarative programming particularly suitable for settings where correctness, transparency, and interoperability are more important than in-depth control over execution [3].

Reasoning in open or closed environments. An important distinction within declarative programming concerns the assumptions made about available information. In many applications, one can assume to have access to all relevant information. In a hospital setting, an oxygen administration record may be assumed to be complete: all delivered treatments are logged in the patient's record, so if no oxygen therapy is recorded, the patient is therefore assumed *not* to be receiving oxygen. These settings are said to be operating under a *Closed World Assumption (CWA)*. The world is considered closed in the sense that the system is assumed to have a complete view of all relevant information. In these closed-world settings, systems typically employ *Negation-as-Failure (NAF)*: if no evidence can be found for a fact

(such as "the patient is receiving oxygen"), the opposite conclusion ("the patient is not receiving oxygen") is drawn [24].

This CWA does, however, not apply to every possible setting. For a variety of reasons, it is not always possible to access all relevant information at every possible moment. Message delays, connectivity issues and memory problems are all circumstances that can deny us access to information [21, 23]. While inspecting a patient's medical records, the absence of a recorded oxygen administration event does not necessarily mean that no oxygen was given; the information may be delayed, recorded by another system, or temporarily unavailable. As such, under an Open World Assumption (OWA), the absence of information does not imply falsity. This distinction between open and closed worlds has significant consequences for how negation and incomplete information are interpreted within a system, especially in dynamic and distributed environments.

When conclusions can change. A closely related distinction concerns whether a reasoning system is *monotonic* or *non-monotonic* [18]. In a monotonic setting, adding new information can never invalidate conclusions that were previously drawn: once a conclusion follows from a program, it will continue to do so as more information becomes available.

In contrast, many practical reasoning settings are non-monotonic [24]. Returning to the oxygen treatment scenario, a system may initially conclude that a patient is not receiving oxygen because no administration has been recorded so far. This conclusion may need to be withdrawn later, when delayed information arrives indicating that oxygen therapy was administered earlier [8, 18]. Here, conclusions depend on assumptions about the completeness of the available information and may change as that information evolves. This behaviour is particularly relevant in dynamic or distributed environments, where information arrives over time and earlier conclusions may need to be revised accordingly [23].

Declarative programming in practice. The Semantic Web (SW) has long advocated the use of explicit, machine-interpretable semantics to address data heterogeneity and centralisation challenges, demonstrating that formal and declarative languages can be successfully deployed in real-world systems [31]. Central to this approach is the use of shared vocabularies and *ontologies*: formal, logic-based descriptions of the concepts in a domain and the relationships between them, enabling data to be interpreted consistently across systems [30]. Beyond classical Semantic Web deployments, these principles are being carried forward into decentralised data-sharing infrastructures, most notably SOLID-based data spaces and European data space initiatives [28, 32]. In SOLID, machine-interpretable vocabularies and policies are used to describe data resources and access conditions across independently controlled pods, while the European Health Data Space explicitly promotes semantic interoperability through shared

standards, ontologies, and governance frameworks to enable cross-border health data exchange [19, 32]. Declarative languages are also gaining renewed attention in multi-agent and distributed systems, where formally specified semantics provide a common ground for coordination, negotiation, and reasoning between autonomous agents [4]. In domains such as healthcare and the IoT, this enables the integration of heterogeneous data streams with background knowledge, including domain models, medical guidelines, or contextual information, while remaining agnostic to the underlying infrastructure. Crucially, these developments illustrate that the rise of data-driven and machine-learning approaches has not displaced formal and declarative methods, but rather reinforced their role as a complementary layer for semantic integration, interpretation, and reasoning in complex, real-world applications.

1.3 Challenges

To reiterate, the overarching goal of this doctoral dissertation is to establish a rigorous semantic foundation for interoperable SR, enabling meaningful exchange of reasoning results between systems that operate over evolving data. To ground this goal in a concrete and realistic setting, this section introduces a running example that will be used throughout the remainder of the challenges: the continuous monitoring of a single patient, Alice, within a distributed healthcare network.

The medical needs of Alice. Alice is monitored as part of a regional healthcare network that spans multiple locations and technological layers, as illustrated in Figure 1.2. At home, Alice wears a smartwatch that continuously measures her heart rate and blood oxygen saturation, and carries a smartphone that records activity levels, such as step counts. During hospital visits, Alice is additionally connected to bedside monitoring equipment that records clinical measurements and interventions, including oxygen administration. Together, these devices generate multiple continuous data streams that describe Alice's evolving physiological state.

The purpose of this distributed monitoring setup is not merely to collect and visualise measurements, but to reason over Alice's evolving physiological state. Such reasoning supports tasks like detecting early signs of deterioration, distinguishing transient anomalies from sustained conditions, and deciding when clinical intervention or follow-up is required. To perform this reasoning, these streams are sent outwards and processed by different components of the healthcare network. Some processing takes place close to the data source, for example on edge devices that provide immediate feedback or alerts based on recent sensor readings. Other components operate within hospital information systems, where longer histories of observations are combined with clinical protocols and medical expertise. Cloud-based services may further enrich incoming streams with background knowledge, such as patient history, risk models, or population-level statistics. Crucially, these components are developed

independently, deployed in different environments, and operate under different performance and reliability constraints.

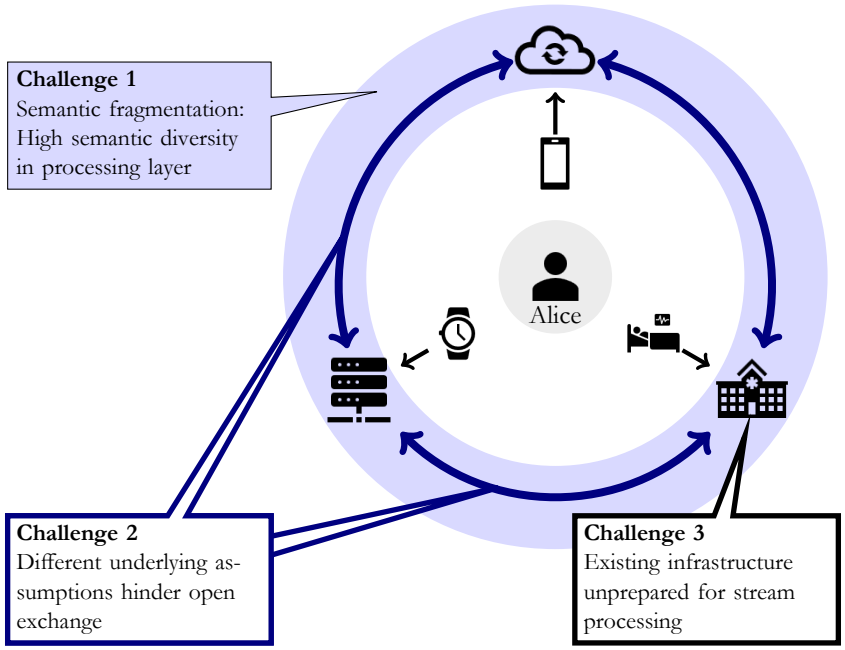


Figure 1.2: Illustration of a distributed patient monitoring scenario where streaming data from local sources is processed across edge, hospital, and cloud systems, and temporal reasoning results are exchanged between them.

1.3.1 Challenge C1: Semantic Fragmentation of SR Formalisms

In practice, interoperability problems rarely stem from data alone, but from the meaning assigned to that data by different systems. Within a distributed application, SR components may interpret the same observations using different reasoning formalisms, since systems are free to adopt whatever semantics best suit their specific purpose rather than conforming to a single shared language for conditions and conclusions. These formalisms diverge in what they can express, the distinctions they make, and the inferences they support. Consequently, conclusions produced by one component cannot be assumed to carry the same meaning for another, even when both operate on overlapping data and pursue similar objectives.

In the healthcare scenario introduced above, semantic fragmentation manifests itself in a very concrete way. Different components of the patient monitoring system, i.e. the monitoring systems, edge-level processing nodes, and cloud-based analytics services, are typically built on different software platforms, developed by different

vendors, and tailored to different operational constraints. As a result, they often employ different SR formalisms, each with its own assumptions about streams, time, and inference. A hospital system may rely on a rule-based temporal logic to detect sustained abnormalities, an edge component may apply window-based SR to provide low-latency alerts, while a cloud-based service may use continuous queries over semantic data to integrate background knowledge from the Web. Although these components aim to reason about the same underlying patient data and support similar clinical goals, they do so using incompatible semantic frameworks.

Letting such components "talk" to each other is therefore anything but straightforward. In the absence of enforceable standardisation at the semantic level, conclusions derived by one component cannot be assumed to have a clear or equivalent meaning for another. A temporal condition detected by an edge component, for example, may not correspond directly to a condition inferred by a hospital system, even if both are phrased informally in similar terms. This lack of semantic alignment complicates the exchange of reasoning results, hinders reuse of specifications, and makes system integration labor-intensive and error-prone.

What this illustrates is that semantic fragmentation in SR is not an incidental byproduct of implementation diversity, but a structural problem at the level of meaning. When different components rely on distinct SR formalisms, there is no principled basis for assuming that similar-looking temporal expressions refer to the same concept, or that they can be safely exchanged and combined. In the absence of shared semantic foundations, integration depends on informal interpretation and manual alignment, making interoperability fragile and hard to scale. This inability to determine when different formalisms agree, differ, or are incomparable constitutes a core obstacle to interoperable SR.

> **C1.** The first challenge addressed in this dissertation is the semantic fragmentation of SR formalisms, which obstructs the principled exchange and reuse of reasoning results across systems.

1.3.2 Challenge C2: Foundational Choices in Temporal Semantics, Negation and Their Consequences

In the Alice patient monitoring scenario, reasoning about Alice's condition depends not only on the incoming data streams, but fundamentally on how time and absence of data are interpreted by different components of the system. Whether an elevated heart rate is considered a brief anomaly or a sustained deterioration depends on how long observations are deemed relevant, how the passage of time is defined, and how gaps in the data are treated. These questions arise routinely in practice, for example when sensor data is delayed, missing, or aggregated at different points in the system. They directly influence the conclusions drawn about Alice's state. Crucially, these

differences do not stem from errors or noise, but from deliberate semantic choices embedded in the reasoning formalisms used by different components.

Two such choices play a particularly decisive role in shaping temporal reasoning. The first concerns how time is modelled and manipulated. In the Alice example, an edge component may reason over a sliding window of recent measurements (e.g., "heart rate above threshold in the last five minutes"), while a hospital system may reason using explicit metric constraints over time (e.g., "heart rate persistently high for at least 300 seconds"). Although these formulations appear intuitively similar, they rely on different temporal abstractions and can lead to different entailments. The choice between window-based and metric-based temporal semantics therefore induces concrete trade-offs between expressivity, precision, simplicity, and analysability.

The second choice concerns the treatment of negation and incomplete information. Some components may operate under a closed-world assumption, concluding that a condition does not hold when no supporting evidence is available, often realised via negation-as-failure. In contrast, components operating in open environments, as is the case for the Web-based services integrated into Alice's care, cannot safely make such assumptions: missing measurements do not imply absence, but may reflect delayed transmission, partial observability, or distribution across systems. This distinction directly affects which temporal properties can be expressed, how conclusions evolve as new data arrives, and which reasoning fragments remain robust in a distributed healthcare setting.

Taken together, these foundational choices divide SR formalisms into families with distinct semantic commitments, making it essential to understand how such choices shape temporal expressivity and what their consequences are for interoperable reasoning [7, 24, 33].

> **C2.** The second challenge addressed in this dissertation concerns foundational differences in temporal semantics and in the treatment of incomplete information, in particular the lack of formal understanding of how reasoning under different temporal abstractions (such as window-based and metric-based semantics) and different assumptions about negation aligns.

1.3.3 Challenge C3: Bridging Theoretical Expressivity and Practical Reasoning Infrastructure

A different, but equally constraining tension arises when expressive reasoning requirements meet real-world deployment constraints. In the Alice patient monitoring scenario, conclusions about Alice's condition may trigger alerts to clinicians, adaptations of care plans, or actions by autonomous agents coordinating care across settings. Such decisions place strict requirements on the underlying software infrastructure: systems

must be scalable, robust, responsive, and reliable over long periods of continuous operation. At the same time, the reasoning tasks involved, i.e. detecting temporal patterns, reasoning over evolving state, and handling incomplete information, often require expressive temporal semantics that go beyond simple stream processing.

Many SR formalisms capable of expressing such temporal semantics are realised in specialised or prototype reasoning engines, while deployed infrastructure in hospitals and clinical networks typically relies on mature, non-temporal reasoning platforms. This reflects a research trend within SR, where the novelty and expressivity of new temporal semantics of the formalism lead to the development of specialised software architectures. As a result, expressive temporal reasoning is frequently tied to bespoke engines, e.g. MeTeoR [34], Laser [6] and TALPlanner [17], while existing reasoning infrastructure remains underutilised. This stands in contrast to mature, non-temporal reasoning systems such as RDFox [25] and Vadalog [9], which already provide highly optimised, even commercial-grade platforms with features such as efficient indexing, incremental materialisation, parallel execution, and robustness guarantees. This mismatch forces practitioners to choose between expressive temporal reasoning and trusted infrastructure, making it difficult to deploy advanced SR capabilities without sacrificing operational guarantees. In the Alice example, hospital information systems and cloud-based analytics are far more likely to rely on mature infrastructure than on experimental SR engines.

This mismatch poses a concrete barrier to adoption in safety-critical domains such as healthcare. Even when expressive temporal reasoning would be desirable, deploying prototype-level engines alongside production systems is often infeasible [33]. Bridging theoretical expressivity and practical reasoning infrastructure therefore becomes a central challenge: temporal reasoning capabilities must be realised in a way that allows them to be embedded into existing, trusted software platforms, rather than requiring wholesale replacement of infrastructure [25]. Addressing this challenge is essential to ensure that advances in SR theory can translate into scalable, sustainable, and interoperable systems in real-world settings such as the Alice monitoring scenario [9].

> **C3.** The third challenge addressed in this dissertation is to bridge the gap between theoretical expressivity and practical reasoning infrastructure, enabling expressive SR without sacrificing deployment viability.

1.4 Research Questions and Hypotheses

In order to address the three distinct challenges above and to contribute to a coherent formal theory for interoperable SR, I connect each of the aforementioned challenges to several concrete research questions addressed in this dissertation.

1.4.1 RQ1: Characterising Semantic Fragmentation in SR

The first challenge C1 addressed in this dissertation concerns the fragmented landscape of SR formalisms. Due to the high diversity in SR approaches, it is difficult to compare existing SR formalisms or to understand under which conditions their reasoning outcomes are semantically compatible [12].

> **Research RQ1.** Which core semantic dimensions are required to systematically characterise and compare SR formalisms?

Addressing semantic fragmentation requires more than introducing new reasoning languages or execution models. Instead, it calls for a principled way of analysing existing formalisms in terms of their underlying semantic assumptions. Without such a framework, it remains unclear whether differences between SR approaches reflect fundamental semantic divergence, alternative encodings of similar concepts, or trade-offs aligned with specific usage scenarios.

To investigate this question, I conduct a systematic analysis of SR formalisms proposed since the inception of the field. This analysis examines both formal definitions and reported usage characteristics, focusing on aspects such as stream modelling choices, temporal interpretation, entailment behaviour, and reasoning objectives. By studying representative approaches drawn from different research traditions, including rule-based reasoning, temporal logics, and Semantic Web technologies, I aim to identify recurring semantic patterns as well as systematic sources of divergence. Based on this investigation strategy, I formulate the following hypotheses regarding the nature of semantic diversity in SR.

> *Hypothesis H1.* The diversity of existing SR formalisms can be captured using a finite and recurring set of semantic dimensions, rather than requiring a bespoke characterisation for each individual approach.

> *Hypothesis H2.* The identified dimensions allow us to trace whether the differences in formalisms are primarily attributable to variations in underlying semantic assumptions, such as time representation, stream modelling, and entailment behaviour, or to superficial syntactic or implementation-level differences.

> *Hypothesis H3.* I hypothesise that the identified semantic dimensions allow for a systematic assessment of whether a single SR formalism can subsume the semantics of others across all dimensions, or whether different formalisms necessarily embody trade-offs that make them more or less suitable for particular classes of applications and reasoning objectives.

1.4.2 RQ2: Relating Metric-Based and Window-Based Temporal Semantics

Temporal interpretation lies at the heart of SR and constitutes one of the two foundational contrasts identified in Challenge C2. While Challenge C2 highlights both temporal semantics and the treatment of incomplete information as sources of semantic divergence, this research question focuses specifically on the former.

As mentioned in Section 1.1, two dominant temporal paradigms are prevalent in SR: window-based approaches and metric-based approaches. Although these paradigms are often used to express intuitively similar temporal conditions, their underlying semantics differ substantially, which complicates the comparison and interoperability of reasoning results across systems that adopt different temporal abstractions.

> **Research Question RQ2.** Under which conditions can window-based and metric-based temporal semantics be formally related in a semantics-preserving manner?

Understanding the relationship between these paradigms is essential for clarifying when conclusions drawn under one temporal abstraction can be meaningfully interpreted under another.

To investigate this question, I study representative SR formalisms that embody window-based and metric-based temporal semantics. I analyse their formal definitions, with particular attention to temporal operators, evaluation contexts, and entailment conditions. Based on this analysis, I construct formal translations between restricted fragments of the respective formalisms and examine whether and under which assumptions key semantic properties are preserved.

> *Hypothesis H4.* I hypothesise that well-defined subfragments of window-based and metric-based temporal formalisms can be delineated such that relationships can be devised between these fragments that enable semantics-preserving translations, while retaining sufficient expressivity to support practically relevant temporal reasoning tasks.

1.4.3 RQ3: Temporal Reasoning under Open-World Assumptions

This research question addresses the second foundational dimension identified in Challenge C2, namely the treatment of negation and incomplete information in temporal reasoning. While RQ2 focuses on aligning different temporal abstractions, RQ3 investigates how temporal reasoning behaves when conclusions must be derived under open-world assumptions, where absence of information does not imply falsity.

Many SR applications operate in open and distributed environments in which complete knowledge of the system state cannot be assumed. In such settings, reasoning paradigms that rely on CWA or non-monotonic negation may lead to conclusions

that are fragile under delayed or partial observations. This motivates the study of temporal reasoning fragments that avoid reliance on negation and remain robust under OWA, where conclusions are derived solely from available positive information.

Investigating this question across all existing SR formalisms is infeasible. Instead, I study this issue in the setting of Metric Temporal Logic (MTL), a well-established temporal logic that underpins several SR frameworks. By focusing on negation-free fragments of MTL, I examine whether temporal expressivity can be preserved when negation operators are eliminated or pushed out, and which temporal operators remain essential under this restriction. Insights obtained at this level can serve as a template for understanding similar trade-offs in other temporal reasoning frameworks.

> **Research Question RQ3.** Can negation be eliminated from metric temporal logics in a semantics-preserving manner, so as to support expressive temporal reasoning under open-world assumptions?

Clarifying this question is essential for both theoretical and practical reasons. From a theoretical perspective, it contributes to a clearer understanding of the expressive core of temporal reasoning under open-world assumptions. From a practical perspective, it informs the design of SR languages and systems that avoid reliance on closed-world reasoning while retaining sufficient expressivity to capture relevant temporal patterns.

The investigation focuses on whether temporal expressivity can be preserved when negation is excluded, by analysing how temporal operators relate to one another under this restriction and whether certain operators can be eliminated without loss of semantic power.

> *Hypothesis H5.* I hypothesise that negation-free fragments of MTL can be characterised using reduced sets of temporal operators that preserve temporal expressivity, making them suitable for temporal reasoning under open-world assumptions.

1.4.4 RQ4: Realising SR on Non-Temporal Reasoning Infrastructure

Challenge C3 raises the question of whether expressive SR necessarily requires specialised temporal engines, or whether temporal semantics can instead be realised on top of existing non-temporal reasoning platforms through suitable semantic transformations.

> **Research Question RQ4.** Can SR semantics be mapped to non-temporal reasoning infrastructure in a semantics-preserving manner?

Table 1.1: Correspondence between Challenges, Research Questions and Chapters

	Hypotheses	Ch1	Ch2	Ch3	
RQ 1	H1, H2, H3	✓			Chapter 2
RQ 2	H4		✓		Chapter 3
RQ 3	H5		✓		Chapter 5
RQ 4	H6, H7			✓	Chapter 4

Answering this question requires addressing two complementary aspects. From a theoretical perspective, it concerns whether fragments of SR semantics can be identified that admit semantics-preserving translations to non-temporal reasoning formalisms. From a practical perspective, it raises the question of whether such fragments retain sufficient expressivity to support realistic reasoning tasks encountered in real-world applications.

To investigate this question, I study formal mappings from expressive SR formalisms to non-temporal reasoning languages. This involves defining translations that encode temporal reasoning constructs into non-temporal representations and analysing whether key semantic properties, most notably entailment and model correspondence, are preserved under these mappings when evaluated by existing reasoning engines. At the same time, I assess the expressivity of the resulting fragments, in order to gauge their applicability to real-life use cases.

> *Hypothesis H6.* I hypothesise that a well-defined fragment of SR semantics can be identified for which semantics-preserving translations to non-temporal reasoning formalisms can be established.

> *Hypothesis H7.* I hypothesise that the fragment identified in H6 retains sufficient expressivity to support temporally rich reasoning tasks encountered in practical SR applications.

1.5 Outline

This doctoral dissertation is structured around a set of core research challenges (C1-C3) and corresponding research questions (RQ1-RQ4) aimed at establishing a semantic foundation for interoperable SR. The work is composed of several chapters, each addressing a specific aspect of this goal. Figure 1.2 and Table 1.1 position the individual chapters with respect to the challenges and research questions introduced earlier in this chapter.

Chapter 2 provides a systematic survey of SR formalisms proposed since the inception of the field. Addressing Challenge C1, RQ1, and hypotheses H1, H2 and H3. This chapter analyses a broad range of formalisms along a set of clearly defined semantic and usage dimensions. The survey exposes recurring design choices, sources

of semantic divergence, and trade-offs between expressivity, complexity, and applicability, thereby offering a structured view of a field that is otherwise fragmented.

Chapter 3 focuses on the relationship between metric-based and window-based temporal semantics, addressing the temporal dimension of Challenge C2, Research Question RQ2, and hypothesis H4. Building on the systematic survey presented in Chapter 2, which identifies DatalogMTL and Logic-based framework for Analytic Reasoning over Streams (LARS) as two of the most prominent and well-founded SR formalisms, the chapter adopts these languages as representative examples of metric-based and window-based SR, respectively. Using DatalogMTL and LARS, the chapter develops bidirectional, semantics-preserving translations between the two approaches, identifying the fragments of each language for which such translations are possible. These translations clarify how entailment and minimal-model semantics can be preserved across paradigms and demonstrate that the identified fragments retain sufficient expressive power to capture temporally rich patterns arising in practical SR use cases.

Chapter 5 returns to foundational questions of temporal expressivity under open-world assumptions as identified in Challenge C2, addressing Research Question RQ3 and hypothesis H5. The chapter studies negation-free fragments of MTL and investigates which temporal operators are essential and which can be eliminated without loss of expressive power. The resulting operator simplifications contribute to a clearer understanding of temporal expressivity and support the development of robust reasoning frameworks suitable for open and distributed environments.

Chapter 4 addresses Challenge C3 and Research Question RQ4 by investigating whether expressive temporal reasoning necessarily requires dedicated temporal reasoning engines. This chapter evaluates the question from both a theoretical and a practical perspective, addressing hypotheses H6 and H7. It presents a semantics-preserving embedding of DatalogMTL into plain Datalog, enabling metric temporal reasoning to be realised on existing, non-temporal reasoning infrastructure. An implementation and experimental evaluation demonstrate the feasibility of this approach and illustrate how theoretical expressivity can be reconciled with practical deployment constraints.

Finally, Chapter 6 concludes the dissertation by summarising the main contributions, revisiting the research challenges, questions, and hypotheses, and reflecting on their implications for interoperable SR. The chapter also outlines open problems and directions for future research.

1.6 Publications

The research results obtained during this dissertation have been published in or are currently under review in scientific journals and presented at different international conferences. This section presents an overview of these publications.

1.6.1 Publications in international journals (A1, listed in the Science Citation Index[1])

1. **Mathijs van Noort**, Femke Ongenae, and Pieter Bonte. *On Syntactical Simplification of Temporal Operators in Negation-Free Metric Temporal Logic.* Published in Mathematics, Volume 14, Issue 7, Article 1124, 2026.

2. **Mathijs van Noort**, Pieter Bonte, and Femke Ongenae. *Bridging Metric and Window-Based SR: Translation Between DatalogMTL and LARS*. Submitted for review to the Theory and Practice of Logic Programming Journal, May 2026.

3. **Mathijs van Noort**, Pieter Bonte, and Femke Ongenae. *Navigating Streams: A Survey on 15 Years of Advances in Stream Reasoning Formalisms*. Submitted for review to the Semantic Web Journal, January 2026.

4. Patrick Hochstenbach, **Mathijs van Noort**, Dörthe Arndt, Rebekka Martens, Jos De Roo, Ruben Verborgh, Pieter Bonte, and Femke Ongenae. *RDF Surfaces: Enabling Classical Negation on the Semantic Web*. Submitted for review to the Semantic Web Journal, June 2024.

1.6.2 Publications in international conferences (P1, listed in the Science Citation Index[2])

1. **Mathijs van Noort** and Przemysław Andrzej Wałęga. *Efficient Temporal Reasoning with Non-Temporal Engines: Embedding DatalogMTL into Datalog.* Accepted for publication in the Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning (KR 2026).

2. Dörthe Arndt, Jos De Roo, Patrick Hochstenbach, Rebekka Martens, Femke Ongenae, and **Mathijs van Noort**. *RDF Surfaces as a First-Order Language for the Semantic Web.* Published in the Proceedings of the International Joint Conference on Rules and Reasoning (RuleML+RR 2024), pages 200–216. Springer, 2024.

[1] The publications listed are recognised as "A1 publications", according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

[2] The publications listed are recognised as "P1 publications", according to the following definition used by Ghent University: P1 publications are proceedings listed in the Conference Proceedings Citation Index – Science or Conference Proceedings Citation Index – Social Science and Humanities of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper, except for publications that are classified as A1.

1.6.3 Publications in other peer-reviewed conferences (C1)

1. **Mathijs van Noort**. *Formalizing Stream Reasoning for a Decentralized Semantic Web*. Accepted as part of the Doctoral Symposium at the Extended Semantic Web Conference (ESWC 2023), pages 277–287. Springer, 2023.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN: 0-201-53771-0.

[2] Tyler Akidau, Slava Chernyak, and Reuven Lax. *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing*. O'Reilly Media, Inc., 2018. ISBN: 978-1-4919-8382-9.

[3] Alessandro Artale and Enrico Franconi. "A survey of temporal extensions of description logics". In: *Annals of Mathematics and Artificial Intelligence* 30.1 (June 1, 2000), pp. 171–210.

[4] Matteo Baldoni et al. "BSPL and Commitments for Modeling Interaction in Agentic AI". In: *Proceedings of the One Day Workshop on Rethinking Multi-Agent Systems in the Era of LLMs*. University of Oxford. Oxford, United Kingdom, 2025, pp. 1–11.

[5] Marcello Balduccini, Yuliya Lierler, and Stefan Woltran, eds. *Logic Programming and Nonmonotonic Reasoning: 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3–7, 2019, Proceedings*. Vol. 11481. Lecture Notes in Computer Science. Cham: Springer, 2019. ISBN: 978-3-030-20527-0.

[6] Hamid R. Bazoobandi, Harald Beck, and Jacopo Urbani. "Expressive stream reasoning with Laser". In: *International Semantic Web Conference*. Springer. 2017, pp. 87–103.

[7] Harald Beck, Minh Dao-Tran, and Thomas Eiter. "LARS: A logic-based framework for analytic reasoning over streams". In: *Artificial Intelligence* 261 (2018), pp. 16–70.

[8] Harald Beck et al. "Stream reasoning with LARS". In: *KI-Künstliche Intelligenz* 32.2 (2018), pp. 193–195.

[9] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. "The Vadalog System: Datalog-based Reasoning for Knowledge Graphs". In: *Proceedings of the VLDB Endowment* 11.9 (2018).

[10] Pieter Bonte et al. "Grounding Stream Reasoning Research". In: *Transactions on Graph Data and Knowledge* 2 (2024), pp. 1–47.

[11] Sebastian Brandt et al. "Querying log data with metric temporal logic". In: *Journal of Artificial Intelligence Research* 62 (2018), pp. 829–877.

[12] Minh Dao-Tran, Harald Beck, and Thomas Eiter. "Contrasting RDF Stream Processing Semantics". In: *Semantic Technology*. Ed. by Guilin Qi et al. Vol. 9544. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 289–298. ISBN: 978-3-319-31675-8 978-3-319-31676-5.

[13] Mathias De Brouwer et al. "Context-aware & privacy-preserving homecare monitoring through adaptive query derivation for IoT data streams with DIVIDE". In: *Semantic Web Journal* (2023).

[14] Jos De Bruijn, Enrico Franconi, and Sergio Tessaris. "Logical reconstruction of normative RDF". In: *1st Workshop on OWL: Experiences and Directions, OWLED 2005*. Vol. 188. CEUR. 2005, pp. 1–9.

[15] Daniele Dell'Aglio et al. "Stream Reasoning: a Survey and Outlook: A summary of ten years of research and a vision for the next decade". In: *Data Science* 1.1-2 (2017), pp. 59–83.

[16] Emanuele Della Valle et al. "It's a streaming world! Reasoning upon rapidly changing information". In: *IEEE Intelligent Systems* 24.6 (2009), pp. 83–89.

[17] Patrick Doherty, Fredrik Heintz, and Jonas Kvarnström. "Robotics, Temporal Logic and Stream Reasoning". In: *International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-19), 14-19 December 2013, Stellenbosch, South Africa*. 2013.

[18] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. "Answer set programming: A primer". In: *Reasoning Web International Summer School*. Springer, 2009, pp. 40–110.

[19] European Commission. *Proposal for a Regulation of the European Parliament and of the Council on the European Health Data Space*. COM(2022) 197 final. May 3, 2022.

[20] Martin Hirzel et al. "Stream Processing Languages in the Big Data Era". In: *ACM Computing Surveys* 51.3 (2018), pp. 1–39. DOI: 10.1145/3299887.3299892.

[21] Patrick Hochstenbach et al. *RDF Surfaces: Enabling Classical Negation on the Semantic Web*. 2024. arXiv: 2406.10659 [cs.AI].

[22] John W. Lloyd. *Foundations of Logic Programming*. 2nd ed. Berlin, Heidelberg: Springer Science & Business Media, 2012. ISBN: 978-3-642-83189-8.

[23] Alessandra Mileo. "Web stream reasoning: From data streams to actionable knowledge". In: *Reasoning Web International Summer School*. Springer, 2015, pp. 75–87.

[24] Alessandra Mileo et al. "Streamrule: A nonmonotonic stream reasoning system for the semantic web". In: *International Conference on Web Reasoning and Rule Systems*. Springer. 2013, pp. 247–252.

[25] Yavor Nenov et al. "RDFox: A highly-scalable RDF store". In: *International Semantic Web Conference*. Springer. 2015, pp. 3–20.

[26] Sari Palojoki, Lasse Lehtonen, and Riikka Vuokko. "Semantic interoperability of electronic health records: Systematic review of alternative approaches for enhancing patient information availability". In: *JMIR Medical Informatics* 12.1 (2024), e53535.

[27] Gözde Ayşe Tataroğlu Özbulak et al. "A comprehensive survey of stream reasoning and its integration with knowledge graphs". In: *Knowledge and Information Systems* (2025), pp. 1–33.

[28] Ruben Verborgh. *Let's talk about pods*. Blog post. Dec. 2022. URL: https://ruben.verborgh.org/blog/2025/08/12/inside-the-insight-economy/.

[29] Rick Villars et al. *IDC FutureScape: Worldwide IT Industry 2024 Predictions*. Tech. rep. US50435423. International Data Corporation (IDC), Oct. 2023.

[30] W3C. *OWL 2 Web Ontology Language*. W3C Recommendation. World Wide Web Consortium (W3C), Dec. 2012.

[31] W3C. *RDF 1.2 Semantics*. W3C Recommendation. 2024.

[32] W3C Solid Community Group. *Solid Protocol*. Tech. rep. World Wide Web Consortium (W3C), 2021.

[33] Przemysław Andrzej Wałęga, Mark Kaminski, and Bernardo Cuenca Grau. "Reasoning over streaming data in metric temporal datalog". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 3092–3099.

[34] Dingmin Wang et al. "Meteor: Practical reasoning in datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.5 (2022), pp. 5906–5913.

[35] Yukiharu Yorifuji et al. *IDC FutureScape: Worldwide Digital Transformation 2022 Predictions*. Tech. rep. US47115521. International Data Corporation (IDC), Oct. 2021.

2

Navigating Streams: A Survey on 15 Years of Advances in Stream Reasoning Formalisms

As discussed in Research Challenge C1, the expansion of Stream Reasoning (SR) research over the past fifteen years has produced a diverse set of formalisms grounded in differing assumptions about time, data, and inference. Developing at the intersection of knowledge representation, temporal logic, and data stream processing, the field has progressed along multiple conceptual trajectories rather than converging on a single framework, which complicates systematic comparison and reuse. This chapter addresses Research Question RQ1 through a survey of SR formalisms published since "It's a Streaming World! Reasoning upon Rapidly Changing Information", which is widely regarded as a foundational reference for articulating the shift from static to evolving knowledge and for motivating the need for formal semantic foundations encompassing inference, temporal interpretation, and stream modelling. Taking its publication as a natural starting point, the survey examines how subsequent work has responded to, refined, or diverged from this initial vision, and provides a structured semantic perspective on the resulting diversity.

Rather than cataloguing approaches in isolation, the chapter organises the literature along a concise set of theoretical and usage dimensions. This analysis addresses Hypothesis H1 by showing that the diversity of SR formalisms can be characterised using a finite and recurring set of semantic dimensions. It further supports Hypothesis H2 by demonstrating that observed fragmentation primarily reflects differences in underlying semantic assumptions, rather than superficial syntactic or implementation-level variation. Finally, the comparative perspective speaks to Hypothesis H3 by assessing whether any single formalism subsumes the semantics of others across these

dimensions, and shows instead that existing approaches embody distinct trade-offs that make them suitable for different classes of applications and reasoning objectives.

⋆ ⋆ ⋆

M. van Noort, P. Bonte, and F. Ongenae

Submitted for review to the Semantic Web Journal, January 2026.

2.1 Abstract

SR has emerged as a crucial paradigm for enabling real-time, intelligent decision-making over dynamic data streams, which are increasingly prevalent in domains such as Internet-of-Things (IoT), edge computing, and decentralised systems. Formalisms for SR are essential because they define the theoretical and practical foundations for reasoning under continuous, time-sensitive conditions. This paper presents a comprehensive review of SR formalisms introduced over the past 15 years, evaluating their evolution and impact. We propose a set of dimensions for assessing SR formalisms, considering both theoretical properties, such as expressivity, underlying paradigm, and stream representation, as well as practical metrics like citation count, software development, and real-world applications. Through a detailed analysis of the literature since 2009, at which the foundational paper "It's a Streaming World" was published, we score the existing SR formalisms based on these dimensions, highlighting the field's considerable progress. Our findings indicate that recent advancements have led to sophisticated SR formalisms capable of tackling increasingly complex reasoning tasks. However, further research remains essential, as no single formalism satisfies all possible requirements. The best choice depends on the specific needs of the intended application. To support this, we provide a broad overview of currently existing formalisms to help practitioners select the most suitable approach. Formalisms like DatalogMTL and LARS stand out for their strong theoretical foundations and promise for supporting advanced applications, while query-based formalisms show potential for addressing intricate reasoning challenges beyond basic querying. Additionally, SR formalisms that integrate RDF streams are particularly well-positioned to enhance interoperability across heterogeneous systems, opening concrete opportunities for deployment in IoT and edge computing scenarios.

2.2 Introduction

SR refers to the ability to perform logical reasoning over data streams in real time, combining the principles of knowledge representation with continuous data

processing. Unlike traditional temporal reasoning, which focuses on reasoning over time-stamped facts in a static dataset, SR addresses the challenges posed by unbounded, rapidly changing streams of information. This distinction is crucial because SR not only considers when events occur but also how to reason efficiently and correctly under strict time constraints. Formal SR approaches enable guarantees regarding correctness, complexity, and optimisation, which are essential for building scalable and reliable systems.

In recent years, reasoning over streaming data has rapidly evolved from a niche research topic into a critical capability for modern data-intensive applications. The explosive growth in both the volume and velocity of data is reshaping how organisations operate, with businesses increasingly relying on real-time streams to drive agility, enhance customer experiences, and improve operational efficiency [132]. Unlike traditional batch processing, streaming data arrives continuously and without a predetermined endpoint, making it virtually unbounded [32, 67]. Moreover, its lifecycle is short: new information often invalidates previous observations, creating a dynamic environment where timely reasoning is essential. These characteristics underscore why SR is not just an extension of temporal reasoning but a necessity for today's data-driven systems, and why robust theoretical foundations are needed to ensure correctness, scalability, and optimisation in reasoning systems.

The year 2009 marks a rise in interest towards the topic of Stream Reasoning. The first Stream Reasoning workshop (SRW 09) was organized in conjunction with the European Semantic Web Conference. This rise can be observed in Figure 2.1, where a clear uptake in "Stream Reasoning" research works from 2009 onwards. Following SRW 09, Della Valle et al. [67] outlines the most prominent challenges within the area of SR, and subsequently suggests multiple routes into which said research should best proceed. One of said proposed routes is the formalisation of the theoretical aspects of SR. A model theory for SR which covers inference methodology, the representation of a stream and of information within a stream had not been thoroughly explored at the time, but greatly facilitates the study of the behaviour and properties of a reasoning system.

The practical interest in SR often lies not in isolated facts but in reasoning over patterns and relationships across time and multiple streams. For example, consider a smart traffic management system: knowing the location of a single vehicle at one moment is of limited use, but reasoning over the evolving positions of thousands of vehicles allows the system to infer congestion patterns, predict traffic jams, and optimise routing decisions. This requires deductive reasoning over temporal and spatial constraints, which goes beyond detection of isolated events and highlights the added value of SR .

Back in 2009, the lack of formalisation of SR's theoretical aspects was cited as a major hurdle to delivering robust solutions for emerging use cases. Formalisms for

SR provide a structured foundation for both theoretical exploration and practical implementation, by describing the different conceptual elements and relations between them in a formal, typically mathematical way. They enable guarantees about correctness and complexity, facilitate optimisation, and ensure that systems remain reliable under scaling and evolving requirements. This is analogous to relational algebra in databases, which underpins SQL's scalability and optimisation [1]. By adhering to formal principles, SR systems can adapt to increasing data volumes and complexity without sacrificing performance or reliability.

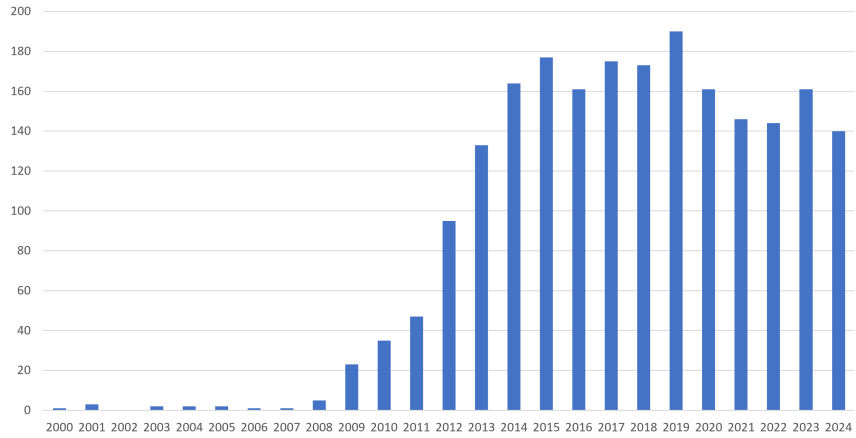


Figure 2.1: Amount of papers found by Google Scholar on search term 'Stream Reasoning' between 2000 and 2024, grouped by year of publication.

The scope of this survey is limited to deductive SR, which focuses on logic-based reasoning over streams [33]. In this context, deductive SR refers to approaches whose behaviour is defined through formal logical semantics, regardless of whether these formalisms are also used for other purposes such as querying, planning, search, or constraint solving. While SR research also includes areas such as Complex Event Processing (CEP), Streaming Linked Data (SLD), and inductive SR approaches (e.g., machine learning), these are outside the focus of this work. To our knowledge, no prior survey has systematically examined SR formalisms at the level of knowledge representation and reasoning.

In this paper, we summarize the current state of the art in SR formalisms, highlighting their strengths and weaknesses. Furthermore, we review their adoption in real-world applications. Collecting and structuring these contributions provides a reliable gateway for new researchers, showcases the current capabilities of the field, and helps identify gaps for future research.

Our paper is structured as follows. Our approach to this literature review is outlined in Section 2.3. The necessary background to understand the formalisms covered

in this paper is grouped in Section 2.4. Section 2.6 details the theoretical & practical usage dimensions that are used to evaluate the different SR formalisms that are subject to our study. The subsequent sections each cover the properties of the different formalisms in function of the described dimensions, where Section 2.7 covers the theoretical dimensions and Section 2.8 covers the usage dimensions. Our overall discussion of the outlined formalisms is bundled in Section 2.9. Lastly, we consider future research directions in the concluding Section 2.10.

2.3 Research Questions and Protocol

To clearly define the scope and methodology of this survey, we first outline the research questions that guide our analysis. These questions establish the focus of our review and ensure that the evaluation of SR formalisms is systematic and objective. Following this, we describe the protocol adopted for conducting the literature review, which provides transparency and reproducibility in our approach.

2.3.1 Research Questions

To pinpoint the exact scope of this survey, we focus on answering the following questions:

Q1. What are the common constructions and expressions of a formalism for SR?

Q2. What are sensible criteria to evaluate a logical formalism, in particular a formalism's suitability for SR?

Q3. Given the criteria from the previous question, how do recently developed SR formalisms meet these criteria and how do these formalisms compare to one another?

Q4. Among the formalisms that have been developed since '*It's a Streaming World*' [67], which have been adopted in real-life applications?

Q5. What areas of SR are currently present, but have been explored only sparsely?

2.3.2 Protocol

Inspired by Brereton et al. [37], who discusses how to set up an adequate literature review within the software engineering domain, we abide by a strict protocol for the execution of the intended systematic literature review. This protocol aims to provide structure to our literature search and to minimize subjectivity.

First, we conduct a preliminary study to explore our search engine of choice, i.e. Google Scholar, with regards to various keywords combinations. We selected Google

Scholar as our primary source because it provides comprehensive coverage of academic publications across multiple disciplines, including computer science and semantic technologies. Compared to alternatives such as Scopus or Web of Science, Google Scholar offers broader indexing of conference proceedings and preprints, which are highly relevant for SR research [107].

To do so, we identified a total of 14 key papers in the field of SR. This set of papers is constructed by taking the papers that cite 'It's a Streaming World' by Della Valle et al. [67] and selecting the most cited paper from each year from 2010 up until 2021, where the upper bound of 2021 reflects the inception of this survey's research work. These 12 papers are further supplemented by two prominent survey works by Barbieri et al. [24] and Falzone, Tommasini, and Della Valle [75]. The full list of papers can be consulted in Appendix 2.

To determine an appropriate search strategy, we first examined broad primary keywords commonly used to describe reasoning over temporally evolving data, such as "Stream Reasoning" and "Temporal Reasoning". This initial test showed that the broader terms returned an excessively large number of results. In particular, "Temporal Reasoning" yielded approximately 28.000 hits, covering a wide conceptual space that includes static temporal reasoning and temporal description logics rather than reasoning over streams. Even the more targeted terms "Reasoning over time" (∼2.090 results) and "Reasoning about time" (∼7.370 results) still produced literature whose scope extended beyond SR and led to substantial noise in the result set.

To narrow the scope further, we then combined these primary keywords with secondary terms related to formal specification, such as "formalism", "formal framework" or "formal semantics". Examples included queries such as "Stream Reasoning AND formalism". While this greatly reduced the number of results, it also proved too restrictive: key papers were retrieved only when the query was tailored almost exactly to those papers, indicating that such combinations lacked the generality required for a systematic survey and risked missing relevant contributions.

Based on these observations, the keyword that achieved the best balance between scope, size of the result set, and consistency with the key papers was the general term "Stream Reasoning". It was specific enough to avoid the conceptual drift present in broader temporal-reasoning keywords, yet broad enough to retrieve the central literature in the field without relying on overly narrow query formulations.

Table 2.1: Approximate amount of results per keyword between 2009 and 2024, as retrieved from Google Scholar

Keyword	Search results
'Stream Reasoning'	∼ 2.420
'Temporal Reasoning'	∼ 28.900
'Reasoning over time'	∼ 2.090
'Reasoning about time'	∼ 7.370

To subsequently gather the material relevant to our topic, the following protocol to identify relevant formalisms is followed:

(1) All search results (listed by Google Scholar) for the term "Stream Reasoning" are taken into consideration.

(2) The set of papers is reduced using a fixed set of boundary conditions to ensure that they are both informative and sufficiently recent:

- Given the stimulus to the field of SR provided by Della Valle et al. [67], we consider research articles from the year 2009 up until (and including) 2024.
- Only scientific articles or PhD theses are considered. Other resources, such as slideshows and extended abstracts, are not taking into consideration due to their limited capacity to serve as a stand-alone source of information.
- Search results that are collections of multiple articles, such as conference proceedings and journal editions, are not included as a whole. In case of relevant articles within these collections, we assume these will turn up in the search results individually.

(3) Based on an article's title, keywords, and abstract, it is evaluated in which context(s) the term "Stream Reasoning" is mentioned with respect to its relevance to the defined research questions above. It is consequently either included in our final set of articles or discarded. Papers are discarded when the term "Stream Reasoning" is mentioned only tangentially or in a non-technical context, such as in introductions without formal contributions. This ensures that the final set includes only works that provide substantive insights into SR formalisms or their theoretical foundations, maintaining the relevance and rigor of the review.

(4) Given the set of remaining articles after application of steps (1) - (3), an article cited by one of the selected articles that was not included in the initial search results, is additionally included into the final set if it provides essential underpinning for the citing paper and meets the criteria of steps (2) and (3).

The resulting amount of papers throughout each step of the protocol is shown in Figure 2.2.

Lastly, we reconsider the set of all search results obtained after step (2) to identify systems and applications based on the formalisms selected through our protocol. For each of the papers, we examine whether the paper covers systems or real-life applications that implement or rely on one of the formalisms extracted via steps (1) - (4). This step aims to gauge the degree of adoption and research popularity of the identified formalisms.

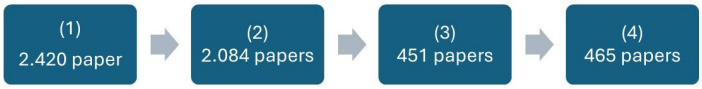


Figure 2.2: Number of papers that remains after each step of the defined protocol

2.4 Preliminaries

Below, we introduce several key concepts from the field of SR. This includes the notion of a data stream and the different ways this can be formally represented. Furthermore, we discuss several techniques to express pieces of data. Most notably, we summarize the most common ways of shaping and expressing temporal information.

2.4.1 Streams and time

The concept of a 'data stream' plays a central role in SR. Despite this, the concept can be interpreted in multiple different ways, depending on the context and intended purpose. Nonetheless, most works on SR agree on several different factors that stipulate a data stream and associated concepts. We outline this common basis below.

2.4.1.1 Time Representation

Throughout the SR literature, a 'data stream' is unanimously understood as a collection of data which arrives sequentially over a period of time [33, 66, 67]. The sequential nature implies a certain order of the data. Said order can be imposed directly on the data itself, i.e. pieces of data A, B and C arrive back-to-back in sequence (A, B, C). Alternatively, time can be made explicit via the use of a *time domain*; an ordered set of numbers used to represent different points in time. The arrival of pieces of data is subsequently indicated using time points drawn from the time domain. Via the order of the time domain, an ordering of the (arrival of) the pieces of data follows.

Typical choices for a *time domain* include the sets $\mathbb{N}, \mathbb{Q}$ and $\mathbb{R}$, along with their natural order $\leq$. Discrete domains, such as $\mathbb{N}$ and $\mathbb{Z}$, can be considered more 'manageable', as the amount of time points is countable and, in case of bounded intervals over the domain, finite. Comparatively, dense domains, such as $\mathbb{Q}$ and $\mathbb{R}$, may better represent the continuous flow of time. Such a representation, with infinite time points over every considered interval, may however greatly increase the complexity of a reasoning system.

2.4.1.2 Representation of a stream

As considered, a data stream encompasses a collection of data which arrives sequentially over a period of time. Time can be modelled as a time domain T in several different ways. We cover the most prevalent ways below;

First, given the 'collection' aspect of a data stream, one can opt to represent a data stream S as a timestamped set (or bag) of data points, for example $S = \{(t, \alpha)\}$, where $t \in T$ and α denotes a piece of information. Alternatively, to incorporate the sequential aspect directly, a stream can be represented as an ordered sequence, such as $S = (\alpha_1, \alpha_2, \ldots)$, with α_i pieces of information. A third perspective views a stream as a function that maps time points from a time domain T to the data arriving at those points, expressed as $S : T \rightarrow \Omega$,where Ω denotes the universe of all possible pieces of information.

Throughout this paper, we refer to these elemental pieces of information as *atoms*, with Ω representing the set of all possible atoms. These representations illustrate the flexibility in modelling streams within SR. While a formalism does not need to adopt one of these forms exactly, most approaches adhere to one of these underlying principles. For clarity, we distinguish four broad categories of stream representation:

- timestamped collections (e.g. sets)
- ordered sequences without explicit timestamps
- functions mapping time to atoms that hold true at said time
- other representations that deviate from the above

2.4.1.3 Windowing

In real-time scenarios where SR is deployed, there is typically no knowledge of when a data stream will deliver its final element and terminate. Consequently, streams are often treated as theoretically infinite. This infinite nature not only conflicts with the finite storage resources of reasoning engines, but also with the practical need to produce timely answers rather than waiting indefinitely for all data to arrive. To resolve these challenges, engines may partition streams into finite segments called windows, a process known as *windowing* [67]. A window represents a manageable subset of the stream, enabling reasoning over recent or relevant data. The most common windowing strategies are *time-based* and *tuple-based* windows, which slide over the stream to extract sub-streams. Time-based windows select data within a specific time interval, while tuple-based windows select a fixed number of elements. Another way to characterize windows is by how they progress over the stream rather than by what they cover. *Sliding* windows advance continuously, allowing overlapping segments, whereas *tumbling* windows move in discrete steps, producing consecutive non-overlapping segments. By choosing an appropriate windowing strategy, reasoning engines can process data incrementally, ensuring timely results while still considering enough context for accurate reasoning. For illustration, consider a time-based sliding window of 10 seconds that updates every second, or a tumbling window that processes events in discrete 5-second blocks—both strategies enable reasoning engines to deliver incremental results without waiting for the entire stream to finish.

2.4.2 Static relations

As SR is a field of study with a wide range of application areas, the exact information that is carried by the streams can vary greatly. RDF triples and predicate expressions are only two types of information that can be carried by a stream. We discuss several elemental structures and relations of these pieces of information commonly used within the SR environment. Here, our focus is on relations that do not consider a temporal dimension, which we refer to as *static relations*. This in contrast to temporal relations, discussed in Section 2.4.3.

2.4.2.1 First-Order Logic, Description Logics and Public Announcement Logic

First-Order Logic (FOL) [140] represents information as logical expressions, often called propositions, which evaluate to either true or false, making it a Boolean logic. FOL uses several logical connectives to express relations between pieces of information:

1. conjunction $\wedge$ expresses the concurrence of two pieces of information, i.e. they hold simultaneously.

2. negation $\neg$ indicates the contrary of a given fact, i.e. when expression ϕ is true, $\neg\phi$ is false and vice versa.

3. disjunction $\vee$ expresses that one or both out of two facts hold.

4. $\phi \rightarrow \psi$, the implication, specifies that when ϕ holds, so does ψ. This construction is often employed to express a rule, stating that when certain conditions (the rule body) are met, the conclusion (the rule head) holds.

5. $\forall$, the universal quantification, conveys a considered expression ϕ holds for all possible values for a specified variable x.

6. $\exists$, the existential quantification, denotes the expression ϕ holds for at least one (yet unspecified) instance for the given variable x.

When formulae use only the first four operators, the logic is called propositional logic. Its abstract nature has made FOL widely used in reasoning systems. Furthermore, FOL includes equivalences such as:

$$\phi \wedge \psi = \neg(\neg\phi \vee \neg\psi) \qquad \forall x\phi = \neg(\exists x \neg\phi)$$
$$\phi \vee \psi = \neg(\neg\phi \wedge \neg\psi) \qquad \exists x\phi = \neg(\forall x \neg\phi)$$
$$\phi \rightarrow \psi = \neg\phi \vee \psi$$

These equivalences show that some operators can be omitted without losing expressivity. For example, $\phi \vee \psi$ can be consistently replaced by $\neg(\neg\phi \wedge \neg\psi)$ to eliminate every occurrence of the $\vee$ operator. Likewise, it is possible to omit $\rightarrow$, one of the two operators $\wedge$ and $\vee$ and one of the quantifiers $\forall$ and $\exists$ without loss of expressivity. For example, a logic that only considers operators $\neg, \wedge, \exists$ is able to construct a "semantic equivalent" for each formula constructed using all of the above FOL operators.

Full FOL is undecidable for logical consequence derivation, so fragments are often used to ensure decidability. Description Logics (DL) are such fragments, designed to balance expressivity and computational tractability. DL underpins many Semantic Web (SW) technologies [133, 135] and divides knowledge into two parts: the Terminological Box (TBox) (terminological knowledge, e.g., class hierarchies and constraints) and the Assertional Box (ABox) (assertional knowledge, e.g., individuals and their properties). DL-Lite is a lightweight member of this family, limiting the syntax for efficient query answering while maintaining decidability [47], making it particularly relevant for SR applications. Epistemic logic is a form of modal logic focused on reasoning about knowledge and belief in multi-agent systems. Public Announcement Logic (PAL) [130] builds on epistemic logic by modelling how knowledge changes after truthful announcements. For example, $K_a\phi$ means "agent a knows ϕ." Introspection ($K_a\phi \rightarrow K_a(K_a\phi)$) means if an agent knows something, it also knows that it knows it. Truthfulness ($K_a\phi \rightarrow \phi$) means agents cannot know something false. PAL adds $[!\phi]\psi$, meaning "after announcing ϕ, ψ holds," which updates the knowledge state by removing worlds where ϕ is false.

2.4.2.2 Set theory

Instead of writing a single logical expression such as for FOL, one can group elements sharing a property into sets. For example, if C is the set of all sensors reporting temperature and D is the set of sensors reporting humidity, then:

- union $C \cup D$ = sensors reporting temperature or humidity,
- intersection $C \cap D$ = sensors reporting both,
- $\bar{C}$ sensors not reporting temperature,
- $C \subseteq D$ = all temperature sensors also report humidity.

This clarification specifies what C and D contain. These operators resemble logical connectives ($\wedge, \vee, \neg, \rightarrow$), but their semantics differ across formalisms.

2.4.2.3 Answer Set Programming (ASP)

ASP is a declarative problem-solving paradigm rooted in non-monotonic logic programming. Unlike traditional logic programming, which typically relies on procedural

interpretation of rules, ASP adopts a model-theoretic approach where solutions correspond to stable models (also called answer sets) of a logic program. This paradigm was introduced by Marek and Truszczyński [104] as an alternative to classical logic programming, emphasizing reasoning under incomplete or evolving information.

An ASP program consists of a finite set of rules of the form:

$$a \leftarrow b_1, \cdots, b_m, \text{not} c_1, \cdots, \text{not} c_n$$

Here, a, b_i, c_j are atoms, and "not" denotes negation-as-failure, a key feature of non-monotonic reasoning. Intuitively, the rule states that a is true if all b_i are true and none of the c_j can be proven true. This differs from classical negation ($\neg$), as "not" reflects the absence of evidence rather than explicit falsity. ASP programs can express defaults, exceptions, and constraints naturally, making them suitable for complex reasoning tasks.

The semantics of ASP is based on the idea of finding self-consistent sets of beliefs. Given a set of facts as a candidate solution, to check if this set is acceptable, you ask: "If I assume these facts are true, do they justify themselves according to the rules?" If the answer is yes, and no smaller set would work, then this set is a stable model. In other words, an answer set is a collection of facts that does not contradict the program and does not rely on unsupported assumptions. This intuitive notion replaces the need for procedural execution with a declarative interpretation: the program describes constraints, and the solver finds all consistent worlds that satisfy them. ASP supports expressive constructs such as choice rules, constraints, and aggregates, enabling concise modelling of combinatorial problems. For example, a scheduling problem can be encoded by rules that generate candidate assignments and constraints that eliminate invalid ones. The solutions correspond to answer sets satisfying all constraints. Unlike traditional inference, ASP does not aim to derive all logical consequences; instead, it computes complete models that represent possible worlds consistent with the program. The theoretical foundation of ASP lies in Here-and-There Logic (HT) and Equilibrium Logic (EL), which generalise classical logic to accommodate non-monotonic reasoning [73]. HT introduces a three-valued interpretation (here, there, and classical truth), while EL characterizes stable models as equilibrium points in this extended semantic space. These logics provide a rigorous basis for ASP's semantics and guarantee properties such as consistency and minimality. ASP has become a powerful tool for knowledge representation and reasoning, particularly in domains requiring default reasoning, planning, and decision-making under uncertainty. Its declarative nature, combined with efficient solvers like Clingo [78], makes it practical for real-world applications, including SR scenarios where evolving data and constraints must be handled dynamically. For a more in-depth discussion of ASP, we refer to the primer by Eiter, Ianni, and Krennwallner [73].

2.4.3 Expressing temporal relations

The section above considers the truthfulness of information in a static context, i.e. a piece of information, e.g. "His name is John", is assumed to be true, or it is not. Information within a SR environment, however, often has a temporal dimension tied to it, i.e. a measurement is measured at a certain time, a person's address can change or a traffic light can change colour. As a result, the veracity of a piece of information is in part dependent on the moment it is considered.

Based on this 'temporal dimension' of the data, we consider two types of reasoning over data streams. On the one hand, a system can be oblivious to temporal connections between the data in the stream. These systems 'collect' a stream's data as it arrives, but do not take into account whether a data piece ϕ arrived before or after data piece ψ. We say these systems perform *reasoning over time*. On the other hand, there can be systems that do take into account the temporal dimension of the data by observing that data piece ψ arrived after data piece ϕ, a system can perhaps infer valuable (new) information, e.g. $(\psi \text{ after } \phi) \rightarrow \chi$. For these types of systems, we say the system performs *reasoning about time*.

When information can be true at a given point in time, several formulations follow intuitively, i.e. some pieces of information hold regardless of the point in time, other pieces hold over only a limited time interval. A system that wishes to incorporate this type of temporal expressions can do so in different manners.

As discussed in Section 2.4.2, FOL provides a foundation for expressing static relations but lacks constructs for time-dependent truth. To overcome this limitation, temporal logics extend FOL with operators that explicitly capture temporal aspects of data. Linear Temporal Logic (LTL), introduced by Pnueli [117], is one such extension. As a temporal logic, LTL has several additional operators over FOL, summarized in Table 2.2 and Table 2.3 for future and past temporal operators respectively. In short, $\bigcirc\phi$ and $\bullet\phi$ express that ϕ holds at the next, respectively the previous, point in time. $\phi\mathcal{U}\psi$ states ϕ holds until ψ holds, i.e., it stops holding as soon as ψ appears, likewise for $\phi\mathcal{S}\psi$. $\diamondsuit\phi$ and $\boxplus\phi$ express that ϕ holds at some point, respectively at all points in time, in the future. $\blacklozenge$ and $\boxminus$ are their mirror operators towards the past.

Table 2.2: LTL future temporal operators

Symbol	Name
$\bigcirc$	'next' operator
$\mathcal{U}$	'until' operator
$\boxplus$	'always in the future' operator
$\diamondsuit$	'once in the future' operator

Table 2.3: LTL past temporal operators

Symbol	Name
$\bullet$	'previous' operator
$\mathcal{S}$	'since' operator
$\boxminus$	'always in the past' operator
$\diamondminus$	'once in the past' operator

In short, $\bigcirc\phi$ and $\bullet\phi$ express ϕ holds at the next, respectively the previous, point in time. $\phi\mathcal{U}\psi$ states ϕ holds until ψ holds, i.e. it stops holding as soon as ϕ appears, likewise for $\phi\mathcal{S}\psi$. $\boxplus\phi$ and $\diamondplus\phi$ express that ϕ holds at some point, respectively at all points in time, in the future. $\diamondminus$ and $\boxminus$ are their mirror operators towards the past. Alternatively to $\diamondplus, \boxminus, \boxplus, \diamondminus$, where 'past' and 'future' are relative to a single time point, some formalisms employ an absolute version, $\Box$ and $\Diamond$, that hold over the entire timeline, rather than merely the past and/or future, i.e $\Diamond\phi$ means that ϕ is true at some point.

Some information holds true over a specific interval over the time domain T. To capture this, Metric Temporal Logic (MTL) [92] extends LTL by annotating its temporal operators with explicit time bounds. In LTL, operators such as "eventually" ($\boxplus, \diamondplus$), "always" ($\diamondplus, \boxplus$), $\mathcal{S}$ and $\mathcal{U}$ express properties relative to an unbounded future or past. MTL refines these by introducing intervals $[a, b]$, allowing statements like $\boxplus_{[a,b]}\phi$, stating "ϕ will holds at some point between a and b time units in the future", or $\boxminus_{[a,b]}\psi$, i.e. "ψ held at every time point between a and b time units in the past". Similarly, $\phi\mathcal{U}_{[a,b]}\psi$ requires that ψ becomes true somewhere between a and b time units in the future, while ϕ stays true until ψ holds true.

This addition of metric bounds transforms LTL's qualitative reasoning into quantitative reasoning, making MTL suitable for real-time systems with strict timing constraints. For example, $\boxplus_{[0,10]}(\texttt{temperature} < 80)$ states that a given temperature remains below 80°C for the next 10 seconds, while $\boxplus_{[2,5]}\texttt{alarm}$ expresses an alarm will trigger within 2 to 5 seconds. By enabling precise timing specifications, MTL provides the foundation for monitoring, verification, and scheduling in time-critical applications.

Lastly, several logic-based temporal frameworks consider another temporal operator, i.e. the "holds at" operators $@_t$, which denotes a subsequent expression holds at a given time point t. For example, $@_5$ states that the formula ϕ is true specifically at time $t = 5$, regardless of the current evaluation point. This operator is useful for expressing absolute time references rather than relative ones, enabling comparisons across different timestamps or anchoring conditions to specific moments in time.

This operator makes an appearance in frameworks such as the Logic-based framework for Analytic Reasoning over Streams (LARS) [27], where it allows rules to refer explicitly to time points within a stream, and in STARQL [113], which

supports timestamp-based bindings in Resource Description Framework (RDF) stream queries. Variants of this concept also occur in Metric LARS [77] and other timed extensions of ASP, as well as in event-based languages like ETALIS Language for Events (ELE) [7], where explicit time anchoring is essential for complex event detection. By providing a way to pin formulas to exact time instants, $@_t$ complements interval-based operators like those in MTL, offering finer-grained temporal control in reasoning and querying tasks.

Note that the LTL and MTL operators both employ a notion of 'past' and 'future'. This indicates the notion of a certain 'present' t, meaning the operators are to be taken relative to this present. The intricacies of this relativity depend on the semantics deployed by the formalism.

2.4.3.1 Signal Temporal Logic (STL)

STL is a formalism specifically designed for systems where behaviour is described by continuous signals rather than discrete events. It was introduced by Maler and Nickovic [101] as an extension of Metric Interval Temporal Logic (MITL), combining closed-interval temporal operators with real-valued signal predicates. STL formulas are evaluated over signals $s : \mathcal{R}_{\geq 0} \to \mathcal{R}^n$, mapping time to an n-dimensional vector of real values. Atomic predicates typically express constraints on these signals, such as $s_i(t) > c$, where $s_i(t)$ denotes the value of the i-th component at time t and c is a constant. These predicates allow STL to capture quantitative properties of dynamic systems, such as temperature thresholds or velocity bounds.

STL covers only future temporal operators, including $\mathcal{U}_I$, "always" ($\boxplus_I$), and "eventually" ($\oplus_I$), each parameterized by a time interval I similarly to MTL considered above. Analogously to MTL, this boundedness is crucial for monitoring and verification, as it ensures that satisfaction can be checked over finite traces rather than requiring infinite horizons.

A distinctive feature of STL is its robust semantics, which assigns a real-valued robustness score to each formula instead of a simple Boolean truth value. This score measures how strongly a signal satisfies or violates a specification. For instance, if a predicate requires $s_i(t) > 10$ and the actual value is 15, the robustness is positive and proportional to the margin (here, 5). Conversely, if the value is 8, the robustness is negative, indicating violation. Robust semantics enable quantitative reasoning, sensitivity analysis, and optimisation, which are essential in control and verification tasks.

2.4.4 Datalog

Datalog is a declarative query and rule-based language rooted in FOL, widely used for knowledge representation and reasoning. At its core, Datalog expresses information through *facts* and *rules*. Facts represent atomic statements about the domain, such as `parent(alice,bob)`, while rules define logical implications of the form:

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

Here, the head of the rule (`ancestor(X,Y)`) is inferred whenever the body conditions hold. Variables in rules are universally quantified, and safety conditions require that all variables in the head appear in the body.

Unlike full FOL, Datalog omits function symbols, ensuring that its Herbrand universe is finite and that reasoning remains decidable. This restriction makes Datalog suitable for efficient query evaluation while still supporting recursion, which enables expressing transitive relationships such as reachability or hierarchical dependencies. The semantics of Datalog is based on the notion of *minimal models* and is often computed via a fixpoint procedure: starting from the known facts, rules are applied iteratively until no new facts can be derived.

Datalog serves as the foundation for many reasoning systems and SR formalisms, where its rule-based structure is extended with temporal operators or windowing constructs. Its declarative nature allows users to specify *what* should hold rather than *how* to compute it, making it a natural choice for expressing complex queries and constraints in dynamic environments. For a comprehensive introduction to Datalog and its theoretical underpinnings, we refer the reader to Abiteboul, Hull, and Vianu [1].

2.4.5 Resource Description Framework (RDF)

RDF is a standard model for representing information on the Web in a structured, machine-readable way. At its core, RDF expresses data as a set of triples, each consisting of a subject, predicate, and object, which together form statements about resources. These resources are identified using Internationalized Resource Identifiers (IRIs), ensuring global uniqueness and interoperability. RDF graphs, which are collections of such triples, provide a flexible and extensible way to describe relationships between entities. This graph-based representation enables integration of heterogeneous data sources and supports reasoning over linked data. The formal concepts underlying RDF, including IRIs, literals, blank nodes, and the notion of an RDF graph, are defined in the W3C RDF 1.2 Concepts and Abstract Syntax specification [134].

Beyond its abstract syntax, RDF is grounded in a formal semantics that defines the meaning of RDF graphs and ensures consistency across implementations. RDF semantics introduces mappings, referred to as *interpretations*, from IRIs and literals to elements in a domain This formal foundation guarantees that RDF-based systems can reason correctly and predictably, supporting tasks such as ontology alignment and data integration. The details of these semantic conditions and entailment rules are provided in the W3C RDF 1.2 Semantics [135].

2.4.6 Querying

Querying is a fundamental operation in reasoning systems, as it allows users to retrieve relevant information from a knowledge base or a data stream. In the context of SR, queries are typically evaluated over dynamic, time-sensitive data rather than static datasets, which introduces additional complexity. Before discussing specific stream querying formalisms, we outline the basic concepts underlying query languages. A query can be understood as a declarative specification of the information a user wishes to obtain. Rather than describing how to compute the result procedurally, a query expresses the desired pattern or condition that the data must satisfy. For example, in a relational setting, a query might ask for all tuples where a certain attribute equals a given value. In logic-based settings, queries often take the form of a formula or rule whose satisfaction determines whether a candidate answer is valid. Unlike rule-based or logic-based reasoning formalisms, query languages are primarily concerned with retrieving existing information that satisfies a specification, rather than deriving new entailed facts through materialisation or logical inference.

Queries typically involve *variables*, which act as placeholders for unknown values. When a query is evaluated, these variables are assigned concrete values drawn from the underlying domain, a process known as variable binding. For instance, in a query `?x hasTemperature ?t`, the variables `?x` and `?t` will be bound to specific entities and temperature values that satisfy the stated condition. The set of all such bindings that make the query true constitutes the *answer set* of the query. In stream querying, bindings are not computed over a static dataset but over a windowed view of the stream, as introduced in Section 2.4.1. This means that the query engine continuously evaluates the query against the most recent subset of the stream, producing updated bindings as new data arrives and old data expires. This incremental evaluation is essential for real-time applications, where answers must reflect the current state of the stream rather than its historical entirety. Query languages differ in their expressivity and underlying data model. For example, SPARQL [136] operates over RDF graphs and supports graph pattern matching, while Datalog-based languages use rule-based syntax to express recursive queries. In case of the latter, a query can be expressed in the form of rule `Head :- Body`, where where the head represents the query goal and the body specifies conditions that must hold for the goal to be satisfied. For example,

```
HighTemp(x) :- Sensor(x), Temperature(x,t), t > 30.
```

retrieves all sensors `x` reporting a temperature above 30°C.

Stream-oriented extensions such as C-SPARQL [21] and CQELS [115] augment these languages with temporal constructs, enabling queries like "retrieve all sensors reporting a temperature above 30°C in the last 10 seconds". These extensions often introduce operators for specifying windows, ordering, and temporal joins, which are crucial for reasoning about evolving data. Finally, the semantics of query evaluation in a streaming context can vary. Some formalisms adopt snapshot semantics, where

the query is evaluated over the current window as if it were a static dataset. Others employ continuous semantics, where the query is treated as a standing request that produces new answers whenever the underlying stream changes. Understanding these distinctions is key to comparing stream querying formalisms, as they impact both performance and correctness guarantees.

2.5 Related Surveys

Surveying the landscape of SR has been the focus of several prior efforts, each emphasizing different aspects of the field. In this section, we briefly review existing surveys and related work, and position our contribution within this context.

The cooperative work "Grounding Stream Reasoning research" [33] summarizes the field of SR, more specifically the results and discussion of the yearly Streaming Reasoning workshops since 2015. It presents a breakdown of the SR field into four areas of research: (i) Stream Processing, (ii) Streaming Linked Data (SLD), (iii) deductive Stream Reasoning, and (iv) inductive Stream Reasoning. For each of these, Bonte et al. [33] provides a high-level overview of the field, but does not provide a detailed study of specific existing frameworks and their properties.

Several recent surveys do provide a more in-depth analysis of the current state-of-the-art in some subfields of SR. For example, Bonte et al. [34] surveys systems enabling RDF Stream Processing (RSP). It thereby focuses most strongly on the querying aspects of the field. They categorize the current RDF Stream Processing (RSP) systems based on their technical properties, e.g system architecture, operational semantics and accepted data formats. In this work, the field of SR is positioned as an extension of the field of RSP, by supplementing RSP systems with advanced reasoning capabilities. However, this extension is not discussed in depth, nor are the available formalisms surveyed. The earlier work of Bonte, Ongenae, and Tommasini [31] likewise covers aspects of RSP, more precisely the key ontologies used in RSP applications. They provide an overview of the modelling and knowledge representation properties of the considered ontologies, but similar to Bonte et al. [34], they do not cover the aptitude and ability for advanced reasoning of the considered systems. In 2019, Isah et al. [83] composed a comparison on existing open-source and commercial stream processing systems. Central in this comparison is the architectural composition and data flow models. From a practical perspective, this work arguments the inherent complexity of streaming data and the need for advanced reasoning capabilities, but does not focus on the underpinning formalisms or their properties.

Isah et al. [83], Bonte, Ongenae, and Tommasini [31] and Bonte et al. [34] each focus on different aspects of the SR field, but mainly fit within areas (i) and (ii) outlined by Bonte et al. [33]. As a result, the current literature on areas (iii) and (iv) remains dispersed across the field's growing range of academic works. With this work, we aim to provide a reference work for the research area of deductive SR. We consider

the coverage of inductive SR to be beyond the scope of this work as despite their linguistic similarity, deductive and inductive SR employ vastly different techniques and tackle inherently different problems.

Lastly, several earlier works have contributed to a proper outline and definition of the SR field, and thereby to the scope of this survey. Both Margara et al. [106] and Dell'Aglio et al. [66] have identified requirements of SR systems. In particular, Dell'Aglio et al. [66] points out the need for more expressive frameworks, in order to better encode user needs. They stressed the emerging interest for ASP temporal logics for streaming data, as well as several other promising avenues such as Temporal Action Logic (TAL) and Event Calculus. As will be covered in Sections 2.7 and 2.8, these avenues were successfully explored and led to promising new SR formalisms such as LARS, Event Calculus for Run-Time reasoning (RTEC), DatalogMTL and others.

While Dell'Aglio et al. [66] focus on system-level requirements such as scalability, timeliness, heterogeneity, and robustness, our survey evaluates SR formalisms along theoretical and usage dimensions. Their requirements define what SR systems must achieve operationally, whereas our dimensions study the semantic foundations, expressivity, complexity, and adoption of the underlying formalisms in a more formal manner. Together, these perspectives are complementary: system requirements ensure practical feasibility, while formalism dimensions guarantee the conceptual rigor needed to support those capabilities.

2.6 Dimensions of a SR formalism

Within this work, we focus on the formalisms that provide syntax and semantics for streaming data. In order to compare multiple different SR formalisms in a consistent manner, we outline 10 dimensions upon which to evaluate the formalisms: 7 dimensions focused on the theoretical properties and 3 dimensions capturing a formalism's usage and adoption.

These 7 theoretical dimensions were derived by analyzing recurring design choices and challenges across the surveyed literature. Specifically, they originate from common aspects that repeatedly appear in SR formalism proposals: (i) the underlying mathematical foundation, (ii) representation of streams and time, (iii) expressivity for static relations, (iv) expressivity for temporal relations, (v) handling of uncertainty and incomplete data, (vi) the intended reasoning objective (querying vs. reasoning), and (vii) computational complexity and decidability. Each of these aspects was identified during the preliminary review of key papers and refined to ensure coverage of both conceptual and practical concerns.

Related surveys such as Bonte et al. [34], Margara et al. [106] and Dell'Aglio et al. [66] touch upon several of these aspects, such as temporal operators [34, 66, 106], mathematical foundation [34], stream representation [34], and handling of uncertainty [34, 66, 106], but given their different scope, their treatment is less formal and

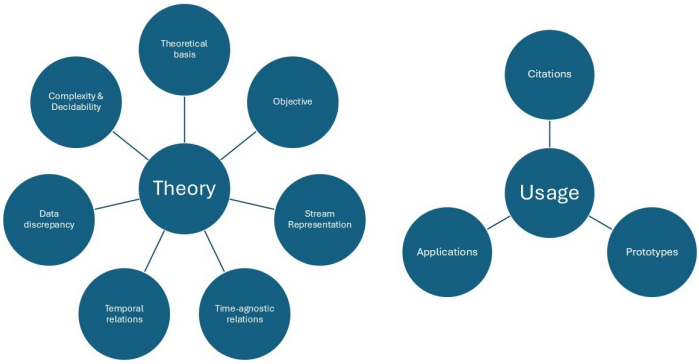


Figure 2.3: Overview of evaluation dimensions

granular; these works emphasize system architectures and application-level considerations rather than the theoretical underpinnings of SR formalisms (see Section 2.5). By contrast, our dimensions aim to provide a rigorous framework for analyzing the conceptual properties of SR formalisms alongside their practical adoption. This distinction motivates the depth and structure of our evaluation approach.

To ensure transparency and traceability, each dimension is explicitly linked to the comparative tables in Section 2.7 and 2.8. For example, Table 2.4 summarizes T1 (Theoretical Basis) and T3 (Static Expressivity), Table 2.6 addresses T4 (Temporal Expressivity), Table 2.8 covers T2 (Stream Representation and Windowing), and Table 2.10 captures T5 (Uncertainty Handling). Dimensions T6 (Objective) and T7 (Complexity) are discussed in the narrative of Section 2.8, while the usage dimensions U1-U3 are reflected in Table 2.14. These cross-references allow readers to easily navigate from the conceptual definition of a dimension to its empirical evaluation across formalisms. Together, these dimensions serve as different axes for our analysis of the various formalisms. They allow us to:

- Position newly developed formalisms relative to the state of the art.
- Identify the most suitable formalism for a given application.
- Compare multiple formalisms for specific use cases.
- Expose gaps in current SR research and opportunities for extension.

These objectives directly address the research questions introduced in Section 2.3.1. In particular, the theoretical dimensions in Section 2.7 operationalise **Q1** by revealing common constructions and expressions across formalisms, and they address **Q2** by providing sensible criteria for evaluating a formalism's suitability for SR. Furthermore, by applying these criteria to each formalism and comparing them across categories, the theoretical dimensions also contribute to answering **Q3**. The

usage dimensions of Section 2.8 complement this by answering **Q4** and **Q5**, focusing on adoption in real-life applications and identifying directions for future research.

2.6.1 Structure of the Discussion

To structure our discussion of SR formalisms according to these dimensions, we adopt a two-tiered approach that distinguishes theoretical aspects from practical considerations. First, in Section 2.7, we address the seven theoretical dimensions, which capture the conceptual foundations and formal properties of each formalism. Next, in Section 2.8, we turn to the three usage dimensions, which reflect adoption and applicability in real-world contexts. This separation provides a clear distinction between what a formalism can express and how it is used in practice.

Within both sections, formalisms are grouped into five categories based on observed characteristics in the surveyed literature: (a) DL-based formalisms, (b) Temporal Logic-based formalisms, (c) SR query languages, (d) ASP-based formalisms, and (e) Miscellaneous formalisms. These categories emerged from the systematic protocol described in Section 2.3.2 and allow for a structured comparison across families of approaches rather than isolated formalisms.

In Section 2.7, we discuss each formalism in detail within its category, evaluating it against the seven theoretical dimensions T1–T7. This enables readers to consult the characteristics of a specific formalism while preserving comparability across related approaches. In Section 2.8, we summarize how the formalisms in each category score on the three usage dimensions U1–U3, highlighting patterns of adoption and maturity. Complementing the narrative, comparative tables provide a concise overview of theory dimensions (Tables 2.4–2.10) and usage dimensions (Table 2.14), enabling two complementary modes of analysis: direct comparison across formalisms and in-depth examination of individual characteristics. This structure ensures transparency, facilitates navigation, and supports both high-level synthesis and detailed evaluation.

2.6.2 Theoretical Dimensions

The theoretical dimensions capture the conceptual and formal properties that define the essence of a SR formalism. They allow us to assess the underlying logic or mathematical foundation, the way streams and time are represented, and the expressivity offered for both static and temporal relations. In addition, these dimensions consider how a formalism addresses practical challenges such as data discrepancies, its intended reasoning objectives, and its computational characteristics in terms of complexity and decidability. Together, these aspects provide a structured lens for comparing formalisms beyond surface syntax, focusing on their ability to model dynamic environments rigorously and efficiently.

T1. Theoretical basis

A formalism can be rooted in different mathematical theories. Some formalisms are rooted in mathematical logic, whereas others might stem from set theory, ASP, algebraic structures or other. A SR formalism can be developed as an extension of an existing formalism for static data. Alternatively, a SR formalism can be developed 'from scratch', i.e. no existing language or formalism for static data is taken as a basis, but rather an envisioned system or real-life application is modelled directly.

T2. Time and Stream representation

As introduced in Section 2.4.1, there are multiple suitable ways of modelling the concept of a data stream. Understanding these differences is crucial because the chosen representation directly influences the semantics of reasoning, the complexity of algorithms, and the applicability of a formalism to real-world scenarios. For example, whether time is modelled explicitly or implicitly determines how temporal operators behave, while the choice of stream structure (set, sequence, or function) affects how incremental updates and windowing are implemented. These design choices ultimately impact expressivity, performance, and interoperability between systems. Within this dimension, we identify the structure of a data stream within the considered formalism. To describe said structure, we consider the following elements:

- The choice of mathematical construction, i.e. set, collection, function or other, as introduced in Section 2.4.
- How is time represented, e.g. explicitly via time points or implicitly by (chronological) ordering of the elements?
- What are the atoms α that populate the stream?
- If applicable, over what time domains does the formalism consider the streams? What restrictions are imposed upon an eligible time domain?
- Whether or not the stream representation allows for (theoretically) infinite data streams or in case of finite streams, what determines their (maximal) length?
- Whether windowing is used and which operators are employed to define these windows, i.e. tuple-based, time-based, or other as introduced in Section 2.4.1?

Analysing these elements provides insight into the ability of a formalism to represent dynamic data and reason over it effectively. It also reveals potential limitations, such as constraints on time granularity or lack of support for infinite streams, which can be decisive for applications requiring real-time responsiveness or interoperability across heterogeneous systems.

T3. Expressivity of static relations

The ability of a formalism to express relationships between atomic facts is a fundamental determinant of its reasoning power. Once the atoms $\alpha \in \Omega$ are identified, it becomes necessary to examine how these atoms can be combined into more complex statements. Because syntactic conventions differ substantially across formalisms, a purely syntactic comparison offers limited insight. Instead, we adopt a semantic perspective and evaluate whether a formalism supports three core logical constructs: conjunction (AND), disjunction (OR), and negation (NOT).

The choice of these three operators is motivated by their status as the minimal basis of classical propositional logic. Together, they constitute the foundational operations required for compositional reasoning. Conjunction enables the representation of simultaneous conditions, disjunction captures alternative possibilities, and negation allows for the explicit modelling of absence or contradiction. These operators are not only intuitive from a human reasoning perspective but also sufficient to derive other common constructs such as implication and equivalence. Consequently, their presence or absence provides a meaningful baseline for assessing the expressive adequacy of a formalism.

- **AND:** the concurrence of two atoms, i.e. two 'facts' that **hold true at the same time**. Examples include properties that hold simultaneously or measurements taken parallel to each other. This is represented by $\wedge$ and $\cap$ in FOL and set theory respectively.

- **OR:** the discretion of two atoms, i.e. the discretion is true if and only if **at least one of the atoms is true**, commonly known as (inclusive) `OR`. Within FOL, this is displayed via disjunction $\vee$. Set theory employs the union $\cup$ of two sets.

- **NOT:** the contrary or **opposite of an atom** α. Knowledge that a fact or assumption is not true can be equally valuable as the knowledge it is true. As such, being able to express the contrary can be an appreciable asset of a formalism's expressivity. Negation $\neg$ and complement $\overline{C}$ are known realisations within FOL and set theory.

Evaluating these operators allows us to determine whether a formalism supports the essential mechanisms for combining facts, which is indispensable for any non-trivial reasoning task.

T4. Temporal expressivity

A broader interpretation of "reasoning over streams of data" is that reasoning may extend beyond isolated facts to include temporal dependencies between them. This distinction is captured by two complementary notions:

Reasoning over time refers to applying static reasoning techniques to data that arrives incrementally, without considering temporal relationships between facts. Reasoning about time goes further by explicitly modelling and exploiting temporal relations, enabling the detection of patterns such as persistence, succession, and causality.

The dimension of temporal expressivity evaluates whether a formalism supports reasoning about time and, if so, to what extent. To this end, we focus on a set of temporal operators that have become canonical in temporal logics such as LTL and MTL. These operators form a principled baseline for expressing temporal properties:

- A fact holds over a prolonged period of time, denoted by box-shaped operators $\boxplus$, $\boxminus$.
- A fact is known to hold at some point, but the exact point in time is not known, or only approximately), i.e. diamond-shaped operators $\diamondplus$, $\diamondminus$.
- A fact is known to hold continuously (at least) since or until a specified other fact is known to be true, operators $\mathcal{S}$ and $\mathcal{U}$.
- A fact will be true at the next point in time ($\bigcirc$), or was true at the previous point in time ($\bullet$).

This choice reflects a balance between expressivity and analysability: these operators are widely adopted in formal verification and SR, and they provide sufficient coverage for most practical scenarios without introducing excessive complexity.

Allen's Interval Algebra offers an alternative, interval-centric perspective by defining 13 qualitative relations between time intervals (e.g., before, meets, overlaps, during). While our selected operators are point-based and metric in nature, Allen's relations can often be encoded using combinations of these operators when intervals are represented explicitly. For example, A before B can be expressed by requiring that the end of A precedes the start of B, which corresponds to constraints formulated with $\mathcal{U}$. Conversely, formalisms that natively support Allen's relations (e.g., ELE) exhibit high temporal expressivity because they allow reasoning about interval relationships without reducing them to point-based semantics. In our evaluation, such formalisms are considered to support the full spectrum of temporal relations defined in this dimension.

Assessing these operators provides a principled baseline for comparing formalisms on their ability to reason about temporal patterns. The comparative results for this dimension are summarized in Table 2.6. In particular, dimensions T4 and T6 operationalise the distinction between reasoning over time and reasoning about time, which explicitly exploits temporal relations between facts. This distinction underpins our evaluation of temporal expressivity and formalism objectives, as it reflects two different interpretations of SR.

T5. Uncertainty in the data and in the stream

Real-world streaming environments rarely provide complete or perfectly ordered data; instead, they exhibit uncertainty, vagueness, and temporal irregularities that challenge traditional SR approaches. Numerous works have identified uncertainty management as a key requirement for practical applications, as data can be imprecise, missing, contradictory, noisy, or unstructured [106]. For example, sensor networks may suffer from data loss or imprecise measurements, while social media streams often contain incomplete conversations or ironic statements that complicate sentiment analysis [66]. These issues demand reasoning models that explicitly account for different sources of uncertainty and incorporate them during computation. Beyond uncertainty, advanced capabilities such as fuzzy semantics and multi-valued approaches enable reasoning over vague concepts and conflicting information, moving beyond strict true/false dichotomies. Handling out-of-order or delayed events is equally essential in distributed settings to maintain temporal consistency. Several modelling paradigms have been proposed to address these challenges, including probabilistic frameworks for uncertainty, fuzzy set theory [144] for graded truth values, and logics such as Gödel or Łukasiewicz for multi-valued reasoning [80]. Collectively, these capabilities and their underlying formalisms significantly enhance the robustness and applicability of SR in complex, uncertain environments, forming the conceptual foundation for the implementation and evaluation strategies discussed in subsequent sections.

T6. Formalism objective

In designing SR formalisms, we distinguish three fundamental objectives: stream querying, reasoning over time, and reasoning about time. Throughout this survey, these objectives are identified based on the tasks that the original authors explicitly define, formalise, and analyse within the formalism. Consequently, the categories are defined according to the primary design objective of the formalism as stated in the original literature, rather than all tasks that may be expressible within the formalism, capabilities introduced by later implementations, or application-specific extensions.

Stream querying focuses on extracting existing information from continuously evolving data. A query specifies a pattern that data must satisfy, and the result consists of bindings or records matching that pattern. Although sophisticated temporal patterns may appear in queries, the primary outcome remains data retrieval rather than the derivation of new facts.

By contrast, reasoning, both over time and about time, aims at deriving new knowledge from existing information through logical entailment. The result of reasoning is therefore not merely the selection of existing data, but the production of conclusions that were not explicitly present in the input. With *reasoning over time*, we refer to the practice of performing traditional (non-temporal) inference on data, which

just so happens to arrive sequentially. A typical example includes incremental updating of a knowledge base. For example, a reasoning engine may infer `sensor1 is operational` based on sequential temperature measurements `temperature(sensor1, 22°C)`, `temperature(sensor1, 23°C)`, ... and update the knowledge base accordingly. The inference itself is static, i.e. it does not depend on temporal patterns or relations between the observations, but it occurs incrementally as new data arrives. New information from data that arrives incrementally, while the inference rules themselves remain non-temporal. Time determines when information becomes available, but not why a conclusion follows. In *reasoning about time*, temporal relationships themselves participate in the entailment process. Conclusions depend on temporal constructs such as duration, succession, persistence, windows, intervals, or temporal operators. For instance, identifying that a traffic jam occurred after a series of congestion alerts and a road closure. CEP, aimed at recognizing higher-level event patterns over time, falls under the umbrella of reasoning about time, as well as other frameworks which consider temporal relations in their inference process, for example via LTL operators $\boxplus$, $\diamondsuit$ etc. The formalism derives new knowledge based on both the content of observations and their temporal relationships.

Distinguishing these objectives is crucial in a SR context: while querying provides the foundation for accessing dynamic data, reasoning over time ensures consistency and incremental enrichment of knowledge, and reasoning about time enables richer interpretations of temporal structure. Although these categories are conceptually useful, they are not strictly disjoint. For instance, one might carefully craft queries in a stream querying framework such as C-SPARQL to detect a specific temporal pattern, which effectively constitutes a form of reasoning about time. Nonetheless, specifying the primary objective of a formalism helps clarify its strengths, weaknesses, and underlying design philosophy, and provides an additional perspective when comparing approaches with different goals. Each of these objectives has its proper place in the SR research landscape, ensuring SR systems can support both low-level data access and high-level temporal inference.

T7. Complexity and Decidability

Complexity and decidability analyses play a crucial role in understanding the theoretical properties of SR formalisms. They provide insight into the trade-off between expressiveness and computational feasibility, guiding both the design of new approaches and the selection of suitable formalisms for practical applications. For instance, knowing whether query evaluation is tractable or whether reasoning remains decidable under certain restrictions helps anticipate scalability and implementation challenges. However, such results are still relatively scarce in SR literature, and when available, they often apply to specific fragments or under bounding criteria such as fixed window sizes or restricted recursion. Given these theoretical results are nonetheless an important characteristic of a SR formalism, we consider for each formalism whether reasoning

tasks are decidable or undecidable, and we indicate known complexity bounds using standard classes such as P, NP, or EXP, when available in the considered literature.

2.6.3 Usage Dimensions

The usage dimensions capture the practical footprint of a SR formalism within the research community and in real-world applications. While theoretical properties determine what a formalism can express and how it behaves, usage dimensions reveal whether these capabilities translate into adoption and impact. They measure academic visibility through citation counts, the availability of software prototypes or implementations that demonstrate feasibility, and evidence of deployment in concrete application domains. Together, these indicators provide insight into the maturity and relevance of a formalism beyond its conceptual design, highlighting which approaches have gained traction and which remain largely theoretical.

U1. Citations

As each of the formalisms considered in our survey stems from a research paper, the amount of citations of the paper provides a direct and tangible metric for the degree in which the formalism is valued by the research community. The primary indicator in this regard is the amount of citations of the reference paper for the formalism. In case no single source paper can be identified, e.g. due to the formalism being introduced (by the same authors) in multiple venues and/or journals, we take each of these papers into consideration and sum their citation counts. The results are summarized in Table 2.12.

U2. Software prototypes

To assess the availability of prototypes for a given SR formalism, we define a prototype as a piece of generic software that can be used across different applications to perform SR. In our analysis, we consider all prototypes referenced in the papers included in the secondary study (see Subsection 2.3.2). We restrict this dimension to high-level implementations and exclude solutions developed exclusively for one specific application, as those are addressed in the next dimension. In addition to dimension U1, Table 2.12 holds the prototypes of the considered formalisms.

U3. Applications

To gauge a formalism's contribution toward tangible applications and products, we examine the real-life systems and solutions built upon the considered SR formalisms. In particular, we investigate the domains in which these formalisms have been applied, as summarized in Table 2.14, since application areas provide insight into their practical relevance and versatility. For this purpose, we rely on the material collected in the

final step of our protocol, the secondary study, which reports implementations and deployments referenced in the literature.

2.7 Survey Results: Theory Dimension

This section examines the seven theoretical dimensions introduced in Section 2.6, which capture the conceptual foundations and formal properties of SR formalisms. By analyzing each formalism against these dimensions, we address **RQ1**, **RQ2**, and **RQ3**: identifying common constructions and expressions, establishing criteria for evaluating suitability for SR, and comparing recently developed formalisms with respect to these criteria.

We begin with an overview of the surveyed formalisms grouped by dimension and category, supported by comparative tables for quick reference. Specifically, dimension T1 (Theoretical Basis) and T3 (Static Expressivity) are covered in Table 2.4, dimension T2 (Stream Representation and Windowing) is covered in Table 2.8, dimension T4 (Temporal Expressivity) is covered in Table 2.6, and dimension T5 (Uncertainty Handling) is covered in Table 2.10. Dimensions T6 and T7 are discussed in detail in the narrative, as their interpretation requires more context than tabular representation.

Following this overview, the subsequent subsections provide a detailed discussion of each category of formalisms: (a) DL-based, (b) Temporal Logic-based, (c) SR query languages, (d) ASP-based, and (e) Miscellaneous approaches. For each category, we describe how individual formalisms address the seven theoretical dimensions, highlighting design choices, strengths, and limitations. This structure enables readers to either consult the tables for direct comparison or explore the narrative for an in-depth understanding of specific formalisms.

2.7.1 DL-based Formalisms

Description Logics (DL) have long been established as a cornerstone of knowledge representation and underpin many SW technologies, most notably Web Ontology Language (OWL) [133]. Their distinction between terminological knowledge (TBox) and assertional knowledge (ABox) makes them well-suited for scenarios where static domain knowledge must be combined with dynamic data streams. This separation allows DL-based approaches to incorporate evolving observations while preserving a structured conceptual model. Consequently, DL-based formalisms have been explored for SR both through direct application [45] and through extensions that introduce temporal or predictive capabilities [89].

2.7.1.1 DL-Lite TQL and its fragments

Klarman and Meyer [89] combine the DL-Lite family of description logics with a set of temporal rules, taken from the Temporal Query Language (TQL) [79]. TQL is

Table 2.4: Summary of the T1 (theoretical basis) and T3 (expressivity of static relations) dimensions for SR formalisms

Formalism	**Basis**	**Logical relations**				
		$\neg$	$\wedge$	$\vee$	$\rightarrow$	other operators
DL-Lite TQL	DL	✓	✓	R	R	-
DL-Lite TQL^{+}	DL	-	✓	-	-	-
DL-Lite TQL$^{\exists}$	DL	✓	✓	R	R	-
DL-Lite TQL$^{\exists,+}$	DL	-	✓	-	-	-
LTL$^{\mathbf{X},\mathbf{G}}_{\mathcal{EL}}$	DL ($\mathcal{EL}$)	-	✓	-	-	$\exists$
$\mathcal{EL}^{++}$ ontology streams	DL $\mathcal{EL}^{++}$	-	✓	-	-	$\exists$
DatalogM(I)TL	Datalog	-	✓	-	✓	-
DatalogM(I)TL^{FP}	Datalog	-	✓	-	✓	-
DatalogM(I)TL$^{\neg}$	Datalog	✓	✓	-	✓	-
DatalogM(I)TL$^{\exists}$	Datalog	-	✓	-	✓	$\exists$
MSTL	RCC-8	✓	✓	✓	✓	$\exists, \forall$
P-MTL	MTL	✓	✓	✓	✓	$\exists, \forall$
PMLTL	MLTL	✓	✓	✓	R	-
TEL	EL	✓	✓	✓	✓	$\exists, \forall$
TEL + past operators	EL	✓	✓	✓	✓	$\exists, \forall$
MEL	EL	✓	✓	✓	✓	-
LTPAL	PAL	✓	✓	R	R	-
ASTL	STL	✓	✓	✓	-	-
ProbSTL	STL	✓	R	✓	R	-
EvTL	STL	✓	R	✓	R	-
RobTL	STL	✓	✓	R	R	-
SDRL	-	✓	✓	R	R	-
TESLA	TRIO	✓	✓	R	R	$\forall$
TAL	PMON	✓	✓	✓	✓	$\forall, \exists$
C-SPARQL	SPARQL	✓	✓	✓	R	$\exists, \forall$
STARQL	SPARQL	✓	✓	✓	R	$\exists, \forall$
SPARQL_stream	SPARQL	✓	✓	✓	R	$\exists, \forall$
CT-SPARQL	SPARQL	✓	✓	✓	R	$\exists, \forall$
ELE	-	✓	✓	✓	R	-
EP-SPARQL	SPARQL	-	✓	✓	-	$\exists, \forall$
CQELS	SPARQL	✓	✓	✓	R	$\exists, \forall$
RSP-QL	SPARQL	✓	✓	✓	R	$\exists, \forall$
RSP-QL*	SPARQL	✓	✓	✓	R	$\exists, \forall$
C-ASP Language	ASP	✓	✓	-	✓	-
LARS	ASP	✓	✓	✓	✓	-
Metric LARS	ASP	✓	✓	✓	✓	-
wLARS	ASP	✓	✓	✓	✓	-
PrASP	ASP	✓	✓	✓	✓	-
Temporal Datalog	Datalog	-	✓	-	✓	-
RTEC	Event Calculus	-	✓	-	-	-

Formalism	Temporal Relations										
	○	●	$\mathcal{S}$	$\mathcal{U}$	□	⊞	⊟	◇	⟐	⟑	Interval
DL-Lite TQL	-	-	✓	✓	-	R	R	-	R	R	-
DL-Lite TQL^{+}	-	-	✓	✓	-	-	-	-	R	R	-
DL-Lite TQL$^{\exists}$	-	-	-	-	-	R	R	-	✓	✓	-
DL-Lite TQL$^{\exists,+}$	-	-	-	-	-	-	-	-	✓	✓	-
LTL$_{\mathcal{EL}}^{\mathbf{X},\mathbf{G}}$	✓	-	-	-	-	✓	-	-	-	-	-
DL $\mathcal{EL}^{++}$ + ontology stream	-	-	-	-	-	-	-	-	-	-	-
DatalogM(I)TL	-	-	✓	✓	-	✓	✓	-	✓	✓	✓
DatalogM(I)TL^{FP}	-	-	-	-	-	✓	✓	-	-	✓	✓
DatalogM(I)TL$^{\neg}$	-	-	✓	✓	-	✓	✓	-	✓	✓	✓
DatalogM(I)TL$^{\exists}$	-	-	✓	✓	-	✓	✓	-	✓	✓	✓
MSTL	✓	-	-	-	✓	-	-	✓	-	-	✓
P-MTL	✓	-	-	-	✓	-	-	✓	-	-	✓
PMLTL	-	-	-	✓	-	✓	-	-	✓	-	✓
TEL	✓	-	-	✓	-	✓	-	-	✓	-	-
TEL + past operators	✓	✓	✓	✓	-	✓	✓	-	✓	✓	-
MEL	✓	✓	✓	✓	-	R	R	-	R	R	R
LTPAL	✓	-	-	✓	-	R	-	-	R	-	-
ASTL	-	-	-	✓	-	✓	-	-	✓	-	✓
ProbSTL	-	-	-	✓	-	R	-	-	R	-	✓
EvTL	-	-	-	✓	-	R	-	-	R	-	✓
RobTL	-	-	-	✓	-	R	-	-	R	-	✓
SDRL	✓	✓	R	R	-	R	R	-	R	R	✓
TESLA	R	R	R	R	R	R	R	R	R	R	✓
TAL	-	-	-	✓	-	✓	-	-	✓	-	✓

Table 2.5: Summary of the T4 dimension (temporal expressivity) for the different surveyed formalisms. — Part A

Formalism	Temporal Relations										
	○	●	$\mathcal{S}$	$\mathcal{U}$	□	⊞	⊟	◇	⟐	⟑	Interval
C-SPARQL	-	-	-	-	-	-	-	-	-	-	-
STARQL	-	-	-	-	-	✓	✓	-	✓	✓	-
SPARQL_stream	-	-	-	-	-	-	-	-	-	-	-
CT-SPARQL	-	-	-	-	-	-	-	-	-	-	-
ELE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP-SPARQL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
CQELS	-	-	-	-	-	-	-	-	-	-	-
RSP-QL	-	-	-	-	-	-	-	-	-	-	-
RSP-QL*	-	-	-	-	-	-	-	-	-	-	-
Continious ASP (C-ASP) Language	-	-	-	-	-	-	-	-	-	-	-
LARS	-	-	-	-	✓	-	-	✓	-	-	-
wLARS	-	-	-	-	✓	-	-	✓	-	-	-
Metric LARS	-	-	-	-	✓	✓	✓	✓	✓	✓	✓
PrASP	-	-	-	-	-	-	-	-	-	-	-
Temporal Datalog	-	-	-	-	-	-	-	-	-	-	-
RTEC	✓	✓	✓	✓	-	-	-	-	-	-	-

Table 2.6: Summary of the T4 dimension (temporal expressivity) for the different surveyed formalisms. — Part B

Formalism	Representation		Windowing		
	Time	Stream	Tuple	Time	Other
DL-Lite TQL	$\mathbb{Z}$	sequence	-	-	-
DL-Lite TQL^{+}	$\mathbb{Z}$	sequence	-	-	-
DL-Lite TQL$^{\exists}$	$\mathbb{Z}$	sequence	-	-	-
DL-Lite TQL$^{\exists,+}$	$\mathbb{Z}$	sequence	-	-	-
LTL$_{\mathcal{EL}}^{\mathbf{X},\mathbf{G}}$	$\mathbb{N}$	sequence	-	-	-
DL $\mathcal{EL}^{++}$ + ontology stream	$\mathbb{N}$	sequence	-	-	-
DatalogM(I)TL	$\mathbb{Z}$ or $\mathbb{Q}$	mapping function	-	-	-
DatalogM(I)TL^{FP}	$\mathbb{Z}$ or $\mathbb{Q}$	mapping function	-	-	-
DatalogM(I)TL$^{\neg}$	$\mathbb{Z}$ or $\mathbb{Q}$	mapping function	-	-	-
DatalogM(I)TL$^{\exists}$	$\mathbb{Z}$ or $\mathbb{Q}$	mapping function	-	-	-
MSTL	ordered set $(T, <)$	mapping function	-	-	-
P-MTL	$\mathbb{N}$	mapping function	-	-	-
PMLTL	$\mathbb{N}$	sequence	-	-	-
TEL	Unspecified	sequence	-	-	-
TEL + past operators	Unspecified	sequence	-	-	-
MEL	Unspecified	sequence	-	-	-
LTPAL	$\mathbb{N}_{<n}$	sequence	-	-	-
ASTL	$\mathbb{N}$	mapping function	-	-	-
ProbSTL	$\mathbb{N}$	mapping function	-	-	-
EvTL	$\mathbb{N}$	sequence	-	-	-
RobTL	$\mathbb{N}$	sequence	-	-	-
SDRL	Unspecified	mapping function	-	-	-
TESLA	Unspecified	mapping function	-	-	-
TAL	linear $(T, <)$	set	-	-	-

Table 2.7: Summary of the T2 dimension (time and stream representation) for SR formalisms. — Part A

Formalism	Representation		Windowing		
	Time	Stream	Tuple	Time	Other
C-SPARQL	Unspecified	sequence	✓	✓	-
STARQL	linear $(T, <)$	sequence	-	✓	-
SPARQL_stream	Unspecified	sequence	-	✓	-
CT-SPARQL	Unspecified	sequence	-	-	-
ELE	$\mathbb{Q}_{\geq 0}$	mapping function	-	-	-
EP-SPARQL	Unspecified	sequence	✓	✓	-
CQELS	$\mathbb{N}$	sequence	✓	✓	-
RSP-QL	$\mathbb{N}$	sequence	✓	✓	-
RSP-QL*	$\mathbb{N}$	sequence	✓	✓	-
C-ASP Language	Unspecified	sequence	✓	✓	-
LARS	$\mathbb{N}$	mapping function	✓	✓	partition
Metric LARS	$\mathbb{N}$	mapping function	✓	✓	partition
wLARS	$\mathbb{N}$	mapping function	✓	✓	partition
PrASP	$\mathbb{N}$	sequence	-	-	-
Temporal Datalog	$\mathbb{N}$	sequence	-	-	-
RTEC	$\mathbb{Z}$	sequence	-	✓	-

Table 2.8: Summary of the T2 dimension (time and stream representation) for SR formalisms. — Part B

Formalism	Uncertainty			
	Probabilistic	Fuzzy	Multi-valued	Out-of-order data
DL-Lite TQL	-	-	-	-
DL-Lite TQL^{+}	-	-	-	-
DL-Lite TQL$^{\exists}$	-	-	-	-
DL-Lite TQL$^{\exists,+}$	-	-	-	-
LTL$_{\mathcal{EL}}^{\mathbf{X},\mathbf{G}}$	-	-	-	-
DL $\mathcal{EL}^{++}$ + ontology stream	-	-	-	-
DatalogM(I)TL	-	-	-	-
DatalogM(I)TL^{FP}	-	-	-	-
DatalogM(I)TL$^{\neg}$	-	-	-	-
DatalogM(I)TL$^{\exists}$	-	-	-	-
MSTL	-	-	-	-
P-MTL	✓	-	-	-
PMLTL	-	-	-	-
TEL	-	-	-	-
TEL + past operators	-	-	-	-
MEL	-	-	-	-
LTPAL	-	-	-	-
ASTL	-	-	$\mathbb{R}$	-
ProbSTL	✓	-	-	-
EvTL	✓	-	-	-
RobTL	-	-	-	✓
SDRL	-	-	-	-
TESLA	-	-	-	-
TAL	-	-	-	-

Table 2.9: Summary of the T5 dimension (uncertainty in the data and in the stream) for SR formalisms. — Part A

Formalism	Uncertainty			
	Probabilistic	Fuzzy	Multi-valued	Out-of-order data
C-SPARQL	-	-	-	-
STARQL	-	-	-	-
SPARQL_stream	-	-	-	-
CT-SPARQL	-	-	-	-
ELE	-	-	-	-
EP-SPARQL	-	-	-	-
CQELS	-	-	-	-
RSP-QL	-	-	-	-
RSP-QL*	✓	✓	✓	-
C-ASP Language	-	-	-	-
LARS	-	-	-	-
wLARS	✓	✓	✓	-
Metric LARS	-	-	-	-
PrASP	✓	✓	semiring $\mathcal{R}$	-
Temporal Datalog	-	-	-	-
RTEC	-	-	-	-

Table 2.10: Summary of the T5 dimension (uncertainty in the data and in the stream) for SR formalisms. — Part B

composed of formulae that tie together multiple queries using conjunction $\wedge$, negation $\neg$, $\mathcal{S}$ and $\mathcal{U}$, i.e. $\neg q, q_1 \wedge q_2, q_1 \mathcal{S} q_2, q_1 \mathcal{U} q_2$. This results in **DL-Lite TQL**, a SR formalism with DL fragment DL-Lite as a theoretical basis (T1). With temporal rules of the form $\phi \Rightarrow \psi$, with ϕ and ψ formulae as above, it is able to infer new information ψ when ϕ is met. As ϕ and ψ can capture temporal relations, DL-Lite TQL is therefore aimed at reasoning *about* time (T6). It thereby distinguishes two strands of inference, i.e. inference over data streams towards the future ('prediction') and towards the past ('explanation').

The concept of a data stream (T2) is modeled as a sequence of (ABox) data $(\mathcal{A}_i)_{i \in \mathbb{Z}}$ over time domain $\mathbb{Z}$. As such, time is explicitly modeled as a discrete timeline $\mathbb{Z}$. The ABox sequence is tied to a fixed TBox and can theoretically be infinite. Each ABox A_i has to be consistent with the TBox of the stream. Klarman and Meyer [89] make no mention of windowing mechanisms for DL-Lite TQL.

The stream and TBox can be queried at a time $i \in \mathbb{Z}$. TQL takes a set of conjunctive queries, recursively combined with logical operators $\neg, \wedge, \mathcal{U}$ and $\mathcal{S}$. In terms of non-temporal operators, DL-Lite TQL is thus equipped with negation, conjunction. Via recursion, it can also express disjunction and implication (T3). In terms of temporal operators, it has $\mathcal{S}$ and $\mathcal{U}$ at disposal. Via recursion, relations $\boxplus, \boxminus, \oplus$ and $\ominus$ can be expressed as well (T4). All temporal relations apply to either the past or future as a whole, with no possibility to specify time intervals.

All formulae are evaluated to be either true or false, thus with no provisions for uncertainty in the data. The streaming data sequence $(\mathcal{A}_{i \in \mathbb{Z}})$ is considered to in-order. As such, unreliability of the stream such as out-of-order arrival is not taken into account (T5).

Furthermore, Klarman and Meyer [89] introduces several fragments of the aforementioned TQL, restricting the available operators within either the static (T3) or temporal (T4) dimension, or both. A first fragment, TQL^{+}, consists of the formulae that do not contain negation $\neg$. As expressivity of disjunction and implication relies on negation, only disjunction and implication are lost (T3) as well as $\boxplus$ and $\boxminus$ (T4). A second fragment TQL$^{\exists}$ limits the $\mathcal{S}$ and $\mathcal{U}$ operators to occur only with the general truth $\top$ as their first argument, i.e. $\top \mathcal{U} \phi$ and $\top \mathcal{S} \phi$, which coincides with $\oplus$ and $\ominus$ respectively. This restriction removes the ability to express the $\mathcal{S}$ and $\mathcal{U}$ relations fully, but retains expressivity of the other temporal relations (T4). A third and last fragment proposed by Klarman and Meyer [89] is TQL$^{\exists,+}$, the intersection of TQL$^{\exists}$ and TQL^{+}. In this case, the only relations remaining are $\wedge$, $\oplus$ and $\ominus$ (T3, T4).

The complexity of fact checking in DL-Lite TQL when it comes to deduction, with the fact encoded as a TQL formula, is proven to be PSPACE-complete (T7). Regarding its fragments, deduction is NP-complete for TQL$^{\exists}$, TQL^{+} and TQL$^{\exists,+}$. Consequently, the above tasks are decidable in DL-Lite TQL as well as fragments TQL$^{\exists}$, TQL^{+} and TQL$^{\exists,+}$.

2.7.1.2 $\text{LTL}_{\mathcal{EL}}^{\mathbf{X,G}}$

Extending DL with temporal operators is also done in $\mathbf{LTL}_{\mathcal{EL}}^{\mathbf{X,G}}$, introduced by Tirtarasa and Turhan [127], where LTL operators $\bigcirc$ (next) and $\Box$ (global always) are incorporated into DL $\mathcal{EL}$ (T1). It aims to construct temporalised query concepts, which in turn enables one to answer temporal queries, more specifically temporal conjunctive queries as seen in the context of ontology-based data access. With a stream representation $I = (A_i)_{i\in\mathbb{N}}$, having A_i ABoxes and time domain $\mathbb{N}$ (T2) , $\text{LTL}_{\mathcal{EL}}^{\mathbf{X,G}}$ uses sets to represent concepts, e.g. the concept 'house' is represented by the set of all elements that are a house (e.g. the set of townhouses, villas, apartments, houseboats etc.). These concept are subsequently combined with static operators $\cap$ and $\exists$ (T3) and temporal operators $\mathbb{X}$ and $\mathbb{G}$, expressing relations $\bigcirc$ and $\boxplus$ (T4). By asserting whether instances adhere to these concepts, $\text{LTL}_{\mathcal{EL}}^{\mathbf{X,G}}$ perform an inference over temporal relations, i.e. reasoning about time (T6).

With roots in DL, $\text{LTL}_{\mathcal{EL}}^{\mathbf{X,G}}$ relies on binary assertion of statements over a sequence $(\mathcal{A}_\rangle)_{i\in\mathbb{N}}$, not accounting for uncertainty in either data or the stream (T5). Tirtarasa and Turhan [127] does not consider concrete results on the complexity or decidability of $\text{LTL}_{\mathcal{EL}}^{\mathbf{X,G}}$ (T7).

2.7.1.3 $\mathcal{EL}^{++}$ ontology streams

Another DL-based formalism (T1) is **ontology streams over** $\mathcal{EL}^{++}$ [94], which aims to identify the nature and cause of changes on an ontology level, where the ontology is seen as the combination of ABox and TBox. Here, a stream is a sequence of ontologies $(O_{t_0}, O_{t_1}, \ldots, O_{t_n})$ where O_t are ontologies. The sequence is finite and ranges from time point t_0 to t_n, taken from time domain $\mathbb{N}$ (T2). Aside from the intersection of concepts $\cap$ and the existential operator $\exists$ inherent to $\mathcal{EL}^{++}$ [15] (T3), Lécué [94] does not introduce new operators, neither temporal or non-temporal (T4). The overall objective of detecting changes in ontologies is modeled via the deduction of new information, whereas model checking is used to spot inconsistencies across ontologies in the stream. As it incrementally updates the ontology, but does not consider temporal relations between the incoming data, the $\mathcal{EL}^{++}$ ontology streams fall under reasoning over streams (T6).

With regards to the stream and the data within the stream, the formalism assumes the order of the stream is correct and the data itself, i.e. the ontology, is trustworthy (T5). From $\mathcal{EL}^{++}$, the formalism employs the concept of General Concept Inclusion axioms (CGIs). Checking if a given GCI stays invariant k consecutive moments over an ontology stream O_m^n is decidable, with complexity PTIME-complete [94] (T7).

2.7.2 Temporal Logic Formalisms

The existence of temporal logic languages predates the field of SR. As such, a recurring approach to SR formalism design is to enhance a state-of-the-art static formalism or language with temporal capabilities.

2.7.2.1 DatalogMTL and its fragments

One such formalism is DatalogMTL [36], which starts from Datalog (T1) as theoretical basis for static data and incorporates operators from MTL into the syntax. DatalogMTL aims to specify and reason over complex events in real-time systems. As a recursive rule language, it is well-suited to tackle analysis involving recursive propagation of information. In DatalogMTL, the syntax remains rooted in Datalog's foundational principles, with formulae comprising of atomic assertions and temporal constraints expressed through temporal operators. These operators, specifically $\mathcal{S}_I$, $\mathcal{U}_I$, $\boxplus_I$, $\boxminus_I$, $\diamondplus_I$, $\diamondminus_I$, introduce temporal logic into the framework, enabling the specification of temporal patterns and constraints over data streams (T4). The integration of temporal logic enhances Datalog's expressive power, allowing for sophisticated reasoning over dynamic and evolving data streams. Furthermore, rules in DatalogMTL maintain the familiar structure of traditional Datalog, with a clear separation between rule heads and bodies. The rule body consists of formulae built using conjunction $\wedge$ and the six aforementioned temporal operators, whereas in the rule head, only formulae with operators $\boxplus$ and $\boxminus$ are permitted. Compound expressions are thereby allowed, such as $\left(\boxplus_{I_1} \phi\right)\mathcal{U}_{I_2}\left(\diamondminus_{I_3} \psi\right)$ ("ϕ always holds interval I_1 in the future up until ψ holds somewhere in I_2 in the past") in rule bodies and $\boxplus_{I_2} \boxminus_{I_2} \psi$ ("everywhere in interval I_1 in the future, ψ holds in interval I_2 in the past") in both bodies and heads. Since DatalogMTL only allows $\wedge$ in the body and supports implication via the rule construct, the only supported static relations are $\wedge$ and $\rightarrow$ (T3). As rules can contain temporal expressions, both in the rule head and body, DataloMTL is capable of reasoning about time (T6). Since it only considers logical entailment and a correct order of the stream, it does not account for uncertainty in the data or the stream (T5).

DatalogMTL considers a timeline T, which can either be an integer timeline $\mathbb{Z}$ ('unary timeline') and rational timeline $\mathbb{Q}$ ('binary timeline'). A stream is modeled as a function S which maps a point t from the timeline to a set of grounded atoms, $S(t)$ (T2).

Brandt et al. [36] provides several complexity and decidability results for DatalogMTL. For a given DatalogMTL program, i.e. set of DatalogMTL rules, consistency checking is proven to be EXPSPACE-complete, even when only variable-free propositions are considered for atoms (T7).

DatalogMTL sports several fragments as well as extensions, each aimed at specific tasks.

DatalogMTL^{FP}[137] is dedicated to deriving information about the future based on information from the past. To achieve this, the only operators allowed in the rule body are the past operators $\boxminus$ and $\diamondminus$, whereas the rule head only includes the future operator $\boxplus$, limiting temporal expressivity (T4). Recall diamond and since/until operators are already disallowed in rule heads by regular DatalogMTL.

DatalogMTL$^{\neg}$ has been proposed by Cucala et al. [55] as a supplement to DatalogMTL, catering to a growing need of non-monotonic negation, where previous conclusions can possibly be revoked due to newer information [41]. The rule bodies of original DatalogMTL are composed of multiple *atoms* connected using $\wedge$. The atoms are strictly *positive*, i.e. they do not contain negation $\neg$. DatalogMTL$^{\neg}$ introduces stratified negation as failure by extending the format of rule bodies to $A_1 \wedge \ldots \wedge A_n \wedge \neg A'_1 \wedge \neg A'_m$, thus adding operator $\neg$ (T3). The term stratified indicates that negation is applied in layers (strata), ensuring that rules only negate facts computed in lower strata. This prevents cyclic dependencies through negation and preserves tractability. Importantly, this extension does not increase the complexity of fact entailment [55] (T7).

The atoms that provide the basis for the rules can contain variables, but DatalogMTL operates under the assumption that every variable in the rule head must also occur in (an atom of) the rule body. This prohibits 'existential rules', where one can conclude the existence of an (unspecified) instance based on several pieces of information. Nissl [112] introduces a template to incorporate existential rules into DatalogMTL, leading to the extension DatalogMTL$^{\exists}$. DatalogMTL$^{\exists}$ can be used with two semantics, either 'natural semantics' that align with those of Datalog fragment Datalog$^{\exists}$ which similarly allows existential rule in Datalog (cfr. Leone et al. [99]), or 'uniform' semantics. In short, with the latter an existential quantification $\exists$ must hold for a single instance over a given interval, whereas the former allows for multiple instances throughout an interval, as long as the existential quantification $\exists$ holds at each time point separately (T3). Nissl [112] provides complexity results for model checking under both semantics, being EXPSPACE-complete for uniform semantics under the Closed World Assumption (CWA), and undecidability in all other cases (T7).

DatalogMITL is a fragment of DatalogMTL identical in syntax, save for the exclusion of singleton intervals (e.g. $[1, 1]$) in the operators (T4) [137]. This follows from the MTL fragment MITL, which only allows for non-singleton intervals. A benefit of this limitation is that the required memory for an arbitrary reasoning algorithm can be controlled independently of the chosen time domain [137]. For DatalogMITL^{FP}, the forward-propagating fragment of DatalogMTL, Wałęga, Kaminski, and Grau [137] provides complexity results for fact entailment. In case of a binary encoding of the time points, checking whether a single piece of information at a specific time, denoted $\alpha@t$, holds for the given dataset, is PSPACE-complete. In case of a unary encoding, it is P-complete (T7).

As DatalogMITL^{FP} is a restricted fragment of both DatalogMTL^{FP} and of DatalogMTL, the complexity results of DatalogMITL^{FP} also serve as lower bounds on the complexity of the parent formalisms (T7).

2.7.2.2 Metric Spatio-Temporal Logic (MSTL)

Another MTL-extended formalism is MSTL [98], which combines MTL with Region Connection Calculus (RCC-8) [118], a qualitative spatial reasoning framework aimed at describing spatial relations between regions (e.g. overlap, bordering, ...) (T1). The aim of MSTL is to perform spatio-temporal reasoning, more specifically to infer unobservable qualitative spatial relations and their evolution over time via the introduction of the 'landmark' concept, i.e. a region that remains unchanged throughout time.

Focusing on the temporal part, we observe MSTL covers operators **next**, $\boxplus$ and $\oplus$. Additionally, it sports operators $\Box_{[i_1,i_2]}$ and $\Diamond_{[i_1,i_2]}$. These operators define the 'always' and 'eventually/once' temporal relations, but the semantics of MSTL dictate that the interval subscripts specify an absolute time interval, compared to the relative time intervals seen in other MTL-based formalisms (cfr. [126, 137, 138]). As a result, MSTL does not support the 'traditional' MTL-operators $\boxminus$ and $\ominus$, despite its ability to describe 'always/once' relations that may occur before a considered time point (T4). Furthermore, MSTL has static operators $\neg, \wedge, \vee, \veebar, \rightarrow, \forall$ and $\exists$ (T3), to be combined freely with the aforementioned temporal operators, to construct their spatio-temporal formulae.

MSTL models time explicitly by means of an ordered set of time points $(T, <)$, which can thus be $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ or other. In the semantics of MSTL, a stream is not defined explicitly, but rather implicitly via an interpretation which maps every time point t and predicate P to a set of constants for which P holds at time t. As a result, the implicit stream has form $S : T \times Pred \rightarrow Const^*$, with $Pred$ the set of predicates and $Const$ the set of constants. No window operators are considered (T2).

By assessing the truth value of these spatio-temporal formulae, MSTL is able to perform reasoning about time over streams of data (T6). Problems of uncertainty of the data or the stream order are not considered in Leng and Heintz [98] (T5), nor does it consider any results in terms of complexity or decidability (T7).

2.7.2.3 Predictive MTL (P-MTL)

A final MTL-based SR formalism concerns P-MTL [126], a formalism for reasoning over uncertain states within a SR context. It aims to fill the gap between lower-level probabilistic inference on sensor data and the higher-level logical reasoning performed on the same information. Starting from MTL (T1), P-MTL incorporates states that can be predicted, thereby enabling temporal reasoning over uncertain information, in contrast to only certain information. It considers stochastic variables, for example the location of a ball, and predicts the value of said variable in a future state, based

on a given state. The probabilistic aspect of the logic is embedded via probabilistic statements, say ϕ, which take values on $[0, 1]$, and evaluating these expressions using comparative expressions such as $\phi > 0.85$ or $\phi < 0.05$ (T5).

The stochastically evaluated formulae are formed by recursively combining predicate terms such as $P(\tau_1, \tau_n)$ with both static operators $\neg, \wedge\vee, \rightarrow, \forall, \exists$ (T3) and temporal operators $\bigcirc, \boxplus_{[t_1,t_2]}, \diamondplus_{[t_1,t_2]}, \Box$ and $\Diamond$ (T4). To this end, it is able to perform (stochastic) reasoning over time (T6).

Similar to MSTL above, a stream is implicitly defined as a mapping $S : T \times Pred \rightarrow Const^*$ from a time point t and predicate $P \in Pred$ to a set of constants $Const$ for which P holds at time t. The time points t are taken from a natural timeline $T = \mathbb{N}$. Formulae are evaluated on a single stream, without implementing window operators (T2).

Tiger and Heintz [126] states the construction of probabilistic statements ϕ being evaluated in comparative expressions such as $\phi < 0.05$, does not increase complexity or affect decidability compared to FOL expressions. Concrete effects on complexity and decidability of the evaluation over the temporal dimension are not given by Tiger and Heintz [126] (T7).

2.7.2.4 Predictive Mission-time Linear Temporal Logic (PMLTL)

PMLTL [14] is an extension of Mission-time Linear Temporal Logic (MLTL) (T1) aimed at predicting the environment of a system. It does so in a multimodal approach, i.e. by considering and evaluating multiple different scenarios (models) of the future.

PMLTL adopts the time representation of MLTL of finite timed traces $\pi_0\pi_1 \ldots \pi_n$, a finite sequence of sets of atoms. As a result, the time domain follows from the indexing of these timed traces, i.e. $\mathbb{N}$ (T2). Starting from the atoms which populated the timed traces π_j, PMLTL constructs expressions using static operators $\neg, \wedge, \vee$ and by recursion, $\rightarrow$ (T3), and temporal operators $\boxplus_I, \diamondplus_I, \mathcal{U}$ (T4).

In order for a PMLTL formula to hold by a certain time point in the future (the 'deadline'), it needs to either be inferred from the observed data, or if it is predicted to be true in all of the different models of the future, based on the given timed trace. As such, it thus models the uncertainty of a statement concerning the future by considering multiple different models (T5). As formulae are able to express various temporal relations between the atoms, PMLTL is capable of performing reasoning about time (T6). No analysis of complexity or decidability of PMLTL is performed in Aurandt et al. [14] (T7).

2.7.2.5 Temporal Equilibrium Logic (TEL)

TEL extends EL by incorporating past temporal operators from LTL. As a result, TEL can be used to tackle non-monotonic reasoning problems with a temporal dimension. From a theoretical point of view, Cabalar and Pérez Vega [38] extend HT

with LTL operators, producing Temporal Here-and-There Logic (THT). Using the same minimisation principle that produces EL from HT (cfr. Section 2.4.2), TEL is produced from THT. In TEL, formulae consist of atomic assertions and temporal constraints expressed through formulae with static operators $\wedge, \vee$ and $\rightarrow$ (T3) and temporal operators $\bigcirc, \mathcal{U}, \boxplus$ and $\diamondplus$ (T4). Negation $\neg\phi$ is additionally available as shorthand for the formula $\phi \rightarrow \bot$. By integrating (future) temporal operators into EL, TEL provides a natural and intuitive framework for reasoning about time (T6), specifically for temporal relations concerning the future, given only the 'future'-operators are considered. $\langle H, T\rangle$ Building upon EL, the basis for a stream in TEL is a pair $\langle H, T\rangle$. For TEL, $\langle H, T\rangle$ is a possibly infinite sequence of pairs of atom sets $\langle H_i, T_i\rangle$. The exact nature of 'time point' i and thereby the timeline T is not specified, but allows for both finite and infinite timelines (T2).

Similar to EL, TEL is able to deal with uncertainty in the data by distinguishing between a 'here' part H_i of what is known to be true at time i and a 'there' part T_i which could potentially be true at i (T5). Cabalar and Pérez Vega [38] notes complexity and decidability analyses as potential future work following the introduction of TEL (T7).

In later work, Aguado et al. [2] presented a method of extending TEL with operators $\mathcal{S}$, $\boxminus$, $\diamondminus$ and $\bullet$ (T4). The extension can subsequently be reduced to the original TEL syntax via the introduction of auxiliary atoms.

2.7.2.6 Metric Equilibrium Logic (MEL)

In a similar vein to TEL, Cabalar et al. [39] extends EL (T1) with MTL operators. It does so via the same procedure as TEL, starting from adding MTL operators to HT to obtain Metric Here-and-There Logic (MHT) and subsequently consider the minimal timed traces to arrive at the proposed MEL. MEL is equipped with the interval-variants of the operators **next**, **previous**$_I$, $\mathcal{S}_I$ and $\mathcal{U}_I$. Via recursive definition of the formulae, operators $\boxplus_I$, $\boxminus_I$, $\diamondplus_I$ and $\diamondminus_I$ also follow (T4). MEL retains the same static operators as TEL, i.e. $\wedge, \vee$ and $\rightarrow$ (T3). Likewise, dimensions (T2) and (T5-7) remained unchanged compared to TEL discussed above.

2.7.2.7 Linear Temporal Public Announcement Logic (LTPAL)

LTPAL [63] extends PAL (T1) with operators from LTL, enabling reasoning about temporal properties and events within a framework of public announcements and temporal constraints. It is designed to reason over the output of AI systems, such as classifiers, on a uniform basis. In LTPAL, formulae consist of public announcements and temporal constraints expressed through LTL temporal operators **next** and $\mathcal{U}$ (T4). This in addition to static operators $\neg$, $\wedge$ (T3) inherited from PAL.

Via recursion, LTPAL is also fitted with static operators $\vee$ and $\rightarrow$ (T3) and temporal operators $\boxplus$ and $\oplus$ (T4). The resulting set of operators allow for the specification of temporal patterns and dependencies over data streams, thus enabling reasoning about time in LTPAL (T6).

LTPAL models the flow of time via a model, referred to as a transition system. Here, a finite set of states describes changes over time and the LTPAL formulae are evaluated across the various states. As a result, the time domain is implicitly assumed to be a finite set of time points $(0, \ldots, n)$ (T2).

The formulae of LTPAL are evaluated on a binary basis, not accounting for uncertainty of the expressed relations. The modelling of the transition system as a set of finite states is assumed to be correct, i.e. no measures are taken w.r.t out-of-order data (T5). Dehkordi, Alizadeh, and Movaghar [63] does not provide details on the complexity or decidability of any evaluation (T7).

2.7.2.8 Accumulative Signal Temporal Logic (ASTL)

Zhao et al. [145] considers service monitoring of IoT systems. It thereby focuses on Quality-of-Service (QoS) constraints and how one can estimate the satisfaction of these type of constraints. In order to quantify this satisfaction, Zhao et al. [145] adds a robustness function to STL (T1), leading to a new framework named Accumulative STL. As an extension, ASTL inherits most of STL's expressivity, both in terms of temporal and non-temporal relations. Negation $\neg$, conjunction $\wedge$ and disjunction $\vee$ are supported (T3), as well as future-oriented temporal operators $\mathcal{U}_I$, $\boxplus_I$ and $\oplus_I$ with interval semantics (T4).

Likewise, ASTL inherits the stream representation of a stream as a function over time to a set for formulae $S : T \rightarrow \mathcal{F}$ from STL, where $T = \mathbb{N}$ and formulae F are constructed by combining signal statements such as "signal x is lower than c" ($x \leq c$) with aforementioned operators $\neg, \wedge, \vee, \mathcal{U}_I, \boxplus_I$ and $\oplus_I$. (T2).

The key innovation in ASTL is the introduction of the robustness function $\hat{\rho}$, which measures the degree to which a given signal ω satisfies an ASTL formula ϕ at a specific time point t. The robustness function assigns a real-valued score to the satisfaction of the formula: high positive values indicate a strong satisfaction of the constraint, while negative values suggest a violation. This approach allows ASTL to handle uncertainties in dynamic, real-time IoT environments, where data is subject to variability and noise (T5). No complexity and decidability results are provided by Zhao et al. [145] (T7).

2.7.2.9 Probabilistic Signal Temporal Logic (ProbSTL)

ProbSTL [125] extends STL (T1) to better model uncertain or probabilistic events in signal-based systems. It is mainly catered towards the domain of Signal Reasoning, where the goal is to analyse and reason about continuous signals under uncertainty.

ProbSTL focuses on uncertainty in individual statements or conditions, particularly by introducing probability distributions over signals. It extends STL by allowing formulas to express probabilistic constraints, meaning that instead of a rigid Boolean outcome (true or false), the satisfaction of a formula is represented probabilistically, i.e. over interval $[0, 1]$ (T5). For example, ProbSTL allows one to specify constraints like "the probability that the signal stays above a threshold for at least 10 seconds is greater than 0.9.". As a result, ProbSTL is an example of a SR formalism for (probabilistic) reasoning about time (T6).

The most elemental stream in ProbSTL is referred to as a *signal*, modeled as a function from time domain T to a value domain ($\mathbb{D}$) for said signal (T2). ProbSTL assume a discrete time domain $T = \mathbb{N}$. Statements ϕ in ProbSTL are comprised of static operators $\neg$ and $\vee$ and temporal operator $\mathcal{U}_I$, applied to atoms $f_{\mathbb{B}}(l)$, which represent a Boolean evaluation of a statement l. Via recursion, the total set of static operators can be extended to $\neg, \vee, \wedge$ and $\rightarrow$ (T3), whereas the available temporal operators can be extended to $\boxplus_I$ and $\oplus_I$, in addition to $\mathcal{U}_I$ (T4).

Tiger and Heintz [125] identifies two factors that determine the complexity of evaluation a ProbSTL statement ϕ. The first is the length of ϕ, i.e. the amount of steps it takes to construct ϕ given the static and temporal operators, and the second complexity factor depends on the properties of the atoms. As a result, satisfiability checking is linear in terms of the length of formula ϕ, implying decidability of the satisfiability problem (T7).

2.7.2.10 Evolution Temporal Logic (EvTL)

Building on the theory of STL, Castiglioni, Loreti, and Tini [50] introduces a probabilistic variant of STL (T1) called EvTL. Similar to ProbSTL, EvTL addresses uncertainty that affects the system state as a whole. While ProbSTL deals with uncertainty in individual signal values, EvTL aims to model the probability distribution of data within a streaming system. It evaluates whether the observed distribution of the data aligns sufficiently with a proposed distribution. It models the stochastic behaviour of a stream, rather than the contents of a stream itself (T5). By comparing the observed stochasticity to other distributions, EvTL performs reasoning about time over stochastic information (T6).

A data stream is not modeled explicitly in EvTL. Instead, Castiglioni, Loreti, and Tini [50] considers an *evolution sequence*, which is a countable sequence of distributions of the data within a system. Following this, time is implicitly assumed to be discrete, more specifically $\mathbb{N}$, as each element of the sequence can be assigned an index $i \in \mathbb{N}$ (T2).

Statements in EvTL are centered around the atoms, which are expressions which compare the observation distribution of the data to a chosen distribution. To build statement formulae, EvTL makes use of the STL operators $\neg$, $\vee$ and $\mathcal{U}_I$. Via recursion, the available set of operators in hence $\neg, \vee, \wedge, \rightarrow$ for static operators (T3) and

$\mathcal{U}_I, \boxplus_I, \oplus_I$ for temporal operators (T4). Castiglioni, Loreti, and Tini [50] does not go into detail regarding complexity or decidability of the EvTL formalism (T7).

2.7.2.11 Robustness Temporal Logic (RobTL)

RobTL [49] is another extension of STL (T1) designed to deal with uncertainty in systems that process continuous signals, particularly in scenarios where the signals are subject to varying degrees of noise, error, or delay. While formalisms like STL and its variants, such as ProbSTL, focus on modelling specific conditions or probabilistic events, RobTL emphasizes robustness against perturbations of the data, i.e. against out-of-order arrival (T5).

In line with ProbSTL and EvTL, RobTL opts to represent time implicitly by means of a sequence of data states, the aforementioned evolution sequence. As such, time is modeled implicitly as a set $\mathbb{N}$ (T2). RobTL uses atoms that express the distance between the considered evolution sequence and a perturbation of the sequence. With these atoms, RobTL constructs formulae using the static operators $\neg, \wedge$ and by recursion $\vee$ and $\rightarrow$ (T3), and temporal operator $\mathcal{U}_I$, which in turn induces $\boxplus_I$ and $\oplus_I$ (T4). By evaluating these expressions, RobTL performs inference over the temporal relations between behaviour of data within a given system over time, thus performing reasoning about time (T6). The complexity or decidability of RobTL is not evaluated in Castiglioni, Loreti, and Tini [50].

2.7.2.12 Signal Diagnostic Processing Language (SDRL)

SDRL [122] was proposed to enable the formulation of complex diagnostic tasks to detect anomalies in engineering and maintenance processes at runtime. In contrast to the other SR formalism discussed, SDRL is not based on a pre-existing formalism (e.g. EL for TEL and MEL, STL for ProbSTL, EvTL and RobTL) (T1). The recursively defined signal processing expressions only include temporal operators $\mathbf{next}^i$ and $\mathcal{S}^i$, where $\mathbf{next}^i := \mathbf{next}(\mathbf{next}^{i-1})$. Furthermore, SDRL sports aggregator functions (min, max, avg, sum) for signals. The divergent choice of temporal operators makes it difficult to compare to the previously considered formalisms.

SDRL reasons on signals, which take on different value at different time points. For a single signal s, the value at time point t is defined as $f_s(t)$. As a result, the concept of a stream is thus represented as a function from a time point to a value (T2). No specifics on the time domain are considered in Savkovic et al. [122].

The temporal operators of SDRL express the duration ($duration(\geq, t)$) and how long before or after ($before[t]$, $after[t]$) one signal occurs relative to another. Using these operators, relations akin to $\mathcal{U}_I, \mathcal{S}_I, \boxplus_I, \boxminus_I, \oplus_I, \ominus_I$ can be expressed (T4). Additionally, SDRL has static operators $\wedge$ and $\neg$, which in turn induce operators $\vee$ and $\rightarrow$ (T3). By evaluating the various temporal expressions across different signals, SDRL performs reasoning about time (T6).

Savkovic et al. [122] encodes SDRL into a custom extension of DatalogMTL, DatalogMTL$_{NR}$ with aggregator functions. Via DatalogMTL$_{NR}$, Savkovic et al. [122] shows fact checking (i.e. checking whether a given signal processing rule is fired) in SDRL is PSPACE-complete and decidable. Furthermore it is AC^0 in terms of the size of the signal and ontological data considered (T7).

Within SDRL itself, concepts of uncertainty of the data or out-of-order arrival of the data is not taken into account (T5).

2.7.2.13 TRIO-based Event Specification Language (TESLA)

The CEP language TESLA [56] was introduced with the objective to provide a formal specification of real-time systems, such as the control of industrial processes. It is rooted in the temporal logic TRIO, introduced by Morzenti, Mandrioli, and Ghezzi [109] (T1). TESLA was introduced to overcome the limitations of several CEP and data stream processing languages, notably the lack of rigorous semantics of the former and the limited complex event recognition in the latter.

TESLA assumes an influx of timestamped events, for example `Temp`(10°)@12, to arrive in order of ascending timestamps. As such, a data stream is (implicitly) modeled as a sequences of timestamped data. No specific assumptions are made regarding the nature of the time domain (T2).

Central in TESLA are the patterns used to infer a complex event. As a CEP formalism, it performs reasoning about time (T6). The patterns themselves are constructed using connectives `and`, `not` as well as constructions such as `first-within`, `last-within`, k-`within` or `each-within`. Patterns furthermore allow for explicit quantification over time, quoted to be one of TESLA's main strengths [56]. As a result, the static relations expressible by TESLA are $\neg, \wedge, \vee$ and $\rightarrow$ (T3). In terms of temporal relations, the ability to quantify directly over time allows one to construct expression that capture the same relations as $\boxminus_I, \boxplus_I, \diamondminus_I, \diamondplus_I, \mathcal{S}_I, \mathcal{U}_I$, as illustrated in Cugola and Margara (T4).

As considered above, TESLA assumes an in-order arrival of data. Furthermore, it does not consider any uncertainty regardign the data itself (T5). Lastly, Cugola and Margara [56] does not analyze the complexity or decidability of TESLA (T7).

2.7.2.14 Temporal Action Logic (TAL)

Within the field of Knowledge Representation, we find the class of logics called 'Temporal Action Logics', designed to represent reasoning over such actions and change. TAL [70] has found its way into the field of SR via systems such as DyKnow and TALPlanner [69]. Although it does not build upon existing (non-temporal) formalisms directly, it has been translated into FOL (T1). For its representation of time, TAL does not impose conditions on the time domain except linearity, meaning the most common time domains ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ etc.) are valid under TAL, but branching

time domains are not allowed. The concept of a stream in TAL is most closely embodied by a *narrative*, which is a set of timestamped expressions (T2).

TAL is expressed via a base language, $\mathcal{L}(ND)$, which considers *timed formulae*. These timed formulae are constructed starting from atoms (t, f), which represents the value of an expression f at time t. Formulae are built using static connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$ (T3), as well as additional operators $\mathcal{U}_I$, $\boxplus_I$ and $\oplus_I$ (T4). Given its ability to express and infer from temporal patterns, TAL performs reasoning about time (T6). However, this explicit quantification induces an increase in reasoning complexity, because formulas are expressed in full first-order logic with arithmetic constraints over time variables. This means that satisfiability and entailment require handling nested quantifiers, variable instantiations, and constraint solving, which significantly raises the computational burden compared to propositional or modal temporal logics.

Within TAL, the data at hand is strictly assumed to be correct and correctly timestamped. Therefore, TAL does not incorporate handling of uncertainty of data or of the data stream into account (T5). Doherty and Kvarnström [70], nor the later work of Doherty, Heintz, and Kvarnström [69] explore the theoretical complexity or decidability of the formalism (T7).

2.7.3 SR Query Languages

Query languages play an important role in SR by enabling efficient querying and filtering of streaming data in real time. They focus on retrieving information rather than complex reasoning. However, they are included in our survey as selecting relevant data based on specified conditions can itself be seen as a form of reasoning. This perspective aligns with the broader understanding of SR as discussed in Bonte et al. [33].

As mentioned before, the domain of SR is strongly embedded in the Semantic Web domain. With the prominent position of SPARQL as query language for RDF data, it is unsurprising that SPARQL has frequently been chosen as a basis for a query language over RDF streams. As covered in 2.4.6, stream querying formalisms retrieve data from a stream by means of specific queries, which stipulate conditions to be met. Once retrieved by the query engine, batches of data, for example induced by windowing, can be reasoned upon using non-temporal reasoning engines (e.g. DL or Datalog reasoners). As stream querying formalisms do not perform explicit reasoning, complexity and decidability results, when provided, are not comparable to other SR formalisms discussed. As such, we do not consider dimension T7 in the discussion of this category.

2.7.3.1 Continuous SPARQL (C-SPARQL)

Introduced in 2009, C-SPARQL [22] takes inspiration from earlier stream query languages, such as CQL [11], and applies it to the RDF-specific SPARQL language (T1). C-SPARQL considers RDF Streams $S = \left(\langle{:}s_i{:}p_i{:}o_i\rangle, t_i\right)$ of RDF triples $\langle{:}\ s : p : o\rangle$,

timestamped with $t_i \in T$. The time domain T from which the timestamps are taken can be chosen freely. In the examples of Barbieri et al. [22], RDF literals of type `:xsd:dateTime` are used. Note that the RDF stream is ordered by its timestamps in a non-decreasing manner. The RDF stream can be infinite in length. In order to query the infinite stream, C-SPARQL employs time- and tuple-based windows (T2).

In the querying of the data, C-SPARQL provides several operators in order to specify the desired data. Using the `WHERE` clause, conditions can be imposed on the data to subsequently match these conditions to specific instances. The implicit conjunction of conditions within the `WHERE` clause, in addition to SPARQL keywords `NOT` and `UNION`, allows one to express negation $\neg$, conjunction $\wedge$ and disjunction $\vee$ of various conditions (T3). In C-SPARQL, the windows of the queries are the main way of extracting the desired temporal data. Beyond the window operators, C-SPARQL does not provide explicit means to express temporal operators [142] (T4).

With the introduction of RDF data streams and syntax for windowing and aggregation, C-SPARQL enables us to reason over data that arrives over time, in contrast to the static data of (base) SPARQL. Via the timestamp function and `WHERE` clause, C-SPARQL allows to retrieve data based on temporal patterns, thus allowing for reasoning about time, where the reasoning is encapsulated inside the query conditions (T6). Our study did not provide us with papers that research the computational complexity of C-SPARQL (T7). Furthermore, C-SPARQL does not have dedicated techniques to handle uncertainty of data or out-of-order arrival of data (T5).

2.7.3.2 Streaming and Temporal ontology Access with a Reasoning-based Query Language (STARQL)

Özçep, Möller, and Neuenstadt [113] observed existing stream data management systems at the time employed a "bag of assertions" semantics, i.e. these systems pooled incoming information together in a " bag" and disregarded their chronological order. As a result these systems suffer some loss of (temporal) information. STARQL was introduced [113] to resolve this, building on a basis of SPARQL (T1). STARQL's semantics are defined over *synchronous streams*. These synchronous streams are represented by an infinite sequence of chronologically ordered timestamped atoms, where the atoms take on the form of DL-Lite ABox assertions and the timestamps stem from a linear timeline $T(,<)$. STARQL proceeds by applying time-based window operators to this infinite sequence to extract a *temporal Abox*, a set of timestamped Aboxes, which is subsequently converted to a finite set of (non-temporal) Aboxes (T2). This allows for classical SPARQL queries within a single ABox, while enabling temporal reasoning, i.e. reasoning about time, by performing inter-ABox reasoning (T6). Similar to C-SPARQL, STARQL is equipped with conjunction, disjunction and negation (T3). Keywords `FORALL` and `EXISTS` provide ways to universally or existentially quantify. The `HAVING` clause allows to define conditions for inter-ABox reasoning, thus allowing to impose conditions that mimic $\boxplus, \boxminus, \diamondplus, \diamondminus$ to be imposed on the data

queries (T4). For example, $\boxplus A(x)$ can be modeled as `HAVING FORALL i > NOW IN SEQ: A(x)`, where `SEQ` is the sequence used to represent the data stream.

STARQL assumes the data to be true, i.e. without uncertainty, and to arrive in order. It does not consider methodology for handling data uncertainty in its formalism (T5).

2.7.3.3 SPARQL$_{\text{stream}}$

SPARQL$_{\text{stream}}$ is an extension to SPARQL 1.1 (T1) proposed by Calbimonte, Corcho, and Gray [44], designed to provide ontology-based access to streaming data sources stored in relational models. Similar to C-SPARQL, it represent streams by means of RDF streams $S = \big(\langle : s_i : p_i : o_i \rangle, t_i\big)$ of RDF triples $\langle : s : p : o \rangle$. Windowing operators over time allow for information extraction from a data stream. Compared to C-SPARQL, SPARQL$_{\text{stream}}$ only allows for time-based windows, argumenting tuple-based windows may not be able to capture graph patterns spanning multiple triples (T2). SPARQL$_{\text{stream}}$ inherits the static relations from SPARQL, i.e. $\wedge$, $\neg$ and $\vee$ as well as quantification (T3). Despite the representation of streams as sets of time-stamped triples, Calbimonte, Corcho, and Gray [44] makes no note of being able to express temporal relations on the data, suggesting the window operators work in a "bag of assertions" fashion, which leaves one unable to identify temporal relations between data points within the same window (T4). As a result, SPARQL$_{\text{stream}}$ is used for stream querying and can, in conjunction with an RDF reasoner, be used for reasoning over time (T6). SPARQL$_{\text{stream}}$ does not tackle problems concerning uncertainty in data or data streams (T5).

2.7.3.4 Continuous Temporal SPARQL (CT-SPARQL)

Although with C-SPARQL one can reason about data that arrives in the form of a stream, reasoning about temporal relations between the data is not accommodated, i.e. C-SPARQL is able to reason *over* time, but not *about* time. Argumented by Yang [142], C-SPARQL windowing causes the data to lose its (time-)sequential nature, as windows transform the section of the RDF stream into an RDF graph without time annotation. CT-SPARQL aims to retain the sequential nature using a SEQ relation between events [142]. Furthermore, it breaks down an RDF stream into multiple indexed RDF graphs, representing snapshots over time.

CT-SPARQL shares its SPARQL foundation (T1), stream representation (T2) and static expressivity (T3) with C-SPARQL. The expression 'A `SEQ` B' captures the tuples (A_i, B_j) where A_i and B_j are events of type A and B respectively and B_j is the first of type B to come after A_i. Yang [142] state any complex temporal relation can be constructed using the `SEQ` operator, with a formal discussion thereof left for future

work. No further work of said author has been published which considers such discussion (T4). Despite temporal expressivity being not yet formally explored, the `SEQ` operator provides a basis for derivation of temporal patterns. As a result, CT-SPARQL is able to derive data which matches temporal patterns, i.e. reasoning about time (T6).

Uncertainty in data or data streams are not addressed within the CT-SPARQL framework (T5).

2.7.3.5 ETALIS Language for Events (ELE)

ELE, introduced by Anicic et al. [8], defines the formal syntax and semantics of Event-Driven Transaction Logic Inference System (ETALIS), a system for CEP. It is rooted in a Logic Programming (LP)-approach to CEP (T1), thus oriented towards reasoning about time (T6). It represents streams by means of *event streams* ϵ, a mapping of grounded event predicates to a set of time points in $\mathbb{Q}^+$, indicating at which time points a given grounded event predicate. As a result, the time domain for ELE is $\mathbb{Q}^+$, and event stream ϵ can de defined as $\epsilon : Event \rightarrow \mathcal{P}(\mathbb{Q})$. (T2). It supports the static operations of $\wedge$, $\vee$ and $\neg$ via constructs `AND`, `OR` and `NOT` (T3). For its temporal relations, it relies on Allen's interval algebra (see Allen [4]). Allen's interval algebra uses 13 temporal relations, e.g. `SEQ`, `DURING`, `FINISHES` and others, to describe the temporal ordering and dependency of events specifically. ELE is stated to support all of the temporal relations of Allen's interval algebra (T4). Temporal patterns akin to the temporal relations considered in T4 can be constructed using constructins with Allen's interval relations, e.g. $A\mathcal{U}_{[5,10]}B$ aligns with expression A `DURING` (0 `SEQ` (B `DURING` (0 `SEQ` 5))).

ELE itself does not capture uncertainty of data or out-of-order problems with event streams. Outside of the specification of the ELE formalism, Anicic [7] considers extensions of the ETALIS system which addresses out-of-order arrival algorithmically (T5).

2.7.3.6 EP-SPARQL

The order of events is of substantial importance for CEP applications, as complex events are oftentimes defined by a specific sequence of (basic) events. Event Processing SPARQL (EP-SPARQL) [9] is a SPARQL-based language, grounded in the operational semantics of ELE [7], (T1) which goes beyond just the `SEQ` operator of CT-SPARQL. In line with the SPARQL Protocol and RDF Query Language (SPARQL)-based formalisms above, EP-SPARQL employs RDF streams $S = \big(\langle : s_i : p_i : o_i \rangle, t_i\big)$ and supports time- and tuple-based windows (T2). It furthermore inherits its static expressivity from pure SPARQL, i.e. $\wedge$ via juxtaposition and $\vee$ via `UNION`, but no `NOT` construct. Anicic et al. [9] notes negation can nonetheless be expressed by combination of the `OPTIONAL` and `FILTER` clauses (T3). In terms of temporal expressivity, EP-SPARQL uses four binary temporal operators: `SEQ`, `EQUALS`,

`OPTIONALSEQ` and `OPTIONALEQUALS`. Additionally, new functions `getDURATION()`, `getSTARTTIME()` and `getENDTIME()` allow to specify conditions on duration, start time and end time of a specified event pattern (T4). Further research is required to assess if and how the temporal operators can be used to express temporal patterns such as $\boxplus_I$, $\diamondsuit_I$, $\mathcal{S}_I$ and others.

With the above, EP-SPARQL is able to extract data from RDF streams based on temporal pattern matching, inducing a form of reasoning about time (T6). The data itself is implicitly assumed to be correct and to be arriving in-order, hence no methods for data uncertainty are integrated in EP-SPARQL (T5)

2.7.3.7 Continuous Query Evaluation over Linked Streams (CQELS)

C-SPARQL and its derivatives above are engineered towards querying of streams, but provide no integrated data processing capabilities [115]. As a result, the processing engine in charge of the windowing and the query engine in charge of answering queries are decoupled. To avoid the overhead involved in the communication between the two separate engines, the SPARQL-based framework (T1) of CQELS [115] was proposed to serve as a unified theoretical framework for modelling, querying and processing RDF Streams and (static) data. Similar to C-SPARQL, an RDF stream is represented as a set of timestamped RDF triples over a time domain of $\mathbb{N}$. A window is represented by a function $\omega : \mathbb{N} \rightarrow 2^{\mathbb{N}}$ mapping a time point to a set of time points. CQELS is equipped to handle both time-based and triple-based window operators (T2). It supports conjunction $\wedge$ and disjunction $\vee$ via juxtaposition and UNION ($\cup$) respectively (T3). Aside from selection using window operators, CQELS does not support temporal operators (T4). As a result, CQELS only performs reasoning over time (T6). Phuoc [115] does not address challenges of data uncertainty or possible out-of-order arrival of data (T5).

2.7.3.8 RSP-QL and RSP-QL*

Dell'Aglio et al. [65] proposes RSP-QL as overarching model for multiple RSP systems, including the aforementioned C-SPARQL and CQELS. The motivation behind RSP-QL's development is to address the limitations and inconsistencies between different RSP models. With its role as an overarching abstract model for other RSP systems, RSP-QL is a formalism in the most abstract sense, as it is not a language itself, but rather an abstraction model that can be shaped into multiple different languages. Simultaneously, it can be seen as significant step towards standardisation of RSP systems.

RSP-QL extends the formalism of SPARQL (T1). It queries RDF streams $S = \big(\langle : s_i : p_i : o_i \rangle, t_i\big)$ over time domain $T = \mathbb{N}$. Windows are modeled using a general mapping from the set of streams $\mathcal{S}$ to the powerset of RDF triples, $\mathcal{P}(\texttt{RDF})$, i.e. $w : \mathcal{S} \rightarrow \mathcal{P}(\texttt{RDF})$, with further specification for both time- and tuple-based windows

(T2). For static expressivity, RSP-QL inherits the conjunction $\wedge$ and disjunction $\vee$ from SPARQL (T3). It does not provide means to express temporal relations, focussing only on retrieval of the temporal data (T4). As a result, RSP-QL is oriented to enable reasoning over time, but not about time (T6).

RSP-QL does not provide built-in methods for handling uncertainty in data (T5). Following this observation Keskisärkkä et al. [85] introduces the extension RSP-QL* with statement-level annotations for RDF statements. These annotations can serve a variety of purposes, most notably as measure of uncertainty or fuzziness. To realise these annotations, one employs nesting of triple patterns as seen in RDF* and SPARQL* [81].

2.7.4 ASP-based Formalisms

ASP-based formalisms excel in SR scenarios that require non-monotonic reasoning and dynamic adaptation, as they naturally support rule-based modelling where new information can override previous assumptions. This makes them particularly effective for handling evolving knowledge and conflicting constraints in streaming environments.

2.7.4.1 Continious ASP (C-ASP)

Pham, Ali, and Mileo [114] considers several limitation of C-SPARQL and CQELS with regards to SR, most notably their limited ability to handle non-determinism and 'default' values, i.e. values that could be overwritten later. To resolve these shortcomings, they propose the ASP-based reasoning system (T1), C-ASP. C-ASP takes RDF Streams $(\langle : s : p : o \rangle, t)$ for input. The exact time domain is not specified by Pham, Ali, and Mileo [114]. Using time- and tuple-based windows, triples are extracted. On the selected triples, C-ASP applies a set of ASP rules via the ASP reasoner Clingo and subsequently outputs a new RDF stream.

Within the ASP rules C-ASP follows the ASP-core2 language standard for ASP rules [46]. As a result, C-ASP has static expressivity of $\wedge$ and $\neg$ within its rules. In this case, recursion can not be applied to obtain $\vee$. The $\rightarrow$ is captured by the concept of a rule itself (T3). Due to its reliance on non-temporal ASP rules (T4), a triple's timestamp can not be taken into account in the entailment, thus C-ASP can not reason about time, only over time (T6).

As considered above, the ASP approach to non-deterministic data one form of data uncertainty, i.e. where conclusions are not 'final' and can be rescinded by future information. Pham, Ali, and Mileo [114] illustrates this approach to tackle incomplete information problems (T5). Complexity results of reasoning using C-ASP are not considered in Pham, Ali, and Mileo [114] (T7).

2.7.4.2 Logic-based framework for Analytic Reasoning over Streams (LARS) and its extensions

C-ASP has demonstrated the ability to process data at high frequencies, which suggests scalability. However, as stressed by Della Valle et al. [67], a theoretical basis for SR needs to find a balance between expressivity and scalability. With the aim of providing formal semantics for SR, LARS, was introduced by Beck, Dao-Tran, and Eiter [27], rooted in ASP (T1).

Streams are modelled by means of a function $v : T \subseteq \mathbb{N} \rightarrow \mathcal{P}(\mathcal{G})$, which projects a time $t \in T$ onto a set of atoms of the form $p(c_1, \ldots, c_n)$, with p a predicate and c_i constants. The time domain T is fixed to be $\mathbb{N}$. Furthermore, LARS introduces an abstract definition of a window on a stream as a function that projects a stream S and time point $t \in \mathbb{N}$ to a substream $S' \subseteq S$. Beck, Dao-Tran, and Eiter [27] further stipulates time-based, tuple-based and partition-based windows. The latter cuts several snippets out of a given streams and pastes them together into a new stream (T2).

Using formulae, LARS is able to denote and assert meaningful statements about the data within a stream. With atoms $p(x_1, x_n)$, with p a predicate and x_i a constant or variable, LARS formulae are inductively defined using the static operators $\neg$, $\wedge$, $\vee$ and $\rightarrow$ (T3) and the temporal operators $\Box$, $\Diamond$ and $@_t$. LARS employs absolute semantics for the temporal operators, i.e. $\Box$ and $\Diamond$ express "always" and "once" considered over the whole timeline, independent of the time point at which the formula is evaluated (T4). Additionally, LARS provides two window operators to be used in the formulae, denoted in Beck, Dao-Tran, and Eiter [27] by $\boxplus^w$ and $\triangleright$, integrating window operators into the formulae to create a substream of an originally considered stream. With the ability to express a wide range of temporal patterns, LARS is equipped to perform reasoning about time (T6), but does not account for data uncertainty (T5).

In terms of complexity, Beck, Dao-Tran, and Eiter [27] discusses both model checking and model satisfiability. First for LARS formulae, the monotone core of the LARS language, Beck, Dao-Tran, and Eiter [27] notes both model checking and satisfiability are PSPACE-complete in case of grounded formulae. Model checking and satisfiability diverge in case of non-grounded formulae: model checking remains PSPACE-complete, whereas satisfiability increases to NExpTime-completeness. Regarding the LARS programs for non-monotonic reasoning, grounded programs likewise have a model-checking and satisfiability complexity of PSPACE-complete, as well as model checking for non-grounded LARS programs. Satisfiability increases to $\texttt{NExpTime}^{\text{NP}}$ for non-grounded programs. With the above results, satisfiability and model checking for both LARS formulae and LARS programs, regardless of (non-)grounded status, are decidable. Via restrictions imposed upon the window operators, the complexity can be reduced further. For details we refer to Beck, Dao-Tran, and Eiter [27]. Each of these complexity results assume the necessary window evaluation is executable in polynomial time (T7).

LARS imports the LTL operators $\Box$ and $\Diamond$. However, in combination with the LARS window operator $\boxplus^w$, temporal relations akin to the (more granular) MTL operators $\boxplus_I$, $\boxminus_I$, $\diamondplus_I$, $\diamondminus_I$ can be deduced. This is shown in Funk [77], who subsequently proposed a fragment of MTL called 'Metric LARS'. Metric LARS captures the MTL formulae that can be mapped to LARS formulae. This mapping shows LARS also has access to the MTL operators, with relative semantics, in addition to the absolute temporal operators $\Box$ and $\Diamond$ (T4). Furthermore, complexity bounds can be derived via MTL. From results presented in Alur and Henzinger [5], it follows that satisfiability checking of Metric LARS formulae is decidable and `ExpTime`-complete (T7).

Intent on use cases with uncertain and probabilistic data, Eiter and Kiesel [72] imports ideas from Weighted Logic by expanding the classical Boolean evaluation of LARS formulae to an evaluation over a semiring[1] $\mathcal{R}$, referred to as Weighted LARS (wLARS). By extending the truth values of a formula from a Boolean $\{0, 1\}$ to a (larger) semiring, wLARS tailors to several quantitative challenges, such as probabilistic and preferential reasoning, as well as reasoning under weak constraints (T5).

As an extension of LARS, the `PSPACE` -completeness of LARS provides a lower bound of `PSPACE`-hardness for evaluating wLARS expressions, as any choice of semiring gives raise to an evaluation problem at least as complex as the Boolean case. Via a restriction of wLARS to 'efficiently encodeable' semirings, more accurate complexity bounds can be obtained. For details on exact decidability and complexity conditions, we refer to Eiter and Kiesel [72] (T7).

2.7.4.3 Probabilistic ASP (PrASP)

Moving away from LARS, a different approach to ASP-based (T1) probabilistic reasoning is given by Nickles and Mileo [111] in the form of PrASP, which allows for non-monotonic probabilistic reasoning. Here, formulae (FOL formulae or AnsProlog rules) are annotated with point-valued probabilities (T5). PrASP works in conjunction with CQELS and performs reasoning over RDF streams, represented via mapping $S = (\langle : s_i, : p_i : o_i \rangle, t_i)$ (T2). As PrASP uses FOL syntax, the static expressivity is given through operators $\neg, \wedge, \vee$ and $\rightarrow$ (T3), and no temporal operators are included (T4). As a result, PrASP is suited for probabilistic reasoning over data streams, but not about time (T6). Among the publications considered within this survey, none cover complexity results for PrASP (T7).

2.7.5 Miscellaneous Formalisms

The subsections above collect several formalism which start from a common basis, i.e. DL, temporal logic, query languages and ASP. There is, however, a significant portion of SR formalisms which cannot be allocated to one of such categories or warrant to

[1] A set with addition $+$ and multiplication $\times$, where both $+$ and $\times$ are associative and have a neutral element, $+$ is commutative and $\times$ is distributive over $+$.

be seen as a category of its own. The lack of a category does not imply the formalism in question is less significant to the field of SR.

2.7.5.1 Temporal Datalog

DatalogMTL has not been the only endeavour which extends Datalog for SR. Temporal Datalog [54] extends the syntax of (negation-free) Datalog (T1) with time constants and time variables, in addition to the 'object' constants and variables. Cruz-Filipe, Nunes, and Gaspar [54] stipulates the ability of Temporal Datalog to formulate hypothetical answers, i.e. answers based on currently available data that can be overturned by future data, as a means of resolving issues created by out-of-order arrival (T5). Streams are represented as a dataset of atoms $P(c_1, \ldots, c_n)$, where c_i are constants of which at most one (say c_n) is a temporal constant that serves as a timestamp, taken from $\mathbb{N}$. The set of atoms, and thereby the stream, can be infinite in nature. Window operators are not defined for Temporal Datalog (T2). Temporal Datalog relies on Datalog rules of the form $\alpha \leftarrow \alpha_1 \wedge \ldots \alpha_n$ where α, α_i are expressions $P(x_1, \ldots, x_n)$ with x_i constants or variables. As a result, Temporal Datalog only considers static operators $\wedge$ and $\rightarrow$ (T3). However, Cruz-Filipe, Nunes, and Gaspar [54] propose an extension which adds negation $\neg$ to the rule body. By recursion, disjunction $\vee$ can also be expressed in this extension. Temporal Datalog does not accommodate temporal operators directly (T4). However, Temporal Datalog does nonetheless provide a means for reasoning about time (T6), via the use of temporal constants and variables. Cruz-Filipe, Nunes, and Gaspar [54] provides no complexity results for Temporal Datalog (T7).

2.7.5.2 Event Calculus for Run-Time reasoning (RTEC)

In order to perform event recognition at run-time, Artikis, Sergot, and Paliouras [13] introduces RTEC. RTEC is a dialect of Event Calculus (T1), a formalisation by Kowalski and Sergot [91] based the Horn fragment of FOL. A stream is represented as a sequence of events over integer timeline $\mathbb{Z}$, where the elements are (simple) events represented by predicate expressions. A windowing mechanism is applied to these streams, with only time-based windows of uniform length WM (T2). Inference is done in a rule-based manner where rule bodies are formed by conjunctive predicate expressions. As such, RTEC only supports conjunction $\wedge$. Opposite predicates, e.g. $\neq$ as opposite of the predicate $=$, may serve as negation is specific cases, but no general negation operator is available in RTEC (T3). The interval predicates of RTEC follow the expressivity of Event Calculus. In terms of our temporal expressivity dimension T4, near all temporal relations can be modelled. `holdsFor` can be used to model $\bigcirc$ or $\bullet$, the operators that refer to the next and previous time point respectively. Likewise, a combination of `holdsAt` and `terminatedAt` can model $\mathcal{S}$ or $\mathcal{U}$ relations

(T4). Oriented towards CEP, RTEC is capable of reasoning about time (T6). There is no accommodation for data uncertainty handling considered (T5).

Artikis, Sergot, and Paliouras [13] does not report on the complexity of RTEC in terms of fact entailment or satisfiability (T7). The focus of its complexity results lie in the interval manipulation introduced by RTEC to improve the overall reasoning efficiency. For details, we refer to Artikis, Sergot, and Paliouras [13].

RTEC has been extended by future work. In the later works of Mantenoglou, Pitsikalis, and Artikis [103] and Mantenoglou and Artikis [102], aimed at resolving challenges regarding cyclic temporal dependencies of events. Hpwever, as these works focus on the algorithms rather than the underlying formalism, the expressivity is not impacted directly.

2.8 Survey Results: Usage Dimension

A formalism is in the first place a theoretical description. Studying this theoretical side provides valuable insight into the properties of the formalism, but SR is by nature an application-oriented field. Putting the SR formalisms into practice is where a formalism's added value can truly be put to the test. Testing every SR formalisms considered in this study goes beyond the scope of this literature study. Instead, we resort to three metrics as indication of a formalism's applicability: the amount of citations a given formalism has generated (U1), the amount of prototype implementations of the formalism (U2) and the amount of application-based papers that build upon the formalism (U3). The latter two are distilled from the "secondary study", where the full set of considered papers is revisited. For further details, we refer to subsection 2.3.2. The results across these dimensions are summarized in Table 2.14 and discussed in detail in the following subsections.

2.8.1 DL-based Formalisms

The category of DL formalisms has comparatively fewer citations compared to the other formalism categories (U1). Lécué [94], $\mathcal{EL}++$ ontology streams accumulated a total of 18 citations since 2012. The DL Lite TQL data streams proposed by Klarman and Meyer [89] have 8 citations since 2013. The formalism $\text{LTL}_{\mathcal{EL}}^{\mathbf{X},\mathbf{G}}$ has only been cited once thus far.

Among the papers considered in our survey, none introduce application-independent software prototypes (U2).

Several real-life use cases using a DL formalism were identified (U3), each of which employ the description logic $\mathcal{EL}^{++}$ with ontology streams. Via a multitude of traffic-related data sources on traffic in the city of Dublin, Ireland, Lécué et al. [96] diagnose

Formalism	Year of publication	Citations	Software Prototypes
DL-Lite TQL and fragments	2013	8	
$\text{LTL}_{\mathcal{EL}}^{\mathbf{X},\mathbf{G}}$	2021	1	-
DL $\mathcal{EL}^{++}$ + ontology stream	2012	20	-
DatalogMTL, DatalogMTL^{FP}	2019	60	[139]
DatalogMTL$^{\neg}$	2021	33	-
DatalogMTL$^{\exists}$	2023	0	-
MSTL	2016	24	-
P-MTL	2016	33	-
PMLTL	2024	1	-
TEL	2007	57	[40]
TEL + past operators	2017	6	-
MEL	2022	9	-
LTPAL	2020	1	-
ASTL	2022	8	-
ProbSTL	2020	37	-
EvTL	2022	0	[100]
RobTL	2022	4	-
SDRL	2018	1	-
TESLA	2010	329	[105]
TAL	2013	59	[62, 69]

Table 2.11: Summary of citations (U1) and software prototypes (U2) of SR formalisms. — Part A

Formalism	Year of publication	Citations	Software Prototypes
C-SPARQL	2009	372	[21]
STARQL	2014	24	[86]
SPARQL_stream	2013	15	[42]
CT-SPARQL	2018	1	-
ELE	2012	244	[6, 119, 148]
EP-SPARQL	2011	547	[6, 68]
CQELS	2011	587	[115]
RSP-QL	2014	164	[128]
RSP-QL*	2019	17	[85]
C-ASP Language	2019	14	[114]
LARS	2015	242	[25, 26, 28]
wLARS	2020	26	-
Metric LARS	2021	0	-
PrASP	2014	20	-
Temporal Datalog	2020	7	-
RTEC	2012	59	[12]

Table 2.12: Summary of citations (U1) and software prototypes (U2) of SR formalisms. — Part B

Table 2.13: Summary of dimension U3, listing the applications per formalism, grouped per application domain. Formalisms without application-based papers in the considered literature are omitted from the table. — Part A

Formalism	Urban Traffic	Sensor Monitoring	Smart Grid	Industrial Production	Social Media	Cloud Computing
DL $\mathcal{EL}^{++}$ ontology streams	[53, 95, 96]	[53]	0	0	0	0
ASTL	0	[145]	0	0	0	0
SDRL	0	[87]	0	0	0	0
TAL	0	[71]	0	0	0	0
C-SPARQL	0	[84, 90]	0	[76]	[17, 97]	0
STARQL	0	[86, 124]	0	0	0	0
$\text{SPARQL}_{\text{stream}}$	[43]	[43]	0	0	0	0
ELE	0	0	[141]	[119]	0	0
EP-SPARQL	0	0	[68]	[76]	0	0
CQELS	0	0	0	0	0	0
RSP-QL	0	[131]	0	[51, 52]	0	0
LARS	0	0	0	0	0	[30]
RTEC	[13]	0	0	0	0	0

traffic congestion, Lécué and Pan [95] perform knowledge discovery on bus delays and Chen et al. [53] counteract concept drift within bus delay predictions. Within the same work, Chen et al. [53] employ the same principles for concept drift to a different application, i.e. air quality predictions in the city of Beijing, China based on factors such as humidity, wind speed and direction, and air pollutants.

2.8.2 Temporal Logic Formalisms

In terms of academic citations (U1), we observe the following results for temporal logic-based formalisms; The work of Wałęga, Kaminski, and Grau [137] on DatalogMTL and the fragments DatalogMITL and DatalogMTL^{FP}, has accumulated 60 citations since 2019. Fragments $\text{DatalogMTL}^{\neg}$ [55] and $\text{DatalogMTL}^{\exists}$ [112] have received 33 and 0 citations since 2021 and 2023 respectively. MSTL [98] and P-MTL [126] received 22 and 30 citations respectively. For the EL-based formalisms of TEL and MEL, we observe a citation count of 57 for TEL [38]. The subsequent works on extending TEL with past operators, i.e. Aguado et al. [2] and with metric operators, i.e. MEL [39], have gathered 6 and 9 citations respectively.

Table 2.14: Summary of dimension U3, listing the applications per formalism, grouped per application domain. Formalisms without application-based papers in the considered literature are omitted from the table. — Part B

Formalism	Human Activity Tracking	Maritime Traffic	Robotics
DL $\mathcal{EL}^{++}$ ontology streams		0	0
ASTL	0	0	0
SDRL	0	0	0
TAL		0	0
C-SPARQL	[16, 35, 57, 59, 61, 82, 110]	[64]	[108]
STARQL	0	0	0
$\text{SPARQL}_{\text{stream}}$	0	0	0
ELE	0	0	[146, 148]
EP-SPARQL	0	0	0
CQELS	[3]	0	0
RSP-QL	[58]	0	0
LARS	0	0	0
RTEC	0	[116, 121, 129]	0

ProbSTL [125] has been cited 37 times since 2010. With a total of 329 citations, the work of Cugola and Margara [56] on TESLA has gathered significant attention in the research community since publication in 2010. Furthermore, the work of Doherty, Heintz, and Kvarnström on TAL has accumulated 59 citations since 2013. The remaining temporal logic-based formalisms fall within the single-digit citation range. For a detailed overview, we refer to Table 2.12.

Several formalisms have seen software implementations (U2). TAL is supported by two notable prototypes: TALPlanner, a temporal planner, and DyKnow, a middleware for SR from multiple sources [69]. DatalogMTL has an implementation in the form of MeTeoR [139]. EvTL is implemented via SPEAR for ULTRON[2], which extends the SPEAR tool for model checking [100]. TEL has been realised in the temporal ASP solver Telingo[3] [40]. Finally, DOTR combines the RDFox store with the T-Rex engine for temporal reasoning based on TESLA [105].

[2] https://github.com/vale-castiglioni/Ultron
[3] https://github.com/potassco/telingo

The secondary study identified three distinct use cases (U3) employing temporal logic-based formalisms. One of these focuses on monitoring the sensors within Unmanned Aerial Vehicle (UAV) control systems, where HDRC3 [71] uses TAL to ensure correct semantics in planning components such as TALPlanner and TF-POP, modelling constraints as TAL statements to eliminate unsafe or infeasible plans. SDRL is applied in two industrial IoT scenarios within the work of Kharlamov et al. [87], monitoring the sensors of trains and of turbines, encoding diagnostic programs more efficiently than previous rule languages. The work of Zhao et al. [145] illustrates the ability of ASTL to combine qualitative and quantitative constraints via a case study within the domain of sensor monitoring.

2.8.3 SR Query Languages

When it comes to academic popularity, multiple query languages rank highly in terms of citations (U1). Most notably are CQELS, EP-SPARQL, published in 2011, and ELE, published in 2012, with respectively 558, 547 and 244 citations. The work of Barbieri et al. [21] on C-SPARQL has only been cited 372 times since 2009. However, we note that several different publications in the years 2009-2010 may serve as potential source for citing C-SPARQL. In addition to the publication Barbieri et al. [21] (372 citations), two conference publications [19, 23] (251 and 197 citations) and a journal publication [20] (381 citations) may each serve as a citation source for C-SPARQL. RSP-QL's introductory publication [65] has received 164 citations since 2014 and its extension RSP-QL* [85] has accumulated 17 citations since 2019. The formalisms of $\text{SPARQL}_{\text{stream}}$, STARQL and CT-SPARQL have been cited considerably less, with citation counts of 15, 24 and 1 since 2013, 2014 and 2018.

Several query-based formalisms have been introduced alongside their software implementations (U2). C-SPARQL was introduced together with its engine [21], and the term "C-SPARQL" is used interchangeably for both the language and the engine throughout the literature. Similarly, CQELS was introduced with its native engine [115]. EP-SPARQL and ELE are both implemented in the ETALIS system [6], which serves as their primary implementation. Additional engines building on ETALIS include Retalis [148], μCEP [119] and μReasoner [68], . Retalis integrates ELE with Synchronized Logical Reasoning Language (SLR) [147] and is available on GitHub[4]. μCEP explicitly refers *the ETALIS language*, which most likely implies μCEP is a software implementation for ELE. i.e. Anicic et al. We therefore consider it a software implementation for ELE. Diwold et al. notes EP-SPARQL as language within ETALIS. As a result, we consider μReasoner a software implementation for EP-SPARQL. STARQL has been implemented in the EXASTREAM system [86], which is part of the OptiqueVQS framework [123]. $\text{SPARQL}_{\text{stream}}$ relies on the

[4] https://github.com/procrob/Retalis

morph-stream implementation[5] [42]. As an abstract reference model, RSP-QL does not produce a materialised engine itself but provides standardisation for RSP engines. To this end, the RSP4J library [128] has been developed. RSP-QL* has a software prototype in the RSPQLStarEngine, available on GitHub[6] [85]. For CT-SPARQL, Yang only notes a design architecture for an implementation AIMRS, without reference to an existing implementation.

Within our secondary study, a total of 30 use case studies were found that build upon query-based formalisms (U3). These use cases span several different formalisms, most notably C-SPARQL, CQELS, EP-SPARQL, STARQL and $\text{SPARQL}_{\text{Stream}}$. The majority of 14 use cases employ C-SPARQL, followed by 4 for ELE, 4 for RSP-QL, 3 for $\text{SPARQL}_{\text{stream}}$, 2 for EP-SPARQL, 2 for STARQL and 1 for CQELS.

With regards to C-SPARQL, a significant portion of its use cases focuses on human activity tracking; we observe several medical and E-health applications [35, 59], sport and athleticism applications [57, 61, 110] and more generally the monitoring of person location and activity [16, 82]. Furthermore, our secondary study contained two applications in the social media domain [17, 97], two in sensor monitoring [84, 90] and one in industrial production [76], maritime traffic monitoring [64] and IoT [108] respectively. CQELS has been used by Ali et al. [3] as a basis for an IoT-enabled meeting management application. Several use cases employ the ETALIS system, and thereby either ELE or EP-SPARQL, for CEP. Xu et al. [141] considers smart grid applications, more specifically for efficient energy consumption in office buildings, which uses ELE as their underlying formalism. Likewise, use cases in robotics [146] and industrial production [76] show the applicability of ELE and EP-SPARQL respectively to robotics and industrial automation. Other applications of ELE include the robotics use case considered in Ziafati et al. [148] and the industrial production setting considered in Ren, Anicic, and Runkler [119]. STARQL-based systems have been applied to monitoring large networks of appliances, illustrated by Siemens gas and steam turbines [86, 88, 124]. In Calbimonte [43], $\text{SPARQL}_{\text{stream}}$ has been used in sensor monitoring, such as for flood detection networks and environmental hazard forecasting platforms, and urban traffic monitoring, in the form of a bike sharing platform. Lastly, several applications allow for any RSP-QL-compliant language to be used within their system. These include the DIVIDE system for E-Health monitoring via IoT devices [58, 60] and Streaming MASSIF, used for sensor monitoring to gauge COVID-19 risk in office settings [131]. Additionally, the human activity tracking application considered in Calvaresi and Calbimonte [48] is RSP-QL-compliant. The works Charbonnier et al. [52] and Charbonnier et al. [51] employ a system build upon RSP4J for quality assurance in industrial production.

[5] https://github.com/jpcik/morph-streams
[6] https://github.com/keski/RSPQLStarEngine

2.8.4 ASP-based Formalisms

When considering the academic citations (U1), we observe for the ASP-based formalisms that PrASP has received 20 citations since 2014,C-ASP 14 and LARS 242. In addition, wLARS and Metric LARS account for 26 and 0 citations. C-ASP, wLARS and Metric LARS are thereby relatively recent, dating between 2019 and 2021. The work of Beck et al. [29] on LARS was published in 2015.

The most notable ASP-based formalism in terms of developed software prototype appears to be LARS (U2). Within our secondary study, three software implementations of LARS were found; Ticker[7] [28], Laser[8] [26] and Laser2[9] [25]. The chronologically first implementation of Ticker employs the ASP solver Clingo to repeatedly solve a static version of the LARS program at hand. Laser is a subsequent implementation (in Python), optimised to avoid duplicate computations. As a result, Bazoobandi, Beck, and Urbani [26] reports a significant difference in run-time between Ticker and Laser. Laser2 further optimises Laser by identifying rules that can, over a finite time interval, not be satisfied. Bazoobandi et al. [25] reports an improved run-time over Laser. The C-ASP language is implemented by Pham, Ali, and Mileo [114] into a C-ASP engine, which uses an internal Clingo ASP solver as seen in LARS' Ticker. Nickles and Mileo [111] states the development of an (early-stage) prototype of PrASP, but does not provide details on the public availability of said prototype.

With regards to applications (U3) of the ASP-based formalisms considered in Section 2.7.4, we note that PrASP was introduced by Nickles and Mileo [111] in 2015, LARS in 2017 and C-ASP in 2019. As a result, each of the covered ASP-based formalisms have been introduced recently relative to the other formalisms. The DL- and query-based formalisms considered in the previous two subsection were introduced between the years 2009 and 2014. The community has subsequently had more time to develop (and report on) use case cases based on these formalisms than on the aforementioned ASP-based formalisms. This is subsequently reflected in the amount of use cases found in our secondary study, as only one ASP-based use case was found: Beck et al. [30] implements LARS for intelligent administration of content-centric networks by leveraging LARS' expressivity in caching strategies.

2.8.5 Miscellaneous Formalisms

Regarding academic citations (U1), the formalisms of Temporal Datalog, introduced in 2020, and RTEC, introduced in 2012, have gathered 7 and 59 citations respectively.

[7] https://github.com/hbeck/ticker
[8] https://github.com/karmaresearch/Laser
[9] https://bitbucket.org/hrbazoo/laser/src/master/

Artikis, Sergot, and Paliouras [13] notes RTEC has implemented, available on GitHub[10] [12]. No software implementations for Temporal Datalog were observed in the considered literature.

The software prototype of RTEC is subsequently put to use in both the domain urban traffic monitoring and in the domain of maritime traffic monitoring. The application in the former domain stems from the introductory paper of Artikis, Sergot, and Paliouras [13] as a primary proof-of-concept on the traffic of Helsinki. RTEC has thereafter been implemented in several scenarios, aimed at recognizing the activity of one or more vessels. RTEC is used to model the event patterns used to monitor the various activities a ship can partake in [116, 121, 129].

2.9 Discussion

Within this work, we set out to provide a systematic overview of the theoretical underpinning of SR. We observe that the propositions for SR formalisms have been numerous, but the highly diverse demands and explored directions have left the academic field in an unstructured state. Based on the directions indicated by the literature, we have outlined a series of expectations upon which SR formalisms, both old and new, can be assessed. The existing SR formalisms were subsequently evaluated across these different dimensions in order to obtain a global overview of the research results obtained thus far. This overview helps to highlight the capabilities of existing technologies as well as to identify research gaps.

The resulting overview of prerequisites for a SR formalism as presented in Section 2.6 aims to serve as a guideline for both past and future formalisation efforts. The structure we have presented here serves as a first step towards a 'comparative framework' of various SR formalisms. By means of the proposed dimensions, one is able to more strongly argument a choice for a particular formalism, by formally comparing different options for a given use case. Furthermore, it allows to position a newly introduced formalism within the current state-of-the-art.

Throughout the surveying process, several elements naturally emerged as common traits shared by all or most SR formalisms. First, it stands out that not every single formalism has the same objective when it comes to SR. the term 'Stream Reasoning' can be interpreted in different ways by different formalisms. Some formalisms are only aimed at intelligently extracting and delivering data from streams, whereas others aim to extensively process and derive complex conclusions from such data. This large diversity is highly determinant for a formalism's capabilities. As such we classify it via dimension T6. Although not unique to SR, most formalisms have their roots in pre-existent theories, which can range from mathematical theories (e.g. set theory, FOL) to existing reasoning formalisms. We capture the roots of the various SR formalisms

[10] https://github.com/aartikis/RTEC

in the dimension T1, labelled "theoretical basis". Furthermore, we consider the representation of a stream. All formalisms have some way of representing streams, either explicitly or implicitly, as the subject of their operations. Given the high variability observed and practical implications of different choices, this shaped into our dimension T2 "stream representation". Another element is the ability to relate different pieces of information to one another. In order to deduce meaningful insights from data, one cannot look at every piece of data in a vacuum, but has to consider how the different information can be connected to one another. The wide variety to relate different pieces of information, both with and without taking account the temporal aspect, has lead to dimensions T3 and T4, "static relations" and "temporal relations" respectively. In addition to these expressive aspects, many SR settings must contend with imperfect, incomplete or disorderly data, which motivates dimension T5. This dimension captures whether a formalism can accommodate uncertainty, probabilistic information or out-of-order arrival in the stream, an increasingly important consideration given the volatility and heterogeneity of real-world data sources.

Coming back to our research questions considered in Section 2.3, we can thus answer **Q1**: "What are the common constructs and expression of a formalism for SR?". Based on the different formalisms introduced over the past 15 years, we have identified five elements that are common to most SR formalisms; a specific objective (T6) a theoretical basis (T1), a representation of streams (T2) and methods of expressing relations between pieces of information, either time-agnostic (T3) or time-dependent (T4).

In the second research question **Q2**, "What are sensible criteria to evaluate a logical formalism, in particular a formalism's suitability for SR?", we set out identify the criteria upon which a SR formalism can be evaluated. In this search for tangible comparative metrics, the dimensions that best fit this description are dimensions T7, the complexity of a formalism, and U1, U2 and U3, the dimensions that describe the usage and popularity of a formalism. Additionally, dimensions T3 en T4 also lend themselves to this evaluation; T3 and T4 chart the expressivity of a formalism, which allows to compare different formalisms in terms of their capabilities.

Recall research questions **Q3** and **Q4**, "Given the criteria from the previous question, how do recently developed SR formalisms meet these criteria and how do these formalisms compare to one another?" and "Among the formalisms that have been developed since *'It's a Streaming World'* [67], which have been adopted in real-life applications ?" (Q4). There we have asked ourselves how the formalisms introduced in the past 15 years score on the various dimensions, and which ones have been translated to real-life applications. A detailed breakdown for each SR formalism is given in the sections 2.7 and 2.8 for the theoretical and usage dimensions respectively. For each of the dimensions, the results can be summarized as follows.

T1. Dissection of the existing state-of-the-art of SR formalisms according to dimension T1, the theoretical basis of the formalism, shows a relatively diverse spectrum, with several clusters as shown in Table 2.4. The field of SR is not dominated by a single mathematical paradigm upon which all formalisms are built. Instead, we observe different clusters such as formalisms based on DL, on ASP, on query languages and other. Each of these clusters stems from different backgrounds and application areas (i.e. ASP from non-monotonic reasoning, DL being tied to Semantic data), but each of the observed areas has grown towards supporting reasoning over streams in their own manner. It is this diversity that allows the field of SR to be applied to many different application areas.

T2. In terms of dimension T2, stream representation, we considered the different approaches to modelling streams in the context of SR, as summarized in Table 2.8. First, the representation of time itself shows three dominant options in specifying the time domain; (a) a discrete domain, such as $\mathbb{N}$ or $\mathbb{Z}$, (b) dense time domain $\mathbb{Q}$ and (c) the time domain is not specified. In addition, MSTL, TAL and STARQL have very little assumptions on the time domain. This could allow for more exotic time representations, such as branched timelines, but examples thereof have not yet been explored within the literature. Second, on the representation of a stream, many of the covered formalisms rely on the model of an RDF stream as a set of time-stamped RDF triples. This is the case for many of the query languages, such as C-SPARQL, CQELS and ELE, as well as for the answer set based PrASP and C-ASP formalisms. Other formalisms prefer to not annotate each piece of information individually, but group the information per time point. This leads to a chronological sequence of sets of information. Examples of this approach can be found in DL Lite TQL and its fragments, TEL and MEL, Temporal Datalog and $\mathcal{EL}++$ ontology streams. A last common approach consists of describing a data stream as a mapping from a given time point to a set of information that holds at said time point. This is seen in DatalogMTL (and its fragments), LARS (and its fragments and extensions), STARQL and ProbSTL. For most observed formalisms, the concept of a stream in not restricted to finite streams and therefore naturally supports the idea of endlessly continuing data flows, in line with real-world streaming environments. Exceptions exist in formalisms such as PMLTL and LTPAL, which operate over finite timed traces or finite transition systems. Lastly, in terms of windowing, we observe that the query-based formalisms support this practice the most, with time-based windows being the most popular mechanism. In addition, LARS also supports windowing. With LARS, the windowing is incorporated into the syntax itself via a dedicated window operator, incorporating the concept of windows deeper into the reasoning process.

The high diversity in time and stream representation allows one to pick the representation most suitable to an intended use case. Simultaneously, this diversity can

be an obstacle when attempting to communicate between systems of different formalisms, in particular those with different stream representations.

T3, T4. When it comes to expressivity, we can observe from Tables 2.4 and 2.6 that nearly all of the considered formalisms are able to express the basic concepts of conjunction, disjunction and negation, with or without relying on recursion. Among all static relations, it stands out that conjunction $\wedge$ is the most prominent, with every observed formalism supporting it (albeit via recursion in the case of ProbSTL and EvTL). Negation appears to not be as vital as conjunction, with many but not all formalism supporting it. Most notably, the ASP-based formalisms consistently support negation as the only category considered, which can be linked to their disposition towards non-monotonic reasoning. A more prominent disparity can be found in their temporal expressivity. We observe the whole spectrum, ranging from none (e.g. PrASP, $\mathcal{EL}$ + + streams) to roughly all of our specified temporal relations (e.g. MEL, DatalogMTL, EP-SPARQL). For each of the different categories aside from the 'remainder' category, at least one SR formalism meets a fair amount of our predetermined criteria. Among DL-based formalisms, (unfragmented) DL-Lite TQL possesses its own variations of $\mathcal{S}$, $\mathcal{U}$, $\boxplus$, $\boxminus$, $\diamondplus$ and $\diamondminus$. Likewise for query-based formalisms, C-SPARQL, EP-SPARQL and ELE allow for various complex temporal expressions either via direct timestamp specification or via temporal comparatives. When it comes to temporal logic-based formalisms, the best scoring formalisms are TEL and TESLA, along with MEL and DatalogMTL for those that support interval semantics. Within the ASP-based formalisms, LARS and its extensions are the only ones equipped with temporal expressivity. When comparing the best formalisms within each category, we observe that the query-based languages provide the most extensive tools for temporal expressions. None of the query languages does however support interval semantics. DatalogMTL, on the other hand, does. DatalogMTL only compromises on 'next' and 'previous' operators, which by themselves only apply to discrete time domains (or otherwise strictly ordered sets of information). LARS contrasts with DatalogMTL by employing an absolute time semantics, compared to the relative semantics of DatalogMTL; the operators $\Box$ and $\Diamond$ are used instead of $\boxplus$, $\boxminus$, $\diamondplus$ and $\diamondminus$. Similar to dimension T2, the variability in expressivity lends itself well to tailored solutions and closed systems, but can become an obstacle for open-ended systems containing multiple different formalisms. This even extends to formalisms with seemingly identical operators, e.g. DatalogMTL's $\boxplus$, $\boxminus$ and MEL's $\boxplus$, $\boxminus$, due to differences in underlying semantics.

T5. When it comes to handling various types of discrepancy in data streams, we observe in Table 2.10 that there exists only limited support in the current literature. The formalisms that account for data uncertainty are mostly oriented towards probabilistic uncertainty, as is the case for temporal logic formalisms P-MTL, ProbSTL and EvTL.

Three formalisms, being RSP-QL*, wLARS and PrASP, support a versatile approach to tackling uncertainty, as they can be used to express probability, as well as fuzziness or other other uncertainty measures. Among the considered formalisms, only RobTL explicitly incorporates the handling of out-of-order data into the formalism.

T6. Across the surveyed landscape, a clear division can be observed with respect to a formalism sets out to do: (i) stream querying, (ii) reasoning over time, or (iii) reasoning about time. Query languages (e.g., C-SPARQL, STARQL, CQELS, RSP-QL) are engineered primarily for continuous, windowed retrieval and filtering of evolving RDF data, yet carefully crafted queries can sometimes emulate simple temporal inferences; this makes them broadly adoptable but typically limits temporal semantics to what can be encoded as selection conditions over windows. In contrast, logics like DatalogMTL, TEL/MEL, TESLA, TAL, and LARS target inference that is intrinsically temporal; operators such as $\mathcal{S}$, $\mathcal{U}$, $\boxplus$, $\diamondplus$ etc. are part of the language itself, so the "when" becomes a first-class citizen alongside the "what". A third group (e.g., EL++ ontology streams) performs incremental, non-temporal updates to knowledge bases as data arrives—useful for change tracking and explanation—without committing to temporal pattern semantics. From a larger perspective, we observe a degree of sequentiality in the objectives of the various formalisms; stream querying formalisms were introduced relatively early, primarily between 2009 and 2014. This compared to formalisms for reasoning about time, of which the majority was introduced after 2016. An explanation for this contrast can be found in the sequentiality of data processing: before complex reasoning about time can be considered, one must first be able to retrieve, i.e. query, said data. As a result, SR query languages were introduced first, Furthermore, reasoning over time is more common for earlier SR formalisms, whereas later formalisms move towards reasoning about time, with more detailed temporal relations (see T4) as well. As a result, there is a noticeable trend towards more complex reasoning over the past 15 years. In addition, many SR formalisms are motivated by concrete needs within specific application domains rather than from purely theoretical motivation. The broader categories as well as several stand-alone formalisms each reflect this orientation, with examples such as TAL for modelling actions and change, SDRL for industrial diagnostics and LTPAL for interpreting knowledge produced by AI systems.

T7. A synopsis of the complexities of the SR formalisms covered in this work is hindered by sparsity. The amount of SR formalisms for which complexity results were retrieved, either through their introductory paper(s) or subsequent papers, is limited. Across the limited results, we observe that both DatalogMTL, SDRL and LARS are shown to be PSPACE-complete with regards to fact checking, suggesting PSPACE-completeness is not merely characteristic for these three formalisms, but more generally of reasoning about time.

U1–U3. The theoretical side of SR formalisms does not necessarily reflect their added value to real-life applications. From the results of our study into the formalism's applications and academic reception, summarized in Tables 2.14 and 2.12, we observe first that the two metrics of citations and applications exhibit largely the same patterns: formalisms with many applications tend to also have a large amount of citations, and formalisms without observed applications also tend to have a lower citation count. The most notable example thereof is C-SPARQL, with both a high amount of applications (7) as well as citations (372). Nonetheless, we must not ignore several promising SR formalisms who do not (yet) have a noteworthy amount of applications, but whose citation rate indicates significant traction in the SR community. Most notably, the collections of LARS and DatalogMTL formalisms have received a discerning amount of citations in the time since their publication in 2015 and 2018 respectively, which forebodes a growing impact on future SR developments.

Furthermore, the existence and availability of software prototypes does not necessarily translate to a high amount of applications. TAL and C-SPARQL are examples of formalisms with both software prototypes and a high amount of applications, but on the contrary LARS, TEL and TESLA do have software prototypes, but do not boast a significant amount of applications. Reasons for the lack of a one-on-one relation between software availability and observed applications may include, but are not limited to, the narrow scope of our secondary study, the properties of the formalisms themselves or the recency of introduction of a formalism.

Looking at the usage trends across the different categories of SR formalisms, we observe that the category of query-based SR formalisms outranks the other categories in terms of both applications and citations. As such, we conclude that query-based formalisms have thus far been more popular. A possible explanation for this popularity lies in their intended use; the focus on querying data streams makes these formalisms applicable to a wide range of applications. However, the "reasoning" within these formalisms is often-times limited to reasoning by means of query constructions. This in contrast to other, especially logic-based, SR formalisms which employs rule constructs to perform reasoning. As the query-based formalisms are not strictly tailored towards reasoning, but more broadly to any type of stream querying (and processing), the metrics for the usage dimension of observed applications and academic citations is skewed in favour of the query-based formalisms.

Aside from the theoretical properties of SR formalisms, we have studied the adoption of the covered formalisms towards real-life use cases. To this end, we considered the implementations seen in the total set of papers we have taken into account for this survey. This set yielded a large diversity of application domains, illustrating the wide applicability of SR as a field. Between the formalisms themselves, we have observed that some formalisms have seen a significantly higher implementation rate than others, indicating that several formalisms are favoured by the (research) industry over others.

In examining the adoption of the surveyed SR formalisms across real-life applications, we observe that query-based formalisms stand out with a substantially higher number of use cases than other categories. Several factors contribute to this trend. First, many query languages are comparatively mature: C-SPARQL, EP-SPARQL, STARQL, CQELS and SPARQL$_{\text{stream}}$ were introduced between 2009 and 2014, providing nearly a decade or more for the research community to explore implementations and deploy these systems in practical settings. In contrast, a number of more novel formalisms such as LARS, SDRL and DatalogMTL were introduced later, between 2015 and 2018, and therefore have had fewer years in which to be translated to real-world applications. As such, numerical comparisons between formalism adoption must be interpreted in light of these differing timelines. Lastly, we note that the category of query formalisms typically functions at a low level within the applications; the continuous queries defined in these languages often operate close to the source of the streams, such as the direct monitoring of IoT devices, health parameters or small sensors (e.g. [3, 90, 141]). As such, the primary objective of the formalisms in these applications is the retrieval of specific data. The requirements of querying on data streams are possibly less high compared to more high-level reasoning on streaming data that is targeted by other formalisms such as ASP- or temporal logic based SR formalisms.

In addition to overall popularity, we also observe that several formalisms are strongly represented in one or more application domains. ELE has found its way into the robotics and smart grid domains, STARQL's use cases are each focused on monitoring industrial processes and C-SPARQL is a common choice for various forms of human activity tracking. This shows in part that different formalisms tailor to the needs of different fields of application, thus that SR is by no means limited to select cases or areas. Looking at such formalisms individually, it is only natural a formalism sees increasing use in one or more domains; the development of said formalism could have been driven by a need in a specific area, resulting in a evident candidate field for real-life applications. Alternatively, a prominent standalone use case can emphasise a formalism's suitability for a certain application area, inducing subsequent use cases. The presence of such a specialisation towards certain use cases can thus be seen as an asset towards further SR development.

2.10 Conclusion & Future Perspectives

As discussed in Section 2.5, several parts of the SR ecosystem have already been the subject of benchmarking and comparative studies. Most notably, RSP systems have been surveyed with a focus on technical properties such as architecture, operational semantics and supported data formats [34]. Complementary system-level perspectives, such as those of Margara et al. [66] and Dell'Aglio et al. [66], have further clarified requirements and helped bring SR out of isolated case studies and towards broader applicability. Building on these efforts, the dimensions outlined in this work serve as a

first step towards a comparison framework specifically targeted at SR formalisms, enabling an objective analysis at the level of syntax, semantics and reasoning objectives.

The current state of the art shows significant progress in expressivity; however, support for different kinds of uncertainty remains limited. Where uncertainty is addressed, it is often through specialised extensions tailored to particular niches. Extending more general formalisms to accommodate incomplete, noisy or out-of-order data remains an important avenue for future work. At present, wLARS and RSP-QL* stand out as prominent examples of such extensions, and similar directions could be explored for other formalisms.

In recent years, several SR formalisms with strong temporal capabilities have emerged, enabling increasingly complex reasoning. To translate these theoretical advances into practice, the field would benefit from a greater number and variety of real-world deployments. DatalogMTL and LARS are among the strongest formalisms from a theoretical standpoint, making them prime candidates for complex applications. In parallel, it is worth exploring how query-based formalisms can support richer reasoning patterns that go beyond straightforward stream querying. Formalisms that operate over RDF streams are especially well-positioned for interoperability in domains such as IoT, edge processing and decentralised ecosystems. It is up to the SR community to identify the most promising opportunities for demonstration and implementation of these advances.

Despite the high level of abstraction of the above formalisms, there is no single formalism that covers all requirements. Our analysis shows that different formalisms exhibit different strengths and that these strengths are closely tied to their primary objectives. Query-oriented approaches prioritise windowed access to evolving data, which favours scalability and deployment but limits temporal expressivity. Temporally expressive logics enable rich pattern detection and causality, yet they often carry higher computational costs and stricter modelling assumptions. Non-monotonic rule frameworks excel at revising conclusions as streams evolve, but their semantics can be difficult to reconcile with snapshot-based query evaluation. In several cases, objectives can even hinder one another. For example, grafting MTL interval operators and stratified negation from DatalogMTL$^\neg$ onto C-SPARQL would require reasoning across window boundaries and fixpoint strata, which undercuts C-SPARQL's snapshot-over-windows execution model.

In light of these trade-offs, the creation of a single SR formalism that "checks all the boxes" across objectives, expressivity and complexity appears highly unfeasible. A more practical route for building SR systems that support multiple objectives is to construct mappings between state-of-the-art formalisms, so that different components specialise in complementary tasks. For instance, one subsystem can provide robust stream querying and window management, another can perform temporal rule-based inference and a third can handle uncertainty. Crucial to this approach is the correctness of the mappings: translations must preserve meaning across languages

so that temporal relations, static operators and uncertainty annotations align semantically rather than merely syntactically. The dimensional analysis provided in this work (T1–T7) offers a principled basis for designing and validating such mappings, by making explicit which constructs and semantics must be preserved when moving between formalisms and where objective-driven constraints are likely to interact.

The field of SR has undergone a period of substantial growth, yielding a wide range of ideas and tools. While this diversity underscores SR's broad applicability, it can also appear overwhelming to new researchers and external stakeholders. In this light, the usage dimensions offer a practical guide: C-SPARQL, EP-SPARQL and EL++ ontology streams emerge as prominent options with a high degree of adoption, which likely reflects their versatility and level of abstraction across many use cases. A deeper understanding of the theoretical capabilities that underpin such formalisms will aid the development of SR systems that are both expressive and accessible. The systematic review of the state of the art enables us to identify research gaps that warrant further attention. As the field has been largely application-driven, we highlight common, cross-cutting requisites, including temporal expressivity, flexible windowing and the treatment of probabilistic and fuzzy data. Several aspects remain comparatively underexplored, such as delayed or out-of-order arrival and its implications for temporal reasoning. Although delayed data can be mitigated algorithmically, for example through buffering or caching, its theoretical implications can only be fully understood when accompanied by sufficiently formal semantic treatments.

We hope this work will provide a basis for new researchers to participate in the formal aspects of SR and support seasoned researchers in future efforts. By synthesising 15 years of developments in SR formalisms, we aim to provide a stable point of reference and structure for ongoing research. Open challenges remain, but recent advances also show the field's potential. In particular, LARS and DatalogMTL exhibit extensive temporal expressivity. Future work may focus on showcasing these capabilities through impactful real-life applications.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN: 0-201-53771-0.

[2] Felicidad Aguado et al. "Temporal equilibrium logic with past operators". In: *Journal of Applied Non-Classical Logics* 27.3-4 (2017), pp. 161–177.

[3] Muhammad Intizar Ali et al. "Real-time data analytics and event detection for IoT-enabled communication systems". In: *Journal of Web Semantics* 42 (2017), pp. 19–37.

[4] James F Allen. "Maintaining knowledge about temporal intervals". In: *Communications of the ACM* 26.11 (1983), pp. 832–843.

[5] Rajeev Alur and Thomas A Henzinger. "Real-time logics: complexity and expressiveness". In: *Information and Computation* 104.1 (1993), pp. 35–77.

[6] D. Anicic and P. Fodor. *ETALIS*. https://code.google.com/archive/p/etalis/. 2009.

[7] Darko Anicic. "Event Processing and Stream Reasoning with ETALIS". PhD thesis. Karlsruhe, Germany: Karlsruher Institut für Technologie (KIT), 2012.

[8] Darko Anicic et al. "A rule-based language for complex event processing and reasoning". In: *International Conference on Web Reasoning and Rule Systems*. Springer. 2010, pp. 42–57.

[9] Darko Anicic et al. "EP-SPARQL: a unified language for event processing and stream reasoning". In: *Proceedings of the 20th international conference on World wide web* (2011). DOI: 10.1145/1963405.1963495.

[10] Darko Anicic et al. "Stream reasoning and complex event processing in ETALIS". In: *Semantic web* 3.4 (2012), pp. 397–407.

[11] Arvind Arasu, Shivnath Babu, and Jennifer Widom. "The CQL continuous query language: semantic foundations and query execution". In: *The VLDB Journal* 15.2 (2006), pp. 121–142.

[12] Alexander Artikis, Manolis Pitsikalis, and Periklis Mantenoglou. *RTEC: Run-Time Event Calculus*. GitHub repository. 2016. URL: https://github.com/aartikis/RTEC.

[13] Alexander Artikis, Marek Sergot, and Georgios Paliouras. "Run-time composite event recognition". In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. 2012, pp. 69–80.

[14] Alexis Aurandt et al. "Multimodal model predictive runtime verification for safety of autonomous cyber-physical systems". In: *International Conference on Formal Methods for Industrial Critical Systems*. Springer. 2024, pp. 220–244.

[15] Franz Baader, Sebastian Brandt, and Carsten Lutz. "Pushing the EL Envelope". In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*. 2005, pp. 364–369.

[16] Amin BakhshandehAbkenar and Seng W Loke. "Myactivity: Cloud-hosted continuous activity recognition using ontology-based stream reasoning". In: *2014 2nd IEEE international conference on mobile cloud computing, services, and engineering*. IEEE. 2014, pp. 117–126.

[17] Radu Balaj and Adrian Groza. "Detecting influenza epidemics based on real-time semantic analysis of Twitter data". In: *Proc. 3rd Intern. Conf. on Modelling and Development of Intelligent Systems (MDIS 2013)*. 2013, pp. 30–39.

[18] Marco Balduini et al. "Social listening of city scale events using the streaming linked data framework". In: *International Semantic Web Conference*. Springer. 2013, pp. 1–16.

[19] Davide Francesco Barbieri et al. "An execution environment for C-SPARQL queries". In: *Proceedings of the 13th International Conference on Extending Database Technology*. 2010, pp. 441–452.

[20] Davide Francesco Barbieri et al. "C-SPARQL: a continuous query language for RDF data streams". In: *International Journal of Semantic Computing* 4.01 (2010), pp. 3–25.

[21] Davide Francesco Barbieri et al. "C-SPARQL: SPARQL for continuous querying". In: *Proceedings of the 18th international conference on World Wide Web*. 2009, pp. 1061–1062.

[22] Davide Francesco Barbieri et al. "Continuous Queries and Real-time Analysis of Social Semantic Data with C-SPARQL". In: *Workshop on Social Data on the Web*. 2009, pp. 1–12.

[23] Davide Francesco Barbieri et al. "Querying RDF streams with C-SPARQL". In: *ACM SIGMOD Record* 39.1 (2010), pp. 20–26. DOI: 10.1145/1860702.1860705.

[24] Davide Francesco Barbieri et al. "Stream Reasoning: Where We Got So Far". In: *Proceedings of the 4th Workshop on New Forms of Reasoning for the Semantic Web (Scalable and Dynamic)*. 2010.

[25] Hamid R Bazoobandi et al. "Handling impossible derivations during stream reasoning". In: *European Semantic Web Conference*. Springer. 2020, pp. 3–19.

[26] Hamid R. Bazoobandi, Harald Beck, and Jacopo Urbani. "Expressive stream reasoning with Laser". In: *International Semantic Web Conference*. Springer. 2017, pp. 87–103.

[27] Harald Beck, Minh Dao-Tran, and Thomas Eiter. "LARS: A logic-based framework for analytic reasoning over streams". In: *Artificial Intelligence* 261 (2018), pp. 16–70.

[28] Harald Beck, Thomas Eiter, and Christian Folie. "Ticker: A system for incremental ASP-based stream reasoning". In: *Theory and Practice of Logic Programming* 17.5-6 (2017), pp. 744–763.

[29] Harald Beck et al. "LARS: A logic-based framework for analyzing reasoning over streams". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015, pp. 1431–1438.

[30] Harald Beck et al. "Rule-based stream reasoning for intelligent administration of content-centric networks". In: *European Conference on Logics in Artificial Intelligence*. Springer. 2016, pp. 522–528.

[31] Pieter Bonte, Femke Ongenae, and Riccardo Tommasini. "A holistic view over ontologies for Streaming Linked Data". In: *Semantic Web* 15.5 (2024), pp. 2005–2033.

[32] Pieter Bonte and Riccardo Tommasini. "Streaming linked data: A survey on life cycle compliance". In: *Journal of Web Semantics* 77 (2023), p. 100785.

[33] Pieter Bonte et al. "Grounding Stream Reasoning Research". In: *Transactions on Graph Data and Knowledge* 2 (2024), pp. 1–47.

[34] Pieter Bonte et al. "Languages and systems for RDF stream processing, a survey". In: *The VLDB Journal* 34.4 (2025), p. 50.

[35] Mathieu Bourgais, Franco Giustozzi, and Laurent Vercouter. "Detecting situations with stream reasoning on health data obtained with IoT". In: *Procedia Computer Science* 192 (2021), pp. 507–516.

[36] Sebastian Brandt et al. "Querying log data with metric temporal logic". In: *Journal of Artificial Intelligence Research* 62 (2018), pp. 829–877.

[37] Pearl Brereton et al. "Lessons from applying the systematic literature review process within the software engineering domain". In: *Journal of systems and software* 80.4 (2007), pp. 571–583.

[38] Pedro Cabalar and Gilberto Pérez Vega. "Temporal equilibrium logic: a first approach". In: *International Conference on Computer Aided Systems Theory*. Springer. 2007, pp. 241–248.

[39] Pedro Cabalar et al. "Metric temporal answer set programming over timed traces". In: *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer. 2022, pp. 117–130.

[40] Pedro Cabalar et al. "telingo= ASP+ time". In: *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer. 2019, pp. 256–269.

[41] Pedro Cabalar et al. "Towards metric temporal answer set programming". In: *Theory and Practice of Logic Programming* 20.5 (2020), pp. 783–798.

[42] Jean-Paul Calbimonte. *Morph-stream*. GitHub repository. 2016. URL: https://github.com/jpcik/morph-streams.

[43] Jean-Paul Calbimonte. "Ontology-based access to sensor data streams". PhD thesis. Informatica, 2013.

[44] Jean-Paul Calbimonte, Oscar Corcho, and Alasdair JG Gray. "Enabling ontology-based access to streaming data sources". In: *The Semantic Web–ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I 9*. Springer. 2010, pp. 96–111.

[45] Jean-Paul Calbimonte, Jose Mora, and Oscar Corcho. "Query Rewriting in RDF Stream Processing". In: *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains-Volume 9678*. 2016, pp. 486–502.

[46] Francesco Calimeri et al. *ASP-Core-2 Input Language Format*. 2019. arXiv: 1911.04326 [cs.AI].

[47] Diego Calvanese et al. "Tractable reasoning and efficient query answering in description logics: The DL-Lite family". In: *Journal of Automated reasoning* 39.3 (2007), pp. 385–429.

[48] Davide Calvaresi and Jean-Paul Calbimonte. "Real-time compliant stream processing agents for physical rehabilitation". In: *Sensors* 20.3 (2020), p. 746.

[49] V Castiglioni, M Loreti, and S Tini. "RobTL: A Temporal Logic for the Robustness of Cyber-Physical Systems". In: *arXiv preprint arXiv:2212.11158* (2022).

[50] Valentina Castiglioni, Michele Loreti, and Silvia Tini. *EvTL: A Temporal Logic for the Transient Analysis of Cyber-Physical Systems*. 2022. arXiv: 2204 . 13357 [cs.LO].

[51] Léa Charbonnier et al. "Explainability of Quality Issues in Manufacturing: a Semantic Based Approach". In: *23nd International Semantic Web Conference (ISWC 2024)*. Vol. 3828. 2024.

[52] Léa Charbonnier et al. "Towards a Semantic Approach to Detection of Quality Issues in Manufacturing 4.0". In: *Procedia Computer Science* 246 (2024), pp. 2439–2448.

[53] Jiaoyan Chen et al. "Knowledge graph embeddings for dealing with concept drift in machine learning". In: *Journal of Web Semantics* 67 (2021), p. 100625.

[54] Luís Cruz-Filipe, Isabel Nunes, and Graça Gaspar. "Hypothetical Answers to Continuous Queries over Data Streams". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.03 (2020), pp. 2798–2805.

[55] David J. Tena Cucala et al. "Stratified negation in Datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.7 (2021), pp. 6488–6495.

[56] Gianpaolo Cugola and Alessandro Margara. "TESLA: a formally defined event specification language". In: *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*. 2010, pp. 50–61.

[57] Mathias De Brouwer, Femke Ongenae, and Filip De Turck. "Demonstration of a stream reasoning platform on low-end devices to enable personalized real-time cycling feedback". In: *European Semantic Web Conference*. Springer. 2019, pp. 28–32.

[58] Mathias De Brouwer et al. "Context-aware & privacy-preserving homecare monitoring through adaptive query derivation for IoT data streams with DIVIDE". In: *Semantic Web Journal* (2023).

[59] Mathias De Brouwer et al. "Distributed continuous home care provisioning through personalized monitoring & treatment planning". In: *Companion Proceedings of the Web Conference 2020*. 2020, pp. 143–147.

[60] Mathias De Brouwer et al. "Optimized continuous homecare provisioning through distributed data-driven semantic services and cross-organizational workflows". In: *Journal of Biomedical Semantics* 15.1 (2024), p. 9.

[61] Mathias De Brouwer et al. "Personalized real-time monitoring of amateur cyclists on low-end devices: proof-of-concept & performance evaluation". In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 1833–1840.

[62] Daniel De Leng and Fredrik Heintz. "DyKnow: A dynamically reconfigurable stream reasoning framework as an extension to the robot operating system". In: *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE. 2016, pp. 55–60.

[63] Amirhoshang Hoseinpour Dehkordi, Majid Alizadeh, and Ali Movaghar. *Linear Temporal Public Announcement Logic: A New Perspective for Reasoning the Knowledge of Agents*. 2020. arXiv: 2009.03793 [cs.LO].

[64] Alexander Dejonghe et al. "C-GeoSPARQL: streaming GeoSPARQL support on C-SPARQL". In: *Joint Proceedings of the 2nd RDF Stream Processing (RSP 2017) and the Querying the Web of Data (QuWeDa 2017) Workshops co-located with 14th ESWC 2017 (ESWC 2017)*. 2017, pp. 1–10.

[65] Daniele Dell'Aglio et al. "RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems". In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.4 (2014), pp. 17–44.

[66] Daniele Dell'Aglio et al. "Stream Reasoning: a Survey and Outlook: A summary of ten years of research and a vision for the next decade". In: *Data Science* 1.1-2 (2017), pp. 59–83.

[67] Emanuele Della Valle et al. "It's a streaming world! Reasoning upon rapidly changing information". In: *IEEE Intelligent Systems* 24.6 (2009), pp. 83–89.

[68] Konrad Diwold et al. "Grid watch dog: a stream reasoning approach for lightweight SCADA functionality in low-voltage grids". In: *Proceedings of the 8th International Conference on the Internet of Things*. 2018, pp. 1–8.

[69] Patrick Doherty, Fredrik Heintz, and Jonas Kvarnström. "Robotics, Temporal Logic and Stream Reasoning". In: *International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-19), 14-19 December 2013, Stellenbosch, South Africa*. 2013.

[70] Patrick Doherty and Jonas Kvarnström. "Temporal action logics". In: *Foundations of Artificial Intelligence* 3 (2008), pp. 709–757.

[71] Patrick Doherty et al. "HDRC3-a distributed hybrid deliberative/reactive architecture for unmanned aircraft systems". In: *Handbook of unmanned aerial vehicles* (2014), pp. 849–952.

[72] T Eiter and R Kiesel. "Weighted LARS for quantitative stream reasoning". In: *ECAI 2020* (2020). DOI: 10.3233/FAIA200160.

[73] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. "Answer set programming: A primer". In: *Reasoning Web International Summer School*. Springer, 2009, pp. 40–110.

[74] Thomas Eiter, Paul Ogris, and Konstantin Schekotihin. "A distributed approach to LARS stream reasoning (system paper)". In: *Theory and Practice of Logic Programming* 19.5-6 (2019), pp. 974–989.

[75] Emanuele Falzone, Riccardo Tommasini, and Emanuele Della Valle. "Stream reasoning: From theory to practice". In: *Reasoning Web International Summer School*. Springer, 2020, pp. 85–108.

[76] Borja Ramis Ferrer et al. "Towards processing and reasoning streams of events in knowledge-driven manufacturing execution systems". In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE. 2015, pp. 1075–1080.

[77] Nikolaus Manes Funk. "Real time logics and reasoning answerset semantics for metric temporal logic". PhD thesis. TU Wien, 2021.

[78] Martin Gebser et al. "Potassco: The Potsdam answer set solving collection". In: *Ai Communications* 24.2 (2011), pp. 107–124.

[79] Víctor Gutiérrez-Basulto and Szymon Klarman. "Towards a unifying approach to representing and querying temporal data in description logics". In: *International conference on web reasoning and rule systems*. Springer. 2012, pp. 90–105.

[80] Petr Hájek. *Metamathematics of fuzzy logic*. Vol. 4. Springer Science & Business Media, 2001.

[81] Olaf Hartig. "Foundations of RDF* and SPARQL*:(An alternative approach to statement-level metadata in RDF)". In: *AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017*. Vol. 1912. Juan Reutter, Divesh Srivastava. 2017.

[82] Thomas Hassan, Fano Ramparany, and Grégoire Lefebvre. "A semantic event approach to enrich device-free indoor localization data". In: *Proceedings of the 4th EAI International Conference on Smart Objects and Technologies for Social Good*. 2018, pp. 106–111.

[83] Haruna Isah et al. "A survey of distributed data stream processing frameworks". In: *IEEE Access* 7 (2019), pp. 154300–154316.

[84] Edmond Jajaga, Lule Ahmedi, and Figene Ahmedi. "StreamJess: a stream reasoning framework for water quality monitoring". In: *International Journal of Metadata, Semantics and Ontologies* 11.4 (2016), pp. 207–220.

[85] Robin Keskisärkkä et al. "RSP-QL: Enabling Statement-Level Annotations in RDF Streams". In: *International Conference on Semantic Systems*. Springer. 2019, pp. 140–155.

[86] Evgeny Kharlamov et al. "Semantic access to streaming and static data at Siemens". In: *Journal of Web Semantics* 44 (2017), pp. 54–74.

[87] Evgeny Kharlamov et al. "Semantically-enhanced rule-based diagnostics for industrial Internet of Things: The SDRL language and case study for Siemens trains and turbines". In: *Journal of Web Semantics* 56 (2019), pp. 11–29.

[88] Evgeny Kharlamov et al. "Towards analytics aware ontology based access to static and streaming data". In: *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference*. Springer. 2016, pp. 344–362.

[89] S Klarman and T Meyer. "Prediction and Explanation over DL-Lite Data Streams". In: *International Conference on Logic for Programming Artificial Intelligence and Reasoning*. 2013, pp. 536–551. DOI: 10.1007/978-3-642-45221-5_36.

[90] Matthias Klusch et al. "iCM-Hydraulic: Semantics-empowered cobrandtndition monitoring of hydraulic machines". In: *Proceedings of the 11th international conference on semantic systems*. 2015, pp. 81–88.

[91] Robert Kowalski and Marek Sergot. "A logic-based calculus of events". In: *New generation computing* 4 (1986), pp. 67–95.

[92] Ron Koymans. "Specifying real-time properties with metric temporal logic". In: *Real-time systems* 2.4 (1990), pp. 255–299.

[93] Pradeep Kumar and H. Howie Huang. "Graphone: A data store for real-time analytics on evolving graphs". In: *ACM Transactions on Storage (TOS)* 15.4 (2020), pp. 1–40.

[94] Freddy Lécué. "Diagnosing changes in an ontology stream: A DL reasoning approach". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 26.1 (2012), pp. 80–86.

[95] Freddy Lécué and Jeff Pan. "Consistent knowledge discovery from evolving ontologies". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (2015).

[96] Freddy Lécué et al. "Predicting severity of road traffic congestion using semantic web technologies". In: *The Semantic Web: Trends and Challenges: 11th International Conference*. 2014, pp. 611–627.

[97] Tony Lee et al. "Location-based Mobile Recommendations by Hybrid Reasoning on Social Media Streams". In: *Joint International Semantic Technology Conference*. Springer. 2013, pp. 261–273.

[98] Daniel de Leng and Fredrik Heintz. "Qualitative spatio-temporal stream reasoning with unobservable intertemporal spatial relations using landmarks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (2016).

[99] Nicola Leone et al. "Efficiently Computable Datalog∃ Programs". In: *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*. AAAI Press, 2012, pp. 13–23.

[100] Michele Loreti and Valentina Castiglioni. *SPEAR for ULTRON*. GitHub repository. 2021. URL: https://github.com/gitUltron/Ultron.

[101] Oded Maler and Dejan Nickovic. "Monitoring Temporal Properties of Continuous Signals". In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166. ISBN: 978-3-540-30206-3.

[102] Periklis Mantenoglou and Alexander Artikis. "Extending the Range of Temporal Specifications of the Run-Time Event Calculus". In: *31st International Symposium on Temporal Representation and Reasoning (TIME 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2024, pp. 6–1.

[103] Periklis Mantenoglou, Manolis Pitsikalis, and Alexander Artikis. "Stream reasoning with cycles". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning* 19.1 (2022), pp. 544–553.

[104] Victor W. Marek and Miroslaw Truszczyński. "Stable models and an alternative logic programming paradigm". In: *The logic programming paradigm: A 25-year perspective*. Springer, 1999, pp. 375–398.

[105] Alessandro Margara et al. "Efficient temporal reasoning on streams of events with DOTR". In: *European Semantic Web Conference* (2018), pp. 384–399.

[106] Alessandro Margara et al. "Streaming the web: Reasoning over dynamic data". In: *Journal of Web Semantics* 25 (2014), pp. 24–44.

[107] Alberto Martín-Martín et al. "Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories". In: *Journal of informetrics* 12.4 (2018), pp. 1160–1177.

[108] Takeshi Morita et al. "Implementing Customer Reception Service in a Robot Cafe Using Stream Reasoning and ROS Based on PRINTEPS". In: *Proceedings of the 15th International Semantic Web Conference (ISWC 2016): Posters and Demos*. 2016.

[109] Angelo Morzenti, Dino Mandrioli, and Carlo Ghezzi. "A model parametric real-time logic". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 14.4 (1992), pp. 521–573.

[110] Esteban Municio et al. "Continuous athlete monitoring in challenging cycling environments using IoT technologies". In: *IEEE Internet of Things Journal* 6.6 (2019), pp. 10875–10887.

[111] Matthias Nickles and Alessandra Mileo. "Web stream reasoning using probabilistic answer set programming". In: *International Conference on Web Reasoning and Rule Systems*. Springer. 2014, pp. 197–205.

[112] Markus Nissl. "Temporal Reasoning in Knowledge Graphs. Artificial Intelligence Systems for Reasoning with Time in Vadalog". PhD thesis. Technische Universität Wien, 2023.

[113] Özgür Levent Özçep, Ralf Möller, and Christian Neuenstadt. *OBDA Stream Access Combined with Safe First-Order Temporal Reasoning*. Tech. rep. Hamburg University of Technology, 2014.

[114] Thu-Le Pham, Muhammad Intizar Ali, and Alessandra Mileo. "C-ASP: continuous ASP-based reasoning over RDF streams". In: *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer. 2019, pp. 45–50.

[115] D Le Phuoc. "A native and adaptive approach for linked stream data processing". PhD thesis. NUI Galway, 2013.

[116] Manolis Pitsikalis and Alexander Artikis. "Maritime Event Recognition". In: *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*. 2019, pp. 232–235.

[117] Amir Pnueli. "The temporal logic of programs". In: *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*. IEEE. 1977, pp. 46–57.

[118] David A Randell, Zhan Cui, and Anthony G Cohn. "A spatial logic based on regions and connection". In: *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*. 1992, pp. 165–176.

[119] Haoyu Ren, Darko Anicic, and Thomas A Runkler. "The synergy of complex event processing and tiny machine learning in industrial IoT". In: *Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems*. 2021, pp. 126–135.

[120] Luca Roffia et al. "A semantic publish-subscribe architecture for the Internet of Things". In: *IEEE Internet of Things Journal* 3.6 (2016), pp. 1274–1296.

[121] Georgios M. Santipantakis et al. "A Stream Reasoning System for Maritime Monitoring". In: *Proceedings of the 25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*. OASIcs. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, 20:1–20:12.

[122] Ognjen Savkovic et al. "Theoretical Characterization of Signal Diagnostic Processing Language". In: *Proceedings of the 31st International Workshop on Description Logics*. Vol. 2211. CEUR Workshop Proceedings. 2018.

[123] Ahmet Soylu et al. "OptiqueVQS: A visual query system over ontologies for industry". In: *Semantic Web* 9.5 (2018), pp. 627–660.

[124] Ahmet Soylu et al. "Querying industrial stream-temporal data: An ontology-based visual approach". In: *Journal of Ambient Intelligence and Smart Environments* 9.1 (2017), pp. 77–95.

[125] Mattias Tiger and Fredrik Heintz. "Incremental reasoning in probabilistic signal temporal logic". In: *International Journal of Approximate Reasoning* 119 (2020), pp. 325–352.

[126] Mattias Tiger and Fredrik Heintz. "Stream reasoning using temporal logic and predictive probabilistic state models". In: *2016 23rd International Symposium on Temporal Representation and Reasoning (TIME)*. IEEE. 2016, pp. 196–205.

[127] Satyadharma Tirtarasa and Anni-Yasmin Turhan. "Towards Reverse Engineering Temporal Queries: Generalizing EL Concepts with Next and Global". In: *Informal Proceedings of the Stream Reasoning Workshop*. Ed. by Emanuele Della Valle et al. 2021, p. 14.

[128] Riccardo Tommasini et al. *RSP4J: an API for RDF stream processing*. 2021, pp. 565–581.

[129] Alexandros Troupiotis-Kapeliaris et al. "Experimental Comparison of Complex Event Processing Systems in the Maritime Domain". In: *2020 21st IEEE International Conference on Mobile Data Management (MDM)*. IEEE. 2020, pp. 293–298.

[130] Johan Van Benthem, Jan Van Eijck, and Barteld Kooi. "Logics of communication and change". In: *Information and computation* 204.11 (2006), pp. 1620–1662.

[131] Jelle Vanhaeverbeke et al. "Real-time estimation and monitoring of COVID-19 aerosol transmission risk in office buildings". In: *Sensors* 23.5 (2023), p. 2459.

[132] Rick Villars et al. *IDC FutureScape: Worldwide IT Industry 2024 Predictions*. Tech. rep. US50435423. International Data Corporation (IDC), Oct. 2023.

[133] W3C. *OWL 2 Web Ontology Language*. W3C Recommendation. World Wide Web Consortium (W3C), Dec. 2012.

[134] W3C. *RDF 1.2 Concepts and Abstract Syntax*. W3C Recommendation. 2024.

[135] W3C. *RDF 1.2 Semantics*. W3C Recommendation. 2024.

[136] W3C. *SPARQL 1.1 Query Language*. W3C Recommendation. World Wide Web Consortium (W3C), Mar. 2013.

[137] Przemysław Andrzej Wałęga, Mark Kaminski, and Bernardo Cuenca Grau. "Reasoning over streaming data in metric temporal datalog". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 3092–3099.

[138] Przemysław Andrzej Wałęga et al. "DatalogMTL: computational complexity and expressive power". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2019, pp. 1886–1892.

[139] Dingmin Wang et al. "Meteor: Practical reasoning in datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.5 (2022), pp. 5906–5913.

[140] William Weiss and Cherie D'Mello. *Fundamentals of model theory*. Topology Atlas, 2000.

[141] Yongchun Xu et al. "An approach for more efficient energy consumption based on real-time situational awareness". In: *Proceedings of the 8th extended semantic web conference*. 2011, pp. 270–284.

[142] Xuanxing Yang. "Query for streaming information: dynamic processing and adaptive incremental maintenance of RDF stream". In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 843–847.

[143] Juan Ye et al. "Semantic web technologies in pervasive computing: A survey and research roadmap". In: *Pervasive and Mobile Computing* 23 (2015), pp. 1–25.

[144] Lotfi A Zadeh. "Fuzzy sets". In: *Information and control* 8.3 (1965), pp. 338–353.

[145] Deng Zhao et al. "ASTL: Accumulative STL with a novel robustness metric for IoT service monitoring". In: *IEEE Transactions on Mobile Computing* 22.10 (2022), pp. 5751–5768.

[146] Pouyan Ziafati et al. "Event-processing in autonomous robot programming". In: *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. 2013, pp. 95–102.

[147] Pouyan Ziafati et al. "Reasoning on robot knowledge from discrete and asynchronous observations". In: *2014 AAAI Spring Symposium Series*. 2014.

[148] Pouyan Ziafati et al. "Retalis Language for Information Engineering in Autonomous Robot Software". In: *Journal of Logics* (2015), p. 65.

Appendix A: Key Papers

[1] Darko Anicic et al. "EP-SPARQL: a unified language for event processing and stream reasoning". In: *Proceedings of the 20th international conference on World wide web* (2011). DOI: 10.1145/1963405.1963495.

[2] Darko Anicic et al. "Stream reasoning and complex event processing in ETALIS". In: *Semantic web* 3.4 (2012), pp. 397–407.

[3] Marco Balduini et al. "Social listening of city scale events using the streaming linked data framework". In: *International Semantic Web Conference*. Springer. 2013, pp. 1–16.

[4] Davide Francesco Barbieri et al. "Querying RDF streams with C-SPARQL". In: *ACM SIGMOD Record* 39.1 (2010), pp. 20–26. DOI: 10.1145/1860702.1860705.

[5] Davide Francesco Barbieri et al. "Stream Reasoning: Where We Got So Far". In: *Proceedings of the 4th Workshop on New Forms of Reasoning for the Semantic Web (Scalable and Dynamic)*. 2010.

[6] Harald Beck, Minh Dao-Tran, and Thomas Eiter. "LARS: A logic-based framework for analytic reasoning over streams". In: *Artificial Intelligence* 261 (2018), pp. 16–70.

[7] Daniele Dell'Aglio et al. "Stream Reasoning: a Survey and Outlook: A summary of ten years of research and a vision for the next decade". In: *Data Science* 1.1-2 (2017), pp. 59–83.

[8] Emanuele Della Valle et al. "It's a streaming world! Reasoning upon rapidly changing information". In: *IEEE Intelligent Systems* 24.6 (2009), pp. 83–89.

[9] Thomas Eiter, Paul Ogris, and Konstantin Schekotihin. "A distributed approach to LARS stream reasoning (system paper)". In: *Theory and Practice of Logic Programming* 19.5-6 (2019), pp. 974–989.

[10] Emanuele Falzone, Riccardo Tommasini, and Emanuele Della Valle. "Stream reasoning: From theory to practice". In: *Reasoning Web International Summer School*. Springer, 2020, pp. 85–108.

[11] Alessandro Margara et al. "Streaming the web: Reasoning over dynamic data". In: *Journal of Web Semantics* 25 (2014), pp. 24–44.

[12] Mohd Halim Mohd Noor, Zoran Salcic, and Kevin I-Kai Wang. "Ontology-based sensor fusion activity recognition". In: *Journal of Ambient Intelligence and Humanized Computing* 11.8 (2020), pp. 3073–3087.

[13] Luca Roffia et al. "A semantic publish-subscribe architecture for the Internet of Things". In: *IEEE Internet of Things Journal* 3.6 (2016), pp. 1274–1296.

[14] Riccardo Tommasini et al. *RSP4J: an API for RDF stream processing*. 2021, pp. 565–581.

[15] Juan Ye et al. "Semantic web technologies in pervasive computing: A survey and research roadmap". In: *Pervasive and Mobile Computing* 23 (2015), pp. 1–25.

3

Bridging Metric and Window-Based Stream Reasoning: Translation Between DatalogMTL and Logic-based framework for Analytic Reasoning over Streams

At the heart of the Stream Reasoning (SR) landscape lies a persistent divide in how time is represented and how temporal relevance is determined. Metric-based approaches such as DatalogMTL reason directly over time points and intervals, while window-based approaches such as Logic-based framework for Analytic Reasoning over Streams (LARS) select finite substreams relative to an evaluation time. Both paradigms are widely used to express temporal dependencies over evolving data, yet they are often treated as embodying incompatible semantic intuitions. As articulated in Research Challenge C2, this perceived incompatibility hampers the exchange of temporal specifications and reasoning results across systems that adopt different abstractions of time.

This chapter explores this divide and examines it at the level of formal semantics rather than surface syntax or implementation style, as such addressing research question RQ2. Using DatalogMTL and LARS as representative formalisms, it characterises precisely which combinations of temporal constructs admit a semantics-preserving correspondence and which do not. The analysis identifies concrete fragments of both languages for which translations preserve temporal entailment and minimal-model semantics, while also exposing the conditions under which such preservation fails. In doing so, the chapter establishes that semantic alignment between metric-based and window-based temporal reasoning is not an all-or-nothing matter, but depends on explicitly identifiable

structural restrictions, thereby satisfying the criteria formulated in Hypothesis H4 and clarifying the semantic relationship between the two paradigms.

⋆ ⋆ ⋆

M. van Noort, P. Bonte and F. Ongenae

Submitted for review to the Theory and Practice of Logic Programming Journal, May 2026.

3.1 Abstract

SR combines temporal rule-based reasoning with continuously arriving data. DatalogMTL and Logic-based framework for LARS have largely developed in parallel and embody fundamentally different temporal intuitions. The former is based on metric temporal operators over time intervals, while the latter is organised around window-based evaluation over streams. In this paper we show that this divide is largely syntactic, rather than semantic.

We establish semantics-preserving translations between DatalogMTL and LARS. For every DatalogMTL program and dataset, we construct an equivalent LARS program and stream representation that preserve truth, models, and minimal-model semantics. For every DatalogMTL program and dataset, we give a bidirectional, semantics-preserving translation between fragments of both formalisms, preserving models and truth at each time point. Conversely, for a well-defined fragment of LARS, we provide a translation back into DatalogMTL with the same model-theoretic guarantees. The correspondence is established at the level of interpretations and models.

Our results establish an exploratory connection between two dominant paradigms in SR. This provides a precise semantic bridge between metric- and window-based approaches to SR, clarifies the design space of temporal languages for SR, clarifies the design space of temporal languages, and enables systematic transfer of semantic results and specifications between the two formalisms without loss of meaning.

3.2 Introduction

Fueled by an increasing shift of information technologies away from static data processing towards fast-paced, real-time, and continuous analytics, SR has become a central component of modern temporal data processing [9]. SR is concerned with logical reasoning over continuously evolving data streams, where observations arrive incrementally and conclusions depend not only on their content but also on their temporal

order, duration, and persistence over time. It supports use cases ranging from financial fraud detection to mobility intelligence and large-scale Internet-of-Things (IoT) infrastructures, where raw streams of sensor readings, wearable measurements, social events, or transactional logs only become useful once temporal patterns and conditions are inferred in the presence of additional context and expert knowledge [5, 6].

To make such temporal interpretation possible, SR relies on formal representation languages [17]. These languages provide a machine-interpretable means to encode temporal dependencies, domain assumptions, causal relations, and aggregation patterns over time. Crucially, they allow experts to specify what temporal conditions and patterns should hold, without committing to how these patterns are operationally detected. In this way, representation languages form the semantic bridge between evolving data streams and declarative reasoning.

Within this landscape, temporal rule-based languages occupy a central position. Rule-based SR formalisms combine explicit temporal semantics with the declarative advantages of logic programming, including transparency of inference, compositionality, and a clear model-theoretic foundation. As shown in recent surveys [17], they are among the most widely adopted approaches for temporal reasoning over streams, as they strike a practical balance between expressivity, analyzability, and implementability.

Among the most prominent temporal rule languages are DatalogMTL [7] and LARS [4]. DatalogMTL extends classical Datalog with the modalities of Metric Temporal Logic (MTL) [13], enabling reasoning over temporal intervals and durations using metric constraints. Its appeal lies in its close connection to well-studied temporal logics, its clear least-model semantics, and its suitability for expressing persistence and bounded temporal properties over discrete or dense timelines [7, 8, 18, 24]. LARS, by contrast, emerges from Answer Set Programming (ASP) [15] and integrates temporal windows and temporal modalities into a rule-based semantics tailored to stream-oriented reasoning. Its window-centric design aligns naturally with incremental evaluation and bounded memory assumptions, which has made it a cornerstone for several practical SR systems [4, 10, 16].

Although both formalisms are designed to reason about temporal dependencies in rule-based settings, they arise from different motivations and adopt different temporal intuitions. DatalogMTL reasons directly over explicit temporal intervals and durations, making metric notions such as persistence and temporal distance part of the semantic interpretation. LARS, in contrast, derives conclusions relative to window-restricted substreams, where temporal relevance is determined by recency and scope rather than by explicit metric constraints [17]. This difference has often been treated as substantive and has encouraged a fragmented view of temporal logic programming, in which each formalism is seen as occupying its own semantic territory.

That fragmentation has concrete and increasingly costly practical consequences. Encoding background knowledge into a temporal rule language requires substantial investment of time and expertise [12]. Once such knowledge is committed to a particular formalism, however, it becomes difficult to reuse or reinterpret in settings governed by different temporal assumptions. This form of representation lock-in hampers semantic interoperability and obstructs the exchange, comparison, and evolution of temporal knowledge across systems. These limitations are becoming more acute with the rise of data space architectures and decentralised reasoning platforms, where conclusions and policies are expected to cross organisational and technical boundaries. Initiatives such as the European Health Data Space explicitly aim to enable cross-border exchange of health data and derived insights while preserving local governance and autonomy [11]. Similarly, SOLID promotes a decentralised Web architecture in which data and its interpretation are distributed across independently controlled agents and services [19, 20]. In such environments, the ability to exchange not only data but also the logic used to derive temporal conclusions becomes essential. When one system relies on window-based semantics and another on metric temporal constraints, the absence of principled semantic translations forces manual re-engineering of rules and undermines the faithful exchange of reasoning results.

A running example helps make the issue concrete. Consider a patient observed over time with respect to vaccination status and symptom-free intervals, adapted from [21]. Suppose we know that the patient is vaccinated at a certain time, that the patient has been symptom-free over a longer interval, and that medical background knowledge links these facts to derived conclusions such as immunity or a negative test result. This kind of example is small, but it captures the structural pattern that appears throughout SR: short-term observations combined with longer-horizon temporal guarantees, with rules that derive higher-level states from both.

The central question of this paper is whether metric temporal operators and window-based stream operators represent genuinely different semantic primitives, or whether they can be related within a common logic-programming framework. We address this question by introducing a formal, bidirectional translation that is guaranteed to preserve semantics between DatalogMTL and LARS. Despite their differing surface syntax and temporal intuitions, we show that the two formalisms overlap on a well-defined class of programs for which temporal behaviour can be transferred without loss of meaning.

Concretely, for every DatalogMTL program Π and dataset D, we construct a corresponding LARS program $\gamma(\Pi)$ and data stream $\gamma(D)$ whose models capture exactly the same temporal conclusions at every time point. Conversely, an inverse translation is established for a restricted LARS fragment that excludes unrestricted nesting, Boolean rule heads, and existential temporal conclusions. Within this fragment, window-based specifications can be translated back to DatalogMTL while preserving the intended temporal inferences.

Rather than relying on syntactic equivalence, the translations maintain agreement on which temporal facts are supported by the rules over time. In intuitive terms, both formalisms derive the same stable set of conclusions from the same evolving information, even though they organise temporal reasoning differently. In this sense, the paper identifies a shared semantic core underlying fragments of two temporal rule languages that are often presented as fundamentally distinct.

The conceptual upshot is that window operators are not a fundamentally new temporal construct but can be understood as derived operators over metric time within a logic-programming semantics. This perspective clarifies the design space of temporal languages for SR and enables the reuse of semantic results across formalisms that have traditionally been developed in isolation. It also has direct implications for data space and decentralised reasoning settings, where insights derived from data must be exchanged across organisational and technical boundaries under heterogeneous assumptions. In such environments, the ability to translate temporal reasoning specifications between metric-based and window-based formalisms without loss of semantics is essential for sharing, reusing, and trusting derived conclusions across systems that employ different temporal abstractions.

The rest of the paper develops this claim in detail. We first review the background on DatalogMTL and LARS and the semantic assumptions on which our result rests in Section 3.3. In Section 3.5, we then present the translation from DatalogMTL to LARS and prove that it preserves the relevant model-theoretic properties. After that, in Section 3.6, we develop the converse translation from LARS to DatalogMTL and establish the corresponding correctness result for the fragment under consideration. Section 3.7 summarizes the work, reflecting on the obtained results and highlighting avenues for future research, before concluding the work in Section 3.8.

3.3 Background

Before outlining the bidirectional translation between DatalogMTL and LARS, we first recall the essential definitions, semantics, and modelling assumptions underlying both formalisms. Although each framework is a temporal extension of rule-based reasoning, they differ substantially in how they treat time, granularity, model construction, and the operational meaning of temporal operators.

3.3.1 DatalogMTL

On the most fundamental level, DatalogMTL considers constants, variables and predicates. In this work, we denote these sets by $\mathcal{C}_{\mathtt{D}}$, $\mathcal{V}_{\mathtt{D}}$ and $\mathcal{P}_{\mathtt{D}}$ respectively, using subscript $\mathtt{D}$ to indicate they are part of DatalogMTL (compared to LARS). *Terms* $\bar{\tau}$ are tuples of constants and variables. A term is grounded if it contains only constants. A *relational atom* is an expression of the form $P(\bar{\tau})$, where $\bar{\tau}$ is a term and P a predicate. Likewise,

$P(\bar{\tau})$ is grounded if $\bar{\tau}$ is grounded. An assignment ν is a mapping from the union of variables and constants to the set of constants, where $\nu(c) = c$ for every constant c.

DatalogMTL uses operator symbols to express temporal relations, borrowed from MTL. To express a statement holds continuously, i.e. "always", it uses the square symbol $\Box$, combined with + or - to refer to either future or past, resulting in symbols $\boxplus$ and $\boxminus$. Likewise an existential statement, e.g. "once upon a time in the past", is expressed using $\diamondminus$ and statement "at some point in the future" is expressed using $\diamondplus$. DatalogMTL supports the relations of 'A *since* B' and 'A *until* B' using operators $\mathcal{S}$ and $\mathcal{U}$. For each of these six operators $\boxplus, \boxminus, \diamondplus, \diamondminus, \mathcal{S}$ and $\mathcal{U}$, DatalogMTL supports 'metric constraints', i.e. the scoping of the relation to a specific interval. It does so by means of an interval subscript, e.g. $\boxplus_{[a,b]}A$ expresses statement A holds continuously between the a and b time units in the future.

Formally, DatalogMTL employs *metric atoms* M, defined recursively as

$$M := \top | \bot | P(\bar{\tau}) | \boxplus_\varrho M | \boxminus_\varrho M | \diamondplus_\varrho M | \diamondminus_\varrho M | M \mathcal{S}_\varrho M | M \mathcal{U}_\varrho M$$

where $P(\bar{\tau})$ is a relational atom, M a metric atom and ϱ is a non-negative interval of the form $[a, b]$ or $[a, +\infty[$ over the rational timeline $\mathbb{Q}$. A metric atom is grounded if each occurring relational atom in the metric atom is grounded.

Example 3.1. *Consider for example a predicate* `NoSympt` *and constant 'Ben'. Then* $\texttt{NoSympt}(Ben)$ *is a relational atom, as well as a metric atom. Adding the operator* $\boxminus$, $\boxminus_{[0,10]}\texttt{NoSympt}(Ben)$ *is a metric atom.*

A *DatalogMTL rule* is of the form $M' \leftarrow M_1 \wedge \ldots \wedge M_n$, where each M_i is a metric atom and M' is a metric atom not mentioning any of $\diamondplus$, $\diamondminus$, $\mathcal{S}$, and $\mathcal{U}$. A *DatalogMTL program* is a finite set of DatalogMTL rules. A program is called *flat* if it does not mention any nesting of temporal operators.

Example 3.2. *Suppose medical experts have determined the infectiousness of individuals based on their symptoms. For example, they concluded that a person who has been symptom-free between three and seven days ago is not at risk of spreading the disease. Even when the person developed symptoms one or two days ago, they are not deemed infectious.*

Using the metric temporal operators of DatalogMTL, we can capture this knowledge as by means of the following DatalogMTL rule:

$$r_{eg} = \texttt{NoRisk}(X) \leftarrow \boxminus_{[3,7]}\texttt{NoSympt}(X)$$

Rules serve as the main carrier for expert knowledge; domain experts are well-versed in the principles that govern their particular field and are able to derive new insights by combining their knowledge with incoming data. By specifying the exact requirements or conditions M_1 to M_n along with the conclusion M', the knowledge of domain experts can be converted into a machine-readable format, to be processed

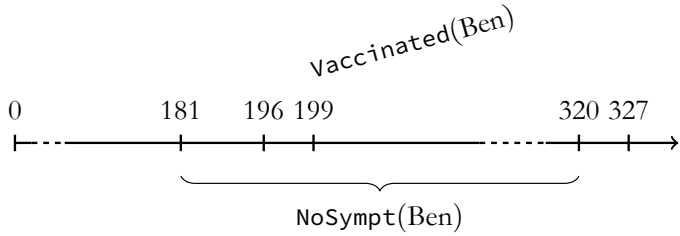


Figure 3.1: Illustration of DatalogMTL dataset $\mathcal{D}_{\text{eg}}$

by a DatalogMTL reasoner, such as MeTeoR [23]. Rules in DatalogMTL allow us to express the expert knowledge, but not the data itself.

To capture observations, such as 'Ben is vaccinated', DatalogMTL considers *facts* of the form $P(\mathbf{c})@\varrho$, where $P(\mathbf{c})$ is a grounded relational atom, P an extensional predicate and ϱ is an interval and the $@$ indicates the atom happens 'at' interval ϱ. In case of punctual intervals, i.e. singleton intervals $[t,t] = \{t\}$, we may write $P(\mathbf{c})@t$ instead of $P(\mathbf{c})@[t,t]$. A *DatalogMTL dataset* $\mathcal{D}$ is a finite set of temporal facts, i.e. a collection of what facts hold and when they hold. We say dataset $\mathcal{D}$ is *finite* if all temporal facts $P(\bar{\tau})@\varrho$ have closed intervals of the form $[a,b]$.

Example 3.3. *We consider the timeline of vaccination and symptoms as illustrated in Figure 3.1. The data of an individual Ben receiving a vaccine and experiencing no symptoms can be captured using a DatalogMTL dataset with two facts,* $\texttt{Vaccinated}(\mathit{Ben})@199$ *and* $\texttt{NoSympt}(\mathit{Ben})@[181,320]$. *As a result, we have DatalogMTL dataset*

$$\mathcal{D}_{\mathit{eg}} = \{\texttt{Vaccinated}(\mathit{Ben})@199, \texttt{NoSympt}(\mathit{Ben})@[181,320]\}.$$

A dataset of observations can be seen as a collection of verified pieces of information, hence 'facts'. Fact $\texttt{NoSympt}(Ben)@[181,320]$ reflects Ben did not have any symptoms within the time frame $[181,320]$, but in our example, we do not have information on Ben's symptoms before time point 181 or after 320. This gives rise to different possible situations; maybe Ben had symptoms on time point 180, or maybe he continued to have no symptoms up until time point 350, but we do not have confirmation of this information. To capture these different possible situations, DatalogMTL considers interpretations.

A *DatalogMTL interpretation* $\mathcal{I}$ is a function mapping each time point (a natural number) to a set of grounded relational atoms. For a $P(\bar{\tau})$ where $\bar{\tau}$ only contains variables, we say $\mathcal{I}$ *satisfies grounded atom* $P(\bar{\tau})$ *at time* t, or alternatively $P(\bar{\tau})$ *holds for* $\mathcal{I}$ *at* t, denoted $\mathcal{I}, t \models P(\bar{\tau})$, if $P(\bar{\tau}) \in \mathcal{I}(t)$. For $\top$ and $\bot$, $\mathcal{I}, t \models \top$ and $\mathcal{I}, t \not\models \bot$ for every $\mathcal{I}$ and t.

We define the range of $\mathcal{I}$ as the minimal and maximal time point, beyond which $\mathcal{I}$ contains no atoms, denoted by $t_{\mathcal{I},\text{min}}$ and $t_{\mathcal{I},\text{max}}$ respectively. In case no maximal time exists within $\mathbb{N}$, $t_{\mathcal{I},\text{max}}$ is defined as $+\infty$.

When domain experts apply their knowledge to a particular situation, they replace an abstract concept, such as 'a person', by one particular instance, say the person 'Ben', and draw conclusions based on this replacement, i.e. "a person immune for 5 days means they will test negative" becomes "Ben being immune for 5 days means Ben will test negative". Formally, the role of such replacements is performed by assignments ν, considered above.

With the use of assignments ν, satisfaction can be extended to non-grounded relational atoms. For an assignment ν and $\bar{\tau}$ containing both constants and variables, we say $\mathcal{I}$ satisfies $P(\bar{\tau})$ under ν at time t, denoted as $\mathcal{I}, t \models^{\nu} P(\bar{\tau})$, if $\mathcal{I}, t \models P(\nu(\bar{\tau}))$ and $\mathcal{I}, t \models P(\bar{\tau})$ if $\mathcal{I}, t \models^{\nu} P(\bar{\tau})$ for every assignment ν. Satisfaction is subsequently extended to all metric atoms as follows. For the metric atoms with temporal operators, we have:

$$
\begin{array}{ll}
\mathcal{I}, t \models^{\nu} \diamondminus_{\varrho} M & \text{if } \mathcal{I}, t' \models^{\nu} M \text{ for some } t' \text{ with } t - t' \in \varrho \\
\mathcal{I}, t \models^{\nu} \diamondplus_{\varrho} M & \text{if } \mathcal{I}, t' \models^{\nu} M \text{ for some } t' \text{ with } t' - t \in \varrho \\
\mathcal{I}, t \models^{\nu} \boxminus_{\varrho} M & \text{if } \mathcal{I}, t' \models^{\nu} M \text{ for all } t' \text{ with } t - t' \in \varrho \\
\mathcal{I}, t \models^{\nu} \boxplus_{\varrho} M & \text{if } \mathcal{I}, t' \models^{\nu} M \text{ for all } t' \text{ with } t' - t \in \varrho \\
\mathcal{I}, t \models^{\nu} M_1 \mathcal{S}_{\varrho} M_2 & \text{if } \mathcal{I}, t' \models^{\nu} M_2 \text{ for some } t' \text{ with } t - t' \in \varrho, \\
 & \quad \text{and } \mathcal{I}, t'' \models^{\nu} M_1 \text{ for all } t'' \in (t', t) \\
\mathcal{I}, t \models^{\nu} M_1 \mathcal{U}_{\varrho} M_2 & \text{if } \mathcal{I}, t' \models^{\nu} M_2 \text{ for some } t' \text{ with } t' - t \in \varrho, \\
 & \quad \text{and } \mathcal{I}, t'' \models^{\nu} M_1 \text{ for all } t'' \in (t, t')
\end{array}
$$

We write $\mathcal{I} \models M@\varrho$ if $\mathcal{I}, t \models M$ for all $t \in \varrho$. An interpretation $\mathcal{I}$ contains an interpretation $\mathcal{I}'$, denoted $\mathcal{I}' \subseteq \mathcal{I}$, if for every t and $P(\bar{\tau})$, $\mathcal{I}', t \models P(\bar{\tau})$ implies $\mathcal{I}, t \models P(\bar{\tau})$.

An interpretation $\mathcal{I}$ is a *model* of a rule $M' \leftarrow M_1 \wedge \ldots \wedge M_n$ if for every time point t in $\mathbb{N}$, it holds that $\mathcal{I}, t \models M'$ holds whenever $\mathcal{I}, t \models M_i$ holds for every $i \in \{1, \ldots, n\}$. Interpretation $\mathcal{I}$ is a model of a program Π if it is a *model* of every rule in Π. An interpretation is a *model* of a dataset $\mathcal{D}$ if for every fact $P(\mathbf{c})@\varrho \in \mathcal{D}$, we have $\mathcal{I} \models P(\mathbf{c})@\varrho$. A pair $(\Pi, \mathcal{D})$ consisting of a program and dataset is *consistent* if there is an interpretation that is a model for both Π and $\mathcal{D}$. In this case, we let $\mathcal{I}_{\Pi,\mathcal{D}}$ be the unique interpretation which is the *least model* of $\mathcal{D}$ and Π, i.e. for any model $\mathcal{I}$ of $\mathcal{D}$ and Π, $\mathcal{I}_{\Pi,\mathcal{D}} \subseteq \mathcal{I}$. A pair $(\Pi, \mathcal{D})$ *entails* a grounded metric atom M at t, written as $(\Pi, \mathcal{D}), t \models M$, if $\mathcal{I}, t \models M$ for every model $\mathcal{I}$ of $(\Pi, \mathcal{D})$. We denote the set of models for a pair $(\Pi, \mathcal{D})$ as $M_{\Pi,\mathcal{D}}$.

Example 3.4. *For the dataset D_{eg} of Example 3.3, we can define a DatalogMTL interpretation $\mathcal{I}_{eg}$ with* `Vaccinated(Ben)` $\in \mathcal{I}_{eg}(199)$ *and for every t between or equal to* 181 *and* 320, `NoSympt(Ben)` $\in \mathcal{I}_{eg}(t)$.

If we consider the domain knowledge captured by Example 3.2, we can observe that $\mathcal{I}_{eg}$ is not a model for the rule r_{eg}. $\mathcal{I}_{eg}$ states Ben has no symptoms over interval $[181, 320]$. *By our domain knowledge, Ben is risk-free over interval* $[188, 323]$. *However, $\mathcal{I}_{eg}$ does not contain fact* $\texttt{NoRisk}(Ben)@[188, 323]$. *$\mathcal{I}_{eg}$ can however be extended to a model for both the rule r_{eg} and dataset $\mathcal{D}_{eg}$ by adding* $\texttt{NoRisk}(Ben)$ *to $\mathcal{I}_{eg}(t)$ for all time points in* $[188, 323]$. *With this extended model, we can also conclude the pair of (one-rule) program $\{r_{eg}\}$ and dataset $\mathcal{D}_{eg}$ is consistent.*

Originally, DatalogMTL is defined over a rational timeline $\mathbb{Q}$ [7]. In this work, we restrict the timeline to $\mathbb{N}$ to match the discrete time model of LARS. This choice has minimal practical impact, as streaming data is typically timestamped at fixed granularities, which means that all relevant temporal information can already be represented using natural numbers. In the vaccination example considered above, we can consider one time unit to be equal to one day, and $t = 0$ equal to January 1st. $t = 199$ then refers to July 19. It is however possible to look more granular, e.g. $t = 199$ then refers to July 19 00:00 and $t = 199.5$ refers to July 19 at 12:00. By counting in hours or minutes instead of days, we once again obtain natural numbers (e.g. 199.5 in days equals 4788 hours). The impact on theory is likewise limited, since the results we rely on, such as satisfaction at time points, bounded metric intervals, and minimal model constructions, depend only on the linear order of time and not on the density of $\mathbb{Q}$.

3.3.2 LARS

Similar to DatalogMTL, LARS considers disjoint sets of constants, variables and predicates, denoted by $\mathcal{C}_{\text{L}}$, $\mathcal{V}_{\text{L}}$ and $\mathcal{P}_{\text{L}}$ respectively, with subscript L for LARS. The set of predicates $\mathcal{P}_{\text{L}}$ is further split into two parts: extensional predicates $\mathcal{P}_{\text{L,ex}}$ and intensional predicates $\mathcal{P}_{\text{L,in}}$. A LARS atom is of the form $p(\bar{t})$, where $p \in \mathcal{P}_{\text{L}}$ and $\bar{t}$ a tuple of constants and variables. In case $p(\bar{t})$ only contains constants, it is called grounded. In this case, we often denote it by $\bar{c}$. We use $\mathcal{G}$ to denote the set of grounded LARS atoms, i.e. $\mathcal{G} := \{p(\bar{c}) | p \in \mathcal{P}_{\text{L}} \text{ and } \bar{c} \text{ a tuple of constants}\}$.

Central in the LARS framework are streams. LARS represents streams by means of a tuple (T, υ), where T is a non-empty interval on $\mathbb{N}$ and υ is an evaluation function $\upsilon : \mathbb{N} \to 2^{\mathcal{G}}$, which maps a time point $t \in T$ to a set of grounded atoms and for each $t \in \mathbb{N} \setminus T$, $\upsilon(t) = \emptyset$. A stream $S' = (T', \upsilon')$ is a substream from S if $T' \subseteq T$ and for all $t' \in T'$, $\upsilon'(t') \subseteq \upsilon(t')$. We say $S' = (T', \upsilon')$ is a *time-based substream* of S if $S' = (T', \upsilon|_{T'})$, i.e. $\upsilon'(t') = \upsilon(t')$ for $t' \in T'$ and $\upsilon'(t') = \emptyset$ otherwise. Note that, unlike [3], the above definition allows for streams over intervals of the form $[0, +\infty[$.

A stream which only contains extensional predicates is called a *data stream*. For a data stream D, we say S is an *interpretation stream* for D if $D \subseteq S$ and all atoms that occur in S but not in D contain only intensional predicates.

Example 3.5. *If we consider the vaccination example within a LARS context, the available data on Ben's vaccination and absence of symptoms can be seen as a stream* $S_{eg} = ([0, 320], \upsilon)$ *where* $\upsilon(199) = \{vaccinated(Ben), noSympt(Ben)\}$, $\upsilon(t) = \{noSympt(Ben)\}$ *for* $t \in [181, 320]$, *but not* $t = 199$, *and* $\upsilon(t) = \emptyset$ *otherwise. Assuming* $vaccinated$ *and* $noSympt$ *are extensional predicates, this stream is a data stream.*

Streams are manipulated by means of *window functions*. In the most abstract sense, a window function is defined as a mapping which takes as input a stream S and time point t and returns a substream S' of S. [4] specifies four types of windows: time-based windows, tuple-based windows, partition windows and filter windows. Time-based windows employ three parameters, lower offset l, upper offset u and hop size d. Given stream $S = (T, \upsilon)$ with $T = [t_{\min}, t_{\max}]$, we define a time-based window function $\tau^{l,u(d)}(S, t) = (T', \upsilon|_{T'})$, where $T' = [\max(t_{\min}, t' - l), \min(t_{\max}, t' + u)]$ and $t' = \left\lfloor \frac{t}{d} \right\rfloor \cdot d$.

Tuple-based windows are defined using two parameters, lower offset l and upper offset u. Window function $\#^{l,u}$ is defined as $\#^{l,u}(S, t) = (T', \upsilon'|_{T'})$, where $\upsilon'(t')$ equals $\upsilon(t')$ as long as the set $\cup_{t' \leq t^* \leq t} \upsilon(t^*)$ contains at most l elements, or $\cup_{t \leq t^* \leq t'} \upsilon(t^*)$ contains at most u elements. If $\upsilon(t')$ causes the union to exceed the set amount of elements, $\upsilon'(t')$ is a subset of $\upsilon(t')$ so the union contains exactly l or u elements.

LARS foresees in the ability to split a single stream into multiple substreams based on the occurring atoms, apply a tuple-based operators on each substream and subsequently rejoin the results. Formally, we consider an index function $idx : \mathcal{G} \to \mathtt{I}$ which assigns an index to each atom, and substream S_i as the stream for which $\upsilon_i(t) = \upsilon(t) \cap \{\alpha | idx(\alpha) = i\}$. With tuple sizes $\bar{n} = (n_1, n_2, \ldots)$ for every $i \in \mathtt{I}$, we define the *partition-based window* as $\mathsf{p}^{idx,\bar{n}}(S, t) = \cup_{i \in \mathtt{I}} \#^{n_i}(S_i, t)$.

For a given set of grounded atoms, say $A \subseteq \mathcal{G}$, a *filter window* $\mathtt{f}^A$ is defined as the function which limits stream S to only the atoms in set A, i.e. every point t' evaluates to set $\upsilon(t') \cap A$. Note the time point parameter t plays no role in the definition of filter window $\mathtt{f}^A$.

Example 3.6. *If we consider the LARS data stream* $S_{eg} = ([0, 320], \upsilon)$ *of Ben's vaccination outlined above, several window functions can be considered. First, time-based window functions allow us to investigate specific time periods. For example, time window* $\tau^{0,30(1)}$ *extracts the data of only the next 30 days from a stream.* $\tau^{7,7(7)}$ *hones in on the data of this week and the past week, for any day of the week. Filter functions can prove useful if our interest goes out to only a specific kind of data. For example retrieving only information on Ben's vaccination and not on his symptoms can be performed by* $\mathtt{f}^A$ *with* $A = \{\mathtt{vaccination}(Ben)\}$.

Tuple- and partition-based windows become relevant if our data is expanded beyond just one person, and questions such as "Who are the last 100 people to get vaccinated?" need to be answered. This can be solved using a partition window which creates a substream of $vaccinated$ *predicates, followed by a tuple-based window* $\#^{100,0}$.

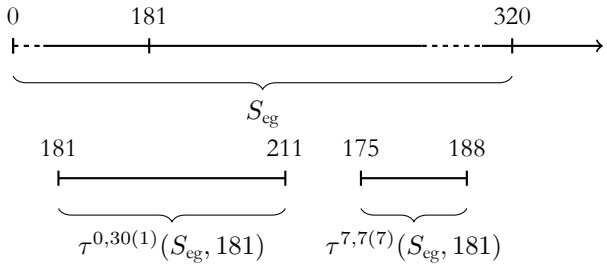


Figure 3.2: Illustration of time-based windows $\tau^{0,30(1)}$ and $\tau^{7,7(7)}$ applied to stream S_{eg} at time 181. Each window application results in a substream of stream S_{eg}.

Consider data stream S_{eg} describing Ben's vaccination and symptoms, as considered in Example 3.5. Applying time-based window $\tau^{0,30(1)}$ to S_{eg} at time 181 results in the substream with timeline $[181, 211]$, the 30 days following day 181. $\upsilon(t)$ is then equal to $\{vaccinated(Ben), noSympt(Ben)\}$ for $t = 199$ and $\{noSympt(Ben)\}$ for all other time points in $[181, 211]$. Applying function $\tau^{7,7(7)}$ to S_{eg} at time 181 retrieves the past week (starting last Monday) and next week (up until next Sunday). If we assume day $t = 0$ is a Monday, all Mondays are equal to the time points divisible by 7. Then day 181 is a Sunday. The start of the week of day 181 is thus 175, and the end of next week is 188. As a result, $\tau^{7,7(7)}(S_{eg}, 181) = ([175, 188], \upsilon|_{[175,188]})$, with $\upsilon|_{[175,188]}(t) = \{noSympt(Ben)\}$ for every $t \in [175, 188]$. Figure 3.2 illustrates both window applications as two segments resulting from the original stream S_{eg}. Note that due to the tumbling nature of window $\tau^{7,7(7)}$, the same substream is obtained for every day of the same week, i.e. for every $t \in [175, 181]$, $\tau^{7,7(7)}(S_{eg}, t) = \tau^{7,7(7)}(S_{eg}, 181)$, because $\left\lfloor \frac{t}{7} \right\rfloor$ rounds down to 25 for each of these time points.

LARS formulae are constructed using recursive application of several operators, starting from atoms $a = p(x_1, \ldots, x_n)$. Four of these operators are the traditional logical operators $\{\neg, \wedge, \vee, \rightarrow\}$. Furthermore, LARS employs three temporal operators: $\Box$ expresses universal quantification over time (i.e. "holds always"), $\Diamond$ expresses existential quantification ("holds at some time point") and $@_t$ expresses something holds at time point t. Lastly, LARS incorporates windows, allowing one to consider formulae over specific ranges or substreams. With operator $\boxplus^w$, the subsequent formula is evaluated on the substream induced by applying window function w to the given stream. Lastly, operator $\triangleright$ indicates the subsequent formula is to be evaluated not on the given stream, which may contain only a portion of the available data, but on all of the available data. As a result, construction of LARS formulae can be defined as

$$\alpha := a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \Box\,\alpha \mid \Diamond\,\alpha \mid @_t\alpha \mid \boxplus^w \alpha \mid \triangleright\,\alpha$$

Consistent with the literature of both LARS and DatalogMTL, the LARS window operator $\boxplus^{\tau}$ and the DatalogMTL "always future" operator $\boxplus_{\varrho}$ employ the same symbol. Throughout this work, the former is distinguished from the latter via its use of superscript for the window function τ, compared to subscript for the interval ϱ.

Within this work, we use the following shorthand for the application of window operator $\boxplus^{w}$ with w a time-based window with hop size 1. For $w = \tau^{l,u(1)}$, we write $\boxplus^{(l)(u)}$ to replace $\boxplus^{\tau^{l,u(1)}}$. In case of $u = 0$, we omit u to obtain $\boxplus^{l}$. Likewise for $l = 0$, we write $\boxplus^{+u}$.

LARS formulae are evaluated on a *structure* $\mathfrak{M} = \langle S, W, B \rangle$, where $S = (T, \upsilon)$ is called the *interpretation stream*, W is a set of window functions and B is a set of grounded facts that are free of temporal and window operators, the *background data*. This structure captures all three parts of the system: all of the available dynamic data in the form of stream S, the possibilities to manipulate the stream via window in W and the static data is captured by B.

Example 3.7. *With regard to the vaccination and symptoms of Ben, the dynamic data consists of his vaccination at time 199 and the absence of symptoms over time period 181 to 320. This thus results in the stream S_{eg} considered in Example 3.5. The set of window functions W is primarily in function of how we would like to manipulate the stream S_{eg}. Suppose we are mainly interested in time- and filter windows, i.e. $W = \{w | w$ a time-based or filter window$\}$. Thus far, we have not considered background data explicitly. A straightforward sample of static data can be found however in Ben's characteristics, such as his gender, genetics and lifestyle (assuming these remain constant over the considered timeline). Say for example we are certain of his gender and lifestyle, i.e. $B = \{man(Ben), healthyLifestyle(Ben)\}$. The structure $\mathfrak{M} = \langle S_{eg}, W, B \rangle$ then captures the dynamic data, ways to manipulate the dynamic data and lastly the static data of our example.*

The semantics of LARS formulae are defined over a structure $\mathfrak{M}$, a stream S and a time point t, where $\mathfrak{M} = \langle S^*, W, B \rangle$ and $S^* = (T^*, \upsilon^*)$. $S = (T, \upsilon)$ is a substream of S^* and t a time point in T^*. The semantics of each formula are given by:

$$
\begin{aligned}
&\mathfrak{M}, S, t \Vdash a && \text{iff } a \in \upsilon(t) \text{ or } a \in B \\
&\mathfrak{M}, S, t \Vdash \neg\alpha && \text{iff } \mathfrak{M}, S, t \nVdash \alpha \\
&\mathfrak{M}, S, t \Vdash \alpha \wedge \beta && \text{iff } \mathfrak{M}, S, t \Vdash \alpha \text{ and } \mathfrak{M}, S, t \Vdash \beta \\
&\mathfrak{M}, S, t \Vdash \alpha \vee \beta && \text{iff } \mathfrak{M}, S, t \Vdash \alpha \text{ or } \mathfrak{M}, S, t \Vdash \beta \\
&\mathfrak{M}, S, t \Vdash \alpha \rightarrow \beta && \text{iff } \mathfrak{M}, S, t \nVdash \alpha \text{ or } \mathfrak{M}, S, t \Vdash \beta \\
&\mathfrak{M}, S, t \Vdash \Box\alpha && \text{iff } \mathfrak{M}, S, t' \Vdash \alpha \text{ for all } t' \in T \\
&\mathfrak{M}, S, t \Vdash \Diamond\alpha && \text{iff } \mathfrak{M}, S, t' \Vdash \alpha \text{ for some } t' \in T \\
&\mathfrak{M}, S, t \Vdash @_{t'}\alpha && \text{iff } \mathfrak{M}, S, t' \Vdash \alpha \\
&\mathfrak{M}, S, t \Vdash \boxplus^{w}\alpha && \text{iff } \mathfrak{M}, S', t \Vdash \alpha \text{ where } S' = w(S, t) \\
&\mathfrak{M}, S, t \Vdash \triangleright\alpha && \text{iff } \mathfrak{M}, S^*, t \Vdash \alpha
\end{aligned}
$$

Note that the LARS temporal operators $\Box$ and $\Diamond$ are not oriented towards either past or future, as is the case for DatalogMTL operators $\boxplus$, $\boxminus$, $\diamondplus$ and $\diamondminus$.

As with DatalogMTL, domain knowledge is encoded in rules. A *LARS rule* is an expression of the form $r = \alpha \leftarrow \beta_1, \ldots, \beta_n$ where α and β_i are LARS formulae. $H(r)$ denotes the *head* α of rule r, whereas $B(r)$ denotes the *body*, $\beta_1, \ldots, \beta_n$. A *LARS program* P is a set of rules.

Example 3.8. *Say we want to infer whether a person is safe against infection, for example if they either have been vaccinated at most 7 days ago or if they are symptom-free for at least 14 days. We can capture this using the LARS rule*

$$safe(X) \leftarrow \boxplus^{7} vaccinated(X) \vee \boxplus^{14} noSympt(X).$$

Furthermore, a person who did not receive a vaccination (yet), but has gone 18 days without experiencing any symptoms, might either be very lucky to not have been infected, or they might be (permanently) naturally immune to the disease. We can capture this using the following rule.

$$
\begin{aligned}
lucky(X) \vee \Box\, naturallyImmune(X) \leftarrow \\
\boxplus^{18}\, noSympt(X) \wedge \neg \Diamond\, vaccinated(X)
\end{aligned}
\tag{3.1}
$$

The rule above gives rise to multiple possible scenarios for a single person, i.e. they are either lucky or naturally immune (or both). Given a data stream D and an interpretation stream $S(T, \upsilon)$ for D, then structure $\mathfrak{M} = \langle S, W, B\rangle$ is called an *interpretation* for stream D. M is called a *model* of rule r for D at time t if $\mathfrak{M}, S, t \Vdash H(r)$ when $\mathfrak{M}, S, t \Vdash B(r)$. $\mathfrak{M}$ is a model of program P for D at t if it is a model on every rule $r \in P$ for D at t.
$\mathfrak{M} = \langle S, W, B\rangle$ is a minimal model of P for D at t if there is no model $\mathfrak{M}' = \langle S', W, B\rangle$ on P for D at time t, for which it holds that $S' = (T, \upsilon')$ and $S' \subset S$.

Example 3.9. *Similar to DatalogMTL, an interpretation in LARS represents one particular possible situation or version of reality. In stream S_{eg} considered above, we observe that Ben has been unvaccinated and symptom-free for 18 days on $t = 199$. According to Rule 3.1, Ben is either lucky or naturally immune. We can therefore consider two scenarios: (a) Ben is lucky or (b) Ben is naturally immune. We can compose an interpretation stream $S_{(a)}$ for option (a) by adding the atom $lucky(Ben)$ to S_{eg} at time $t = 199$, i.e. $S_{(a)}(199) = v(199) \cup \{lucky(Ben)\}$. Likewise, an interpretation stream $S_{(b)}$ can be constructed by adding $naturallyImmune(Ben)$ instead. Both structure $\langle S_{(a)}, W, B\rangle$ and $\langle S_{(b)}, W, B\rangle$ are models of Rule 3.1 for S_{eg} at $t = 199$. However, they represent different 'possible realities'. Note that neither option contains the other, i.e. $S_{(a)} \not\subseteq S_{(b)}$ and $S_{(b)} \not\subseteq S_{(a)}$.*

3.4 Comparative prelude

Table 3.1 summarises the main structural and semantic design choices underlying DatalogMTL and LARS. At a high level, both formalisms follow a similar rule-based paradigm: they build temporal expressions over predicate atoms and organise these into rules evaluated over evolving data. This shared foundation allows both languages to express temporal dependencies and derive conclusions over time in a declarative manner.

The key differences emerge in how temporal scope and evaluation are realised. DatalogMTL adopts a metric perspective, where temporal operators directly constrain intervals on a single, global interpretation. In contrast, LARS evaluates formulae relative to dynamically constructed substreams, obtained via window functions over an underlying stream. As reflected in Table 3.1, this leads to differences in temporal orientation (explicit past- and future-directed operators versus undirected modalities), evaluation objects (interpretations versus structures with substreams), and the role of windows in shaping the data over which reasoning takes place.

Despite these differences, the table also highlights important points of alignment. In particular, both formalisms ultimately perform temporal quantification over selected portions of a stream, whether specified via metric intervals or via window-based substream selection. This correspondence suggests that the distinction between metric and window-based reasoning may lie more in modelling perspective than in fundamental semantic capability, a hypothesis that we study more closely in the following sections.

3.5 Translating DatalogMTL into LARS

The first direction of our bidirectional correspondence concerns the translation of DatalogMTL into LARS. Although both formalisms support rich temporal reasoning,

Table 3.1: Comparative summary of design elements in DatalogMTL and LARS.

Design element	**DatalogMTL**	**LARS**
Basic building blocks	Relational atoms $P(\bar{\tau})$	Atoms $p(\bar{t})$
Temporal intuition	Metric constraints over time intervals	Window-restricted evaluation over streams
Evaluation object	Interpretation $\mathcal{I}$	Structure $\mathcal{M}$ and substream $S \subseteq S^*$
Stream perspective	Single global (implicit) stream	Global stream with derived substreams
Substream construction	Not explicit (always $\mathcal{I}$)	Explicit via window functions
Types of substreams	Time-based	Time-, tuple-, filter-, and partition-based
Temporal restriction	Metric interval subscripts $[a, b]$	Window operators (e.g. time-based)
Temporal orientation	Directed operators ($\boxplus$, $\boxminus$, $\diamondplus$, $\diamondminus$)	Undirected temporal modalities ($\Box$, $\Diamond$)
Semantic viewpoint	Evaluation over full interpretation at each time point	Evaluation over dynamically constructed substreams
Nesting behaviour	Evaluation remains over full timeline (no induced truncation)	Evaluation over progressively restricted substreams
Rule conclusions	Restricted to metric atoms without $\diamondplus$, $\diamondminus$	General LARS formulae

Table 3.2: Correspondence γ of DatalogMTL metric atoms to semantically equivalent LARS formulae.

DatalogMTL metric atom	LARS formula
$\bot$	$\bot$
$\top$	$\top$
$P(\bar{\tau})$	$p(\bar{t})$
$\boxplus_{[a,b]} M$	$\boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box \triangleright \gamma(M)$
$\boxminus_{[a,b]} M$	$\boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)} \Box \triangleright \gamma(M)$
$\diamondplus_{[a,b]} M$	$\boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond \triangleright \gamma(M)$
$\diamondminus_{[a,b]} M$	$\boxplus^{b} \Box \boxplus^{(\infty)(b-a)} \Diamond \triangleright \gamma(M)$
$M_1 \mathcal{U}_{[a,b]} M_2$	$\Big(\boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond \big(\triangleright \gamma(M_2) \wedge \boxplus^{(\infty)(0)} \Box \triangleright \gamma(M_1) \big) \Big) \wedge \big(\boxplus^{+a} \Box \triangleright \gamma(M_1) \big)$
$M_1 \mathcal{S}_{[a,b]} M_2$	$\Big(\boxplus^{b} \Box \boxplus^{(\infty)(b-a)} \Diamond \big(\triangleright \gamma(M_2) \wedge \boxplus^{(0)(\infty)} \Box \triangleright \gamma(M_1) \big) \Big) \wedge \big(\boxplus^{a} \Box \triangleright \gamma(M_1) \big)$

they realise temporal behaviour through fundamentally different mechanisms: DatalogMTL relies on metric operators interpreted over intervals on a (dense or discrete) timeline, whereas LARS evaluates formulae relative to window-restricted substreams combined with modal operators. To connect these views, we develop a translation that maps DatalogMTL metric atoms to semantically equivalent LARS formulae and show how interpretations can be systematically transformed into corresponding LARS structures. This transformation preserves the truth of temporal statements at every time point. The remainder of this section introduces the translation formally, establishes its semantic correctness by structural induction, and shows how the correspondence lifts from atoms to rules, programs, and their respective minimal models.

Definition 3.1. *We consider a mapping γ which maps constants, variables and predicates in DatalogMTL to their counterparts in LARS. With the notation introduced above, we define $\gamma : \mathcal{C}_D \cup \mathcal{V}_D \cup \mathcal{P}_D \to \mathcal{C}_L \cup \mathcal{V}_L \cup \mathcal{P}_L$. We further extend γ to a mapping between DatalogMTL metric atoms and LARS formulae. First for relational atoms, $\gamma(P(x_1, \ldots, x_n)) := \gamma(P)\big(\gamma(x_1), \ldots, \gamma(x_n)\big)$. To extend γ to arbitrary metric atoms, we define γ recursively for each form of metric atom. The expression for each $\gamma(M)$ is given by Table 3.2.*

The LARS formulae in Table 3.2 appear highly verbose. They are, however, chosen with the specific intent of capturing the same semantic meaning as their DatalogMTL counterpart. In Lemmas 3.4 through 3.7 and Theorem 3.9, we will show exactly that these DatalogMTL and LARS statements are semantically equivalent.

Definition 3.2. *For a given DatalogMTL interpretation stream $\mathcal{I}$, we define a LARS structure which captures the semantically same information as $\mathcal{I}$ as follows. With interpretation stream $\mathcal{I} : \mathbb{N} \to 2^{\mathcal{G}}$, we define the LARS structure induced by DatalogMTL interpretation stream $\mathcal{I}$ as $\mathfrak{M}_{\mathcal{I}} = \langle S_{\mathcal{I}}, W_{\mathcal{I}}, B_{\mathcal{I}} \rangle$, where*

$$
\begin{aligned}
S_{\mathcal{I}} &= (T, \upsilon_{\mathcal{I}}), \text{ where } \upsilon_{\mathcal{I}}(t) = \{\gamma(P(\bar{c}) | P(\bar{c}) \in \mathcal{I}(t)\} \\
T &= [t_{\mathcal{I},\min}, t_{\mathcal{I},\max}], \text{ for } t_{\mathcal{I},\max} \in \mathbb{N} \\
&= [t_{\mathcal{I},\min}, +\infty[, \text{ for } t_{\mathcal{I},\max} = +\infty \\
W_{\mathcal{I}} &= \{\tau^{l,u(1)} | l, u \in \mathbb{N} \cup \{\infty\}\} \\
B_{\mathcal{I}} &= \{\top\}
\end{aligned}
$$

The set of windows $W_{\mathcal{I}}$ must contain the time-based windows $\tau^{l,u(d)}$, but can be extended arbitrarily.

The definition above provides a methodology to migrate an interpretation $\mathcal{I}$, a 'version of reality' described in DatalogMTL, into a LARS structure that describes the same reality, but in LARS instead. We illustrate this by revisiting the vaccination example introduced in Section 3.3.

Example 3.10. *Consider the DatalogMTL interpretation $\mathcal{I}_{eg}$ from Example 3.4;*

$$
\begin{aligned}
\mathcal{I}_{eg}(199) &= \{\texttt{Vaccinated}(Ben)\} \\
\mathcal{I}_{eg}(t) &= \{\texttt{NoSympt}(Ben)\} \text{ for } t \in [181, 320] \\
\mathcal{I}_{eg}(t) &= \emptyset \text{ otherwise.}
\end{aligned}
$$

According to Definition 3.2, the LARS structure $\mathfrak{M}_{\mathcal{I}_{eg}}$ has stream $S_{\mathcal{I}_{eg}}$ with

$$
\begin{aligned}
T &= [181, 320] \\
\upsilon_{\mathcal{I}_{eg}}(199) &= \{\gamma(\texttt{Vaccinated}(Ben)), \gamma(\texttt{NoSympt}(Ben))\} \\
\upsilon_{\mathcal{I}_{eg}}(t) &= \{\gamma(\texttt{NoSympt}(Ben))\} \text{ for } t \in [181, 320] \setminus \{199\} \\
\upsilon_{\mathcal{I}_{eg}}(t) &= \emptyset \quad \text{otherwise}
\end{aligned}
$$

With this interpretation $\mathcal{I}_{eg}$, it can be verified the following metric atoms hold;

- $M_1 = \diamondminus_{[1,2]}\texttt{Vaccinated}(Ben)$ *holds over $\mathcal{I}_{eg}$ on days where Ben was vaccinated either the day before or two days before. With $\texttt{Vaccinated}(Ben)@199$, this holds on days $t = 201$ and $t = 202$, i.e. July 21st and July 22nd.*

- $M_2 = \boxminus_{[7,14]}\texttt{NoSympt}(Ben)$ *holds over $\mathcal{I}$ for any day where the data in $\mathcal{I}_{eg}$ specifically states that Ben has had no symptoms between 7 and 14 days before the day at hand. This holds between July 15 and November 23, i.e. $\mathcal{I}_{eg}, t \models \boxminus_{[7,14]}\texttt{NoSympt}(Ben)$ for $t \in [196, 327]$. Note that with the information from $\mathcal{I}_{eg}$, we can not draw conclusions regarding occurrence or absence of symptoms for Ben before July 1st or past November 16.*

- $M_3 = \texttt{NoSympt}(Ben)\mathcal{S}_{[7,14]}\texttt{Vaccinated}(Ben)$ *holds over* $\mathcal{I}_{eg}$ *only if we can with certainty conclude that Ben has been vaccinated between one and two weeks ago and has not experienced any symptoms since his vaccination. This holds true between July 26 (*$t = 207$*) and August 2nd (*$t = 214$*).*

Using γ *as defined above, the corresponding LARS formulae must hold over the structure* $\mathfrak{M}_{\mathcal{I}_{eg}}$ *at the same point in time. With the above,*

- $\gamma(M_1) = \boxplus^{1} \boxdot \boxplus^{(\infty)(1)} \Diamond \triangleright \mathit{vaccinated}(Ben)$
- $\gamma(M_2) = \boxplus^{14} \Diamond \boxplus^{(\infty)(7)} \boxdot \triangleright \mathit{noSympt}(Ben)$
- $\gamma(M_3) = \Big(\boxplus^{14} \boxdot \boxplus^{(\infty)(7)} \Diamond \big(\triangleright \mathit{vaccinated}(Ben) \wedge \boxplus^{(0)(\infty)} \boxdot \triangleright \mathit{noSympt}(Ben) \big) \Big) \wedge \big(\boxplus^{7} \boxdot \triangleright \mathit{noSympt}(Ben) \big)$

Structure $\mathfrak{M}_{\mathcal{I}_{eg}}$ *has stream* $\S_{eg}$ *with* $\upsilon_{\mathcal{I}_{eg}}(199) = \{\mathit{vaccinated}(Ben), \mathit{noSympt}(Ben)\}$ *and* $\upsilon_{\mathcal{I}_{eg}}(t) = \{\mathit{noSympt}(Ben)\}$ *for* $t \in [181, 320] \setminus \{199\}$. *In order for the translation to be correct, each* $\gamma(M_i)$ *should hold over* $\mathfrak{M}_{\mathcal{I}_{eg}}$ *and* $S_{\mathcal{I}_{eg}}$ *for the same time points as* M_i *does. We illustrate this for* M_2.

With $\mathfrak{M}_{\mathcal{I}_{eg}}, S_{\mathcal{I}_{eg}}, t \Vdash \boxplus^{14} \Diamond \boxplus^{(\infty)(7)} \boxdot \triangleright \mathit{noSympt}(Ben)$, *the operator* $\boxplus^{14}$ *limits the timeline to* $[t-14, \min(+\infty, t+\infty)[= [t-14, +\infty[$. *Then by operator* $\Diamond$, *there should be a* t' *on the timeline for which* $\mathfrak{M}_{\mathcal{I}_{eg}}, S', t' \Vdash \boxplus^{(\infty)(7)} \boxdot \triangleright \mathit{noSympt}(Ben)$ *holds, with* $S' = S_{\mathcal{I}}|_{[t-14,+\infty[}$. *Restricting the timeline further to* $[t-14, t'+7]$, *we see* $\mathit{noSympt}(Ben)$ *must hold in the original stream* $S_{\mathcal{I}_{eg}}$ *over the timeline* $[t-14, t'+7]$. *For any possibility for* t' *(in* $[t-14, +\infty[$*) that meets this criterion,* $t' = t-14$ *will also meet it. As a result, it suffices to find the values for* t *for which* $\mathfrak{M}_{\mathcal{I}_{eg}}, S_{\mathcal{I}_{eg}}, t^* \Vdash \mathit{noSympt}(Ben)$ *holds for every* $t^* \in [t-14, t-7]$. *As was the case for DatalogMTL, we obtain that* $\gamma(M_2)$ *holds over* $M_{\mathcal{I}_{eg}}, S_{\mathcal{I}_{eg}}$ *for every* t *in interval* $[196, 327]$.

For ease of notation, we denote the corresponding LARS predicate of a DatalogMTL predicate, say `Predicate`, as $\gamma(\texttt{Predicate}) = \mathit{predicate}$, where we use typewriter font to denote predicate symbols in DatalogMTL and italics for their counterparts in LARS. We use the same notation for a DatalogMTL constant as for its LARS counterpart, e.g. $\gamma(Ben) = Ben$.

3.5.1 Preliminary Lemmas on DatalogMTL Semantics

Before we consider the semantic equivalence of the DatalogMTL metric atoms and LARS formulae of Table 3.2, we introduce two properties regarding DatalogMTL models for programs and datasets;

Lemma 3.1. *For dataset* $\mathcal{D}$ *and program* Π *and interpretation* $\mathcal{I}^*$ *, defined as* $\mathcal{I}^*(t) := \bigcap_{\mathcal{I} \in M_{\Pi,\mathcal{D}}} \mathcal{I}(t)$, *it holds that for every metric atom* M, *time point* $t \in \mathbb{N}$ *and assignment* ν, $\mathcal{I}^*, t \models^{\nu} M$ *if and only if* $\mathcal{I}, t \models^{\nu} M$ *for every* $\mathcal{I} \in M_{\Pi,\mathcal{D}}$.

Proof. Consider arbitrary $t \in \mathbb{N}$ and assignment ν. We employ induction on the construction of metric atom M.

First, consider $M = P(\bar{\tau})$. By the semantics of DatalogMTL and the definition of $\mathcal{I}^*$, it readily follows that $\mathcal{I}^*, t \models^\nu P(\bar{\tau}) \Leftrightarrow \forall \mathcal{I} \in M_{\Pi,\mathcal{D}} : \mathcal{I}, t \models^\nu P(\bar{\tau})$.

Consider $M = \boxplus_{[a,b]} M'$, where the hypothesis holds for M'. Using the semantics and induction hypothesis:

$$
\begin{aligned}
&\Leftrightarrow \mathcal{I}^*, t \models^\nu M \\
&\Leftrightarrow \forall t' \in [t+a, t+b] : \mathcal{I}^*, t \models^\nu M' \\
&\Leftrightarrow \forall t' \in [t+a, t+b] : \forall \mathcal{I} \in M_{\Pi,\mathcal{D}} : \mathcal{I}, t \models^\nu M' \\
&\Leftrightarrow \forall \mathcal{I} \in M_{\Pi,\mathcal{D}} : \mathcal{I}, t \models^\nu M
\end{aligned}
$$

Forms $\boxminus_{[a,b]} M'$, $\diamondplus_{[a,b]} M'$, $\diamondminus_{[a,b]} M'$, $M_1' \mathcal{U}_{[a,b]} M_2'$ and $M_1' \mathcal{S}_{[a,b]} M_2'$ for M follow analogously. □

Lemma 3.2. *For dataset $\mathcal{D}$ and program Π, interpretation $\mathcal{I}^*$ is the least model for Π and $\mathcal{D}$.*

Proof. First, consider a $P(\bar{c})@\varrho \in \mathcal{D}$. Since $\forall t \in \varrho : \forall \mathcal{I} \in M_{\Pi,\mathcal{D}} : P(\bar{c}) \in \mathcal{I}(t)$, it follows that $\forall t \in \varrho : P(\bar{c}) \in \mathcal{I}^*(t)$. Therefore $\mathcal{I}$ is a model for $\mathcal{D}$.

Next, consider rule $r \in \Pi$, $r = M' \leftarrow M_1 \wedge \ldots M_n$, and arbitrary $t \in \mathbb{N}$ and assignment ν. Assume $\mathcal{I}^*, t \models^\nu B(r)$, i.e. $\forall i \leq n : \mathcal{I}^*, t \models^\nu M_i$ By Lemma 3.1, $\forall i \leq n : \forall \mathcal{I} \in M_{\Pi,\mathcal{D}} : \mathcal{I}, t \models^\nu M_i$. Since each $\mathcal{I}$ is a model for Π, $\forall \mathcal{I} \in M_{\Pi,\mathcal{D}} : \mathcal{I}, t \models^\nu H(r) = M'$. Likewise using Lemma 3.1, $\mathcal{I}^*, t \models^\nu H(r)$.

Thus far, $\mathcal{I}^*$ is a model for Π and $\mathcal{D}$. From the definition of $\mathcal{I}^*$, $\mathcal{I}^*$ is contained in every $\mathcal{I} \in M_{\Pi,\mathcal{D}}$, i.e. $\mathcal{I}^* \subseteq \mathcal{I}$ for every model $\mathcal{I}$ of Π and $\mathcal{D}$. $\mathcal{I}^*$ is thus the least model for Π and $\mathcal{D}$. □

3.5.2 Correctness of the Translation by Structural Induction

The recursive definitions of both DatalogMTL metric atoms and LARS formulae lend themselves to a proof by induction. For direction DatalogMTL→LARS, the induction base is provided by relational atoms $P(\bar{\tau})$. The induction step then consists of proving γ correctly translates the semantics of metric atoms $\boxplus_{[a,b]} M | \boxminus_{[a,b]} M | \diamondplus_{[a,b]} M | \diamondminus_{[a,b]} M | M_1 \mathcal{U}_{[a,b]} M_2 | M_1 \mathcal{S}_{[a,b]} M_2$, assuming γ translates M, M_1 and M_2 correctly. We cover this induction step for each form individually in Lemmas 3.3 through 3.8.

3.5.2.1 Since & Until Operators

According to the semantics of DatalogMTL, as described in Section 3.3, the operators $\mathcal{U}$ and $\mathcal{S}$ appear the most complex. Their semantics feature two quantifications, over t' and t'' respectively, compared to only one quantification in case of

$\boxplus, \boxminus, \oplus$ and $\ominus$. Furthermore, as we will discuss below, $\oplus$ and $\ominus$ can be considered as specific sub-cases of $\mathcal{U}$ and $\mathcal{S}$.

Starting with form $M_1 \mathcal{U}_{[a,b]} M_2$. Theorem 3.9 then provides the proof by induction, ensuring that γ retains satisfaction for every DatalogMTL metric atom M.

Lemma 3.3. *Consider interpretation stream $\mathcal{I}$, a time point $t \in \mathbb{N}$ and DatalogMTL metric atom of the form $M_1 \mathcal{U}_{[a,b]} M_2$, where for M_1 and M_2, it holds that $\mathcal{I}', t' \models M_i \Rightarrow \mathfrak{M}_{\mathcal{I}'}, S_{\mathcal{I}'}, t' \Vdash \gamma(M_i)$ for any $\mathcal{I}'$ and t' with $i \in \{1, 2\}$. If $\mathcal{I}, t \models M_1 \mathcal{U}_{[a,b]} M_2$, then it holds that*

$$\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \Big(\boxplus^{+b} \boxdot \boxplus^{(b-a)(\infty)} \Diamond \big(\triangleright \gamma(M_2) \wedge \boxplus^{(\infty)(0)} \boxdot \triangleright \gamma(M_1) \big) \Big) \wedge \big(\boxplus^{+a} \boxdot \triangleright \gamma(M_1) \big)$$

and vice versa.

Proof. Consider an arbitrary interpretation $\mathcal{I}$, time point $t \in \mathbb{N}$ and metric atom $M_1 \mathcal{U}_{[a,b]} M_2$.

$\Rightarrow$. Assume $\mathcal{I}, t \models M_1 \mathcal{U}_{[a,b]} M_2$. We will prove the considered LARS formula holds over $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}$ and t.
According to DatalogMTL semantics, $\mathcal{I}, t \models M_1 \mathcal{U}_{[a,b]} M_2$ is equal to

$$\exists t' \in [t + a, t + b] : \mathcal{I}, t' \models M_2 \text{ and } \forall t'' \in [t, t'] : \mathcal{I}, t'' \models M_1. \tag{3.2}$$

Consider an arbitrary time point $t_1 \in [t, t+b]$. Since $t \leq t+a$ and $t_1 - (b-a) \leq t + b - (b-a) = t + a$, it follows that $[t + a, t + b] \subseteq [\max(t, t_1 - (b-a)), t + b]$. Equation 3.2 thus implies that

$$\begin{aligned} \forall t_1 \in [t, t + b] : \exists t' \in [\max(t, t_1 - (b - a)), t + b] : \\ \mathcal{I}, t' \models M_2 \text{ and } \forall t'' \in [t, t'] : \mathcal{I}, t'' \models M_1 \end{aligned} \tag{3.3}$$

By the assumed induction hypothesis for both M_1 and M_2, it follows that

$$\begin{aligned} \forall t_1 \in [t, t + b] : \exists t' \in [\max(t, t_1 - (b - a)), t + b] : \\ \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t' \Vdash \gamma(M_2) \text{ and } \forall t'' \in [t, t'] : \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M_1) \end{aligned} \tag{3.4}$$

Consider window functions τ^{+b} and $\tau^{(b-a)(\infty)}$. Subsequently define $S_1' := \tau^{+b}(S_{\mathcal{I}}, t)$ and $S_1'' := \tau^{(b-a)(\infty)}(S_1', t_1)$. Then the above can be rewritten into

$$\begin{aligned} \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t' \Vdash & \gamma(M_2) \text{ and } \forall t'' \in [t, t'] : \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M_1), \\ & \text{for some } t' \in T_1'', \text{ where } S_1'' = \tau^{(b-a)(\infty)}(S_1', t_1) \\ & \text{for all } t_1 \in T_1', \text{ where } S_1' = \tau^{+b}(S_{\mathcal{I}}, t) \end{aligned} \tag{3.5}$$

We zoom in on the part $\forall t'' \in [t, t'] : \mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t'' \Vdash \gamma(M_1)$. Consider window function $\tau^{(\infty)(0)}$ and stream $S_1''' := \tau^{(\infty)(0)}(S_1'', t')$. Then $T_1''' = [t''_{1,\text{min}}, t'] = [\max(t'_{1,\text{min}}, t'_1 - (b-a)), t']$. From $t'_{1,\text{min}} = \max(t_\text{min}, t) = t$, we deduce that $[t''_{1,\text{min}}, t'] \subseteq [t, t']$. As a result, $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t'' \Vdash \gamma(M_1)$ holds for every $t'' \in [t''_{1,\text{min}}, t']$. With the definition of S_1''', this is equal to

$$\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t'' \Vdash \gamma(M_1) \text{ for all } t'' \in T_1''', \text{where } S_1''' = \tau^{(\infty)(0)}(S_1'', t') \tag{3.6}$$

Using LARS semantics of the operators $\triangleright$, $\Box$ and $\boxplus$, this coincides with

$$\mathfrak{M}_\mathcal{I}, S_1'', t' \Vdash \boxplus^{(\infty)(0)} \Box \triangleright \gamma(M_1) \tag{3.7}$$

Furthermore, $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t' \Vdash \gamma(M_2)$ can likewise be substituted by $\mathfrak{M}_\mathcal{I}, S_1'', t' \Vdash \triangleright\gamma(M_2)$. Substituting both the latter and Equation 3.7 in Equation 3.5 yields

$$\begin{aligned} \mathfrak{M}_\mathcal{I}, S_1'', t' \Vdash \triangleright\, \gamma(M_2) \wedge \big(\boxplus^{(\infty)(0)} \Box \triangleright \gamma(M_1)\big), \\ \text{for some } t' \in T_1'', \text{where } S_1'' = \tau^{(b-a)(\infty)}(S_1', t_1) \\ \text{for all } t_1 \in T_1', \text{where } S_1' = \tau^{+b}(S_\mathcal{I}, t)\ . \end{aligned} \tag{3.8}$$

This can be compounded using the semantics of $\Box$, $\Diamond$ and $\boxplus$ into

$$\mathfrak{M}_\mathcal{I}, S_1'', t' \Vdash \boxplus^{+b} \Box\, \boxplus^{(b-a)(\infty)} \Diamond \Big(\triangleright\, \gamma(M_2) \wedge \big(\boxplus^{(\infty)(0)} \Box \triangleright \gamma(M_1)\big)\Big) \tag{3.9}$$

This proves the first part of the required conjunction.

For the second part of the conjunction, we return to Equation 3.2, focusing on the part $\forall t'' \in [t, t'] : \mathcal{I}, t'' \models M_1$. Since $t + a \leq t'$, $[t, t + a] \subseteq [t, t']$. As a result, it is implied that $\mathcal{I}, t_2 \models M_1$ holds for any $t_2 \in [t, t + a]$. Using the assumed induction hypothesis for M_1, we obtain

$$\forall t_2 \in [t, t + a] : \mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t_2 \Vdash \gamma(M_1)$$

With window function τ^{+a}, we obtain $S_2' := \tau^{+a}(S_\mathcal{I}, t)$ with $T_2' = [t, t + a]$. With the semantics of operators $\triangleright$, $\Box$ and $\boxplus$, it follows that

$$\begin{aligned} & \mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t_2 \Vdash \gamma(M_1) \text{ for all } t_2 \in T_2', \text{where } S_2' = \tau^{+a}(S_\mathcal{I}, t) \\ \Leftrightarrow\ & \mathfrak{M}_\mathcal{I}, S_2', t_2 \Vdash \triangleright\gamma(M_1) \text{ for all } t_2 \in T_2', \text{where } S_2' = \tau^{+a}(S_\mathcal{I}, t) \\ \Leftrightarrow\ & \mathfrak{M}_\mathcal{I}, S_2', t \Vdash \Box \triangleright \gamma(M_1), \text{ where } S_2' = \tau^{+a}(S_\mathcal{I}, t) \\ \Leftrightarrow\ & \mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t \Vdash \boxplus^{+a} \Box \triangleright\gamma(M_1) \end{aligned} \tag{3.10}$$

We conclude that both Equation 3.9 and 3.10 are consequences of the original assumption $\mathcal{I}, t \models M_1 \mathcal{U}_{[a,b]} M_2$. With help of the semantics of $\wedge$, it thus follows that

$$\begin{aligned} \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \square \boxplus^{(b-a)(\infty} \Diamond \Big(\triangleright \gamma(M_2) \wedge \big(\boxplus^{(\infty)(0)} \square \triangleright \gamma(M_1) \big) \Big) \\ \wedge \boxplus^{+a} \square \triangleright \gamma(M_1) \end{aligned} \tag{3.11}$$

$\Leftarrow$. For the inverse direction, assume Equation 3.11 holds. First, $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+a} \square \triangleright \gamma(M_1)$ leads to $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t' \Vdash \gamma(M_1)$ for all $t' \in T'$, with $S' = \tau^{+a}(S_{\mathcal{I}}, t)$. With $T' = [t, t+a]$ and the induction hypothesis, one quickly obtains $\forall t' \in [t, t+a]$: $\mathcal{I}, t' \models M_1$.

Consider

$$\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \square \boxplus^{(b-a)(\infty} \Diamond \Big(\triangleright \gamma(M_2) \wedge \big(\boxplus^{(\infty)(0)} \square \triangleright \gamma(M_1) \big) \Big).$$

From LARS semantics, it follows that (a) $\mathfrak{M}_{\mathcal{I}}, S_2'', t'' \Vdash \triangleright \gamma(M_2)$ and (b) $\mathfrak{M}_{\mathcal{I}}, S_2'', t'' \Vdash \boxplus^{(\infty)(0)} \square \triangleright \gamma(M_1)$, for some $t'' \in T_2''$, where $S_2'' = \tau^{(b-a)(\infty)}(S_2', t')$, for all $t' \in T_2'$, where $S_2' = \tau^{+b}(S, t)$. The definitions of T_2' and T_2'' lead to

$$\begin{aligned} &\forall t' \in [t, t+b] : \exists t'' \in [\max(t, t' - (b-a)), t+b] : \\ &\quad \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \models \gamma(M_2) \text{ and} \\ &\quad \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t''' \models \gamma(M_1) \text{ for all } t'' \in T_2''', \text{where } S_2''' = \tau^{(\infty)(0)}(S_2'', t'') \end{aligned} \tag{3.12}$$

In particular with the induction hypothesis, definition $T_2''' = [\max(t, t' - (b - a)), t'']$ and for choice $t' = t + b$, it follows that

$$\exists t'' \in [t+a, t+b] : \mathcal{I}, t'' \models M_2 \text{ and } \forall t''' \in [t+a, t''] : \mathcal{I}, t''' \models M_1 \tag{3.13}$$

As we concluded above that $\forall t' \in [t, t+a] : \mathcal{I}, t' \models M_1$, the interval for t''' can be extended to $[t, t'']$, resulting in

$$\exists t'' \in [t+a, t+b] : \mathcal{I}, t'' \models M_2 \text{ and } \forall t''' \in [t, t''] : \mathcal{I}, t''' \models M_1 \tag{3.14}$$

by DatalogMTL semantics, this is equal to $\mathcal{I}, t \models M_1 \mathcal{U}_{[a,b]} M_2$. □

Lemma 3.4. *Consider interpretation stream $\mathcal{I}$, a time point $t \in \mathbb{N}$ and DatalogMTL metric atom of the form $M_1 \mathcal{S}_{[a,b]} M_2$, where for M_1 and M_2, it holds that $\mathcal{I}', t' \models M_i \Rightarrow \mathfrak{M}_{\mathcal{I}'}, S_{\mathcal{I}'}, t' \Vdash \gamma(M_i)$ for any $\mathcal{I}'$ and t' with $i \in \{1, 2\}$.*

If $\mathcal{I}, t \models M_1 \mathcal{S}_{[a,b]} M_2$, then it holds that

$$\begin{aligned} \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \Big(\boxplus^{b} \square \boxplus^{(\infty)(b-a)} \Diamond \big(\triangleright \gamma(M_2) \wedge \boxplus^{(0)(\infty)} \square \triangleright \gamma(M_1) \big) \Big) \\ \wedge \big(\boxplus^{a} \square \triangleright \gamma(M_1) \big) \end{aligned}$$

and vice versa.

$\mathcal{U}$ and $\mathcal{S}$ are each other's mirror image, i.e. $M_1 \mathcal{U}_{[a,b]} M_2$ stretches out M_1 into the future until M_2 holds, versus $M_1 \mathcal{S}_{[a,b]} M_2$ stretches out M_1 into the past until M_2 is found. As a result, the proof of the induction step for $\mathcal{S}$ is near-identical to that of $\mathcal{U}$.

3.5.2.2 Existential Metric Operators

Next, we consider the $\oplus$ and $\ominus$ operators. Observe that under the DatalogMTL semantics, $\oplus_{[a,b]} M$ is equivalent to $\top \mathcal{U}_{[a,b]} M$, that is, existential future modality can be expressed as an $\mathcal{U}$ formula with a trivially true left-hand side. As a consequence, the correctness argument for $\oplus$ can be obtained from the proof for the $\mathcal{U}$ operator by instantiating M_1 with $\top$ and M_2 with M in Lemma 3.3.

Under this instantiation, all occurrences of $\gamma(\top)$ in the translated LARS formula are trivial, as they are satisfied at every time point. These components therefore do not contribute to the semantic content of the translation and can be safely omitted, yielding a simplified LARS formula for $\oplus$. As a result, the proof of the following lemma follows the same structure as that of Lemma 3.3, with the parts corresponding to M_1 eliminated.

Lemma 3.5. *Consider interpretation stream $\mathcal{I}$, a time point $t \in \mathbb{N}$ and DatalogMTL metric atom of the form $\oplus_{[a,b]} M$, where for M; $\mathcal{I}', t' \models M \Rightarrow \mathfrak{M}_{\mathcal{I}'} S_{\mathcal{I}'}, t' \Vdash \gamma(M)$ for any $\mathcal{I}'$ and t'.*

Then $\mathcal{I}, t \models \oplus_{[a,b]} M$ holds if and only if $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \boxdot \boxplus^{(b-a)(\infty)} \Diamond \triangleright \gamma(M)$.

The full proof of Lemma 3.5 is provided in the Appendix . The case of $\ominus$ is entirely analogous to that of $\oplus$, but concerns existential quantification over the past. Under DatalogMTL semantics, $\ominus_{[a,b]} M$ can be expressed as a $\mathcal{S}$ formula with a trivially true right-hand side. As before, this allows the proof to be obtained by reusing the argument for the corresponding binary temporal operator, omitting vacuous conditions involving $\gamma(\top)$.

Lemma 3.6. *Consider interpretation stream $\mathcal{I}$, a time point $t \in \mathbb{N}$ and DatalogMTL metric atom of the form $\ominus_{[a,b]} M$, where for M, $\mathcal{I}', t' \models M \Rightarrow \mathfrak{M}_{\mathcal{I}'} S_{\mathcal{I}'}, t' \Vdash \gamma(M)$ for any $\mathcal{I}'$ and t'.*

If $\mathcal{I}, t \models \ominus_{[a,b]} M$, then it holds that

$$\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{b} \boxdot \boxplus^{(\infty)(b-a)} \Diamond \triangleright \gamma(M)$$

and vice versa.

3.5.2.3 Universal Metric Operators

Lastly, we consider the $\boxplus$ and $\boxminus$ operators. The approach to the proof remains the same, i.e. converting interval restrictions to window applications on streams and translating temporal quantifications to LARS temporal operators. In contrast

to the existential operators, the $\boxplus$ and $\boxminus$ modalities cannot be reduced to $\mathcal{S}$ or $\mathcal{U}$ with a trivially true argument. Their semantics involves universal quantification over metric intervals, which requires a separate treatment rather than a direct reuse of the binary operator proofs.

Lemma 3.7. *Consider interpretation stream $\mathcal{I}$, a time point $t \in \mathbb{N}$ and DatalogMTL metric atom of the form $\boxplus_{[a,b]}M$, where for M, $\mathcal{I}', t' \models M \Rightarrow \mathfrak{M}_{\mathcal{I}'}S_{\mathcal{I}'}, t' \Vdash \gamma(M)$ for any $\mathcal{I}'$ and t'.*

If $\mathcal{I}, t \models \boxplus_{[a,b]}M$, then it holds that

$$\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box \triangleright \gamma(M)$$

and vice versa.

Proof. Consider an arbitrary interpretation stream $\mathcal{I}$, time point $t \in \mathbb{N}$ and metric atom $\boxplus_{[a,b]}M$ in DatalogMTL.

$\Rightarrow$. First, assume $\mathcal{I}, t \models \boxplus_{[a,b]}M$. Via the semantics of DatalogMTL, this is equivalent to

$$\forall t' \in [t+a, t+b] : \mathcal{I}, t' \models M.$$

Looking towards the past from the time point $t+b$, $\mathcal{I}, t' \models M$ holds over the interval stretching back $b-a$: $[(t+b)-(b-a), t+b]$. Since $(t+b)-(b-a) = t+a \geq t$, $\mathcal{I}, t \models M$ also holds over interval $[\max((t+b)-(b-a), t), t+b]$.

With $t' = t+b$ as a witness, we therefore conclude the following must hold:

$$\exists t' \in [t, t+b] : \forall t'' \in [\max(t'-(b-a), t), t+b] : \mathcal{I}, t'' \models M \tag{3.15}$$

By induction, we obtain:

$$\exists t' \in [t, t+b] : \forall t'' \in [\max(t'-(b-a), t), t+b] : \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M) \tag{3.16}$$

Recall $S_{\mathcal{I}} = (\mathbb{N}, \upsilon_{\mathcal{I}})$ with $\upsilon_{\mathcal{I}}(t) = \{\gamma(M) | M \in \mathcal{I}\}$.

Consider time-based window function τ^{+b}. Then $S' := \tau^{+b}(S_{\mathcal{I}}, t)$ with $S' = (T', \upsilon') = ([t, t+b], \upsilon|_{[t,t+b]})$. As a result, Equation 3.16 is equal to

$$\begin{aligned} \forall t'' \in &[\max(t'-(b-a), t), t+b] : \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M) \\ &\text{for some } t' \in T', \text{where } S' = \tau^{+b}(S_{\mathcal{I}}, t). \end{aligned} \tag{3.17}$$

Subsequently, consider window function $\tau^{(b-a)(\infty)}$. Applied to stream S' at time t', the substream $S'' := \tau^{(b-a)(\infty)}(S', t')$ has $T'' = [\max(t, t'-(b-a)), t+b]$ and $\upsilon'' = \upsilon'|_{T'}$. Equation 3.17 can therefore be written as

$$\begin{aligned} \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M) \text{ for all } t'' \in T'', &\text{where } S'' = \tau^{(b-a)(\infty)}(S', t'), \\ &\text{for some } t' \in T', \text{ where } S' = \tau^{+b}(S_{\mathcal{I}}, t), \end{aligned} \tag{3.18}$$

As structure $\mathfrak{M}_{\mathcal{I}}$ is defined as $\langle S_{\mathcal{I}}, W_{\mathcal{I}}, B_{\mathcal{I}} \rangle$, $S_{\mathcal{I}}$ is the principal stream of $\mathfrak{M}_{\mathcal{I}}$. The operator $\triangleright$ evaluates the subsequent formula on the principal stream of the structure. $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M)$ can thus be rewritten into an evaluation of $\triangleright\gamma(M)$ over any substream of $S_{\mathcal{I}}$. Consider stream S'' for this role. Then LARS semantics allow for rewriting Equation 3.18:

$$\begin{aligned} \mathfrak{M}_{\mathcal{I}}, S'', t'' \Vdash \triangleright\gamma(M) \text{ for all } t'' \in T'' \text{,where } S'' = \tau^{(b-a)(\infty)}(S', t'), \\ \text{for some } t' \in T', \text{where } S' = \tau^{+b}(S_{\mathcal{I}}, t) \end{aligned} \tag{3.19}$$

The universal quantification of t'' over the time interval T'' of stream S'' allows for the introduction of operator $\Box$. As the semantics of $\Box$ are indifferent towards the evaluation time point, it can be chosen freely. Consider t', resulting in

$$\begin{aligned} \mathfrak{M}_{\mathcal{I}}, S'', t' \Vdash \Box \triangleright \gamma(M) \text{ where } S'' = \tau^{(b-a)(\infty)}(S', t'), \\ \text{for some } t' \in T', \text{where } S' = \tau^{+b}(S_{\mathcal{I}}, t). \end{aligned} \tag{3.20}$$

Likewise using LARS semantics of operators $\boxplus$ and $\Diamond$, we obtain

$$\begin{aligned} & \mathfrak{M}_{\mathcal{I}}, S', t' \Vdash \boxplus^{(b-a)(\infty)} \Box \triangleright\gamma(M) \text{ for some } t' \in T', \text{where } S' = \tau^{+b}(S_{\mathcal{I}}, t) \\ \Leftrightarrow \quad & \mathfrak{M}_{\mathcal{I}}, S', t \Vdash \Diamond \boxplus^{(b-a)(\infty)} \Box \triangleright \gamma(M) \text{ where } S' = \tau^{+b}(S_{\mathcal{I}}, t) \\ \Leftrightarrow \quad & \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box \triangleright\gamma(M). \end{aligned}$$

We can conclude $\mathcal{I}, t \models \boxplus_{[a,b]} M$ implies $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box \triangleright\gamma(M)$.

$\Leftarrow$. For the inverse direction, assume $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box \triangleright\gamma(M)$. By the semantics of LARS, we can readily return to Equation 3.18. With $T' = [t, t+b]$ and $T'' = [\max(t, t' - (b-a)), t+b]$, Equation 3.16 also holds. Since $[\max(t, t' - (b-a)), t+b] \subseteq [t+a, t+b]$ for any possible $t' \in [t, t+b]$, we obtain $\forall t' \in [t+a, t+b] : \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t' \Vdash \gamma(M)$, leading to $\forall t' \in [t+a, t+b] : \mathcal{I}, t' \models M$ by the induction hypothesis. This is subsequently equal to $\mathcal{I}, t \models \boxplus_{[a,b]} M$. □

The case of $\boxminus$ is dual to that of $\boxplus$ and follows by an entirely analogous argument, exchanging future-directed reasoning for past-directed reasoning. Since no additional technical complications arise, we omit the proof here.

Lemma 3.8. *Consider interpretation stream $\mathcal{I}$, a time point $t \in \mathbb{N}$ and DatalogMTL metric atom of the form $\boxminus_{[a,b]} M$, where for M, $\mathcal{I}', t' \models M \Rightarrow \mathfrak{M}_{\mathcal{I}'} S_{\mathcal{I}'}, t' \Vdash \gamma(M)$ for any $\mathcal{I}'$ and t'.*

If $\mathcal{I}, t \models \boxminus_{[a,b]} M$, then it holds that

$$\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)} \Box \triangleright\gamma(M).$$

3.5.2.4 Assembling the Inductive Argument

With Lemmas 3.4 through 3.7, we have established the inductive steps required to handle each temporal constructor occurring in DatalogMTL metric atoms. It therefore remains only to verify the base case of the induction, namely DatalogMTL relational atoms, to conclude that the translation preserves satisfaction for all metric atoms. Once this base case is established, the induction extends the correctness of the translation uniformly across the entire syntax of DatalogMTL metric expressions.

Theorem 3.9. *For an interpretation $\mathcal{I}$, a time point $t \in \mathbb{N}$ and metric atom M in DatalogMTL, $\mathcal{I}, t \models M$ if and only if $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \gamma(M)$.*

Proof. Consider arbitrary interpretation $\mathcal{I}$, time point $t \in \mathbb{N}$ and metric atom M. M is constructed recursively via $M := P(\bar{\tau}) | \boxplus_{[a,b]} M' | \boxminus_{[a,b]} M' | \diamondplus_{[a,b]} M' | \diamondminus_{[a,b]} M' | M'_1 \mathcal{U}_{[a,b]} M'_2 | M'_1 \mathcal{S}_{[a,b]} M'_2$.

Consider $M = P(\bar{\tau})$. If $\mathcal{I}, t \models P(\bar{\tau})$, then $P(\bar{\tau}) \in \mathcal{I}(t)$ according to DatalogMTL semantics. By definition of $\mathfrak{M}_{\mathcal{I}}$, $\gamma(P(\bar{\tau})) \in \upsilon_{\mathcal{I}}(t)$. By the semantics of LARS, $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \gamma(P(\bar{\tau}))$. The semantics and definition of $\mathfrak{M}_{\mathcal{I}}$ likewise allow us to deduce that $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \gamma(P(\bar{\tau}))$ implies $\mathcal{I}, t \models M$.

For the remaining forms of M, it suffices to assume the hypothesis holds for M', M'_1 and M'_2. By Lemmas 3.3-3.8, the hypothesis holds for M. Thus by induction, $\mathcal{I}, t \models M$ if and only if $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \gamma(M)$ for any M and vice versa. □

The above proof establishes a direct connection between the languages of DatalogMTL and LARS. In particular, it shows that the metric temporal operators of DatalogMTL can be embedded into LARS. This means that, despite LARS not supporting metric temporal constraints natively, a combination of window operators and the LARS temporal operators allows one to encode metric constraints. The relationship between LARS and DatalogMTL in this manner has been hinted at in [4], but up to this point had not been formally established. The theory above provides a direct translation of DatalogMTL datasets into LARS data streams. Theorem 3.9 is formulated in terms of satisfaction rather than explicit falsity: it states that $\mathcal{I}, t \models M$ if and only if $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \gamma(M)$. For example, consider an interpretation $\mathcal{I}$ such that $\mathcal{I}(t) = \{p\}$ for $t \in [0, 3]$ and $\mathcal{I}(t) = \emptyset$ otherwise; then $\boxplus_{[3,5]} p$ is not satisfied at time 0, since p does not hold at all time points in the interval $[3, 5]$. This allows us to conclude only that $\mathcal{I}, 0 \not\models \boxplus_{[3,5]} p$. Likewise, it can be verified using LARS semantics that the expression $\gamma(\boxplus_{[3,5]} p)$ does not hold at $t = 0$ over $\mathfrak{M}_{\mathcal{I}}$ and $S_{\mathcal{I}}$.

3.5.3 Lifting the Translation from Atoms to Rules and Programs

Next, we lift the correspondence established at the level of individual atoms to full rules and programs. While the preceding results show semantic equivalence for DatalogMTL metric atoms and their LARS counterparts, Stream Reasoning specifications

are expressed as collections of rules evaluated over datasets rather than isolated formulae. To bridge this gap, we first define how a finite DatalogMTL dataset is translated into a LARS data stream, and then show how the atom-level correspondence extends compositionally to rule satisfaction and program semantics.

Definition 3.3. *For a finite DatalogMTL dataset $\mathcal{D}$, we define the corresponding LARS data stream $\gamma(\mathcal{D})$ as stream $(T_\mathcal{D}, \upsilon_\mathcal{D})$, where*

$$T_\mathcal{D} := \left[\inf_{P(\bar{c})@\varrho \in \mathcal{D}} \text{left}(\varrho), \sup_{P(\bar{c})@\varrho \in \mathcal{D}} \text{right}(\varrho) \right],$$
$$\upsilon_\mathcal{D}(t) := \{\gamma(P(\bar{c})) \mid \exists P(\bar{c})@\varrho \in \mathcal{D} : t \in \varrho\}.$$

$\gamma(\mathcal{D})$ only contains extensional predicates, making $\gamma(\mathcal{D})$ a data stream in LARS.

Lemma 3.10. *$\mathcal{I}$ is a model for dataset $\mathcal{D}$ and for all $t \in \mathbb{N}$ and any predicate P in $\mathcal{I}(t)$, but not in $\mathcal{D}$, it holds that P is an intensional predicate if and only if $S_\mathcal{I}$ is an interpretation stream for $\gamma(\mathcal{D})$.*

Proof. Consider a DatalogMTL interpretation $\mathcal{I}$ and dataset $\gamma(\mathcal{D})$.

$\Rightarrow$. Assume $\mathcal{I}$ is a model for $\mathcal{D}$ and for all $t \in \mathbb{N}$ and any predicate P in $\mathcal{I}(t)$ but not in $\mathcal{D}$ is an intensional predicate.

Consider an arbitrary $t \in \mathbb{N}$. By definition, $\upsilon_\mathcal{I}(t) := \{\gamma(P(\bar{c}))|P(\bar{c}) \in \mathcal{I}(t)\}$. Consider an element $\gamma(P(\bar{c}))$ in $\upsilon_\mathcal{I}(t)$. Suppose $\gamma(P(\bar{c}))$ is not in $\upsilon_\mathcal{D}(t)$. Then there is no ϱ for which $t \in \varrho$ and $P(\bar{c})@\varrho$ is not in $\mathcal{D}$. As $P(\bar{c})$ is in $\mathcal{I}(t)$, P must be an intensional predicate.

Furthermore, for arbitrary choice of $t \in \mathbb{N}$ and for every $\gamma(P(\bar{c})) \in \upsilon_\mathcal{D}(t)$, there exists a ϱ for which $t \in \varrho$ and $P(\bar{c})@\varrho \in \mathcal{D}$. Since $\mathcal{I}$ is a model for $\mathcal{D}$, we obtain $\mathcal{I}, t \models P(\bar{c})$. Since $\gamma(P(\bar{c})) \in \upsilon_\mathcal{D}(t), \gamma(P(\bar{c})) \in \upsilon_\mathcal{I}(t)$. As a result, $\upsilon_\mathcal{D}(t) \subseteq \upsilon_\mathcal{I}(t)$, resulting in $\gamma(\mathcal{D}) \subseteq S_\mathcal{I}$. $\gamma(\mathcal{D})$ is a substream of $S_\mathcal{I}$ and any predicate in $S_\mathcal{I}$ but not in $\gamma(\mathcal{D})$ is an intensional predicate, so by definition, $S_\mathcal{I}$ is an interpretation stream for $\gamma(\mathcal{D})$.

$\Leftarrow$. Alternatively, assume $S_\mathcal{I}$ is an interpretation stream. Consider an arbitrary $\mathcal{P}(\bar{c})@\varrho \in \mathcal{D}$. Since $\upsilon_\mathcal{D} = \{\gamma(P(\bar{c}))|\exists P(\bar{c})@\varrho \in \mathcal{D} : t \in \varrho\}$, $P(\bar{c}) \in \upsilon_\mathcal{D}(t)$ for every $t \in \varrho$. With $\upsilon_\mathcal{D}(t) \subseteq \upsilon_\mathcal{I}(t)$, it follows that $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t \Vdash P(\bar{c})$ for every $t \in \varrho$. With the help of Theorem 3.9, $\mathcal{I} \models P(\bar{c})@\varrho$, making $\mathcal{I}$ a model for dataset $\mathcal{D}$. Lastly, the condition on intensional predicates in $\mathcal{I}$ readily follows from the definition of an interpretation stream. □

Lemma 3.10 establishes that the translation γ aligns DatalogMTL datasets with LARS data streams at the level of extensional information, ensuring that interpretations on both sides agree on what counts as given data and what must be derived. Combined with Theorem 3.9, which guarantees truth preservation for individual metric atoms under γ, this provides all ingredients needed to lift the correspondence from

atoms and datasets to rules themselves. We now show that γ preserves the notion of rule satisfaction: a DatalogMTL rule holds in an interpretation if and only if its translated LARS rule holds in the corresponding LARS structure. This result constitutes the central correctness theorem for the translation from DatalogMTL to LARS.

Theorem 3.11. *For an interpretation $\mathcal{I}$, a time point $t \in \mathbb{N}$, dataset $\mathcal{D}$ and rule $r = M' \leftarrow M_1, \wedge \ldots \wedge M_n$ in DatalogMTL, $\mathcal{I}$ is a model for dataset $\mathcal{D}$ and for rule r if and only if structure $\mathfrak{M}_\mathcal{I} = \langle S_\mathcal{I}, W, B_\mathcal{I} \rangle$ is a model of rule $\gamma(r)$ for data stream $\gamma(\mathcal{D}) = (\mathbb{N}, \upsilon_\mathcal{D})$ at every time point $t \in \mathbb{N}$.*

Proof. Consider arbitrary time point $t \in \mathbb{N}$, dataset $\mathcal{D}$, rule $r = M' \leftarrow M_1, \wedge \ldots \wedge M_n$ and interpretation $\mathcal{I}$.

$\Rightarrow$. Assume $\mathcal{I}$ is a model for dataset $\mathcal{D}$ and rule r. $\gamma(\mathcal{D})$ is a datastream and by Lemma 3.10, $S_\mathcal{I}$ is an interpretation stream for $\gamma(\mathcal{D})$.

LARS rule $\gamma(r)$ is equal to $\gamma(M') \leftarrow \gamma(M_1), \ldots, \gamma(M_n)$. Consider an arbitrary $t \in \mathbb{N}$ and assume $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t \Vdash \gamma(M_i)$ for every $i \leq n$. By Theorem 3.9, it follows that $\mathcal{I}, t \models M_i$ for every i. $\mathcal{I}$ is a model for rule r and $B(r) = M_1 \wedge \ldots M_n$. As a result, $\mathcal{I}, t \models M'$. Again using Theorem 3.9, now in the other direction, we obtain $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t \Vdash \gamma(M')$. As a result, $\mathfrak{M}_\mathcal{I}$ is a model for rule $\gamma(r)$ for $\gamma(\mathcal{D})$ at time t.

$\Leftarrow$. Conversely, suppose structure $\mathfrak{M}_\mathcal{I} = \langle S_\mathcal{I}, W, B_\mathcal{I} \rangle$ is a model of rule $\gamma(r)$ for data stream $\gamma(\mathcal{D}) = (T_\mathcal{D}, \upsilon_\mathcal{D})$ at every time point $t \in \mathbb{N}$.

Since $\mathfrak{M}_\mathcal{I}$ is a model for data stream $\gamma(\mathcal{D})$, $S_\mathcal{I}$ is an interpretation stream for $\gamma(\mathcal{D})$ From Lemma 3.10, it follows that $\mathcal{I}$ is a model for $\mathcal{D}$.

Consider rule $r = M' \leftarrow M_1 \wedge \ldots \wedge M_n$ as well as an arbitrary assignment ν and time point $t \in \mathbb{N}$. Assume the body $B(r)$ holds under assignment ν, i.e. for every $i \leq n : \mathcal{I}, t \models^\nu M_i$. By DatalogMTL semantics, this is equal to $\mathcal{I}, t \models \nu(M_i)$ for every $i \leq n$, where $\nu(M_i)$ indicates the application of ν to each of the variables in M_i. Since ν binds all variables, $\nu(M_i)$ are grounded. Via Theorem 3.9, $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t \Vdash \gamma(\nu(M_i))$, i.e. under assignment ν and for every $i \leq n$, $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t \Vdash \gamma(M_i)$ holds. Given the body of rule $\gamma(r)$ is $\gamma(M_1) \wedge \ldots \wedge \gamma(M_n)$ and $\mathfrak{M}_\mathcal{I}$ is a model rule r from program Π on $\gamma(\mathcal{D})$ at t, rule head $\gamma(M')$ must hold over $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t$ under the same assignment, i.e. $\mathfrak{M}_\mathcal{I}, S_\mathcal{I}, t \Vdash \gamma(\nu(M'))$. With Theorem 3.9, we obtain $\mathcal{I}, t \models \nu(M')$, equal to $\mathcal{I}, t \models^\nu M'$. Given the arbitrary choice of ν and t, we conclude that $\mathcal{I}$ is a model for rule r. □

Theorem 3.11 shows that the translation γ is sound and complete with respect to rule satisfaction: a DatalogMTL rule holds in an interpretation over a dataset if and only if its translated LARS rule holds in the corresponding LARS structure over the translated stream. Crucially, this equivalence is established uniformly for all time points and under all variable assignments. The result thus goes beyond a syntactic encoding and establishes γ as a genuine semantic correspondence between DatalogMTL rules and LARS rules, grounded in their respective model-theoretic semantics.

Theorem 3.11 establishes the correctness of the translation γ at the level of individual rules. To complete the semantic correspondence between DatalogMTL and LARS, it remains to lift this result to entire programs and to show that the respective notions of preferred models coincide under the translation.

3.5.4 Preservation of Program Semantics and Minimal Models

We now show that the rule-level correspondence established above extends compositionally to full programs evaluated over datasets. In particular, we establish that γ preserves program satisfaction and aligns the least-model semantics of DatalogMTL with the minimal-model semantics of LARS. As a first step, the following corollary lifts Theorem 3.11 from individual rules to entire programs.

Corollary 3.12. *For an interpretation $\mathcal{I}$, dataset $\mathcal{D}$ and program Π in DatalogMTL, interpretation $\mathcal{I}$ is a model for dataset $\mathcal{D}$ and program Π if and only if structure $\mathfrak{M}_{\mathcal{I}} = \langle S_{\mathcal{I}}, W, B_{\mathcal{I}} \rangle$ is a model of program $\gamma(\Pi)$ for data stream $\gamma(\mathcal{D}) = (T_{\mathcal{D}}, \upsilon_{\mathcal{D}})$ at every time point $t \in \mathbb{N}$.*

Corollary 3.12 lifts the rule-level correspondence of Theorem 3.11 to entire programs, showing that γ preserves program satisfaction pointwise in time. At this point, the two formalisms are aligned with respect to what is true in a given interpretation or structure. However, both DatalogMTL and LARS assign special significance to preferred models: DatalogMTL adopts a least-model semantics, whereas LARS is based on minimal models of interpretation streams. To complete the semantic bridge, it therefore remains to show that γ also preserves this notion of minimality. The following result establishes exactly this correspondence.

Corollary 3.13. *$\mathcal{I}$ is the least model for dataset $\mathcal{D}$ and program Π if and only if $\mathfrak{M}_{\mathcal{I}}$ is the minimal model on program $\gamma(\Pi)$ for data stream $\gamma(\mathcal{D})$ at every time point $t \in \mathbb{N}$.*

Proof. Consider arbitrary dataset $\mathcal{D}$ and program Π. By Corollary 3.12, interpretation $\mathcal{I}$ is a model for dataset $\mathcal{D}$ and program Π if and only if $\mathfrak{M}_{\mathcal{I}} = \langle S_{\mathcal{I}}, W, B_{\mathcal{I}} \rangle$ is a model for program $\gamma(\Pi)$ for data stream $\gamma(\mathcal{D})$ at every $t \in \mathbb{N}$. We proceed to verify the correspondence between least and minimal models.

$\Rightarrow$. First, we assume $\mathcal{I}$ is the least model for Π and $\mathcal{D}$. For a proof by contradiction, additionally assume $\mathfrak{M}_{\mathcal{I}}$ is not a minimal model for some time point t'.

By the definition of a minimal model, it must then hold that there exists a model $\mathfrak{M} = \langle S, W_{\mathcal{I}}, B_{\mathcal{I}} \rangle$ with $S \subsetneq S_{\mathcal{I}}$. For every $t \in \mathbb{N}$, $\upsilon'(t) \subseteq \upsilon_{\mathcal{I}}(t)$. Since $S' \neq S_{\mathcal{I}}$, there exists a $t^* \in T'$ and DatalogMTL relational atom a' for which $\gamma(a') \in \upsilon'(t^*) \setminus \upsilon_{\mathcal{I}}(t^*)$. Define the set $A^* := \{P(\bar{c}) | \gamma(P(\bar{c})) \in \upsilon'(t^*)\}$ and interpretation $\mathcal{I}'$ with $\mathcal{I}'(t^*) := A^*$ and $\mathcal{I}'(t) := \mathcal{I}(t)$ for $t \neq t^*$. It follows directly from Definition 3.2 that $\mathfrak{M}_{\mathcal{I}'} = \mathfrak{M}'$.

Since $\mathfrak{M}'$ is a model on $\gamma(\Pi)$ for $\gamma(\mathcal{D})$ at t^*, $\gamma(\mathcal{D}) \subseteq S'$. As a result, for every element $P(\bar{c})@\varrho \in \mathcal{D}$ and every $t \in \varrho$, $\gamma(P(\bar{c})) \in \gamma(\mathcal{D})(t) \subseteq \upsilon'(t)$. With $\upsilon' =$

$\upsilon_{\mathcal{I}'}$, this implies that $P(\bar{c}) \in \mathcal{I}'(t)$, or alternatively $\mathcal{I}', t \models P(\bar{c})$, holds for the same quantification. As a result $\mathcal{I}'$ is a model for $\mathcal{D}$. Analogous to the proof of Theorem 3.11, we obtain that $\mathcal{I}'$ is a model for any rule r in Π.

$\mathcal{I}$ is a model for $\mathcal{D}$ and Π, but with $\mathcal{I}'(t) \subseteq \mathcal{I}(t)$ for every $t \in \mathbb{N}$, $\mathcal{I}$ is not the least model for $\mathcal{D}$ and Π, a contradiction. We conclude $\mathfrak{M}_{\mathcal{I}}$ must be the minimal model of $\gamma(\Pi)$ for $\gamma(\mathcal{D})$ at every $t \in \mathbb{N}$.

$\Leftarrow$. For the opposite direction, consider minimal model $\mathfrak{M}_{\mathcal{I}}$ on program $\gamma(\Pi)$ for $\gamma(\mathcal{D})$ at every $t \in \mathbb{N}$. Likewise for a proof by contradiction, assume $\mathcal{I}$ is not the least model for $\mathcal{D}$ and Π. Then there exists a model $\mathcal{I}'$ of $\mathcal{D}$ and Π for which $\mathcal{I} \not\subseteq \mathcal{I}'$. In particular, there exists a $t^* \in \mathbb{N}$ and $P(\bar{c})$ for which $P(\bar{c}) \in \mathcal{I}(t^*)$ and $P(\bar{c}) \notin \mathcal{I}'(t^*)$. Since $\mathcal{I}$ is a model of Π and $\mathcal{D}$, it follows that $P(\bar{c}) \notin \mathcal{I}^*(t^*)$, with $\mathcal{I}^* = \bigcap_{\mathcal{I} \in M_{\Pi,\mathcal{D}}} \mathcal{I}$ the least model of Π and $\mathcal{D}$. By Corollary 3.12, $\mathcal{M}_{\mathcal{I}^*}$ is a minimal model on $\gamma(\Pi)$ for $\gamma(\mathcal{D})$ at every $t \in \mathbb{N}$. Looking at interpretation streams $S_{\mathcal{I}}$ and $S_{\mathcal{I}^*}$, it follows from Definition 3.2 that $\gamma(P(\bar{c})) \in \upsilon_{\mathcal{I}}(t^*)$ and $\gamma(P(\bar{c})) \notin \upsilon_{\mathcal{I}^*}(t^*)$. In combination with $\mathcal{I}^*(t) \subseteq \mathcal{I}(t)$ for every $t \in \mathbb{N}$, it follows that $S_{\mathcal{I}^*} \subset S_{\mathcal{I}}$, contradicting the fact that $\mathfrak{M}_{\mathcal{I}}$ is a minimal model. We conclude that $\mathcal{I}$ must be the least model for Π and $\mathcal{D}$. □

3.5.5 Translating DatalogMTL Domain Knowledge – a Working Example

The preceding results establish that γ preserves both truth and minimal-model semantics at an abstract, model-theoretic level. To demonstrate how this correspondence applies to realistic reasoning scenarios, we now return to the running vaccination example and enrich it with additional domain knowledge expressed as DatalogMTL rules. This example illustrates how interacting metric constraints are translated into LARS rules and how the resulting least and minimal models coincide in practice.

Example 3.11. *In addition to the vaccination data considered in the examples above, one can also consider several pieces of expert knowledge. Our available expert knowledge tells us a person is immune and will remain so for 90 days if they have been vaccinated between 3 and 4 weeks ago as well as tested negative. In DatalogMTL, we can write this as*

$$\boxplus_{[0,90]} \mathtt{Immune}(X) \leftarrow \mathtt{NoSympt}(X) \mathcal{S}_{[21,28]} \mathtt{Vaccinated}(X).$$

Furthermore, one can safely infer that a person who has been immune for the past 5 days will test negative;

$$\mathtt{NegTest}(X) \leftarrow \boxminus_{[0,5]} \mathtt{Immune}(X)$$

Together, they form a DatalogMTL program Π_{eg}*. By means of Theorem 3.11, DatalogMTL program* Π_{eg} *induces a LARS program with the two following rules:*

$$\boxplus^{90} \Diamond \boxplus^{(\infty)(90)} \boxdot \triangleright immune(X) \leftarrow \Big(\boxplus^{28} \boxdot \boxplus^{(\infty)(7)} \Diamond \big(\triangleright\, vaccinated(X) \wedge \boxplus^{(0)(\infty)} \boxdot \triangleright noSympt(X) \big) \Big) \wedge \big(\boxplus^{21} \boxdot \triangleright\, noSympt(X) \big)$$

and

$$negTest(X) \leftarrow \boxplus^{5} \Diamond \boxplus^{(\infty)(5)} \boxdot \triangleright immune(X)$$

With data from $\mathcal{D}_{eg}$*,* Π_{eg} *allows us to infer that Ben is immune from day 220 up until day 318. With knowledge on Ben's immunity, we can infer Ben will test negatively on days 226 to 318, should he be tested on these days. As a result, any DatalogMTL interpretation* $\mathcal{I}$ *for which*

$$\begin{aligned} \texttt{Vaccinated}(Ben) &\in \mathcal{I}(199) \\ \texttt{NoSympt}(Ben) &\in \mathcal{I}(t) && \text{for } t \in [181, 320] \\ \texttt{Immune}(Ben) &\in \mathcal{I}(t) && \text{for } t \in [220, 318] \\ \texttt{NegTest}(Ben) &\in \mathcal{I}(t) && \text{for } t \in [226, 318] \end{aligned}$$

will be a model for the pair $(\Pi_{eg}, \mathcal{D}_{eg})$*.*

With the help of Corollary 3.12, the corresponding LARS structure $\mathfrak{M}_{\mathcal{I}}$ *is a model of program* $\gamma(\Pi_{eg})$ *for data stream* $\gamma(\mathcal{D}_{eg})$ *at every time point* $t \in \mathbb{N}$*. Moreover, if the DatalogMTL interpretation* $\mathcal{I}$ *contains only the information outlined above, it can be verified it is the minimal model for* $(\Pi_{eg}, \mathcal{D}_{eg})$*. Following Corollary 3.13,* $\mathfrak{M}_{\mathcal{I}}$ *is likewise minimal.*

3.5.6 Syntactic Restrictions and Translation Size Guarantees

Having established that the translation γ is semantically correct and preserves minimal models, we now briefly consider its syntactic consequences. In particular, we examine how the translation affects the size and structure of the resulting LARS specifications and under which syntactic assumptions this impact remains well controlled.

When translating DatalogMTL metric atoms into LARS formulae, the size of the resulting expressions may grow due to the introduction of multiple window operators and nested temporal modalities. However, this growth is structurally bounded, especially when DatalogMTL programs are assumed to be in a normal form, as commonly adopted in the literature [7, 22]. In normal form, programs contain no nesting of metric operators, and all intervals are either bounded or of the restricted unbounded form with lower bound 0. This significantly limits the syntactic complexity of source programs and ensures that each temporal construct expands into only one layer of

LARS windows and modal operators, rather than recursively generating deeper nesting. Because of this, the translation incurs at most a linear blow-up per metric operator: each atom of the form $\boxplus_{[a,b]}M$, $\boxminus_{[a,b]}M$, $\diamondplus_{[a,b]}M$, $\diamondminus_{[a,b]}M$, $M_1\mathcal{U}_{[a,b]}M_2$ or $M_1\mathcal{S}_{[a,b]}M_2$ expands to a LARS formula where nesting only occurs according to the expressions in Table 3.2. Crucially, since nested temporal operators do not occur in normal-form DatalogMTL programs, this expansion does not cascade. In the absence of normal form, deeply nested metric structures could cause multiplicative size increases across successive translation steps, though still without exponential branching. Normal form therefore provides an important syntactic guarantee that the DatalogMTL-to-LARS translation remains semantically well-defined in practice. By results obtained in Brandt et al. [7], any DatalogMTL program can be transformed to a program in normal form in polynomial time. As a result we can pre-normalise any input program before translation, eliminating metric nesting and restricting interval shapes at only polynomial preprocessing cost. This bounds the per-operator expansion to a single (non-cascading) window pattern in LARS, avoids multiplicative blow-ups across layers, and enables straightforward implementation optimisations (e.g., interval/window normalisation and formula sharing) on the normalised ruleset, yielding more compact outputs without altering semantics.

3.6 Translating LARS into DatalogMTL

This section establishes how LARS specifications can be represented within DatalogMTL. Parts of this correspondence were already implicitly covered in the previous section: Theorems and Corollaries 3.9 through 3.13 address semantic preservation in both translation directions. However, the construction and correctness of the translation from LARS to DatalogMTL have so far only been treated as a byproduct of the forward translation. Several aspects specific to the direction LARS→DatalogMTL therefore remain unaddressed.

3.6.1 Representing LARS Streams and Structures in DatalogMTL

We begin by revisiting the mapping γ introduced in Definition 3.1, which can be viewed as a one-to-one correspondence between constants, variables, and predicate symbols in LARS and DatalogMTL. If a predicate $p \in \mathcal{P}_L$ occurring in a LARS specification does not yet have a corresponding DatalogMTL predicate P such that $\gamma(P) = p$, such a predicate can be introduced into the set $\mathcal{P}_D$. The same assumption applies to constants and variables. Since γ is injective on symbols, it also induces a one-to-one correspondence between DatalogMTL relational atoms and LARS atoms.

To support the inverse translation, we introduce a mapping δ as the inverse of γ. Concretely, $\delta(p(\bar{x})) = P(\bar{\tau})$ whenever $\gamma(P(\bar{\tau})) = p(\bar{x})$.

Analogously to Section 3.5, we now define how LARS data streams and LARS structures are represented within DatalogMTL, starting with the translation of data streams.

Definition 3.4. *For a given LARS data stream $D = (T, \upsilon)$ with $T \subseteq \mathbb{N}$, the corresponding DatalogMTL dataset is defined as*

$$\mathcal{D}_D := \{\delta(p(\bar{c}))@[a,b] | [a,b] \text{ a maximal interval for which } \forall t \in [a,b] : p(\bar{c}) \in \upsilon(t)\}.$$

Example 3.12. *The vaccination and symptom data on Ben is captured in a LARS data stream $D = ([181, 320], \upsilon)$ as*

$$\begin{aligned} \upsilon(199) &= \{vaccinated(Ben), noSympt(Ben)\} \\ \upsilon(t) &= \{noSympt(Ben)\} \text{ for } t[181, 320], t \neq 199 \\ \upsilon(t) &= \emptyset \text{ otherwise.} \end{aligned}$$

Definition 3.4 then induces DatalogMTL dataset

$$\texttt{NoSympt}(Ben)@[181, 320]\}.$$

Note the above mirrors Example 3.10, where a DatalogMTL dataset induces a LARS stream.

Furthermore, we consider the conversion of a LARS structure $\mathfrak{M} = \langle S, W, B\rangle$ to a DatalogMTL interpretation stream $\mathcal{I}_\mathfrak{M}$, which mirrors Definition 3.2.

Definition 3.5. *For a LARS structure $\mathfrak{M} = \langle S, W, B\rangle$ with $S = (T, \upsilon)$, we define a DatalogMTL interpretation stream $\mathcal{I}_\mathfrak{M}$ which captures the same semantic information, as*

$$\mathcal{I}_\mathfrak{M}(t) := \{\delta(p(\bar{c})) | p(\bar{c}) \in \upsilon(t) \text{ or } p(\bar{c}) \in B\}.$$

The range $[t_{\mathcal{I}_\mathfrak{M},\min}, t_{\mathcal{I}_\mathfrak{M},\max}]$ (or $[t_{\mathcal{I}_\mathfrak{M},\min}, +\infty[$) is limited by T; For an atom $\delta(p(\bar{c}))$ to be in $\mathcal{I}(t)$ for a t in the range, $p(\bar{c})$ needs to be in $\upsilon(t)$ or in B. For $\upsilon(t)$ to not be empty, t needs to be in T. As a result, the range is a subset of T. Without loss of generality, we can assume that $[t_{\mathcal{I}_\mathfrak{M},\min}, t_{\mathcal{I}_\mathfrak{M},\max}] = T$, or in case of $t_{\mathcal{I}_\mathfrak{M},\max} = +\infty$, $[t_{\mathcal{I}_\mathfrak{M},\min}, +\infty[= T$.

Example 3.13. *The LARS structure introduced in Example 3.7 consists of the data stream S_{eg}, all time-based and filter windows for W and background knowledge*

$$B = \{man(Ben), healthyLifestyle(Ben)\}.$$

By Definition 3.5, the corresponding interpretation stream is given by

$$\begin{aligned} \mathcal{I}(199) = \{&\texttt{Vaccinated}(Ben), \texttt{NoSympt}(Ben), \\ &\texttt{Man}(Ben), \texttt{HealthyLifestyle}(Ben)\} \\ \mathcal{I}(t) = \{&\texttt{NoSympt}(Ben), \texttt{Man}(Ben), \texttt{HealthyLifestyle}(Ben)\} \\ &\text{for } t \in [181, 320], t \neq 199\ . \end{aligned}$$

The set of windows W from LARS structure $\mathfrak{M}$ is not taken into account, as DatalogMTL has no support for windows. Furthermore, Definitions 3.2 and 3.5 share an inverse relation: translating interpretation $\mathcal{I}$ from DatalogMTL to LARS via Definition 3.2 ($\mathfrak{M}_{\mathcal{I}}$) and from LARS to DatalogMTL via Definition 3.5 results in $\mathcal{I}$ again.

Lemma 3.14. *For any DatalogMTL interpretation $\mathcal{I}$, it holds that $\mathcal{I}_{\mathfrak{M}_{\mathcal{I}}} = \mathcal{I}$. For every LARS structure $\mathfrak{M} = \langle S, W, B\rangle$, $\mathfrak{M}_{\mathcal{I}_{\mathfrak{M}}}$ satisfies the same facts as $\mathfrak{M}$, i.e. $\mathfrak{M}, S, t \Vdash \alpha$ if and only if $\mathfrak{M}_{\mathcal{I}_{\mathfrak{M}}}, S_{\mathcal{I}_{\mathfrak{M}}}, t \Vdash \alpha$ for any LARS atom α.*

Proof. Consider the DatalogMTL interpretation $\mathcal{I}$. The corresponding LARS structure, according to Definition 3.2, is $\mathfrak{M}_{\mathcal{I}}$. If we apply Definition 3.5 to $\mathfrak{M}_{\mathcal{I}}$, we obtain

$$\begin{aligned}\mathcal{I}_{\mathfrak{M}_{\mathcal{I}}}(t) &= \{\delta(p(\bar{c}))|p(\bar{c}) \in \upsilon_{\mathcal{I}}(t) \text{ or } p(\bar{c}) \in B_{\mathcal{I}}\}\\ &= \{\delta(\gamma(P(\bar{\tau})))|P(\bar{\tau}) \in \mathcal{I}(t)\}\\ &= \mathcal{I}(t).\end{aligned}$$

Likewise starting from a LARS structure $\mathfrak{M} = \langle S, W, B\rangle$, Definition 3.5 yields $\mathcal{I}_{\mathfrak{M}}$. When Definition 3.2 is applied to $\mathcal{I}_{\mathfrak{M}}$, we obtain for $S_{\mathcal{I}_{\mathfrak{M}}} = (T_{\mathcal{I}_{\mathfrak{M}}}, \upsilon_{\mathcal{I}_{\mathfrak{M}}})$ where

$$\begin{aligned}T_{\mathcal{I}_{\mathfrak{M}}} &= [t_{\mathcal{I}_{\mathfrak{M}},\min}, t_{\mathcal{I}_{\mathfrak{M}},\max}] = T\\ \upsilon_{\mathcal{I}_{\mathfrak{M}}}(t) &= \{\gamma(P(\bar{\tau})|P(\bar{\tau}) \in \mathcal{I}_{\mathfrak{M}}(t)\}\\ &= \{\gamma(\delta(p(\bar{c}))|p(\bar{c}) \in \upsilon(t) \text{ or } p(\bar{c}) \in B\}.\end{aligned}$$

From the semantics of LARS, it follows that an arbitrary α with $\mathfrak{M}, S, t \Vdash \alpha$, α must be in $\upsilon(t)$ or in B. In both instances, $\gamma(\delta(\alpha))$ is in $\upsilon_{\mathcal{I}_{\mathfrak{M}}}$, resulting in $\mathfrak{M}_{\mathcal{I}_{\mathfrak{M}}}, S_{\mathcal{I}_{\mathfrak{M}}}, t \Vdash \gamma(\delta(\alpha))$.

Note that $T_{\mathcal{I}_{\mathfrak{M}}} = T$ holds even when $t_{\mathcal{I}_{\mathfrak{M}},\max} = +\infty$, give the inclusion of the range in T noted in Definition 3.5.

We can conclude that Definitions 3.2 and 3.5 are inverse translations with regards to semantics of formulae. □

3.6.2 A Translatable Fragment of LARS

The above definitions all consider the translation δ on relational atoms. When considering a translation from LARS formulae to DatalogMTL, a direct problem arises. The logical operators $\neg$, $\wedge$, $\vee$ and $\rightarrow$ have no direct counterpart in DatalogMTL. The approach outlined in Section 3.5 can thus not readily be repeated for the direction LARS→DatalogMTL. Furthermore, DatalogMTL does not support the derivation

Table 3.3: Correspondence Δ of LARS rule heads to DatalogMTL rule heads.

LARS rule head	DatalogMTL rule heads
$\top$	$\{\top\}$
$p(\bar{x})$	$\{\delta(P(\bar{\tau}))\}$
$\alpha_1 \wedge \alpha_2$	$\{\delta(\alpha_1), \delta(\alpha_2)\}$
$\Box\alpha$	$\{\boxplus_{[0,+\infty[}\delta(\alpha), \boxminus_{[0,+\infty[}\delta(\alpha)\}$
$\boxplus^{(b_1)(b_2)}\,\Box\,\alpha$	$\{\boxminus_{[0,b_1]}\delta(\alpha), \boxplus_{[0,b_2]}\delta(\alpha)\}$
$\boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)}\,\Box\,\alpha$	$\{\boxplus_{[a,b]}\delta(\alpha)\}$
$\boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)}\,\Box\,\alpha$	$\{\boxminus_{[a,b]}\delta(\alpha)\}$

of existentially qualified temporal information, such as statements involving $\oplus$ or $\mathcal{S}$ in the rule head. These observations indicate that not all expressivity of LARS can be translated to DatalogMTL.

We therefore restrict attention to a well-defined fragment of LARS, referred to in the following as the *LARS metric fragment*. This fragment is chosen precisely so that each permitted rule head admits a direct and semantics-preserving translation into DatalogMTL; the resulting correspondences are made explicit in Table 3.3.

Definition 3.6. *Consider the LARS metric fragment, which captures rules of the form* $\alpha \leftarrow \beta_1, \ldots, \beta_n$*, where the rule head* α *is of the form*

$$\begin{aligned}\alpha := p(\bar{x}) \mid \Box\, p(\bar{x}) \mid \boxplus^{(b_1)(b_2)}\,\Box p(\bar{x}) \mid \boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)}\,\Box p(\bar{x}) \\ \mid \boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)}\,\Box p(\bar{x}) \mid \alpha \wedge \alpha,\end{aligned}$$

with $p(\bar{x})$ *a LARS atom. Body atoms* β_i *are given by*

$$\begin{aligned}\beta := p(\bar{x}) \mid \Box\, p(\bar{x}) \mid \boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)}\,\Box p(\bar{x}) \mid \boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)}\,\Box p(\bar{x}) \\ \mid \boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond p(\bar{x}) \mid \boxplus^{b} \Box \boxplus^{(\infty)(b-a)} \Diamond p(\bar{x}) \\ \mid \Big(\boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond \big(p_1(\bar{x}) \wedge \boxplus^{(\infty)(0)} \Box\, p_2(\bar{x})\big)\Big) \wedge \big(\boxplus^{+a} \Box p_1(\bar{x})\big) \\ \mid \Big(\boxplus^{b} \Box \boxplus^{(\infty)(b-a)} \Diamond \big(p_2(\bar{x}) \wedge \boxplus^{(0)(\infty)} \Box\, p_1(\bar{x})\big)\Big) \wedge \big(\boxplus^{a} \Box p_1(\bar{x})\big).\end{aligned}$$

Most notably, the LARS formulae in both heads and bodies of the LARS metric fragment do not contain any nesting of temporal constructs; for example, expressions such as $\boxplus^{(b_1)(b_2)}\,\Box\,(\Box a)$ are excluded. Table 3.3 summarises the corresponding DatalogMTL rule heads for each admissible LARS rule head in this fragment. Through the following Sections 3.6.3-3.6.6, we adhere to the LARS metric fragment as defined in Definition 3.6.

3.6.3 Translating LARS Formulae to DatalogMTL Metric Atoms

We reiterate our assumption of a natural timeline $\mathbb{N}$ for DatalogMTL. Under this assumption, evaluations of the form $\mathcal{I}, t \models M$ are only defined for $t \geq 0$. When metric operators range over time points outside $\mathbb{N}$, their evaluation is implicitly restricted to the non-negative part of the timeline.

With slight abuse of notation, we may therefore write $\mathcal{I}, 10 \models \boxminus_{[0,30]}\mathtt{NoSympt}(Ben)$ for an interpretation $\mathcal{I}$ with $\mathcal{I}(t) = \{\mathtt{NoSympt}(Ben)\}$ for all $t \in [0, 15]$. Although $\boxminus_{[0,30]}\mathtt{NoSympt}(Ben)$ nominally requires evaluation of $\mathtt{NoSympt}(Ben)$ over the interval $[-20, 10]$, this reduces to $[0, 10]$ under our assumption of a natural timeline, over which the condition is satisfied. Accordingly, we consider $\mathcal{I}, 10 \models \boxminus_{[0,30]}\mathtt{NoSympt}(Ben)$ to hold.

Definition 3.7. *For a LARS rule head formula α belonging to the LARS metric fragment, we define the mapping Δ from LARS formulae in this fragment to DatalogMTL metric atoms; in particular, when α is a LARS atom occurring in either a rule head or rule body, we define*

$$\Delta(\alpha) = \begin{cases} \{\delta(p(\bar{x}))\} & \text{for } \alpha = p(\bar{x}) \\ \{\boxplus_{[0,+\infty[}\delta(p(\bar{x})), \boxminus_{[0,+\infty[}\delta(p(\bar{x}))\} & \text{for } \alpha = \Box p(\bar{x}) \\ \{\boxplus_{[0,b_2]}\delta(p(\bar{x})), \boxminus_{[0,b_1]}\delta(p(\bar{x}))\} & \text{for } \alpha = \boxplus^{(b_1)(b_2)} \Box\, p(\bar{x}) \\ \{\boxplus_{[a,b]}\delta(p(\bar{x}))\} & \text{for } \alpha = \boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box\, p(\bar{x}) \\ \{\boxminus_{[a,b]}\delta(p(\bar{x}))\} & \text{for } \alpha = \boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)} \Box\, p(\bar{x}) \\ \{\boxdiamond_{[a,b]}\delta(p(\bar{x}))\} & \text{for } \alpha = \boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond\, p(\bar{x}) \\ \{\diamondminus_{[a,b]}\delta(p(\bar{x}))\} & \text{for } \alpha = \boxplus^{b} \Box \boxplus^{(\infty)(b-a)} \Diamond\, p(\bar{x}) \\ \{\delta(p_1(\bar{x}))\mathcal{U}_{[a,b]}\delta(p_2(\bar{x}))\} & \text{for } \alpha = \Big(\boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond\big(p_1(\bar{x}) \wedge \boxplus^{(\infty)(0)} \Box\, p_2(\bar{x})\big)\Big) \wedge \big(\boxplus^{+a} \Box p_1(\bar{x})\big) \\ \{\delta(p_1(\bar{x}))\mathcal{S}_{[a,b]}\delta(p_2(\bar{x}))\} & \text{for } \alpha = \Big(\boxplus^{b} \Box \boxplus^{(\infty)(b-a)} \Diamond\big(p_2(\bar{x}) \wedge \boxplus^{(0)(\infty)} \Box\, p_1(\bar{x})\big)\Big) \wedge \big(\boxplus^{a} \Box p_1(\bar{x})\big) \\ \Delta(\alpha_1) \cup \Delta(\alpha_2) & \text{for } \alpha = \alpha_1 \wedge \alpha_2 \end{cases}$$

Lemma 3.15. *For any LARS structure $\mathfrak{M} = \langle S, W, B\rangle$, a time point $t \in \mathbb{N}$, and a LARS formula α belonging to the LARS metric fragment, $\mathfrak{M}, S, t \Vdash \alpha$ holds if and only if for every DatalogMTL atom $M \in \Delta(\alpha)$, it holds that $\mathcal{I}_{\mathfrak{M}}, t \models M$.*

Proof. Consider an arbitrary LARS structure $\mathfrak{M} = \langle S, W, B\rangle$, time point $t \in \mathbb{N}$ and LARS formula α from the metric fragment.

$\Rightarrow$. Assume $\mathfrak{M}, S, t \Vdash \alpha$.

For α of the form $p(\bar{x})$, $\boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box\, p(\bar{x})$ or $\boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)} \Box\, p(\bar{x})$, it holds that $\alpha = \gamma(M)$ for DatalogMTL metric atom M of the form $P(\bar{x})$, $\boxplus_{[a,b]} P(\bar{x})$ and $\boxminus_{[a,b]} P(\bar{x})$ respectively. By Lemma 3.14, $\mathfrak{M}, S, t \Vdash \alpha$ corresponds to $\mathfrak{M}_{\mathcal{I}_\mathfrak{M}}, S_{\mathcal{I}_\mathfrak{M}}, t \Vdash \alpha$. With this in mind, Theorem 3.9 yields the following result for each of the corresponding α and M:

$$\mathfrak{M}, S, t \Vdash \alpha \Rightarrow \mathcal{I}_\mathfrak{M}, t \models M$$

For each of the considered forms of α, it holds that $\{M\} = \Delta(\alpha)$. As a result, the implication direction $\Rightarrow$ holds for α of the form $p(\bar{x})$, $\boxplus^{+b} \Diamond \boxplus^{(b-a)(\infty)} \Box\, p(\bar{x})$ and $\boxplus^{b} \Diamond \boxplus^{(\infty)(b-a)} \Box\, p(\bar{x})$.

Consider $\alpha = \Box p(\bar{x})$. By the semantics of LARS, $\mathfrak{M}, S, t \Vdash \Box p(\bar{x})$ implies $\mathfrak{M}, S, t' \Vdash p(\bar{x})$ for every $t' \in \mathbb{N}$. By Theorem 3.9, it holds that $\mathcal{I}_\mathfrak{M}, t' \models \delta(p(\bar{x}))$ for every $t' \in \mathbb{N}$.

We subsequently split the quantification over t' into two intervals, 'before t' and 'after t':

$$\mathcal{I}_\mathfrak{M}, t' \models \delta(p(\bar{x})) \text{ for } t' \in [0, t], \text{ (before } t) \tag{3.21}$$
$$\mathcal{I}_\mathfrak{M}, t' \models \delta(p(\bar{x})) \text{ for } t' \in [t, +\infty[, \text{ (after } t) \tag{3.22}$$

By the semantics of DatalogMTL, we can introduce the $\boxplus$ and $\boxminus$ operators.

$$\mathcal{I}_\mathfrak{M}, t \models \boxminus_{[0,+\infty[} \delta(p(\bar{x})) \text{ and}$$
$$\mathcal{I}_\mathfrak{M}, t \models \boxplus_{[0,+\infty[} \delta(p(\bar{x}))$$

Likewise for $\alpha = \boxplus^{(b_1)(b_2)} \Box\, p(\bar{x})$, $p(\bar{x})$ holds for $\mathfrak{M}, S$ over time interval $[\max(0, t-b_1), t+b_2]$. Analogous to Equations 3.21 and 3.22, the quantification can be split over intervals $[\max(0, t-b_1), t]$ and $[t, t+b_2]$. With DatalogMTL semantics, this can be reflected by operators $\boxminus_{[0,b_1]}$ and $\boxplus_{[0,b_2]}$ as follows:

$$\mathcal{I}_\mathfrak{M}, t \models \boxminus_{[0,b_1]} \delta(p(\bar{x})) \text{ and}$$
$$\mathcal{I}_\mathfrak{M}, t \models \boxplus_{[0,b_2]} \delta(p(\bar{x}))$$

Lastly for α a conjunction of formulae $\alpha_1 \dots \alpha_k$, $\mathfrak{M}, S, t \Vdash \alpha$ is equal to $\mathfrak{M}, S, t \Vdash \alpha_i$ for every $i \leq k$. As by Definition 3.6, we can assume each α_i is not a conjunction and therefore is of one of the forms considered above (otherwise, redefine $\alpha_i = \alpha_{i1} \wedge \alpha_{i2}$ to $\alpha_i := \alpha_{i1}$ and $\alpha_{k+1} := \alpha_{i2}$). With the results from above, it follows that the corresponding DatalogMTL atoms hold for each α_i.

$\Leftarrow$. Conversely, assume that $\forall M \in \Delta(\alpha) : \mathcal{I}_\mathfrak{M}, t \models M$.

We consider α of the form $\Box p(\bar{x})$, $\boxplus^{(b_1)(b_2)} \Box p(\bar{x})$ or $\alpha_1 \wedge \alpha_2$. For the remaining forms, the implication follows directly from Theorem 3.9.

Consider $\alpha = \Box p(\bar{x})$. Then $\Delta(\alpha) = \{\boxplus_{[0,+\infty[}\delta(p(\bar{x})), \boxminus_{[0,+\infty[}\delta(p(\bar{x}))\}$. From $\mathfrak{I}_{\mathfrak{M}}, t \models \boxplus_{[0,+\infty[}\delta(p(\bar{x}))$, it follows from Theorem 3.9 that $\mathfrak{M}_{\mathfrak{I}_{\mathfrak{M}}}, S_{\mathfrak{I}_{\mathfrak{M}}}, t \Vdash p(\bar{x})$ for every $t' \in [t, +\infty[$. It then holds that $\mathfrak{M}, S, t' \Vdash p(\bar{x})$ for every $t' \in [t, +\infty[$. Likewise, it follows from $\mathfrak{I}_{\mathfrak{M}}, t \models \boxminus_{[0,+\infty[}\delta(p(\bar{x}))$, it follows that $\mathfrak{M}, S, t' \Vdash p(\bar{x})$ for every $t' \in [0, t]$. Together, we infer that $\mathfrak{M}, S, t' \Vdash p(\bar{x})$ for every $t' \in [0, +\infty[$, which by LARS semantics is equal to $\mathfrak{M}, S, t \Vdash \Box p(\bar{x})$.

Consider $\alpha = \boxplus^{(b_1)(b_2)} \Box p(\bar{x})$. Then $\Delta(\alpha) = \{\boxplus_{[0,b_2]}\delta(p(\bar{x})), \boxminus_{[0,b_1]}\delta(p(\bar{x}))\}$. Similar to $\alpha = \Box p(\bar{x})$, we obtain that $\mathfrak{M}, S, t' \Vdash p(\bar{x})$ holds for t' in intervals $[\max(0, t - b_1), t]$ and $[t, t + b_2]$. As a result, $\mathfrak{M}, S, t' \Vdash p(\bar{x})$ holds over interval $[\max(0, t - b_1), t + b_2]$, which equates to $\mathfrak{M}, S, t \Vdash \boxplus^{(b_1)(b_2)} p(\bar{x})$.

Consider $\alpha = \alpha_1 \wedge \alpha_2$, where the lemma holds for α_1 and α_2. From $\forall H \in \Delta(\alpha) : \mathfrak{I}_{\mathfrak{M}}, t \models H$, it follows that both $\forall H \in \Delta(\alpha_1) : \mathfrak{I}_{\mathfrak{M}}, t \models H$ and $\forall H \in \Delta(\alpha_2) : \mathfrak{I}_{\mathfrak{M}}, t \models H$, which each induce $\mathfrak{M}, S, t \Vdash \alpha_1$ and $\mathfrak{M}, S, t \Vdash \alpha_2$. As a result, $\mathfrak{M}, S, t \models \alpha_1 \wedge \alpha_2$. □

3.6.4 From LARS Rules to DatalogMTL Programs

With translations for both rule heads and rule bodies, we now consider a formal translation of LARS programs into DatalogMTL. We therefore extend the mapping function Δ to LARS rules and sets thereof.

Definition 3.8. *For a LARS rule $r = \alpha \leftarrow \beta_1 \wedge \ldots \wedge \beta_k$, where the rule head α and rule bodies β_i are taken from the LARS metric fragment, we define the set of DatalogMTL rules $\Delta(r)$ as follows:*

$$\Delta(r) := \left\{ H \leftarrow \bigwedge_{b \in B} b \;\middle|\; H \in \Delta(\alpha),\ B = \bigcup_{i \leq k} \Delta(\beta_i) \right\}.$$

For a LARS program P over the LARS metric fragment, we define $\Delta(P)$ as $\bigcup_{r \in P} \Delta(r)$.

Example 3.14. *Consider a LARS program expressing that a patient is considered stable if they have been symptom-free throughout a sliding window of the past 10 days, and that a stable patient is declared recovered once stability has held continuously for 5 additional days:*

$$stable(X) \leftarrow \boxplus^{10} \Box\, noSympt(X)$$
$$recovered(X) \leftarrow \boxplus^{5} \Box\, stable(X)$$

Applying Definition 3.7 and Definition 3.8, this program translates into the following DatalogMTL rules:

$$\texttt{stable}(X) \leftarrow \boxminus_{[0,10]}\texttt{noSympt}(X)$$
$$\texttt{recovered}(X) \leftarrow \boxminus_{[0,5]}\texttt{stable}(X)$$

Theorem 3.16. *For a LARS structure $\mathfrak{M} = \langle S, W, B\rangle$, a time point $t \in \mathbb{N}$, and a data stream D for which S is an interpretation stream, consider a LARS rule r belonging to the LARS metric fragment. If $\mathfrak{M}$ is a model of r for D at every time $t \in \mathbb{N}$, then the DatalogMTL interpretation $\mathcal{I}_{\mathfrak{M}}$ is a model for the DatalogMTL program $\Delta(r)$.*

Proof. We assume the LARS structure $\mathfrak{M} = \langle S, W, B\rangle$ and LARS rule $r = \alpha \leftarrow \beta_1 \wedge \ldots \beta_k$.

Consider an arbitrary DatalogMTL rule $\mathbf{r} = H \leftarrow \bigwedge_{b\in B} b$, from the set $\Delta(r)$. The rule head is given by an element $H \in \Delta(\alpha)$ and the rule body is given by $\bigwedge_{b\in B} b$, with $B = \bigcup_{i\leq k} \Delta(\beta_i)$.

Assume that for interpretation $\mathcal{I}_{\mathfrak{M}}$ and arbitrary $t \in \mathbb{N}$, the rule body of $\mathbf{r}$ holds, i.e. $\mathcal{I}_{\mathfrak{M}}, t \models \bigwedge_{b\in B} b$. Since $B = \bigcup_{i\leq k} \beta_i$, it follows that for every i, every DatalogMTL atom $M \in \Delta(\beta_i)$, $\mathcal{I}_{\mathfrak{M}}, t \models M$. By Lemma 3.15, it follows that $\mathfrak{M}, S, t \Vdash \beta_i$, for every $i \leq k$. $\mathfrak{M}$ is a model of rule r for D at every time $t \in \mathbb{N}$ and $\beta_1, \ldots, \beta_k$ is the body of r, resulting in $\mathfrak{M}, S, t \Vdash \alpha$. By Lemma 3.15, it follows that for every DatalogMTL atom $M \in \Delta(\alpha)$, $\mathcal{I}_{\mathfrak{M}}, t \models M$. We can conclude that $\mathcal{I}_{\mathfrak{M}}$ is a model for DatalogMTL rule $\mathbf{r}$. As $\mathbf{r}$ was arbitrarily chosen from program $\Delta(r)$, it follows that $\mathcal{I}_{\mathfrak{M}}$ is a model for program $\Delta(r)$. □

3.6.5 Preservation of Models and Minimality

Theorem 3.16 shows that the notion of a 'model' is preserved in the translation from the LARS metric fragment to DatalogMTL, and is thereby the direct counterpart to Corollary 3.12 considered in Section 3.5.

Corollary 3.17. *For a LARS structure $\mathfrak{M} = \langle S, W, B\rangle$, time point $t \in \mathbb{N}$, a data stream D for which S is an interpretation stream and a LARS program P from the LARS metric fragment, it holds that if $\mathfrak{M}$ is a model of P for D at every time $t \in \mathbb{N}$, then DatalogMTL interpretation $\mathcal{I}_{\mathfrak{M}}$ is a model for DatalogMTL program $\Delta(P)$.*

Lemma 3.18. *If $\mathcal{I}_{\mathfrak{M}}$ is a model for program $\Delta(P)$ and dataset $\mathcal{D}_D$, then $\mathfrak{M}$ is a model of program P and data stream D at every $t \in \mathbb{N}$.*

Proof. Consider a model $\mathcal{I}_{\mathfrak{M}}$ for program $\Delta(P)$ and dataset $\mathcal{D}_D$.

By Theorem 3.12, $\mathfrak{M}_{\mathcal{I}_{\mathfrak{M}}}$ is a model of program $\gamma(\Delta(P))$ for dataset $\gamma(\mathcal{D}_D)$ for every $t \in \mathbb{N}$. Furthermore, by Definition 3.3 we can deduce the following for $\gamma(\mathcal{D}_D)$. First, the timeline $T_{\mathcal{D}_D}$ is a subset of timeline T of data stream $D = (T, \upsilon)$;

Consider $T_{\mathcal{D}_D} = [\inf_{P(\bar{c})@\varrho} \text{left}(\varrho); \sup_{P(\bar{c})@\varrho} \text{right}(\varrho)]$. If $\sup_{P(\bar{c})@\varrho} \text{right}(\varrho) = +\infty$, then T must be of the form $[a, +\infty[$. Otherwise, given ϱ are closed intervals

over $\mathbb{N}$, the upper bound of $T_{\mathcal{D}_D}$ is given by the upper bound $\text{right}(\varrho)$ of an interval ϱ, for which there exists an element $P(\bar{c}) \in \mathcal{D}_D$. As a result, there exists an element $P(\bar{c})$ for which $P(\bar{c}) \in \upsilon(\text{right}(\varrho))$. As a result, $\text{right}(\varrho) \in T$. Analogously for the lower bound, we obtain that $\text{left}(\varrho) \in T$. With boundaries over $\mathbb{N}$, it follows that $T_{\mathcal{D}_D} \subset T$.

Additionally, the evaluation function $\upsilon_{\mathcal{D}_D}$ coincides with υ. For every $t \in T$, we obtain that

$$\begin{aligned}\upsilon_{\mathcal{D}_D}(t) &= \{\gamma(P(\bar{c}))|\exists P(\bar{c})@[a,b] \in \mathcal{D}_D : t \in [a,b]\} \\ &= \{\gamma(\delta(P(\bar{c})))|p(\bar{c}) \in \upsilon(t)\} \\ &= \upsilon(t).\end{aligned}$$

Without loss of generality, we can extend $T_{\mathcal{D}_D}$ to cover T; for each $t \in T \setminus T_{\mathcal{D}_D}$, we define $\upsilon_{\mathcal{D}_D}(t) = \emptyset$ and therefore equate T with $T_{\mathcal{D}_D}$. As a result, data stream $\gamma(\mathcal{D}_D)$ is equal to $D = (T, \upsilon)$ and $\mathfrak{M}$ is a model of program $\gamma(\Delta(P))$ for dataset D for every $t \in \mathbb{N}$.

It remains to show that $\mathfrak{M}$ is also a model of the original LARS program P for the data stream D at every time point $t \in \mathbb{N}$. To this end, consider an arbitrary LARS rule $r = \alpha \leftarrow \beta_1 \wedge \ldots \wedge \beta_k$ from program P.

Assume the rule body of $r \in P$ is satisfied at time t, i.e. for every $i \leq k, \mathfrak{M}, S, t \Vdash \beta_i$. Then by Lemma 3.15, all metric atoms of each set $\Delta(\beta_i)$ must hold over $\mathcal{I}_{\mathfrak{M}}$ at time t, i.e.

$$\forall M \in \Delta(\beta_i) : \mathcal{I}_{\mathfrak{M}}, t \models M.$$

As a result, $\mathcal{I}_{\mathfrak{M}}$ satisfies the conjunction of all elements in set $\Delta(\beta_i)$ at time t, i.e. $\mathcal{I}_{\mathfrak{M}}, t \models \bigwedge_{M \in \Delta(\beta_i)} M$. Since this holds for every $i \leq k$, the conjunction of these conjunctions also holds:

$$\mathcal{I}_{\mathfrak{M}}, t \models \bigwedge_{i \leq k} \bigwedge_{M \in \Delta(\beta_i)} M$$

Consequently, $\mathcal{I}_{\mathfrak{M}}$ satisfies the rule body $\bigwedge_{i \leq k} \bigwedge_{M \in \Delta(\beta_i)} M$ of rule r at time t.

$\mathcal{I}_{\mathfrak{M}}$ is a model for $\Delta(P)$ and $\mathcal{D}_D$, meaning it is also a model for $\Delta(r)$ and $\mathcal{D}_D$, as $\Delta(r)$ is a subset of $\Delta(P)$. As $\mathcal{I}_{\mathfrak{M}}$ satisfies the rule body of every rule in $\Delta(r)$ at time t, $\mathcal{I}_{\mathfrak{M}}$ must also satisfies the rule head of every rule in $\Delta(r)$. By Definition 3.8, the set of rule heads of $\Delta(r)$ is given by $\Delta(\alpha)$. As such, the combination of $\mathcal{I}_{\mathfrak{M}}$ being a model for $\Delta(P)$ and $\mathcal{D}_D$, $\mathcal{I}_{\mathfrak{M}}$ satisfying the rule over every rule in $\Delta(r)$ and Definition 3.8, we obtain that

$$\mathcal{I}_{\mathfrak{M}}, t \models H \text{ for each } H \in \Delta(\alpha).$$

According to Lemma 3.15, with $\mathcal{I}_{\mathfrak{M}}$ satisfying every DatalogMTL atom in $\Delta(\alpha)$

$$\mathfrak{M}_{\mathcal{I}_\mathfrak{M}}, S_{\mathcal{I}_\mathfrak{M}}, t \Vdash \alpha.$$

Rule head α is satisfied, meaning $\mathfrak{M}$ is a model of rule r for D at time t. Since $r \in P$ and $t \in \mathbb{N}$ were chosen arbitrarily, we conclude $\mathfrak{M}$ is a model of program P for D at every time $t \in \mathbb{N}$. □

Similar to Section 3.5, the notion of minimal models in LARS and least models in DatalogMTL is also preserved.

Corollary 3.19. *Consider a LARS structure $\mathfrak{M} = \langle S, W, B \rangle$, LARS program P and a data stream D for which S is an interpretation stream. If $\mathcal{I}_\mathfrak{M}$ is the least model for dataset $\mathcal{D}_D$ and program $\Delta(P)$, then there exists no model $\mathfrak{M}' = \langle S', W, B \rangle$ on program P for D at every $t \in \mathbb{N}$ with $S' \subsetneq S$.*

Proof. Consider $\mathcal{I}_\mathfrak{M}$ the least model for program $\Delta(P)$ and data stream $\mathcal{D}_D$. By Corollary 3.17, $\mathfrak{M}_{\mathcal{I}_\mathfrak{M}}$ is a model on program P for D at every time $t \in \mathbb{N}$. As a result, $\mathfrak{M}$ is likewise a model of P for D at every $t \in \mathbb{N}$.

Assume there exists a model $\mathfrak{M}' = \langle S', W, B \rangle$ on P for D at every $t \in \mathbb{N}$, with $S' \subsetneq S$. Then $\mathcal{I}_{\mathfrak{M}'}$ is a model for $\Delta(P)$ and $\mathcal{D}_D$. Due to $S' \subsetneq S$, there exists a time point $t^* \in \mathbb{N}$ where $\upsilon'(t^*) \subsetneq \upsilon(t^*)$. By Definition 3.5, $\mathcal{I}'(t^*) \subsetneq \mathcal{I}(t^*)$, contradicting $\mathcal{I}$ as the least model.

□

3.6.6 Translating LARS Domain Knowledge – a Working Example

Example 3.15. *We return to the LARS program and its DatalogMTL translation introduced in Example 3.14, and examine the correspondence between minimal models in LARS and least models in DatalogMTL in more detail.*

Consider a data stream D capturing symptom observations for a patient over time. In LARS, the program of Example 3.14 admits interpretation streams in which intensional predicates, such as stable and recovered, are added exactly at those time points required by the rules. By the definition of LARS semantics, a model must extend the data stream with intensional atoms so that all rules are satisfied, but it should not introduce unnecessary atoms: minimal models are those interpretation streams that are minimal under set inclusion among all such extensions.

In the corresponding DatalogMTL program obtained via Δ, interpretations are constructed over the entire timeline, and satisfaction is defined in terms of metric temporal operators. Here, minimality is expressed through the least-model semantics of DatalogMTL: among all interpretations satisfying the dataset and program, the least model contains exactly those atoms that must hold at each time point as a consequence of the rules, and no others.

Corollary 3.19 shows the DatalogMTL least model property provides minimality conditions on the corresponding LARS model, yet simultaneously highlights a discrepancy between the notions of least model in DatalogMTL and minimal model in LARS; a DatalogMTL least model $\mathcal{I}_\mathfrak{M}$ only

ensures a LARS model $\mathfrak{M}$ does not have a model $\mathfrak{M}'$ that undercuts it across all time points. $\mathfrak{M}$ can however have smaller models on individual time points; For a dataset $D = \{A@[0,1]\}$ and rule $C \leftarrow A$, a structure $\mathfrak{M} = \langle S, W, \emptyset \rangle$ with $S = ([0,1], \upsilon)$ and $\upsilon(0) = \upsilon(1) = \{A, C\}$, it is a model in LARS for time points 0 and 1. It also induces a DatalogMTL least model $\mathcal{I}_{\mathfrak{M}}$ over the corresponding program $\Delta(P)$ and dataset $\mathcal{D}_D$. However, a model $\mathfrak{M}' = \langle S', W, \emptyset \rangle$ with $\upsilon(0) = \{A\}$ and $\upsilon(1) = \{A, C\}$ is a model on time point 0, but is not a model on $t = 1$. As a result, $\mathfrak{M}$ is not a minimal model over every time point.

3.6.7 Expressivity Limitations and Asymmetry of the Correspondence

This subsection completes the bidirectional correspondence by identifying expressivity boundaries of the inverse translation from LARS to DatalogMTL. In particular, we have shown that a substantial and semantically natural fragment of LARS admits a faithful translation into DatalogMTL, preserving both truth and minimal-model semantics.

The metric fragment of LARS considered in this chapter captures only the subset of the language for which a semantics-preserving correspondence with DatalogMTL can be established. In particular, the fragment excludes unrestricted nesting of temporal and window operators, Boolean rule heads, and existential temporal conclusions. These language constructs are not covered by the translation developed in this chapter. Furthermore, the wider range of window operators, i.e. tuple-, partition- and filter-based windows, do not have a direct counterpart in DatalogMTL. Within the identified fragment, however, the translations γ and Δ form a semantic bridge that preserves satisfaction, models, and minimality in both directions.

The results presented identify a sufficient fragment of LARS for which a semantics-preserving translation into DatalogMTL exists, yet do not constitute a complete characterisation of all translatable LARS constructs. For example, the DatalogMTL framework does not have methods for filtering information based on the amount of data tuples or particular predicates. As a result, a proper DatalogMTL counterpart to tuple-, partition- and filter-based LARS windows seems infeasible within the current DatalogMTL framework. Determining an exact delineation of LARS fragments that induce a semantics-preserving translation remains an open question for future work.

Overall, the translation above exposes a degree of overlap between the semantic expressivity of the two frameworks, while simultaneously highlighting important differences between them. DatalogMTL supports reasoning over metric temporal constraints on both discrete and dense timelines, including settings based on $\mathbb{Q}$, whereas LARS provides a richer notion of stream selection through its variety of window operators and stream-oriented evaluation semantics. The results of this chapter therefore establish semantic correspondence only for the fragments explicitly identified above.

They identify a common semantic core for which semantics-preserving translations can be established. The resulting correspondence is inherently asymmetric: every DatalogMTL program considered in this chapter admits a translation into LARS, whereas the converse direction is established only for the LARS metric fragment.

3.7 Discussion and Future Work

The results of this chapter show that DatalogMTL and LARS are not isolated temporal rule languages, but formalisms that admit a precise semantic correspondence. Through the bidirectional translations developed in Sections 3.5 and 3.6, we have established a bridge between metric-temporal reasoning and window-based SR at the level of interpretations and models. In particular, these translations preserve truth at every time point as well as the respective notions of least and minimal models, allowing temporal specifications to be transferred across the two frameworks without changing their intended meaning.

This matters because the literature has largely developed metric-temporal and window-based stream languages as parallel lines of work [1, 14, 23]. Our results replace this informal separation with a semantic characterisation: while DatalogMTL and full LARS differ in their native expressivity, as summarized by Table 3.4, they overlap on a well-defined semantic core on which semantics-preserving translations can be established. From this perspective, the choice between DatalogMTL and LARS is less a choice between incompatible temporal ideas than a choice between different representations of the same underlying temporal behaviour. At the same time, Table 3.4 highlights that the correspondence is inherently asymmetric: DatalogMTL provides native support for metric temporal operators and dense timelines, whereas LARS provides stream-oriented constructs such as tuple-, partition-, and filter-based windows, explicit time-point references, Boolean rule heads, and existential temporal conclusions that lie outside the common translatable fragment. The correspondences established in this work connects DatalogMTL over the natural numbers $\mathbb{N}$ with the LARS metric fragment introduced in Section 3.6.2. This fragment captures the common expressive core of the two languages while leaving several language-specific constructs untranslated. As reflected in Table 3.4, the excluded constructs include tuple-, partition-, and filter-based windows, Boolean rule heads, existential temporal conclusions, and unrestricted temporal operator nesting. These results may serve as foundations to further explore the boundaries of exchangeable expressions, for example translating DatalogMTL over $\mathbb{Q}$ to LARS, or by identifying DatalogMTL counterparts for currently unsupported LARS features such as tuple-, partition- and filter-based windows.

A key conceptual consequence is that window operators need not be treated as primitives outside metric temporal reasoning. Instead, within the logic-programming setting considered here, they can be understood as derived constructs over metric time. This insight clarifies the design space of temporal languages for SR and suggests

opportunities for reusing semantic results across formalisms that have traditionally been studied in isolation. Such reuse is particularly relevant in heterogeneous and decentralised settings, including emerging data space architectures, where reasoning results need to be exchanged across systems with different temporal abstractions.

The running vaccination example illustrates these ideas in a concrete setting. Although the example is deliberately simple, it captures the essential interaction between short-term observations and longer-horizon temporal guarantees and shows how higher-level conclusions can be preserved across formalisms. As such, it serves as a compact witness to the general correspondence established in this chapter.

While this chapter establishes a semantics-preserving correspondence at the theoretical level, practical validation remains essential to assess how these results behave when applied to realistic reasoning tasks. In particular, empirical evaluation is needed to study the effect of the translations when applied to representative DatalogMTL programs and LARS programs from the LARS metric fragment, executed over finite but non-trivial data streams using existing reasoning engines. Such testing is necessary to assess the impact of the translation on program size, reasoning performance, and scalability, and to determine whether the semantic correspondence established here between DatalogMTL and Metric LARS can be exploited effectively in practice.

Several directions remain open. From a theoretical perspective, an immediate question is how far the correspondence can be extended when additional language features are admitted, including negation, stratification, and richer windowing or interval constructs. From a broader SR perspective, it would also be interesting to investigate whether similar semantic translations can be established for other families of formalisms, such as event-based approaches like Event Calculus for Run-Time reasoning (RTEC) [2], in order to expose an even larger common semantic core. Finally, while this chapter focuses on semantic foundations, future work should also investigate how these results can be realised and leveraged in practical reasoning systems.

3.8 Conclusion

This work has established a precise semantic relationship between metric-based and window-based stream reasoning by relating DatalogMTL and LARS at the level of formal semantics. By identifying well-defined fragments of both formalisms, it is shown that temporal entailment and minimal-model semantics can be partially preserved through explicit translations, rather than being inherently tied to a specific temporal abstraction. These results demonstrate that the perceived divide between metric time and windowing stems from modelling choices rather than from fundamental incompatibilities. By making the conditions for semantic correspondence explicit, the chapter provides a principled basis for interoperability between two prominent paradigms in Stream Reasoning and lays the groundwork for subsequent analyses of temporal expressivity and negation.

Table 3.4: Comparative summary of expressibility in DatalogMTL, LARS, and the newly defined LARS metric fragment based on the established translations.

Expressivity aspect	DatalogMTL	LARS	LARS metric fragment
Metric temporal operators	✓	Via translation	✓
Window operators			
Time-based	Via translation	✓	✓
Tuple-based	✗	✓	✗
Partition-based	✗	✓	✗
Filter-based	✗	✓	✗
Explicit time-point reference	✗	✓	✗
Temporal conditions			
Universal	✓	✓	✓
Existential	✓	✓	✓
Temporal operator nesting	✓	✓	✗
Boolean rule heads	✗	✓	✗
Existential temporal conclusions	✗	✓	✗
Time domain expressivity			
$\mathbb{N}$	✓	✓	✓
$\mathbb{Q}$	✓	✗	✗

References

[1] Christian Antić. "Fixed point semantics for stream reasoning". In: *Artificial Intelligence* 288 (2020), p. 103370.

[2] Alexander Artikis, Marek Sergot, and Georgios Paliouras. "Run-time composite event recognition". In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. 2012, pp. 69–80.

[3] Harald Beck, Minh Dao-Tran, and Thomas Eiter. "LARS: A Logic-based framework for Analytic Reasoning over Streams". In: *Artificial Intelligence* 261 (2018), pp. 16–70.

[4] Harald Beck, Minh Dao-Tran, and Thomas Eiter. "LARS: A logic-based framework for analytic reasoning over streams". In: *Artificial Intelligence* 261 (2018), pp. 16–70.

[5] Pieter Bonte et al. "Grounding Stream Reasoning Research". In: *Transactions on Graph Data and Knowledge* 2 (2024), pp. 1–47.

[6] Pieter Bonte et al. "Languages and systems for RDF stream processing, a survey". In: *The VLDB Journal* 34.4 (2025), p. 50.

[7] Sebastian Brandt et al. "Querying log data with metric temporal logic". In: *Journal of Artificial Intelligence Research* 62 (2018), pp. 829–877.

[8] Andrea Colombo et al. "Smart derivative contracts in datalogMTL". In: *Bank of Italy: Markets, Infrastructures, Payment Systems* 37 (2023).

[9] Emanuele Della Valle et al. "It's a streaming world! Reasoning upon rapidly changing information". In: *IEEE Intelligent Systems* 24.6 (2009), pp. 83–89.

[10] Thomas Eiter, Paul Ogris, and Konstantin Schekotihin. "A distributed approach to LARS stream reasoning (system paper)". In: *Theory and Practice of Logic Programming* 19.5-6 (2019), pp. 974–989.

[11] European Commission. *Proposal for a Regulation of the European Parliament and of the Council on the European Health Data Space*. COM(2022) 197 final. May 3, 2022.

[12] Ahlem Chérifa Khadir, Hassina Aliane, and Ahmed Guessoum. "Ontology learning: Grand tour and challenges". In: *Computer Science Review* 39 (2021), p. 100339. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100339.

[13] Ron Koymans. "Specifying real-time properties with metric temporal logic". In: *Real-time systems* 2.4 (1990), pp. 255–299.

[14] Nicola Leone et al. *A Formal Comparison between Datalog-Based Languages for Stream Reasoning*. Extended version. 2022. arXiv: 2208.12726 [cs.AI].

[15] Victor W. Marek and Miroslaw Truszczyński. "Stable models and an alternative logic programming paradigm". In: *The logic programming paradigm: A 25-year perspective*. Springer, 1999, pp. 375–398.

[16] Wafaa Mebrek and Amel Bouzeghoub. "Multi-Context Incremental Reasoning over Data Streams". In: *2023 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE. 2023, pp. 150–157.

[17] Mathijs van Noort, Pieter Bonte, and Femke Ongenae. "Navigating Streams: A Survey on 15 Years of Advances in Stream Reasoning Formalisms". Manuscript submitted for publication to the Semantic Web Journal. Jan. 2026.

[18] Mathijs van Noort and Przemysław Andrzej Wałęga. "Efficient Temporal Reasoning with Non-Temporal Engines: Embedding DatalogMTL into Datalog". In: *Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning (KR 2026)*. Accepted for publication. 2026.

[19] Ruben Verborgh. *Let's talk about pods*. Blog post. Dec. 2022. URL: https://ruben.verborgh.org/blog/2025/08/12/inside-the-insight-economy/.

[20] W3C Solid Community Group. *Solid Protocol*. Tech. rep. World Wide Web Consortium (W3C), 2021.

[21] Przemysław Andrzej Wałęga. "Reasoning about Time in DatalogMTL: Course Notes". In: *Joint Proceedings of the 20th and 21st Reasoning Web Summer Schools (RW 2024 and RW 2025)*. Ed. by Alessandro Artale et al. Vol. 138. Open Access Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025, 9:1–9:23. DOI: 10.4230/OASIcs.RW.2024/2025.9.

[22] Przemysław Andrzej Wałęga et al. "DatalogMTL: computational complexity and expressive power". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2019, pp. 1886–1892.

[23] Dingmin Wang et al. "Meteor: Practical reasoning in datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.5 (2022), pp. 5906–5913.

[24] Kaiyue Zhao et al. "Incremental Maintenance of DatalogMTL Materialisations". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 40.23 (Mar. 2026), pp. 19467–19476. DOI: 10.1609/aaai.v40i23.39025.

Appendix

Lemma 3.5. *Consider interpretation stream $\mathcal{I}$, a time point $t \in \mathbb{N}$ and DatalogMTL metric atom of the form $\oplus_{[a,b]} M$, where for M, $\mathcal{I}', t' \models M \Rightarrow \mathfrak{M}_{\mathcal{I}'} S_{\mathcal{I}'}, t' \Vdash \gamma(M)$ for any $\mathcal{I}'$ and t'.*

Then $\mathcal{I}, t \models \oplus_{[a,b]} M$ holds if and only if $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \boxdot \boxplus^{(b-a)(\infty)} \Diamond \triangleright \gamma(M)$.

Proof. Consider an arbitrary interpretation stream $\mathcal{I}$, time point $t \in \mathbb{N}$ and metric atom $\oplus_{[a,b]} M$ in DatalogMTL.

$\Rightarrow$. First, assume $\mathcal{I}, t \models \oplus_{[a,b]} M$. Via the semantics of DatalogMTL, this is equivalent to

$$\exists t'' \in [t+a, t+b] : \mathcal{I}, t'' \models M$$

Consider an arbitrary time point t' in interval $[t, t+b]$. Since $t \leq t+a$ and $t' - (b-a) \leq t+a$, it follows that $[t+a, t+b] \subseteq [\max(t, t'-(b-a)), t+b]$. In particular, $t'' \in [\max(t, t'-(b-a)), t+b]$ for every $t' \in [t, t+b]$, resulting in

$$\begin{aligned} \forall t' \in [t, t+b] : &\exists t'' \in [\max(t, t'-(b-a)), t+b] : \\ &\mathcal{I}, t'' \models M \end{aligned} \tag{3.23}$$

By induction,

$$\begin{aligned} \forall t' \in [t, t+b] : &\exists t'' \in [\max(t, t'-(b-a)), t+b] : \\ &\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M). \end{aligned} \tag{3.24}$$

Consider window function τ^{+b}.Without loss of generality, assume $S_{\mathcal{I}} = (T, \upsilon_{\mathcal{I}})$ with $T \supseteq [t, t+b]$.[1] Then $S' := \tau^{+b}(S_{\mathcal{I}}, t)$ with $S' = (T', \upsilon') = ([t, t+b], \upsilon_{\mathcal{I}}|_{[t,t+b]})$. Equation 3.24 can thus be written as

$$\begin{aligned} &\exists t'' \in [\max(t, t' - (b-a)), t+b] : \\ &\quad \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M) \text{ for all } t' \in T', \\ &\quad \text{where } S' = \tau^{+b}(S, t). \end{aligned} \tag{3.25}$$

Subsequently, consider window function $\tau^{(b-a)(\infty)}$. Applied to stream S' at time t', the substream $S'' := \tau^{(b-a)(\infty)}(S', t')$ has $T'' = [\max(t, t' - (b-a)), t+b]$ and $\upsilon'' = \upsilon'|_{T'}$. Equation 3.25 can therefore be written as

$$\begin{aligned} &\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M) \text{ for some } t'' \in T'', \\ &\qquad \text{where } S'' = \tau^{(b-a)(\infty)}(S', t'), \\ &\qquad \text{for all } t' \in T', \text{where } S' = \tau^{+b}(S, t). \end{aligned} \tag{3.26}$$

As structure $\mathfrak{M}_{\mathcal{I}}$ is defined as $\langle S_{\mathcal{I}}, W_{\mathcal{I}}, B_{\mathcal{I}} \rangle$, $S_{\mathcal{I}}$ is the principal stream of $\mathfrak{M}_{\mathcal{I}}$. As the operator $\triangleright$ evaluates the subsequent formula on the principal stream of the structure, $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t'' \Vdash \gamma(M)$ can thus be rewritten into an evaluation of $\triangleright\gamma(M)$ over any substream of $S_{\mathcal{I}}$. Consider stream S'' for this role. Then LARS semantics allow for rewriting of Equation 3.26:

$$\begin{aligned} &\mathfrak{M}_{\mathcal{I}}, S'', t'' \Vdash \triangleright\gamma(M) \text{ for some } t'' \in T'', \\ &\qquad \text{where } S'' = \tau^{(b-a)(\infty)}(S', t'), \\ &\qquad \text{for all } t' \in T', \text{where } S' = \tau^{+b}(S, t) \end{aligned} \tag{3.27}$$

The existential quantification of t'' over the time interval T'' of stream S'' allows for the introduction of operator $\Diamond$. As the semantics of $\Diamond$ are indifferent towards the evaluation time point, it can be chosen freely. Consider t', resulting in

$$\begin{aligned} &\mathfrak{M}_{\mathcal{I}}, S'', t' \Vdash \Diamond \triangleright \gamma(M) \\ &\qquad \text{where } S'' = \tau^{(b-a)(\infty)}(S', t'), \\ &\qquad \text{for all } t' \in T', \text{where } S' = \tau^{+b}(S, t), \end{aligned} \tag{3.28}$$

Likewise using LARS semantics of operators $\boxplus$ and $\Box$, we obtain

[1] If $T \not\supseteq [t, t+b]$, expand T to $T \cap [t+b]$ and $\upsilon_I(t) = \emptyset$ where previously undefined.

$$\begin{aligned} & \mathfrak{M}_{\mathcal{I}}, S', t' \Vdash \boxplus^{(b-a)(\infty)} \Diamond \triangleright \gamma(M) \text{ for all } t' \in T', \\ & \qquad \text{where } S' = \tau^{+b}(S_{\mathcal{I}}, t) \\ \Leftrightarrow \quad & \mathfrak{M}_{\mathcal{I}}, S', t \Vdash \Box \boxplus^{(b-a)(\infty)} \Diamond \triangleright \gamma(M) \\ & \qquad \text{where } S' = \tau^{+b}(S_{\mathcal{I}}, t) \\ \Leftrightarrow \quad & \mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond \triangleright \gamma(M) \end{aligned}$$

$\Leftarrow$. Assume $\mathfrak{M}_{\mathcal{I}}, S_{\mathcal{I}}, t \Vdash \boxplus^{+b} \Box \boxplus^{(b-a)(\infty)} \Diamond \triangleright \gamma(M)$. The LARS semantics trace back to Equation 3.26. The definition of T' and T'' and the induction hypothesis yield $\forall t' \in [t, t+b] : \exists t'' \in [\max(t, t'-(b-a)), t+b] : \mathcal{I}, t'' \models M$. The choice $t' = t+b$ results in $\exists t'' \in [t+a, t+b] : \mathcal{I}, t'' \models M$, equal to $\mathcal{I}, t \models \diamondplus_{[a,b]} M$. $\square$

4

Efficient Temporal Reasoning with Non-Temporal Engines: Embedding DatalogMTL into Datalog

Expressive temporal Stream Reasoning (SR) semantics offer limited value if they cannot be realised within the reasoning infrastructure used in practice. As formulated in Research Challenge C3, real-world deployments in domains such as healthcare, Internet-of-Things (IoT), and distributed data processing rely heavily on mature, non-temporal reasoning engines that prioritise robustness, scalability, and long-term maintainability. Requiring the adoption of specialised temporal engines in such settings is often infeasible, regardless of the expressive power of the underlying formalism. This motivates the question of whether expressive temporal reasoning can be supported without replacing existing infrastructure.

Focusing on DatalogMTL, this chapter addresses Research Question RQ4 by identifying a precise fragment of the language that admits a semantics-preserving embedding into plain Datalog, thereby substantiating Hypothesis H6. The translation preserves temporal entailment and minimal-model semantics by treating temporal reasoning as a transformation problem rather than a system-level replacement. To evaluate the practical viability of this approach, the embedding is implemented and executed using the Nemo Datalog reasoning engine, allowing temporal specifications to benefit directly from existing optimisation and execution mechanisms. The experimental evaluation demonstrates that the identified fragment retains

sufficient expressive power for representative reasoning tasks, thereby supporting Hypothesis H7 and showing that expressive temporal semantics can be realised on non-temporal infrastructure in a practically meaningful way.

⋆ ⋆ ⋆

M. van Noort, and P. A. Wałęga

Accepted for publication in Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning (KR2026), April 2026.

4.1 Abstract

We present a translation of DatalogMTL, a temporal extension of the Datalog rule language with operators from metric temporal Logic (MTL), into standard, non-temporal Datalog with arithmetics. As we prove, the translation preserves such key semantic properties as entailment, consistency, and finiteness of materialisability. As a result, our approach yields a faithful embedding of DatalogMTL into classical Datalog, enabling complex temporal reasoning tasks to be executed without the need for specialised temporal reasoning engines. We implement and evaluate our translation using three state-of-the-art Datalog systems: Nemo, EYE and Eyelet. Remarkably, non-temporal Datalog engines are able to achieve comparable performance, and in some cases outperform dedicated temporal reasoning engines, demonstrating the practical viability and efficiency of our approach.

4.2 Introduction

Temporal reasoning is central in modern data-intensive systems, where information evolves continuously and analytical systems must detect complex temporal relationships. Domains such as financial monitoring, cybersecurity, transportation analysis, and IoT ecosystems frequently require identifying temporally extended patterns, e.g. recurring behaviours, suspicious event sequences, or causal chains unfolding over time. In many of these settings, temporal constraints must be articulated explicitly and in an interpretable manner, making rule-based temporal languages an attractive option. Yet, building temporal reasoning engines remains a demanding engineering challenge, especially in the presence of dense time, data stamped with intervals, and non-trivial temporal modalities, while maintaining correct semantics and practical efficiency. Temporal engines tend to be significantly less mature than widely deployed non-temporal systems. This mismatch between the practical demand for

temporal reasoning and the limited maturity of temporal systems motivates our exploration of methods that allow for temporal reasoning by leveraging existing and well-optimised non-temporal systems.

In this paper we focus on DatalogMTL [19]—an extension of the state-of-the-art rule-based language Datalog [1] with Metric Temporal Logic (MTL) operators. It provides a versatile temporal reasoning language with applications in areas such as Semantic Web (SW), SR, and Knowledge Graphs. Among others, DatalogMTL has been applied in ontology-based data access [18], economic fact-checking [33], human movement description [40], and verification of banking agreements [36].

To illustrate capabilities of DatalogMTL, consider a simplified banking system that monitors transactions to detect suspicious behaviour. The system logs transactions of the form $Trans(acct1054, acct0321)@11023$, indicating that at timestamp 11023 a transfer was recorded from account $acct1054$ to account $acct0321$. To prevent money laundering, the bank aims to detect transaction loops, where money is sent from one account to another and then returned shortly after. This can be expressed in DatalogMTL using the following rules:

$$\boxplus_{[0,14]} Susp(x) \leftarrow Trans(y, x) \wedge \diamondminus_{[0,5]} Trans(x, y), \tag{4.1}$$

$$RedList(x) \leftarrow \boxminus_{[0,30]} Susp(x). \tag{4.2}$$

Rule (4.1) flags an account x as suspicious continuously for the next interval of 14 days (expressed with $\boxplus_{[0,14]} Susp(x)$) when x receives a transaction from another account y ($Trans(y, x)$) and x had previously sent money to y sometime within the past 5 days ($\diamondminus_{[0,5]} Trans(x, y)$). Rule (4.2), in turn, places x on the red list ($RedList(x)$) if x has been continuously flagged as suspicious for the past 30 days ($\boxminus_{[0,30]} Susp(x)$).

DatalogMTL offers a good balance between the expressive power and computational complexity of reasoning [13, 47, 49], making it a versatile tool for a wide range of temporal reasoning applications. Consequently, research on DatalogMTL has increasingly focused on developing practical reasoning algorithms. This has led to several systems employing diverse reasoning techniques, including ontology-based rewritings [29], automata-theoretic constructions [51], materialisation-based approaches [14, 15], saturation procedures [48], semi-naïve evaluations [52], and magic set rewritings [53].

However, all these approaches require sophisticated mechanisms tailored to the complex setting of DatalogMTL. Developing such systems is challenging, prone to errors, and the resulting tools often suffer from limited maintainability. This stands in contrast to standard (non-temporal) Datalog systems, which are mature, robust, and used in industry. They are maintained and optimised by professional developers, and offered as reliable commercial products [14, 34].

This leads us to the following research questions:

- Can the complex task of temporal reasoning in DatalogMTL be effectively performed using existing, non-temporal Datalog engines?

- If so, which theoretical properties are preserved by such an approach, and how does its practical performance compare to that of dedicated DatalogMTL reasoners?

We address these questions and show a translation to Datalog that provides us with a practical tool for reasoning in DatalogMTL. Our main contributions are as follows:

- In Section 4.5 we construct a translation of a pair of a DatalogMTL program Π and dataset $\mathcal{D}$ into a Datalog (with arithmetic operators) program $\mathbf{tr}(\Pi)$ and a set of Datalog facts $\mathbf{tr}(\mathcal{D})$.

- We also establish a correspondence between materialisation steps in DatalogMTL and in the translated Datalog setting. Materialisation terminates for Π and $\mathcal{D}$ if and only if it terminates for $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$. Moreover, consistency of Π and $\mathcal{D}$ implies consistency of $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$, whereas the converse holds whenever Π and $\mathcal{D}$ are finitely materialisable.

- In Section 4.6 we turn to practical aspects of the translation. We show how it can be adapted to the N3 triple-store format used by engines such as EYE.

- Then, we provide implementations of the translations based on Datalog and on the N3 triple formats. We compare the performance of the state-of-the-art DatalogMTL reasoner MeTeoR with our translated programs executed in the Datalog engines Nemo and Eyelet as well as in the N3 reasoner EYE. As we show, in several cases Nemo outperforms MeTeoR.

Our results indicate that, instead of developing dedicated DatalogMTL reasoners from scratch, a promising alternative is to translate DatalogMTL programs into the standard Datalog setting and exploit existing, optimised reasoning engines. This approach not only simplifies implementation but also enables seamless integration of temporal and non-temporal reasoning within a single reasoning framework.

4.3 Related Work

The extension of traditional reasoning with temporal dimension has been a topic of continued research. Following early work on static knowledge representation and reasoning, several frameworks have incorporated time into established rule-based or logic-based paradigms. Examples include temporal extensions of Answer Set Programming (ASP), such as C-ASP [38] and PrASP [35]. Within the SW community, significant efforts have focused on extending SPARQL to support streaming data, leading to formalisms such as RSP-QL [26], C-SPARQL [7], CQELS [39], and EP-SPARQL [2]. These systems typically rely on windowing mechanisms: temporal data is divided into bounded subsets that can then be processed by non-temporal engines.

For example, DIVIDE [24] relies on internal RDF reasoners like RDFox [34], EYE [25] uses a Prolog reasoner, and some others employ CEL DL reasoner [32].

A well-known limitation of window-based SR is that discarding the full temporal dimension prevents these systems from recognising extended temporal relations or long-range dependencies, constraining their ability to detect and analyse complex patterns, often referred to as Complex Event Processing (CEP). This reflects a broader trade-off: such approaches benefit from established non-temporal engines but lose expressive power for temporal pattern recognition. Another challenge in the SR landscape is the absence of a unifying semantic framework. Existing formalisms differ substantially in how they interpret windows, timestamps, incremental updates, or the interaction between temporal and logical operators. Some adopt windows as preprocessing steps that generate static sub-datasets, while others integrate temporal modalities directly into rule languages, yielding different expressive capabilities and evaluation behaviours. As a consequence, these systems often lack interoperability, complicate comparisons across formalisms, and make it difficult to establish common benchmarks or correctness guarantees.

An alternative line of work integrates temporal reasoning natively into rule-based systems. Examples include LARS [10], DatalogMTL, and RTEC [5], which incorporate additional temporal operators. These approaches require dedicated implementations, such as MeTeoR [51], Ticker [9], and Laser [8]. Temporal reasoning has also been explored in temporal description logics [4, 6, 27, 42] and other temporal variants of Datalog [3, 20, 30].

The line of research on DatalogMTL has provided an in-depth theoretical analysis. Its computational complexity has been established for both data and combined complexity [19, 47, 49]. Alternative semantics have been developed, including event-based semantics [41] and integer-based timelines [44, 45]. Further extensions introduce non-monotonic negation [23, 46], existential rules [31], and temporal aggregation [12]. Finite materialisability, including developing algorithms for its checking and establishing computational complexity of this problem, have been studied for DatalogMTL [43, 44]. DatalogMTL has also been studied in the context of inconsistency handling [17] and temporal rule mining [50]. Applications span smart contracts [22, 36, 37], crypto-activity analysis [21], economic fact-checking [33], and human movement description [40].

Several practical reasoning approaches for DatalogMTL have emerged, including extensions of existing platforms such as Ontop [29] and Vadalog [15]. Standalone implementations include systems based on automata [51], saturation procedures [48], semi-naïve evaluation [52], and magic-set rewriting [53]. A dedicated generator, iTemporal [11], provides benchmarks and synthetic datasets for temporal Datalog reasoning.

4.4 Background

We will introduce syntax and semantics of Datalog$_\mathbb{Q}$, that is, Datalog with arithmetics over rationals, as well as of DatalogMTL, that is, Datalog with metric temporal operators.

Datalog$_\mathbb{Q}$. For Datalog$_\mathbb{Q}$ we assume a two-sorted language, where each constant and variable is either of an *object* or a *numeric* sort. *Object terms* are object constants and variables, whereas *numeric terms* are defined inductively as numeric constants (values in $\mathbb{Q} \cup \{-\infty, +\infty\}$), variables, and arithmetic expressions $|t|, t + t', t - t', t \times t', t/t'$, for any numeric terms t and t'. We will also use $\max(t_1, \ldots, t_k)$ and $\min(t_1, \ldots, t_k)$, which are definable using the other functions, for example, $\max(t, t') = (t + t' + |t - t'|)/2$. *Relational atoms* are of the form $P(\mathbf{s})$, for P a predicate and $\mathbf{s}$ a tuple of terms with arity matching P. *Comparison atoms* are of the forms $(t = t')$ and $(t \leq t')$, for t and t' being numeric terms. A Datalog$_\mathbb{Q}$ *rule* is of the form $A' \leftarrow A_1 \wedge \ldots \wedge A_n$, where the *rule head* A', and each *rule body atom* A_i is a (relational or comparison) atom, $\top$, or $\bot$. The conjunction $A_1 \wedge \cdots \wedge A_n$ is called the *rule body*. A Datalog$_\mathbb{Q}$ *program*, Λ, is a finite set of Datalog$_\mathbb{Q}$ rules. An expression (term, atom, rule, etc.) is *ground* if it does not mention variables.

A Datalog$_\mathbb{Q}$ *interpretation*, I, is a (possibly infinite) set of ground relational atoms. In each interpretation, we let $I \models \top$ and $I \not\models \bot$. For a ground relational atom A we let $I \models A$ if $A \in I$ after the standard evaluation of arithmetic operations in A. For a ground comparison atom, we let $I \models (t = t')$ or $I \models (t \leq t')$, if after evaluation of arithmetic operations, the (in-)equality holds over the rational numbers. An interpretation I is a *model* of a rule $A' \leftarrow A_1 \wedge \cdots \wedge A_n$, if for every *grounding* ν (a mapping of variables to constants of the same sort), $I \models \nu(A_i)$ for all $i \in \{1, \ldots, n\}$ implies that $I \models \nu(A')$. Interpretation I is a *model* of a program Λ if it is a model of all rules in Λ, and Λ is *consistent* if it has a model. Furthermore, Λ *entails* a ground atom A, in symbols $\Lambda \models A$, if $I \models A$ in every model I of Λ.

A procedural counterpart of the above semantics is defined by means of the *immediate consequence operator* T_Λ, for a program Λ. We let T_Λ be a partial function which maps an interpretation I to the unique subset-minimal interpretation $T_\Lambda(I)$ if one exists (otherwise it is undefined), such that $I \subseteq T_\Lambda(I)$ and for any rule $A' \leftarrow A_1 \wedge \cdots \wedge A_n$ in Λ and any grounding ν, the fact that $I \models \nu(A_i)$ for all $i \in \{1, \ldots, n\}$ implies that $T_\Lambda(I) \models \nu(A')$. Note that $T_\Lambda(I)$ is undefined precisely when Λ contains a rule with $\bot$ in the head and with body satisfied in I. We let $T_\Lambda^{k+1}(I) = T_\Lambda\big(T_\Lambda^k(I)\big)$, for $k \in \mathbb{N}$ and $T_\Lambda^\omega(I) = \bigcup_{k \in \mathbb{N}} T_\Lambda^k(I)$. The *canonical interpretation*, $\mathcal{C}_\Lambda$, is defined as $T_\Lambda^\omega(\emptyset)$. It can be shown that Λ is consistent if $\mathcal{C}_\Lambda$ is well-defined; moreover Λ entails A, if Λ is inconsistent or $\mathcal{C}_\Lambda \models A$. We say that Λ is *finitely materialisable* if there is $k \in \mathbb{N}$

for which $T^k_\Lambda(\emptyset) = \mathcal{C}_\Lambda$. Similarly, a pair consisting of a program Λ and an interpretation I is consistent if $T^\omega_\Lambda(I)$ is well-defined, and the pair is finitely materialisable if $\mathcal{C}_\Lambda(I) \coloneqq T^\omega_\Lambda(I) = T^k_\Lambda(I)$, for some $k \in \mathbb{N}$.

DatalogMTL. In contrast to Datalog$_\mathbb{Q}$, the language of DatalogMTL has one (object) sort. Terms are constants and variables, whereas relational atoms are defined as in Datalog$_\mathbb{Q}$. *Metric atoms* are defined recursively as $\top$, $\bot$, relational atoms, $\boxplus_\varrho M$, $\boxminus_\varrho M$, $\diamondplus_\varrho M$, $\diamondminus_\varrho M$, $M \mathcal{S}_\varrho M$, and $M \mathcal{U}_\varrho M$, where M are metric atoms and ϱ is a closed, non-empty, and positive (i.e. including only positive numbers) interval over the rational timeline; it can be right-unbounded, i.e. with $+\infty$ as the right endpoint. A *DatalogMTL rule* is of the form $M' \leftarrow M_1 \wedge \ldots \wedge M_n$, where each M_i is a metric atom and M' is a metric atom not mentioning any of $\diamondplus$, $\diamondminus$, $\mathcal{S}$, and $\mathcal{U}$. A *DatalogMTL program* is a finite set of DatalogMTL rules. A DatalogMTL program is *flat* if it does not mention any nesting of temporal operators. Unlike in Datalog$_\mathbb{Q}$, rules of DatalogMTL do not allow us to express facts. A DatalogMTL *fact* is of the form $P(\mathbf{c})@\varrho$, where $P(\mathbf{c})$ is a ground relational atom and ϱ is a closed interval over rationals; we allow for left- and right-unbounded intervals (with $-\infty$ and $+\infty$ as endpoints). In case of *punctual* intervals, $[t, t]$, we may write $P(\mathbf{c})@t$ instead of $P(\mathbf{c})@[t, t]$. A *DatalogMTL dataset* is a finite set of temporal facts. A *DatalogMTL interpretation* $\mathcal{I}$ is a function mapping each *time point* (a rational number) to a set of ground relational atoms. We have $\mathcal{I}, t \models \top$ and $\mathcal{I}, t \not\models \bot$, for every t. We let $\mathcal{I}, t \models P(\mathbf{c})$ if $P(\mathbf{c}) \in \mathcal{I}(t)$. Then, for every ground metric atom M we have:

$$
\begin{aligned}
\mathcal{I}, t \models \diamondminus_\varrho M \quad & \text{iff } \mathcal{I}, t' \models M \text{ for some } t' \text{ with } t - t' \in \varrho, \\
\mathcal{I}, t \models \diamondplus_\varrho M \quad & \text{iff } \mathcal{I}, t' \models M \text{ for some } t' \text{ with } t' - t \in \varrho, \\
\mathcal{I}, t \models \boxminus_\varrho M \quad & \text{iff } \mathcal{I}, t' \models M \text{ for all } t' \text{ with } t - t' \in \varrho, \\
\mathcal{I}, t \models \boxplus_\varrho M \quad & \text{iff } \mathcal{I}, t' \models M \text{ for all } t' \text{ with } t' - t \in \varrho, \\
\mathcal{I}, t \models M_1 \mathcal{S}_\varrho M_2 \quad & \text{iff } \mathcal{I}, t' \models M_2 \text{ for some } t' \text{ with } t - t' \in \varrho, \\
& \qquad \text{and } \mathcal{I}, t'' \models M_1 \text{ for all } t'' \in (t', t), \\
\mathcal{I}, t \models M_1 \mathcal{U}_\varrho M_2 \quad & \text{iff } \mathcal{I}, t' \models M_2 \text{ for some } t' \text{ with } t' - t \in \varrho, \\
& \qquad \text{and } \mathcal{I}, t'' \models M_1 \text{ for all } t'' \in (t, t').
\end{aligned}
$$

We write $\mathcal{I} \models M@\varrho$ if $\mathcal{I}, t \models M$ for all $t \in \varrho$.

An interpretation $\mathcal{I}$ is a *model* of a rule $M' \leftarrow M_1 \wedge \ldots \wedge M_n$ if for every time point t and for every grounding ν, it holds that $\mathcal{I}, t \models \nu(M')$, whenever $\mathcal{I}, t \models \nu(M_i)$ for every $i \in \{1, \ldots, n\}$. Interpretation $\mathcal{I}$ is a model of a program Π if it is a *model* of every rule in Π. An interpretation is a *model* of a dataset $\mathcal{D}$ if for every fact $P(\mathbf{c})@\varrho \in \mathcal{D}$, we have $\mathcal{I} \models P(\mathbf{c})@\varrho$. We let $\mathcal{I}_\mathcal{D}$ be the unique interpretation which is the minimal model of $\mathcal{D}$. A pair $(\Pi, \mathcal{D})$ of a program and dataset is *consistent* if there is an interpretation that is a model for both Π and $\mathcal{D}$. A pair $(\Pi, \mathcal{D})$ *entails* a ground metric atom M at t, written as $(\Pi, \mathcal{D}), t \models M$, if $\mathcal{I}, t \models M$ for every model $\mathcal{I}$ of $(\Pi, \mathcal{D})$.

Similarly to Datalog$_\mathbb{Q}$, an alternative semantics of DatalogMTL is provided using the *immediate consequence operator* T_Π, which is a partial function mapping an interpretation $\mathcal{I}$ to the unique interpretation $T_\Pi(\mathcal{I})$, if one exists (otherwise it is undefined), such that for each t, $T_\Pi(\mathcal{I})(t)$ is the subset-minimal set satisfying: (i) $\mathcal{I}(t) \subseteq T_\Pi(\mathcal{I})(t)$ and (ii) for every rule $M' \leftarrow M_1 \wedge \ldots \wedge M_n$ in Π and any grounding ν, we have $\mathcal{I}, t \models \nu(M')$ whenever $\mathcal{I}, t \models \nu(M_i)$ for every $i \in \{1, \ldots, n\}$. For any successor ordinal α, we let $T_\Pi^{\alpha+1}(\mathcal{I}) = T_\Pi(T_\Pi^\alpha(\mathcal{I}))$, and for a limit ordinal γ, we let $T_\Pi^\gamma(\mathcal{I}) = \bigcup_{\beta<\gamma} T_\Pi^\beta(\mathcal{I})$. The canonical interpretation for program Π and dataset $\mathcal{D}$ is $\mathcal{C}_{\Pi,\mathcal{D}} = T_\Pi^{\omega_1}(\mathcal{I}_\mathcal{D})$, where ω_1 is the first uncountable ordinal. Then $(\Pi, \mathcal{D})$ is *consistent* if $\mathcal{C}_{\Pi,\mathcal{D}}$ is well-defined, whereas $(\Pi, \mathcal{D})$ *entails* a fact $P(\mathbf{c})@\varrho$, in symbols $(\Pi, \mathcal{D}) \models P(\mathbf{c})@\varrho$, if $(\Pi, \mathcal{D})$ is inconsistent or $\mathcal{C}_{\Pi,\mathcal{D}} \models P(\mathbf{c})@\varrho$. Moreover, $(\Pi, \mathcal{D})$ is *finitely materialisable* if there is $k \in \mathbb{N}$ for which $T_\Pi^k(\mathcal{I}_\mathcal{D}) = \mathcal{C}_{\Pi,\mathcal{D}}$. The canonical interpretation $\mathcal{C}_{\Pi,\mathcal{D}}$ of a consistent pair $(\Pi, \mathcal{D})$ is the minimal model of this pair [18].

4.5 Translating DatalogMTL to Datalog$_\mathbb{Q}$

In this section, we will show how to translate DatalogMTL into Datalog$_\mathbb{Q}$, to simulate temporal reasoning with non-temporal Datalog engines. For readability we assume that DatalogMTL programs are flat, i.e. with no nested temporal operators. Any program can be translated into this form [19]. Moreover, until Proposition 4.7, we assume that programs do not mention $\bot$ in rule heads. In this setting no inconsistency can occur, and $T_\Pi^\gamma(\mathcal{I})$ is well-defined for any γ. In Proposition 4.7 we turn our attention to programs with $\bot$, as we will consider consistency checking.

We start by providing intuition regarding a desired relation between DatalogMTL and its translation into Datalog. To this end, consider again a DatalogMTL program Π consisting of Rules (4.1) and (4.2), which detect circular financial transactions. Assume that the DatalogMTL dataset $\mathcal{D}$ gathered by the bank contains facts $\mathit{Trans}(\mathit{adam}, \mathit{betty})@t$ and $\mathit{Trans}(\mathit{betty}, \mathit{adam})@t + 2$, for time points $t \in \{0, 10, 20\}$. Then, by application of Rule (4.1), $\mathit{Susp}(\mathit{adam})$ is derived in intervals $[2, 16]$, $[12, 26]$, and $[22, 36]$, as depicted in Figure 4.1. Hence, we obtain $\mathit{Susp}(\mathit{adam})@[2, 36]$, and so, an application of Rule (4.2) derives $\mathit{RedList}(\mathit{adam})@[32, 36]$.

The aim is to translate Π into a Datalog$_\mathbb{Q}$ program $\mathbf{tr}(\Pi)$ and $\mathcal{D}$ into a Datalog$_\mathbb{Q}$ interpretation $\mathbf{tr}(\mathcal{D})$ such that facts derived by $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$ can correspond to the temporal facts derived by Π and $\mathcal{D}$. For example, we aim to simulate the derivation of $\mathit{Susp}(\mathit{adam})[2, 36]$ in DatalogMTL with a derivation of $\mathit{Susp}(\mathit{adam}, 2, 36)$ in Datalog$_\mathbb{Q}$. In other words, we want to encode in Datalog$_\mathbb{Q}$ the endpoints of intervals in which a fact $\mathit{Susp}(\mathit{adam})$ holds in DatalogMTL, by adding two numerical positions in predicates.

It is worth noting, however, that due to the continuous timeline and the need to consider coalescing of intervals, the correspondence between DatalogMTL and

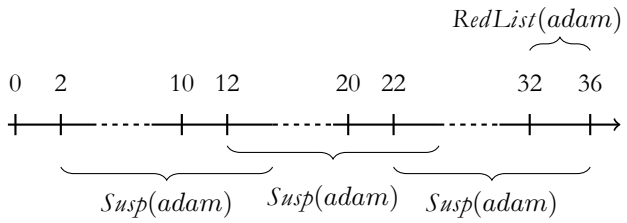


Figure 4.1: Facts derived by Rules (4.1) and (4.2)

$\text{Datalog}_{\mathbb{Q}}$ is not straightforward. For example, it may be tempting to assume that whenever $P(\mathbf{c})$ is entailed at some interval $[a, b]$ in DatalogMTL, then $P(\mathbf{c}, a, b)$ should be derived in $\text{Datalog}_{\mathbb{Q}}$. Notice that if, for instance, $P(\mathbf{c})$ is entailed in $[0, 1]$, this would mean that $P(\mathbf{c}, a, b)$ needs to be derived for all a, b such that $[a, b] \subseteq [0, 1]$. Since we consider a dense (rational) time line, there are infinitely many such intervals, so simulating a single DatalogMTL fact $P(\mathbf{c})@[0, 1]$ would require an infinite $\text{Datalog}_{\mathbb{Q}}$ interpretation, which is not feasible in practice. An alternative option could be to require that $P(\mathbf{c}, a, b)$ is entailed in Datalog only for the maximal interval $[a, b]$ such that $P(\mathbf{c})@[a, b]$ is entailed in DatalogMTL. This would not require infinite $\text{Datalog}_{\mathbb{Q}}$ interpretations. However, it would be too restrictive, as it would not allow for coalescing intervals. Indeed, as presented in the example from Figure 4.1, it is essential to first derive $\mathit{Susp}(adam)@[2, 16]$, $\mathit{Susp}(adam)@[12, 26]$, and $\mathit{Susp}(adam)@[22, 36]$, then coalesce these intervals to obtain $\mathit{Susp}(adam)@[2, 36]$, and only then apply Rule (4.2). We would not be able to simulate such derivations in $\text{Datalog}_{\mathbb{Q}}$ if we did not allow for storing facts over intervals which have non-empty intersections.

This leads us to a more complicated requirement defined next. Intuitively we will consider a DatalogMTL interpretation $\mathcal{I}$ as corresponding to a $\text{Datalog}_{\mathbb{Q}}$ interpretation I if $\mathcal{I}$ pointwise subsumes I, and vice versa. By pointwise subsumption we mean that if $\mathcal{I}, t \models P(\mathbf{c})$ in DatalogMTL, then $P(\mathbf{c}, a, b) \in I$ for some $t \in [a, b]$ in $\text{Datalog}_{\mathbb{Q}}$. For the opposite subsumption, we require that $P(\mathbf{c}, a, b) \in I$ in $\text{Datalog}_{\mathbb{Q}}$ implies $\mathcal{I}, t \models P(\mathbf{c})$ for each $t \in [a, b]$ in DatalogMTL. We formalise these notions in the definition below.

Definition 4.1. *A DatalogMTL interpretation $\mathcal{I}$ is pointwise subsumed by a* $\text{Datalog}_{\mathbb{Q}}$ *interpretation I, denoted $\mathcal{I} \leq_p I$, if for any relational atom $P(\mathbf{c})$ and time point t with $\mathcal{I}, t \models P(\mathbf{c})$, there exists an interval $[a, b]$ with $P(\mathbf{c}, a, b) \in I$ and $t \in [a, b]$. Conversely, we say that I is pointwise subsumed by $\mathcal{I}$, $I \leq_p \mathcal{I}$, if for every time point t and interval $[a, b]$ with $t \in [a, b]$ and $P(\mathbf{c}, a, b) \in I$, it holds that $\mathcal{I}, t \models P(\mathbf{c})$. If $\mathcal{I} \leq_p I$ and $I \leq_p \mathcal{I}$, we write $\mathcal{I} =_p I$.*

In the rest of this section we show how to construct a translation of DatalogMTL programs and datasets into $\text{Datalog}_{\mathbb{Q}}$ so that we obtain a correspondence between DatalogMTL and $\text{Datalog}_{\mathbb{Q}}$ interpretations in terms of the relation $=_{\text{p}}$.

The translation of DatalogMTL facts is straightforward, namely we will translate each fact $P(\mathbf{c})@[a,b]$ into $P(\mathbf{c},a,b)$. For example, $\mathit{Trans}(\mathit{adam},\mathit{betty})@0$ is translated into $\mathit{Trans}(\mathit{adam},\mathit{betty},0,0)$, since a punctual interval 0 starts and ends in the same time point 0. The translation of DatalogMTL rules is much more complicated and is obtained in two steps. The first step is to translate temporal operators in the rule into arithmetic operations. For example, Rule (4.2) is translated into the following $\text{Datalog}_{\mathbb{Q}}$ rule:

$$\begin{aligned}\mathit{RedList}(x,t^-,t^+) \leftarrow &\mathit{Susp}(x,t_1^-,t_1^+) \wedge \\ &(t^- = t_1^- + 30) \wedge (t^+ = t_1^+ + 0) \wedge (t^- \leq t^+)\end{aligned}$$

This rule states that $\mathit{Susp}(x,t_1^-,t_1^+)$ allows us to derive $\mathit{Susp}(x,t_1^- + 30,t_1^+)$ if $t_1^- + 30 \leq t_1^+$. Note that the last condition guarantees that we derive facts only over intervals, that is, the first endpoint cannot be greater than the second.

The second step of the translation is to add $\text{Datalog}_{\mathbb{Q}}$ rules for coalescing intervals. For example to coalesce facts over the predicate Susp, we use the rule

$$\mathit{Susp}(x,t_1,t_4) \leftarrow \mathit{Susp}(x,t_1,t_2) \wedge \mathit{Susp}(x,t_3,t_4) \wedge (t_3 \leq t_2) \wedge (t_1 \leq t_4),$$

which states that from $\mathit{Susp}(x,a,b)$ and $\mathit{Susp}(x,c,d)$ over intersecting intervals $[a,b]$ and $[c,d]$, we can derive $\mathit{Susp}(x,t_1,t_4)$ over the union $[t_1,t_4] = [a,b] \cup [c,d]$.

The full translation requires considering all the possible temporal operators and handling conjunctions in rule bodies. The formal definition is provided below.

Definition 4.2. *For a DatalogMTL dataset $\mathcal{D}$ we define*

$$\mathbf{tr}(\mathcal{D}) \coloneqq \{P(\mathbf{s},a,b) \mid P(\mathbf{s})@[a,b] \in \mathcal{D}\}.$$

For a DatalogMTL rule $r = M' \leftarrow M_1 \wedge \ldots \wedge M_n$, we let

$$\begin{aligned}\mathbf{tr}(r) \coloneqq A' \leftarrow A_1 \wedge \cdots \wedge A_n \wedge (t^- = \max_{i \leq n} \varrho_i^-) \wedge \\ (t^+ = \min_{i \leq n} \varrho_i^+) \wedge (t^- \leq t^+),\end{aligned} \tag{4.3}$$

where A' is as follows:

- $A' \coloneqq M'$, *if* M' *is* $\top$ *or* $\bot$,
- $A' \coloneqq P(\mathbf{s},t^-,t^+)$, *if* M' *is* $P(\mathbf{s})$,
- $A' \coloneqq P(\mathbf{s},t^- + a,t^+ + b)$, *if* M' *is* $\boxplus_{[a,b]} P(\mathbf{s})$,
- $A' \coloneqq P(\mathbf{s},t^- - b,t^+ - a)$, *if* M' *is* $\boxminus_{[a,b]} P(\mathbf{s})$

expressions A_i are as follows:

- $A_i \coloneqq M_i$, *if* M_i *is* $\top$ *or* $\bot$,

- $A_i \coloneqq P(\mathbf{s}, t_i^-, t_i^+)$, *if* M_i *is* $P(\mathbf{s})$, $\oplus_{[a,b]} P(\mathbf{s})$, *or* $\diamondsuit_{[a,b]} P(\mathbf{s})$,
- $A_i \coloneqq P(\mathbf{s}, t_i^-, t_i^+) \wedge (b - a) \leq (t_i^+ - t_i^-)$, *if* M_i *is* $\boxplus_{[a,b]} P(\mathbf{s})$ *or* $\boxminus_{[a,b]} P(\mathbf{s})$,
- $A_i \coloneqq P_1(\mathbf{s}, t_{i1}^-, t_{i1}^+) \wedge P_2(\mathbf{s}, t_{i2}^-, t_{i2}^+) \wedge (t_{i1}^- \leq t_{i2}^- - b) \wedge (t_{i2}^+ \leq t_{i1}^+)$, *if* M_i *is* $P_1(\mathbf{s}) \mathcal{U}_{[a,b]} P_2(\mathbf{s})$,
- $A_i \coloneqq P_1(\mathbf{s}, t_{i1}^-, t_{i1}^+) \wedge P_2(\mathbf{s}, t_{i2}^-, t_{i2}^+) \wedge (t_{i1}^- \leq t_{i2}^-) \wedge (t_{i2}^+ + b \leq t_{i1}^+)$, *if* M_i *is* $P_1(\mathbf{s}) \mathcal{S}_{[a,b]} P_2(\mathbf{s})$

and intervals $\varrho_i \coloneqq [\varrho_i^-, \varrho_i^+]$ *are defined as*

- $\varrho_i \coloneqq [-\infty, +\infty]$ *for* $M_i = \top$ *or* $= \bot$,
- $\varrho_i \coloneqq [t_i^-, t_i^+]$ *if* M_i *is* $P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- - b, t_i^+ - a]$ *if* M_i *is* $\oplus_{[a,b]} P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- + a, t_i^+ + b]$ *if* M_i *is* $\diamondsuit_{[a,b]} P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- - a, t_i^+ - b]$ *if* M_i *is* $\boxplus_{[a,b]} P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- + b, t_i^+ + a]$ *if* M_i *is* $\boxminus_{[a,b]} P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_{i2}^- - b, t_{i2}^+ - a]$ *if* M_i *is* $P_1(\mathbf{c}_1) \mathcal{U}_{[a,b]} P_2(\mathbf{s}_2)$,
- $\varrho_i \coloneqq [t_{i2}^- + a, t_{i2}^+ + b]$ *if* M_i *is* $P_1(\mathbf{c}_1) \mathcal{S}_{[a,b]} P_2(\mathbf{s}_2)$

Finally, for a DatalogMTL program Π *we define*

$$\mathbf{tr}(\Pi) \coloneqq \Pi_{time} \cup \bigcup_{r \in \Pi} \mathbf{tr}(r),$$

where Π_{time} *is the program consisting of the following rule, for every predicate* P *in* Π*, where* $\mathbf{x}$ *matches the arity of* P*:*

$$P(\mathbf{x}, t_1, t_4) \leftarrow P(\mathbf{x}, t_1, t_2) \wedge P(\mathbf{x}, t_3, t_4) \wedge (t_3 \leq t_2) \wedge (t_1 \leq t_4), \tag{4.4}$$

which coalesces overlapping intervals. For example, $Susp(adam)@[2, 16]$ *and* $Susp(adam)@[12, 26]$ *allow us to derive* $Susp(adam)@[2, 26]$.

For instance when applying our translation to Rule (4.1), we obtain the following $\text{Datalog}_{\mathbb{Q}}$ rule:

$$\begin{aligned} Susp(x, t^-, t^+ + 14) \leftarrow & Trans(x, y, t_1^-, t_1^+) \wedge Trans(y, x, t_2^-, t_2^+) \wedge \\ & (t^- = \max\{t_1^-, t_2^-\}) \wedge (t^+ = \min\{t_1^+, t_2^+ + 5\}) \wedge \\ & (t^- \leq t^+) \end{aligned} \tag{4.5}$$

In the rest of this section we will prove correctness of the translation and analyse how the materialisation steps for Π and $\mathcal{D}$ correspond to the materialisation steps of $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$. As we will show, establishing this relation is not straightforward, and poses several technical challenges.

Our analysis will exploit a stronger notion of correspondence between DatalogMTL and $\text{Datalog}_{\mathbb{Q}}$ interpretations than pointwise subsumptions from Definition 4.1. This is because application of rules does not only require that interpretations match on time points, but also on intervals. Hence, the following intervalwise subsumption will be useful.

Definition 4.3. *A DatalogMTL interpretation $\mathcal{I}$ is intervalwise subsumed by a $\text{Datalog}_{\mathbb{Q}}$ interpretation I, in symbols $\mathcal{I} \leq_{int} I$, if for every metric fact $P(\mathbf{c})@\varrho$ satisfied by $\mathcal{I}$ ($\mathcal{I} \models P(\mathbf{c})@\varrho$) there exists an interval $[a, b] \supseteq \varrho$ such that $P(\mathbf{c}, a, b) \in I$. On the other hand, I is intervalwise subsumed by $\mathcal{I}$, in symbols $I \leq_{int} \mathcal{I}$, if for every atom $P(\mathbf{c}, a, b) \in I$, it holds that $\mathcal{I} \models P(\mathbf{c})@[a, b]$. If $\mathcal{I} \leq_{int} I$ and $I \leq_{int} \mathcal{I}$, we write $\mathcal{I} =_{int} I$.*

It follows from Definitions 4.1 and 4.3 that intervalwise subsumption is a stronger notion than pointwise subsumption. Indeed, assume $\mathcal{I} \leq_{\text{int}} I$ and consider an atom $P(\mathbf{c})@t$ such that $\mathcal{I}, t \models P(\mathbf{c})$. Since $\mathcal{I} \leq_{\text{int}} I$, there exists $[a, b]$ with $[t, t] \subseteq [a, b]$ such that $P(\mathbf{c}, a, b) \in I$. As a result, $\mathcal{I} \leq_{\text{p}} I$. Likewise, $I \leq_{\text{int}} \mathcal{I}$ implies $I \leq_{\text{p}} \mathcal{I}$.

Recall that $\mathcal{I}_{\mathcal{D}}$ is the minimal DatalogMTL interpretation satisfying the dataset $\mathcal{D}$. We observe that for each DatalogMTL dataset $\mathcal{D}$ we have $\mathbf{tr}(\mathcal{D}) \leq_{\text{int}} \mathcal{I}_{\mathcal{D}}$. The opposite, however, is not always true.

Proposition 4.1. *For every DatalogMTL dataset $\mathcal{D}$, we have $\mathbf{tr}(\mathcal{D}) \leq_{int} \mathcal{I}_{\mathcal{D}}$. However, the opposite subsumption does not always hold.*

Proof. By definition, $P(\mathbf{c}, a, b) \in \mathbf{tr}(\mathcal{D})$ implies that $P(\mathbf{c})@[a, b] \in \mathcal{D}$. Thus for every $t \in [a, b]$, $P(\mathbf{c})$ holds in $\mathcal{I}_{\mathcal{D}}$ at time t, and so $\mathcal{I} \models P(\mathbf{c})@[a, b]$.

The opposite subsumption, $\mathcal{I}_{\mathcal{D}} \leq_{\text{int}} \mathbf{tr}(\mathcal{D})$, does not always hold. Indeed, consider a dataset $\mathcal{D} = \{Susp(adam)@[0, 20], Susp(adam)@[20, 40]\}$. We have $\mathcal{I}_{\mathcal{D}} \models Susp(adam)@[0, 40]$, but $\mathbf{tr}(\mathcal{D}) = \{Susp(adam, 0, 20), Susp(adam, 20, 40)\}$ does not contain any atom $Susp(adam, a, b)$ with $[0, 40] \subseteq [a, b]$. □

As we show next, the interval subsumption does hold if we apply Π_{time} to $\mathbf{tr}(\mathcal{D})$ a sufficient number of times. In particular, we can show that $\mathcal{I}_{\mathcal{D}} \leq_{\text{int}} T_{\Pi_{\text{time}}}^{\lceil \log(|\mathcal{D}|) \rceil}(\mathbf{tr}(\mathcal{D}))$, where we assume in the paper that log has base 2. The reason why we need to apply Π_{time} multiple times is that the dataset $\mathcal{D}$ may contain multiple adjacent or overlapping intervals for the same fact. Rule (4.4) merges two overlapping intervals, thus repeated applications of this rule perform a binary-style aggregation. Since each application halves the number of remaining intervals to be merged, all intervals contributing to a maximal union can be coalesced in logarithmically many steps. Formally, we obtain the following result.

Proposition 4.2. *For every DatalogMTL dataset $\mathcal{D}$, it holds that*

$$\mathcal{I}_{\mathcal{D}} =_{int} T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|)\rceil}(\mathbf{tr}(\mathcal{D})).$$

Proof. Assume that $[a, b]$ is a subset-maximal interval such that $\mathcal{I}_{\mathcal{D}} \models P(\mathbf{c})@[a, b]$, for some $P(\mathbf{c})$. Then, $[a, b]$ is a union of some intervals $[a_i, b_i]$, with $1 \leq i \leq n$, mentioned in $\mathcal{D}$. Hence, each $P(\mathbf{c}, a_i, b_i)$ belongs to $\mathbf{tr}(\mathcal{D})$. By applying Rule (4.4) at most $\lceil \log(n)\rceil$ times to $\mathbf{tr}(\mathcal{D})$, we can merge the intervals in a binary fashion and derive $P(\mathbf{c}, a, b)$. Since $n \leq |\mathcal{D}|$, we obtain $\mathcal{I}_{\mathcal{D}} \leq_{\text{int}} T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|)\rceil}(\mathbf{tr}(\mathcal{D}))$. Conversely, for each $P(\mathbf{c}, a, b) \in T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|)\rceil}(\mathbf{tr}(\mathcal{D}))$, atom $P(\mathbf{c})$ holds in $\mathcal{I}_{\mathcal{D}}$ at t in $[a, b]$, since every t belongs to some $[a_i, b_i]$. Thus $T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|)\rceil}(\mathbf{tr}(\mathcal{D})) \leq_{\text{int}} \mathcal{I}_{\mathcal{D}}$. □

Propositions 4.1 and 4.2 provide us with a correspondence between $\mathcal{D}$ and $\mathbf{tr}(\mathcal{D})$. Next, we will analyse the correspondence between partial materialisations in DatalogMTL and $\text{Datalog}_{\mathbb{Q}}$. In particular, we will derive bounds on the number of the immediate consequence operator applications which guarantee intervalwise subsumptions between DatalogMTL and $\text{Datalog}_{\mathbb{Q}}$ interpretation. In the bounds below, we will use n_Π to represent the maximal number of metric atoms found in the rule bodies of rules in Π.

Theorem 4.3. *For a DatalogMTL dataset $\mathcal{D}$ and program Π*

$$T_\Pi(\mathcal{I}_{\mathcal{D}}) \leq_{int} T_{\mathbf{tr}(\Pi)}^{\lceil \log(8|\mathcal{D}|^2(n_\Pi \cdot |\Pi|+1))\rceil}(\mathbf{tr}(\mathcal{D})).$$

Proof sketch. Fix $P(\mathbf{c})@\varrho$ such that $T_\Pi(\mathcal{I}_D) \models P(\mathbf{c})@\varrho$. We will show that $P(\mathbf{c}, a, b) \in T_{\mathbf{tr}(\Pi)}^K(\mathbf{tr}(D))$ for some $[a, b] \supseteq \varrho$, where $K = \lceil \log(8|\mathcal{D}|^2(n_\Pi|\Pi| + 1))\rceil$.

By the definition of T_Π, interval ϱ is a union of finitely many intervals $\varrho_1, \ldots, \varrho_m$, such that each ϱ_i occurs explicitly in $\mathcal{D}$ or is an interval over which $P(\mathbf{c})$ is derived by a single rule $r \in \Pi$ applied to $\mathcal{I}_{\mathcal{D}}$.

To compute a bound on m we determine how many endpoints can appear in these intervals. There are $2|\mathcal{D}|$ endpoints mentioned in $\mathcal{D}$. In the case of ϱ_i derived by T_Π, there are $|\Pi|$ rules, each with at most n_Π body atoms which gives rise to at most $4|\mathcal{D}| \cdot n_\Pi|\Pi|$ endpoints. In total, we obtain at most $4|\mathcal{D}|(n_\Pi|\Pi|+1)$ endpoints, so $m \leq 4|\mathcal{D}|(n_\Pi|\Pi| + 1) - 1$.

We next show that for each ϱ_i there is $[a_i, b_i] \supseteq \varrho_i$ such that $P(\mathbf{c}, a_i, b_i) \in T_{\mathbf{tr}(\Pi)}^{K_1+1}(\mathbf{tr}(\mathcal{D}))$, where $K_1 = \lceil \log |\mathcal{D}| \rceil$. If ϱ_i is an interval appearing in $\mathcal{D}$, then the claim holds directly by Proposition 4.2, since $\Pi_{\text{time}} \subseteq \mathbf{tr}(\Pi)$. If ϱ_i is obtained by an application of a rule r with body $M_1 \wedge \cdots \wedge M_n$ satisfied at some ϱ_i' under a grounding ν, then $\mathcal{I}_{\mathcal{D}} \models \nu(M_j)@\varrho_i'$ for each $j \leq n$. By Proposition 4.2, after K_1 applications of $T_{\mathbf{tr}(\Pi)}$ to $\mathbf{tr}(\mathcal{D})$ each translated body atom A_j from Definition 4.2 is derived. One further application of $T_{\mathbf{tr}(r)}$ then derives $P(\mathbf{c}, a_i, b_i)$ with $\varrho_i \subseteq [a_i, b_i]$, as required.

Since the atoms $P(\mathbf{c}, a_i, b_i)$ jointly cover ϱ, applying Rule (4.4) in a balanced binary fashion $\lceil \log m \rceil$ times merges them into a single $P(\mathbf{c}, a, b)$ with $\varrho \subseteq [a, b]$. This is obtained after at most $K_1 + 1 + \lceil \log m \rceil \leq K$ rule applications. □

For the opposite direction it is clear that the following subsumption holds.

Proposition 4.4. *For any DatalogMTL dataset $\mathcal{D}$ and any program Π it holds that*

$$T_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D})) \leq_{int} T_{\Pi}(\mathcal{I}_{\mathcal{D}}).$$

As a consequence of the results established above, and the fact that the intervalwise subsumption implies pointwise subsumption, we obtain the following results.

Corollary 4.5. *For every DatalogMTL dataset $\mathcal{D}$ and any program Π it holds that*

$$T^{\omega}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D})) =_{p} T^{\omega}_{\Pi}(\mathcal{I}_{\mathcal{D}}).$$

We can observe that the above holds for pointwise subsumptions, but not for the intervalwise subsumptions. Indeed, consider a DatalogMTL dataset $\mathcal{D}$ with a single fact $P(\mathbf{c})@0$ and program Π with a single rule $\boxplus_{[0,1]} P(\mathbf{x}) \leftarrow P(\mathbf{x})$. We obtain that $T^{\omega}_{\Pi}(\mathcal{I}_{\mathcal{D}}) \models P(\mathbf{c})@[0, +\infty]$. However, $T^{\omega}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$ cannot contain an atom $P(\mathbf{c}, 0, +\infty)$, since $+\infty$ cannot occur in any atom of $T^{\omega}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$. As a result, $T^{\omega}_{\Pi}(\mathcal{I}_{\mathcal{D}}) \not\leq_{\text{int}} T^{\omega}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$.

The next example shows that for more than ω materialisation steps in DatalogMTL, bounding the corresponding amount of applications in $\text{Datalog}_{\mathbb{Q}}$ becomes infeasible. Consider a DatalogMTL dataset with a single fact $P(\mathbf{c})@[0, 1]$, and a program Π with two rules: $\boxplus_{[0,1]} P(\mathbf{x}) \leftarrow P(\mathbf{x})$ and $Q(\mathbf{x}) \leftarrow \boxplus_{[0,+\infty]} P(\mathbf{x})$. We obtain that $T^{\omega}_{\Pi}(\mathcal{I}_{\mathcal{D}}) \models P(\mathbf{c})@[0, +\infty]$, and $T^{\omega+1}_{\Pi}(\mathcal{I}_{\mathcal{D}}) \models Q(\mathbf{c})@0$. As we have already argued, $P(\mathbf{c}, 0, \infty) \notin T^{\omega}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$. Therefore, $Q(\mathbf{c}, 0, 0) \notin T^{\omega+1}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$.

Now we consider the notion of finite materialisability, that is, the possibility to materialise in a finite number of steps (within a finite number of immediate consequence operator applications). Our results on the bounds on the finite number of materialisation steps imply that the translation preserves finite materialisability: if $(\Pi, \mathcal{D})$ is finitely materialisable in DatalogMTL, then so is $(\mathbf{tr}(\Pi), \mathbf{tr}(\mathcal{D}))$ in $\text{Datalog}_{\mathbb{Q}}$, and vice versa.

Proposition 4.6. *For every DatalogMTL program Π and dataset $\mathcal{D}$, it holds that $(\Pi, \mathcal{D})$ is finitely materialisable in DatalogMTL if and only if $(\mathbf{tr}(\Pi), \mathbf{tr}(\mathcal{D}))$ is finitely materialisable in* $\text{Datalog}_{\mathbb{Q}}$.

We now consider the notion of consistency, for which we need to consider programs with $\bot$ in rule heads (programs without $\bot$ are trivially consistent). Our results on entailment allow us to obtain the following result regarding consistency.

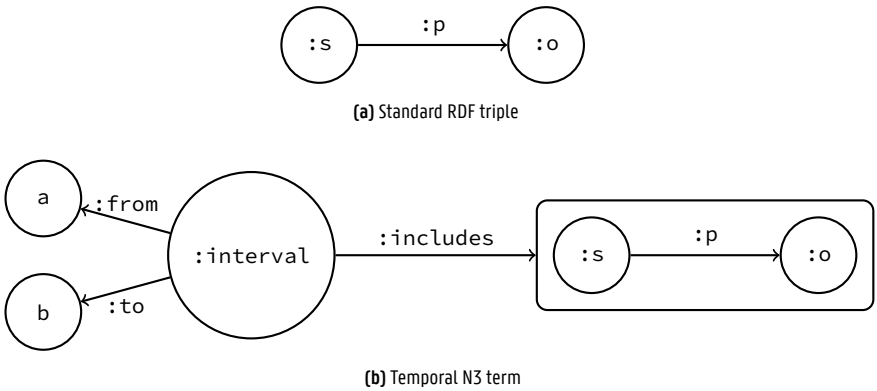
Proposition 4.7. *For every DatalogMTL program Π and dataset $\mathcal{D}$ the following hold:*

- *if* $(\Pi, \mathcal{D})$ *is consistent in DatalogMTL, then* $\big(\mathbf{tr}(\Pi), \mathbf{tr}(\mathcal{D})\big)$ *is consistent in* $\text{Datalog}_{\mathbb{Q}}$,
- *if* $\big(\mathbf{tr}(\Pi), \mathbf{tr}(\mathcal{D})\big)$ *is consistent and* $\mathbf{tr}(\Pi)$ *is finitely materialisable in* $\text{Datalog}_{\mathbb{Q}}$, *then* $(\Pi, \mathcal{D})$ *is consistent in DatalogMTL.*

Proof sketch. We focus on the second item (the first one has a similar proof). Assume that $(\mathbf{tr}(\Pi), \mathbf{tr}(\mathcal{D}))$ is consistent and finitely materialisable. Hence, there exists $k \in \mathbb{N}$ such that $T^k_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$ is the canonical interpretation. We define a DatalogMTL interpretation $\mathcal{I}$ by setting, for each time point t and ground atom $P(\bar{c})$, $\mathcal{I}, t \models P(\bar{c})$ if and only if there exists an interval $[a, b]$ with $a \leq t \leq b$ such that $P(\bar{c}, a, b) \in T^k_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$. We can show that $\mathcal{I}$ is a model of Π and $\mathcal{D}$, so $(\Pi, \mathcal{D})$ is consistent as required. □

Notice that the second item of Proposition 4.7 requires the assumption of finite materialisability. Without this assumption, it may be the case that $\big(\mathbf{tr}(\Pi), \mathbf{tr}(\mathcal{D})\big)$ is consistent, but $(\Pi, \mathcal{D})$ is not. This happens, for example, if $\mathcal{D}$ has a single fact $P(\mathbf{c})@[0, 1]$, whereas Π has two rules: $\boxplus_{[0,1]} P(\mathbf{x}) \leftarrow P(\mathbf{x})$ and $\bot \leftarrow \boxplus_{[0,+\infty]} P(\mathbf{x})$.

4.6 Implementation and Experiments



(a) Standard RDF triple

(b) Temporal N3 term

Figure 4.2: Graphic representation of temporalisation of RDF triples

The aim of this section is to test practical viability of the translation-based approach, by comparing translated DatalogMTL programs executed with existing non-temporal engines with a state-of-the-art dedicated DatalogMTL reasoner. Our experiments focus on acyclic and finitely materialisable programs, which are commonly considered in current DatalogMTL implementations and applications.

We observe that our translation introduces predicates with arities increased by two, to represent interval endpoints. Such a translation can be run directly with solvers such

as Nemo [28], which allow for predicates with arbitrary arities. However, some high-quality Datalog reasoners are based on the triple-based data format, which restricts the arity of allowed predicates to two. In order to be able to perform DatalogMTL reasoning in such reasoners, we will first adapt our translation to this setting.

Translation to N3 The N3 format builds upon the RDF triple patterns, of the form `:s :p :o`, where s and o are the subject and object of predicate P. Atoms $P(s, o)$ thus have predicates P restricted to arity two. N3 introduces graph terms as a means to add metadata to a group of triples. We employ this structure to encode when an atom $P(s, o)$ holds. In order to adapt the DatalogMTL translation to this setting, we encode a temporal fact of the form $P(s, o)@[a, b]$ into N3 as

```
:interval :includes { :s :p :o }.
:interval :from :a.
:interval :to :b .
```

which is depicted in Figure 4.2b. Translation of DatalogMTL rules is likewise adapted to the new temporalised triple structure. Each rule body atom $P(s, o, t_i^-, t_i^+)$ is represented by three N3 triples, as described above. Both subject and object s, o and the temporal bounds ϱ_i^-, ϱ_i^+ are thereby encoded as variables, i.e. `?s`, `?o`, `?start_i` and `?end_i`. Numerical constraints from operators like $\boxplus$, $\boxminus$, $\mathcal{S}$, and $\mathcal{U}$ are expressed using `math:difference` and `math:sum` clauses. To construct the head interval, we use `math:min` and `math:max` to compute the maximum lower bound and minimum upper bound across all body atoms, adjusted by the temporal operator. A fresh interval node `?newInterval` is introduced, along with edges `:from ?start`, `:to ?end` and `:includes { :s :q :o }` for a head atom with predicate $Q(s, o)$.

Systems We test the translation-based approach for reasoning in DatalogMTL using three non-temporal reasoning engines, namely Datalog engines Nemo and Eyelet, and an N3 reasoner EYE. To evaluate practical applicability of this approach for reasoning in DatalogMTL, we ran translated $\text{Datalog}_{\mathbb{Q}}$ programs on Nemo and Eyelet, and translated N3 programs on EYE. As the baseline, we use the dedicated DatalogMTL reasoner MeTeoR.

Experiments In the experiments we have computed all facts entailed by a given pair of a DatalogMTL program and dataset (i.e. we compute the canonical interpretations). We ran experiments on synthetic benchmarks generated using iTemporal [16], the only available generator producing both DatalogMTL programs and datasets. We generate datasets with varying number of relational atoms, (10, 50, 10^2, 500, 10^3, 10^4, 10^5)

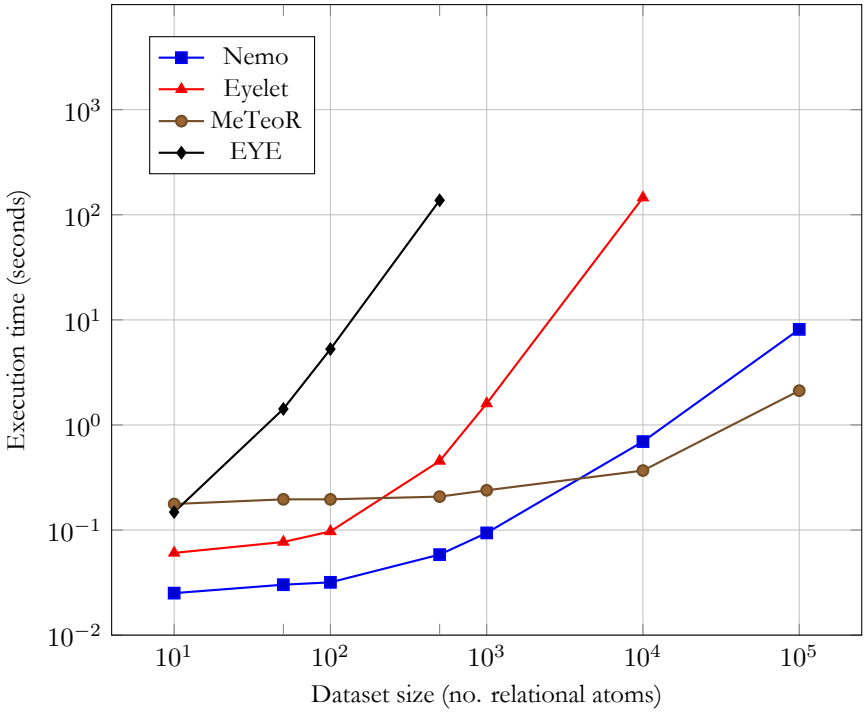


Figure 4.3: Average execution time on iTemporal benchmark

as proxy for dataset size used by iTemporal. Each dataset uses 10 predicate symbols. All datasets are acyclic and finitely materialisable.

Experiments were run on a Dell Latitude 5530 (Intel i7-1265U, 16GB RAM). Each configuration is repeated 5 times, preceded by a warm-up run to avoid initialisation bias. Since applying our translations takes under 800ms even for the largest dataset, we excluded them from runtime comparisons. Runtimes depicted in Figure 4.3 include time for data import and materialisation, as this provides the most consistent basis for comparison across engines with differing execution models. All engines were verified to produce the same outputs, ensuring the correctness of the results. The full set of programs, datasets, and auxiliary files of our experiments are provided via a Github repository[1].

Results Our results, presented in Figure 4.3, show that Nemo outperforms MeTeoR, Eyelet, and EYE on small datasets. The performance gap between Nemo and MeTeoR narrows as dataset size increases, with Nemo surpassing MeTeoR for sizes above 10^4; at size 10^5, MeTeoR is on average 5.994 seconds faster (73.9% of

[1] https://github.com/MathijsvN/DatalogMTL_Embedding_Experiments

Nemo's runtime), but this difference is smaller than Nemo's runtime variability (SD = 0.93). Overall, runtime variability is low for smaller datasets (SD < 0.04 s for all engines). For larger datasets, MeTeoR's variance remains stable (SD $\leq$ 0.13 s), while Nemo (SD up to 0.93 s), Eyelet (SD up to 18.93 s), and EYE (SD up to 2.57 s) show increasing variability with dataset size.

These results demonstrate that, under the considered assumptions, translation to standard Datalog provides a viable alternative to dedicated DatalogMTL reasoners, particularly for moderately sized datasets. Notably, the best performance is observed when using Datalog reasoners. Performance of reasoning on the N3 syntax increases rapidly with dataset size, as experiments only up to dataset size 500 could be obtained. The N3 format thus appears to introduce a substantial performance overhead.

4.7 Conclusion and Future Work

We have presented a translation from DatalogMTL into standard Datalog with arithmetic, which allows us to execute temporal reasoning tasks using mature, non-temporal reasoning engines. The results show that temporal reasoning in DatalogMTL can be effectively performed by translating programs into $\text{Datalog}_{\mathbb{Q}}$ and running standard Datalog reasoning. The theoretical analysis ensures that the translation preserves central semantic properties, including entailment, consistency, and finite materialisability.

Beyond the theoretical correspondence, our empirical evaluation highlights the practical viability of this translation-based method. For small datasets, translated DatalogMTL programs executed on Nemo outperform MeTeoR, a state-of-the-art specialised temporal reasoner. Even when performance is comparable rather than superior, the ability to rely on existing, professionally engineered Datalog platforms represents a major advantage in terms of maintainability, deployment readiness, and ecosystem integration.

As the next step we plan to scale up the empirical evaluation and to determine realistic use-cases where this approach can be applied. Additionally, further steps can be taken with regards to the theoretical aspect of the work. In particular it is interesting to explore the interaction of our translation with stratified negation [23] and incremental reasoning techniques [54] in DatalogMTL. Furthermore, other SR languages such as Logic-based framework for Analytic Reasoning over Streams (LARS) and RTEC can be considered for a similar mapping to non-temporal engines.

Acknowledgements

This research is partially funded by the FWO Travel Grant (Nr. V424925N) and FWO Project FRACTION (Nr. G086822N).

AI Use Disclosure

AI-based tools were used solely for editorial assistance, including improving readability and presentation. All technical content and conclusions are entirely the authors' own.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN: 0-201-53771-0.

[2] Darko Anicic et al. "Stream reasoning and complex event processing in ETALIS". In: *Semantic web* 3.4 (2012), pp. 397–407.

[3] A. Artale et al. "First-order Rewritability of Temporal Ontology-mediated Queries". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2015, pp. 2706–2712.

[4] Alessandro Artale and Enrico Franconi. "A temporal description logic for reasoning about actions and plans". In: *Journal of Artificial Intelligence Research* (1998), pp. 463–506.

[5] Alexander Artikis, Marek Sergot, and Georgios Paliouras. "Run-time composite event recognition". In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. 2012, pp. 69–80.

[6] Franz Baader et al. "Metric Temporal Description Logics with Interval-Rigid Names". In: *Frontiers of Combining Systems*. 2017, pp. 60–76. DOI: 10.1007/978-3-319-66167-4_4.

[7] Davide Francesco Barbieri et al. "C-SPARQL: a continuous query language for RDF data streams". In: *International Journal of Semantic Computing* 4.01 (2010), pp. 3–25.

[8] Hamid R. Bazoobandi, Harald Beck, and Jacopo Urbani. "Expressive stream reasoning with Laser". In: *International Semantic Web Conference*. Springer. 2017, pp. 87–103.

[9] Harald Beck. "Expressive rule-based stream reasoning". PhD thesis. Technische Universität Wien, 2018.

[10] Harald Beck, Minh Dao-Tran, and Thomas Eiter. "LARS: A logic-based framework for analytic reasoning over streams". In: *Artificial Intelligence* 261 (2018), pp. 16–70.

[11] Luigi Bellomarini, Markus Nissl, and Emanuel Sallinger. "iTemporal: an extensible generator of temporal benchmarks". In: *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE. 2022, pp. 2021–2033.

[12] Luigi Bellomarini, Markus Nissl, and Emanuel Sallinger. "Monotonic Aggregation for Temporal Datalog". In: *Proceedings of the International Rule Challenge, Industry Track, and Doctoral Consortium (RuleML+RR 2021)*. 2021, pp. 1–23.

[13] Luigi Bellomarini, Markus Nissl, and Emanuel Sallinger. *Query Evaluation in DatalogMTL: Taming Infinite Query Results*. 2021. arXiv: 2109.10691 [cs.DB].

[14] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. "The Vadalog system: Datalog-based reasoning for knowledge graphs". In: *Proceedings of the VLDB Endowment* (2018), pp. 975–987.

[15] Luigi Bellomarini et al. "The Temporal Vadalog System". In: *International Joint Conference on Rules and Reasoning*. 2022, pp. 130–145.

[16] Luigi Bellomarini et al. "The temporal vadalog system: Temporal datalog-based reasoning". In: *Theory and Practice of Logic Programming* 25.2 (2025), pp. 168–196.

[17] Meghyn Bienvenu, Camille Bourgaux, and Atefe Khodadaditaghanaki. *Inconsistency Handling in DatalogMTL*. 2025. arXiv: 2505.10394 [cs.LO].

[18] Sebastian Brandt et al. "Ontology-Based Data Access with a Horn Fragment of Metric Temporal Logic". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI Press. 2017, pp. 1070–1076.

[19] Sebastian Brandt et al. "Querying log data with metric temporal logic". In: *Journal of Artificial Intelligence Research* 62 (2018), pp. 829–877.

[20] Jan Chomicki and Tomasz Imielinski. "Temporal Deductive Databases and Infinite Objects". In: *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. 1988, pp. 61–73. DOI: 10.1145/308386.308416.

[21] Andrea Colombo et al. "LLM-based DatalogMTL Modelling of MiCAR-compliant Crypto-Assets Markets". In: *Proceedings of the 5th International Workshop on the Resurgence of Datalog in Academia and Industry, Datalog-2.0 2024*. CEUR-WS. 2024, pp. 17–22.

[22] Andrea Colombo et al. "Smart derivative contracts in datalogMTL". In: *Bank of Italy: Markets, Infrastructures, Payment Systems* 37 (2023).

[23] David J. Tena Cucala et al. "Stratified negation in Datalog with metric temporal operators". In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. 2021, pp. 6488–6495.

[24] Mathias De Brouwer et al. "Context-aware & privacy-preserving homecare monitoring through adaptive query derivation for IoT data streams with DIVIDE". In: *Semantic Web Journal* (2023).

[25] Ben De Meester et al. "Event-Driven Rule-Based Reasoning Using EYE". In: *Joint Proceedings of the Semantic Sensor Networks and Terra Cognita Workshop (SSN-TC 2015) and the Ordering and Reasoning Workshop (OrdRing 2015)*. Vol. 1488. CEUR Workshop Proceedings. 2015, pp. 75–86.

[26] Daniele Dell'Aglio et al. "RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems". In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.4 (2014), pp. 17–44.

[27] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Ana Ozaki. "On Metric Temporal Description Logics". In: *22nd European Conference on Artificial Intelligence*. 2016, pp. 837–845. DOI: 10.3233/978-1-61499-672-9-837.

[28] Alex Ivliev et al. "Nemo: your friendly and versatile rule reasoning toolkit". In: *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning*. 2024, pp. 743–754.

[29] Elem Gözel Kalaycı et al. "Ontop-temporal: a tool for ontology-based query answering over temporal data". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 1927–1930.

[30] R. Kontchakov et al. "Temporal and Spatial OBDA with Many-Dimensional Halpern-Shoham Logic". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2016, pp. 1160–1166.

[31] Matthias Lanzinger et al. "Temporal Datalog with existential quantification". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 2023, pp. 3277–3285.

[32] Freddy Lécué. "Diagnosing changes in an ontology stream: A DL reasoning approach". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 26.1 (2012), pp. 80–86.

[33] Marco Mori et al. "Neural Machine Translation for Fact-checking Temporal Claims". In: *Proceedings of the Fifth Fact Extraction and VERification Workshop*. 2022, pp. 78–82. DOI: "10.18653/v1/2022.fever-1.8".

[34] Yavor Nenov et al. "RDFox: A highly-scalable RDF store". In: *International Semantic Web Conference*. Springer. 2015, pp. 3–20.

[35] Matthias Nickles and Alessandra Mileo. "Web stream reasoning using probabilistic answer set programming". In: *International Conference on Web Reasoning and Rule Systems*. Springer. 2014, pp. 197–205.

[36] Markus Nissl and Emanuel Sallinger. "Modelling Smart Contracts with DatalogMTL". In: *Proceedings of the Workshops of the EDBT/ICDT*. 2022.

[37] Markus Nissl and Emanuel Sallinger. "Towards Bridging Traditional and Smart Contracts with Datalog-based Languages." In: *Datalog*. 2022, pp. 68–82.

[38] Thu-Le Pham, Muhammad Intizar Ali, and Alessandra Mileo. "C-ASP: continuous ASP-based reasoning over RDF streams". In: *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer. 2019, pp. 45–50.

[39] D Le Phuoc. "A native and adaptive approach for linked stream data processing". PhD thesis. NUI Galway, 2013.

[40] Katerina El Raheb et al. "BalOnSe: Temporal Aspects of Dance Movement and Its Ontological Representation". In: *European Semantic Web Conference*. 2017, pp. 49–64. DOI: 10.1007/978-3-319-58451-5_4.

[41] Vladislav Ryzhikov, Przemysław Andrzej Wałęga, and Michael Zakharyaschev. "Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics". In: *International Joint Conferences on Artificial Intelligence*. 2019, pp. 1851–1857.

[42] Veronika Thost. "Metric Temporal Extensions of DL-Lite and Interval-Rigid Names". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. 2018, pp. 665–666.

[43] Przemysław Wałęga, Michał Zawidzki, and Bernardo Cuenca Grau. "Finite materialisability of Datalog programs with metric temporal operators". In: *Journal of Artificial Intelligence Research* 76 (2023), pp. 471–521.

[44] Przemysław Andrzej Wałęga, Michał Zawidzki, and Bernardo Cuenca Grau. "Finite materialisability of Datalog programs with metric temporal operators". In: *Journal of Artificial Intelligence Research* 76 (2023), pp. 471–521.

[45] Przemysław Andrzej Wałęga et al. "DatalogMTL over the integer timeline". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. 2020, pp. 768–777.

[46] Przemysław Andrzej Wałęga et al. "DatalogMTL with Negation Under Stable Models Semantics". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. 2021, pp. 609–618. DOI: 10.24963/kr.2021/58.

[47] Przemysław Andrzej Wałęga et al. "DatalogMTL: computational complexity and expressive power". In: *International Joint Conferences on Artificial Intelligence*. 2019, pp. 1886–1892.

[48] Przemysław Andrzej Wałęga et al. "Materialisation-based reasoning in DatalogMTL with bounded intervals". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 2023, pp. 6566–6574.

[49] Przemysław Andrzej Wałęga et al. "Tractable fragments of Datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021, pp. 1919–1925.

[50] Dingmin Wang, Przemysław Andrzej Wałęga, and Bernardo Cuenca Grau. "MTLearn: extracting temporal rules using Datalog rule learners". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 21. 2024, pp. 962–973.

[51] Dingmin Wang et al. "Meteor: Practical reasoning in datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.5 (2022), pp. 5906–5913.

[52] Dingmin Wang et al. "Practical reasoning in DatalogMTL". In: *Theory and Practice of Logic Programming* 25.2 (2025), pp. 225–255.

[53] Shaoyu Wang et al. "Goal-Driven Reasoning in DatalogMTL with Magic Sets". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 2025, pp. 15203–15211.

[54] Kaiyue Zhao et al. "Incremental Maintenance of DatalogMTL Materialisations". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 40. 23. 2026, pp. 19467–19476.

5

On Syntactical Simplification of Temporal Operators in Negation-free Metric Temporal Logic

This chapter addresses the second foundational aspect of Research Challenge C2 by examining the role of negation and operator expressivity in temporal reasoning under open and distributed conditions. In many Stream Reasoning (SR) formalisms, negation is closely tied to the interpretation of absence, often relying on closed-world assumptions that allow missing information to be treated as falsity. In open environments, however, where streams may be incomplete or delayed, the absence of an observation cannot safely be interpreted as evidence that a condition does not hold. This disconnect motivates the study of negation-free temporal reasoning as a robust baseline for environments in which observability is partial and conclusions must remain stable as new information arrives.

Against this background, the chapter addresses Research Question RQ3 by shifting focus from comparing formalisms to analysing the expressive foundations of Metric Temporal Logic (MTL) itself. Specifically, it investigates expressivity relationships between temporal operators when MTL is restricted to negation-free fragments. By identifying which operators can be eliminated without loss of temporal expressivity within the considered negation-free fragments, the chapter substantiates Hypothesis H5, showing that a reduced operator set remains sufficient within negation-free fragments. These results clarify expressivity relationships between temporal operators under open-world assumptions and contribute to the broader goal of interoperability by identifying which operators must be preserved when relating negation-free temporal formalisms.

⋆ ⋆ ⋆

M. van Noort, F. Ongenae and P. Bonte

Published in the MDPI Mathematics Journal, March 2026.

5.1 Abstract

Temporal reasoning in dynamic, data-intensive environments increasingly demands expressive yet tractable logical frameworks. Traditional approaches often rely on negation to express absence or contradiction. In such contexts, negation-as-failure is commonly used to infer negative information from the lack of positive evidence. However, in open and distributed systems such as Internet-of-Things (IoT) networks or the Semantic Web (SW), negation-as-failure semantics become unreliable due to incomplete and asynchronous data. This has led to a growing interest in negation-free fragments of temporal rule-based systems, which preserve monotonicity and enable scalable reasoning. This paper investigates the expressive power of bounded negation-free MTL, a temporal logic framework designed for rule-based reasoning over time. We establish semantics-preserving rewritings of the "always" operators $\boxplus$ and $\boxminus$ in terms of the "once","since" and "until" operators within the bounded, negation-free fragments considered. . Additionally, we show that the bounded "once" operators can likewise be rewritten, yielding a fragment based solely on "until" and "since". These results challenge the assumption that negation is necessary for expressing universal temporal constraints, and reveal a robust fragment capable of capturing both existential and invariant temporal patterns. Furthermore, the results induce a reduction in the syntax of bounded, negation-free MTL, which in turn can provide benefits for both theoretical study as well as for implementation efforts.

5.2 Introduction

Negation is a widely used construct in formal, rule-based reasoning frameworks. It allows for the expression of absence, exclusion, or contradiction, and is often relied upon in a variety of application domains. For instance, in financial systems [3], sensor monitoring [17], or E-health services [13], the ability to express that something did not occur, or is not present, is often necessary. Without some form of negation, it becomes difficult to formally capture notions such as a no-show guest, a missing data point, or the absence of a required condition.

In logical rule-based systems, a common way to support negation is through the Closed World Assumption (CWA) [1]. Under this assumption, if a fact is not explicitly known to be true, it is assumed to be false. This type of negation, referred to as

Negation-as-Failure (NAF) is often a reasonable simplification in systems where the data is centrally managed and assumed to be complete [1].

However, recent developments in information systems have shifted attention toward open and distributed settings, where the CWA becomes harder to maintain [20, 21]. Examples include the World Wide Web, where data is distributed across many sources, and IoT environments [27, 31], where components may be numerous, heterogeneous, and subject to change. In such settings, assuming that all relevant information is available and up-to-date at all times is increasingly difficult to justify. This poses particular challenges for negation: the absence of a fact in one part of the system does not necessarily imply its global absence. For instance, a missing sensor reading may be due to transmission delay, device failure, or simply a temporary disconnection, rather than a true absence of the measured phenomenon. Similarly, in federated or decentralised systems, the lack of a record in one data source does not guarantee that the record does not exist elsewhere. These ambiguities undermine the reliability of NAF semantics, which depend on a complete and static view of the data. Moreover, the dynamic and asynchronous nature of distributed systems – where data sources may evolve independently – complicates the interpretation of negation and calls for more cautious reasoning strategies that take these dynamic and asynchronous aspects into account. As a result, frameworks for open systems require explicit mechanisms to handle partial knowledge and delayed information, further challenging traditional logical foundations of negation.

At the same time, the growing availability of real-time data has led to increased interest in temporal rule-based reasoning [7, 8]. Many phenomena of interest are not characterised by isolated facts, but by patterns that unfold over time. For example, Santipantakis et al. [29] observes the spatio-temporal positions of vessels in order to detect various types of vessel activities, and Bellomarini et al. [6] manages and evaluates evolutions in financial markets, most notably company ownership. This has motivated the development of temporal extensions to classical logic-based systems, based on either Linear Temporal Logic (LTL) or MTL. Some recent developments in this field include Linear Temporal Public Announcement Logic (LTPAL) [14], Metric Spatio-Temporal Logic (MSTL) [26], Logic-based framework for Analytic Reasoning over Streams (LARS) [5] and DatalogMTL [10].

In addition to the considerations of more open and distributed systems, negation also presents specific challenges in streaming environments, where data arrives incrementally and the system must reason over a continuously evolving state [15]. In such settings, the absence of a fact at a given moment does not necessarily imply its permanent absence – it may simply not have arrived yet. This makes it difficult to apply classical NAF semantics, which rely on a fixed and complete dataset. To address this, various stratified or windowed negation approaches have been proposed [12, 22, 34], where negation is restricted to certain layers of the program or applied only within bounded time intervals. These are, however, constructions that inherently treat

negation differently, with good reason: entailment in First-Order Logic (FOL) – and thereby in many rule-based reasoning systems – is undecidable [32]. By introducing negation, either via NAF or stratified negation, a reasoning framework risks undecidability and may compromise its reliability, especially when reasoning process must be both timely and robust in the face of incomplete information.

These challenges have led to a growing interest in identifying expressive yet tractable fragments of temporal rule-based systems that avoid negation altogether. In particular, researchers have explored 'positive fragments' – systems that exclude negation to preserve monotonicity and decidability. For example, Urbani, Krötzsch, and Eiter [33] study positive and stratified LARS programs, while work on LTL and MTL often focuses on negation-free fragments to enable efficient model checking [1, 4, 9].

While such fragments are intuitively less expressive, i.e. negation allows us to express opposites and contradictions, they remain able to express the same temporal expressions of their larger counterparts that include negation: A recurring theme in temporal logic is the equivalence between constructs like "always P" and "not once not P." For instance, in LTL, this is expressed as $\mathbf{G}\phi \equiv \neg\mathbf{F}(\neg\phi)$ [18], and in LARS, $\Box\alpha \equiv \neg \Diamond \neg\alpha$ [5]. These equivalences suggest that temporal "always" operators can be syntactic sugar for expressions involving negation and "once".

This raises an important question: what happens to such constructs when negation is removed? Must we reintroduce "always" as a core operator of the syntax? And if we do not, does this limit the expressive power of the system?

In this work, we study negation-free bounded MTL and show that the bounded "always" operators $\boxplus$ and $\boxminus$ can be rewritten in a semantics-preserving manner despite the absence of negation. This result indicates that a meaningful class of temporal expressions – specifically, those that describe properties that hold continuously over intervals – remains expressible by a limited set of temporal operators within negation-free bounded fragments of MTL. This shows that the expressivity of negation-free MTL is not as limited as intuition might suggest. They can capture not only existential or event-based patterns, but also universal temporal expressions. Furthermore, within the bounded fragments studied in this paper, we identify a characterisation based solely on the temporal operators $\mathcal{S}$ and $\mathcal{U}$.

Classical results in temporal logic relate "always" and "once" operators through negation (e.g. $\mathbf{G}\phi$ as $\neg\mathbf{F}(\neg\phi)$ in LTL), and several works in the MTL literature discuss how temporal operators can be inter-expressed under the assumption that negation is available [5, 19, 23]. However, these existing results do not address whether such relationships continue to hold in negation-free MTL. In particular, the question of whether universal operators such as $\boxplus$ and $\boxminus$ can be eliminated without relying on negation has, to our knowledge, not been answered before. More generally, syntactical reduction of systems, as considered in the wider literature of model theory [16, 24, 25], has remained unexplored with regard to MTL-based systems as a whole.

The results presented in this paper demonstrate that negation-free bounded MTL is more expressive than previously assumed: within this fragment, $\boxplus$ and $\boxminus$ can be rewritten using only $\mathcal{S}$ and $\mathcal{U}$, and the "once" operators are similarly characterised through the remaining operators. Moreover, by providing punctuality-free variants of these transformations, we extend the operator-elimination results to negation-free, bounded Metric Interval Temporal Logic (MITL) and related fragments that forbid singleton intervals for reasons of robustness and decidability. These findings provide a more precise characterisation of operator dependencies in negation-free bounded MTL and identify a reduced set of operators for bounded and negation-free reasoning.

From a practical perspective, the results imply that implementations targeting the studied fragments can be based on two temporal operators rather than the larger operator set typically provided by MTL. . This reduction in the number of primitive operators opens the door towards a slimmer and more maintainable codebase. By characterising expressivity-preserving rewritings within negation-free bounded fragments of MTL, we illustrate a viable negation-free bounded fragment of MTL exists, which adequately balances expressivity with achieving a scalable reasoning system.

The remainder of this paper is structured as follows; We first familiarize the reader with the syntax and semantics of DatalogMTL, as defined in the literature, in Section 5.3. The theoretical rewritings, supported by their formal proofs, are presented in Section 5.4. We discuss the implications of our results in Section 5.5 , with a concluding note in Section 5.6.

5.3 Materials and Methods

MTL, introduced by Koymans [23], introduced a framework for quantitative temporal reasoning in real-time systems. It has since found its way into multiple reasoning systems that deal with temporal data or changing states, such as DatalogMTL [10], Predictive MTL (P-MTL) [30] and Metric Equilibrium Logic (MEL) [11]. We briefly recall the syntax and semantics as it has been adopted throughout the literature [2, 10, 23]. Within the scope of this work, we opt for a model-theoretic semantics and rational timeline $(\mathbb{Q}, <)$.

Definition 5.1. *Given a list of predicate symbols $\mathcal{P}$, a model M is a triple $(\mathbb{Q}, <, V)$ where V is a mapping from time points in $\mathbb{Q}$ to a subset of predicates in $\mathcal{P}$.*

Example 5.1. *By means of example, we consider a setting of healthcare monitoring using IoT data. Information is collected on a single patient's heart rate, blood pressure, whether they received medication. When the medical circumstances require action, an alarm can be sent out. This data is captured using the set of predicates $P = \{hrHigh, bpLow, medGiven, alarm\}$ over a rational timeline $\mathbb{Q}$ with the usual order $<$.*

A model M represents a particular course of events, where pieces of the above information are observed at various moments in time; let $M = (\mathbb{Q}, <, V)$ where V assigns the following (event-time) valuations near $t = 10$ (minutes): $V(7) = \{hrHigh\}$, $V(8) = \{hrHigh, bpLow\}$, $V(9) = \{medGiven\}$, $V(10) = \{hrHigh\}$, and $V(t) = \emptyset$ otherwise.

Within the scope of this work, we consider Bounded MTL (BMTL), which only considers bounded intervals in the subscript of temporal operators:

Definition 5.2. *The set of temporal formulae is defined recursively as follows:*

$$A \coloneqq p|\top|\neg A|A \wedge A|\boxplus_I A|\boxminus_I A|\diamondplus_I A|\diamondminus_I A|A\mathcal{S}_I A|A\mathcal{U}_I A$$

where I is a positive bounded interval over T, i.e. $I = [i_1, i_2]$ with $i_1, i_2 \in T$ and $0 \leq i_1 \leq i_2$.

Example 5.2. *Using Definition 5.2, we are able to describe the medical evolution of our patient in greater detail. Predicates in P capture information of a single moment in time. Formulae allow us to capture temporal patterns, such as*

$\diamondminus_{[0,2]}\,(hrHigh \wedge bpLow)$	*"high heart rate and low blood pressure occurred together at least once in the last 2 min",*
$(alarm)\mathcal{S}_{[0,5]}(hrHigh \wedge bpLow)$	*"the alarm has been on since the last joint occurrence of high heart rate and low blood pressure within 5 min".*

BMTL can be considered a fragment of "full" MTL, which also considers intervals of the form $[0, +\infty[$ in the subscript. A summary of BMTL and several other fragments and respective theoretical properties can be found in Ouaknine and Worrell [28].

Following a model-theoretic approach, we define the semantics of temporal formulae A by Table 5.1, with M an interpretation, t, t' and t'' time points on timeline T and I a positive bounded interval within T.

What can occur is that two formulae appear to be different, but have the same semantics. An example thereof is $\boxplus_{[0,i]}A$ and $A\mathcal{U}_{[i,i]}\top$; the first says A must hold from t (now) up to $t + i$. The latter says A must hold from t up until when $\top$ holds in interval $[t + i, t + i]$. Since $\top$ holds for any time point, it also does so for $t + i$, meaning the latter states A holds from t up until $t + i$. We refer to such formulae with identical semantics as *equivalent*, formally defined as follows:

Definition 5.3. *Two formulae A_1 and A_2 are considered to be equivalent if (and only if) for every model M and for every time point t, it holds that A_1 is true at t in M if A_2 is true at t in M and vice versa. We denote this by writing $A_1 \equiv A_2$.*

Table 5.1: Semantics of MTL formulae

Atom	Semantics
Predicate	$M, t \models p$ if $p \in V(t)$
True	$M, t \models \top$ for each t
Negation	$M, t \models \neg A$ iff $M, t \models A$ does not hold
Conjunction	$M, t \models A_1 \wedge A_2$ iff $M, t \models A_1$ and $M, t \models A_2$
Once (past)	$M, t \models \diamondminus_I A$ iff $M, t' \models A$ for at least one t' with $t - t' \in I$
Once (future)	$M, t \models \diamondplus_I A$ iff $M, t' \models A$ for at least one t' with $t' - t \in I$
Always (past)	$M, t \models \boxminus_I A$ iff $M, t' \models A$ for all t' with $t - t' \in I$
Always (future)	$M, t \models \boxplus_I A$ iff $M, t' \models A$ for all t' with $t' - t \in I$
Since	$M, t \models A_1 \, \mathcal{S}_I \, A_2$ iff $M, t' \models A_2$ for at least one t' with $t - t' \in I$ and $M, t'' \models A_1$ for all $t'' \in [t', t]$
Until	$M, t \models A_1 \, \mathcal{U}_I \, A_2$ iff $M, t' \models A_2$ for at least one t' with $t' - t \in I$ and $M, t'' \models A_1$ for all $t'' \in [t, t']$

Example 5.3. *To illustrate the notion of equivalence from Definition 5.3, we continue the healthcare monitoring scenario introduced above.*

Consider the following two formulae:

$$A \;:=\; \boxminus_{[0,2]}\, hrHigh \qquad \textit{and} \qquad B \;:=\; \big(hrHigh\big)\mathcal{S}_{[0,2]}\top$$

Intuitively, both A and B express the same medical condition: "the patient's heart rate has been continuously high throughout the last two minutes". Formula A states this directly using the past "always" operator. Formula B expresses the same requirement using only the "since" operator: it looks for some time point in the last two minutes where $\top$ held. Since $\top$ always holds, the required time point is found as early as exactly two minutes ago. B then expresses that, since exactly two minutes ago, $hrHigh$ has held up until now, without interruption.

For the model M specified earlier, we indeed have that for each $t' \in [8, 10]$, $M, t' \models hrHigh$, meaning that both A and B hold at $t = 10$. More generally, by evaluating both formulae over all models and time points, one sees that A is true at (M, t) if and only if B is true at (M, t). Therefore, $A \equiv B$.

Table 5.1 does not specify the exact interpretation for "$M, t \models A$ does *not* hold". Each system can decide what is understood as "it does not hold". As considered in the 5.2 Introduction, closed systems often employ a negation-as-failure approach; when there is no conclusive evidence in favour of $M, t \models A$, it assumes $M, t \models A$ does not hold. Imagine for example the reservation list of a restaurant; if there is no evidence you do have a reservation, then the waiter assumes that you do not have a reservation. NAF is not always a desired approach, since it may lead to premature conclusions. Perhaps you do have a reservation, but the waiter's list is outdated. In more open-ended systems, e.g . IoT or the Web, it is preferable to only say $M, t \models A$ does not hold when there is explicit evidence that says so. For example, you may

not have a reservation for eight people, since there is specific evidence against it: your reservation is for two people.

5.4 Results

As indicated in Section 5.2, rewriting operators in function of other operators is a known practice. For example, Gutierrez Basulto, Jung, and Ozaki [19] reiterates the syntax of $\mathrm{LTL}^{\mathrm{bin}}_{\mathcal{ALC}}$, where operations $\Diamond_I C$ and $\Box_I C$ are introduced merely as a shorthand for $\top \mathcal{U}_I C$ and $\neg\big(\Diamond_I \neg C\big)$. Similarly, Finger and Gabbay [18] define their respective temporal operators solely in function of $\mathcal{U}$ and $\mathcal{S}$, i.e. $\mathbf{F}(A) := \mathcal{U}(A, \top)$ and $\mathbf{P}(A) := \mathcal{S}(A, \top)$ for propositional temporal logics. Koymans [23] notes the same relations hold in MTL: $\top \mathcal{U}_I A \equiv \oplus_I A$ and $\top \mathcal{S}_I A \equiv \ominus_I A$. These equivalencies migrate to other frameworks as well, as seen in DatalogMTL [10]. These results can be summarized as follows:

Proposition 5.1. *Suppose $t \in T$, I a non-negative interval and A a formula. Then $\oplus_I A \equiv \top \mathcal{U}_I A$ and $\ominus_I A \equiv \top \mathcal{S}_I A$.*

In Gutierrez Basulto, Jung, and Ozaki [19], the MTL operator $\boxplus A$ is defined via the equivalence $\boxplus A \equiv \neg(\oplus \neg A)$. This formulation relies on the presence of negation to express "always A". However, in the absence of negation, such an equivalence cannot be reproduced within MTL, which raises the natural question: if we exclude negation $\neg$ *and* the "always" operators $\boxplus$ and $\boxminus$ from the syntax of MTL, does this result in a strictly less expressive fragment compared to the full language?

The following theorems and proofs demonstrate that , within the negation-free bounded fragments considered in this chapter, these operators can be rewritten without loss of temporal expressivity. This demonstrates that a range of universal temporal properties remains expressible in these fragments despite the absence of negation.

Theorem 5.2. *Suppose t a time point, $I = [i_1, i_2]$ a non-negative interval and A a formula. Then*

$$\boxplus_I A \equiv \oplus_{[i_1,i_1]}\Big(A \mathcal{U}_{[i_2-i_1,i_2-i_1]} \top\Big) \equiv \top \mathcal{U}_{[i_1,i_1]}\Big(A \mathcal{U}_{[i_2-i_1,i_2-i_1]} \top\Big).$$

Proof. Consider an arbitrary model M and time point t. By the semantics of DatalogMTL, we obtain the following.

$$
\begin{aligned}
& M,t \models \oplus_{[i_1,i_1]}\Big(A\mathcal{U}_{[i_2-i_1,i_2-i_1]}\top\Big) \\
\Leftrightarrow \quad & \exists t' \in [t+i_1, t+i_1] : M,t' \models A\mathcal{U}_{[i_2-i_1,i_2-i_1]}\top \\
\Leftrightarrow \quad & \exists t' \in [t+i_1, t+i_1] : \Big(\exists t^{(2)} \in [t'+(i_2-i_1), t'+(i_2-i_1)] : \\
& (M,t^{(2)} \models \top) \wedge (\forall t^{(3)} \in [t', t^{(2)}] : M,t^{(3)} \models A)\Big) \\
\Leftrightarrow \quad & \exists t^{(2)} \in [t+i_2, t+i_2] : \\
& \Big((M,t^{(2)} \models \top) \wedge (\forall t^{(3)} \in [t+i_1, t^{(2)}] : M,t^{(3)} \models A)\Big) \\
\Leftrightarrow \quad & \forall t^{(3)} \in [t+i_1, t+i_2] : M,t^{(3)} \models A \\
\Leftrightarrow \quad & M,t \models \boxplus_I A
\end{aligned}
$$

The proof works as follows. In order for the equivalence to hold, the two formulae must hold (and not hold) simultaneously for every model and time point. If this can be proven for an arbitrary pair of model and time point, it must hold for every pair. Hence, we consider an unspecified model M and time point t for which $M,t \models \oplus_{[i_1,i_1]}\big(A\mathcal{U}_{[i_2-i_1,i_2-i_1]}\top\big)$. Following the semantics of Section 5.3, there exists a future time point t', such that $t'-t \in [i_1,i_1]$, in other words $t' = t+i_1$, for which $M,t' \models A\mathcal{U}_{[i_2-i_1,i_2-i_1]}\top$. By the semantics of the $\mathcal{U}$ operator, this is equivalent to first the existence of a time point $t^{(2)}$ for which $t^{(2)}-t' \in [i_2-i_1, i_2-i_1]$, in other words $t^{(2)} = t'+(i_2-i_1)$. By substituting the fact that $t' = t+i_1$, we get $t^{(2)} = t+i_2$. For this $t^{(2)}$ $M,t^{(2)} \models \top$ holds. Second, it entails that for every $t^{(3)}$ between t' and $t^{(2)}$, it holds that $M,t^{(3)} \models A$. Since $\top$ holds for any M and $t^{(2)}$ and $t^{(2)}$ can only be $t+i_2$, this is equivalent to $M,t^{(3)} \models A$ for every $t^{(3)}$ in interval $[t+i_1, t+i_2]$. In other words, $M,t \models \boxplus_{[i_1,i_2]}A$. It is easily verified each step likewise holds in the opposite direction. Lastly, the second equivalence follows from Proposition 5.1 by substituting the $\oplus$ operator.

□

We obtain an analogous result for $\boxminus$:

Theorem 5.3. *Suppose t a time point, $I = [i_1,i_2]$ a non-negative interval and A a formula. Then*

$$\boxminus_I A \equiv \ominus_{[i_1,i_1]}\Big(A\mathcal{S}_{[i_2-i_1,i_2-i_1]}\top\Big) \equiv \top\mathcal{S}_{[i_1,i_1]}\Big(A\mathcal{S}_{[i_2-i_1,i_2-i_1]}\top\Big).$$

Following Theorems 5.2 and 5.3, any negation-free formula in BMTL has an equivalent formula free of operators $\boxplus$ and $\boxminus$. Combined with Proposition 5.1, a negation-free BMTL fragment can be defined using only $\mathcal{S}$ and $\mathcal{U}$;

$$A \coloneqq p \mid \top \mid A \wedge A \mid A\mathcal{S}_I A \mid A\mathcal{U}_I A$$

Within negation-free BMTL, the same temporal relations can be expressed using a reduced operator set. The only compromise is situated within the non-temporal relations; $A_1 \vee A_2$, often achieved via $\neg(\neg A_1 \wedge \neg A_2)$ can not be expressed in negation-free BMTL.

Theorems 5.2 and 5.3 express $\boxplus$ and $\boxminus$ using singleton intervals such as $[i_1, i_1]$ and $[i_2 - i_1, i_2 - i_1]$. These punctual constraints are admissible in BMTL but are disallowed in MITL and its subfragments, where the syntax explicitly forbids singleton intervals. Therefore, to obtain analogous eliminations of $\boxplus$ and $\boxminus$ within Bounded MITL, we must develop alternative equivalences that avoid punctual constraints entirely. The next theorems provide such constructions. However, the restriction increases the intricacy of the matter, resulting in equivalencies and proofs that stray further from intuition than the previous theorems.

The idea of Theorem 5.4 is based on the following observation: $\boxplus_{[i_1,i_2]}A$ at time t means A holds over interval $[t + i_1, t + i_2]$, meaning A started somewhere before, or at the latest at, $t + i_1$ and end no earlier than $t + i_2$. We split this expression into two parts, one that states A starts holding at the latest at $t + i_1$ and another that states A stops holding at the earliest at $t + i_2$, with the condition that the two conditions must overlap to ensure that A holds continuously over the entire interval. For the first part, we postulate that A needs to hold for at least a length $i_2 - i_1$, starting somewhere between $t + i_1 - \frac{i_2 - i_1}{2}$ and $t + i_1$, meaning A will hold until at least $t + i_1 + \frac{i_2 - i_1}{2}$, the halfway point of the interval $[t + i_1, t + i_2]$. Likewise, the second part states A should hold for $i_2 - i_1$ time points, since at least $t + i_1 + \frac{i_2 - i_1}{2}$. These two parts are illustrated in Figure 5.1.

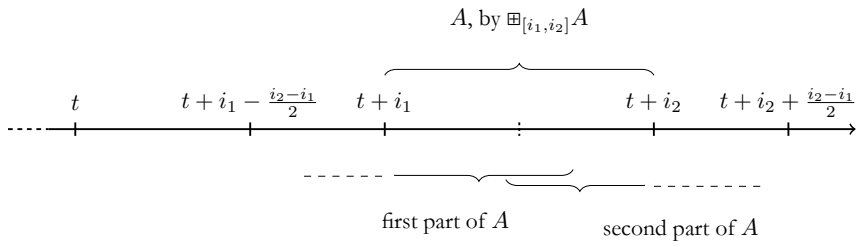


Figure 5.1: Illustration of Proof 5.2. The accolades of parts 1 and 2 have a set length of $i_2 - i_1$, but shift according to the exact location of t^- and t^+. Regardless of the exact location of t^- and t^+, the two accolades will always produce an area spanning at least $[t + i_1, t + i_2]$ in which A holds everywhere.

To clarify the construction further, we explicitly indicate the extremal endpoints enforced by each conjunct in Theorem 5.4. The first conjunct ensures that every witness t^- satisfies

$$t^- \leq t + i_1 \quad \text{and} \quad t^- + (i_2 - i_1) \geq t + i_1 + \tfrac{i_2 - i_1}{2},$$

so the minimal right endpoint covered is exactly $t + i_1 + \frac{i_2 - i_1}{2}$ and the maximal left endpoint is $t + i_1$. Symmetrically, the second conjunct enforces that every t^+ satisfies

$$t^+ \geq t + i_2 \quad \text{and} \quad t^+ - (i_2 - i_1) \leq t + i_1 + \tfrac{i_2 - i_1}{2}.$$

Thus, its maximal left endpoint is $t + i_1 + \frac{i_2 - i_1}{2}$ and its minimal right endpoint is $t + i_2$. Together, these yield exactly the interval $[\,t + i_1,\ \ t + i_2\,]$ as the portion necessarily covered by A.

In the following proof, we show these two parts translate to expressions without box operators or singleton intervals.

Theorem 5.4. *Suppose t a time point, $I = [i_1, i_2]$ a non-negative interval, κ and λ positive time points and A a formula. Then*

$$\begin{aligned}\boxplus_{[i_1,i_2]} A \equiv &\Big(\oplus_{[\frac{3i_1 - i_2}{2}, i_1]} \left(A \mathcal{U}_{[i_2 - i_1, i_2 - i_1 + \kappa]} \top \right) \Big) \\ &\wedge \Big(\oplus_{[i_2, \frac{3i_2 - i_1}{2}]} \left(A \mathcal{S}_{[i_2 - i_1, i_2 - i_1 + \lambda]} \top \right) \Big).\end{aligned}$$

Proof. For clarity, we structure the argument as two implications:

$$\boxplus_{[i_1,i_2]} A \;\Rightarrow\; \Phi \qquad \text{and} \qquad \Phi \;\Rightarrow\; \boxplus_{[i_1,i_2]} A,$$

where Φ denotes the conjunction of the two interval-indexed $\oplus$-$\mathcal{U}$ and $\oplus$-$\mathcal{S}$ components.

Let κ and λ be arbitrary but fixed positive time points. We consider the following two formulae:

$$\oplus_{\left[\frac{3i_1 - i_2}{2}, i_1\right]} \left(A \mathcal{U}_{[i_2 - i_1, i_2 - i_1 + \kappa]} \top \right) \tag{5.1}$$

$$\oplus_{\left[i_2, \frac{3i_2 - i_1}{2}\right]} \left(A \mathcal{S}_{[i_2 - i_1, i_2 - i_1 + \lambda]} \top \right) \tag{5.2}$$

If for a given model M, formula (5.1) holds at time t in M, semantics assert that there exists a time point within the interval $\left[t + \frac{3i_1 - i_2}{2}, t + i_1\right]$, say t^-, at which the formula $A \mathcal{U}_{[i_2 - i_1, i_2 - i_1 + \kappa]} \top$ holds. This means that, starting from t^-, A must hold continuously for a duration of at least $i_2 - i_1$ time units, until a point is reached where the trivially true formula $\top$ holds in M.

Dually, formula (5.2) captures a symmetric condition. It states that if the formula (5.2) holds in M at time t, there exists a time point t^+ within the interval $\left[t + i_2, t + \frac{3i_2 - i_1}{2}\right]$ such that the formula $A \mathcal{S}_{[i_2 - i_1, i_2 - i_1 + \lambda]} \top$ holds in M. This implies that, looking backward from t^+, the proposition A must have held continuously for a duration of at least $i_2 - i_1$, starting from a point where $\top$ is satisfied.

Observe that the lower bound of the interval $\left[t + \frac{3i_1 - i_2}{2}, t + i_1\right]$ can be rewritten as

$$t + \frac{3i_1 - i_2}{2} = t + \frac{3i_1}{2} - \frac{i_2}{2} = t + i_1 + \frac{i_1}{2} - \frac{i_2}{2}$$
$$= t + i_1 + \frac{i_1 - i_2}{2} = t + i_1 - \frac{i_2 - i_1}{2}.$$

This latter expression identifies the time point that lies exactly one interval length of $i_2 - i_1$ before the halfway point of the interval $[t + i_1, t + i_2]$.

From any time point within the range $[t + i_1 - \frac{i_2 - i_1}{2},\ t + i_1]$, the formula in Equation (5.1) requires a forward temporal progression (a "jump") of at least $i_2 - i_1$ time units. This jump begins no later than $t + i_1$ and, in the minimal case permitted by the interval constraints, lands at $t + i_1 - \frac{i_2 - i_1}{2} + (i_2 - i_1) = t + i_1 + \frac{i_2 - i_1}{2}$, i.e. the midpoint of the interval $[t + i_1, t + i_2]$. Since the proposition A is required to hold throughout the entire duration of this jump, it follows that A must hold continuously over at least the first half of the interval $[t + i_1, t + i_2]$, that is, from $t + i_1$ to $t + i_1 + \frac{i_2 - i_1}{2}$. It is important to note that this condition provides no information on the truth value of A prior to $t + i_1$, nor beyond $t + i_1 + \frac{i_2 - i_1}{2}$.

The same method of interpretation applies to formula (5.2). In this case the upper bound interval can be similarly rewritten, i.e.

$$t + \frac{3i_2 - i_1}{2} = t + \frac{3i_2}{2} - \frac{i_1}{2} = t + i_2 + \frac{i_2}{2} - \frac{i_1}{2} = t + i_2 + \frac{i_2 - i_1}{2}.$$

From any time point within the range $[t + i_2, t + i_2 + \frac{i_2 - i_1}{2}]$, the formula in Equation (5.2) requires a backwards "jump" of at least $i_2 - i_1$ time units. As such, this jump back ends at the latest at $t + i_2$ and starts at the latest at $t + i_2 + \frac{i_2 - i_1}{2} - (i_2 - i_1) = t + i_2 - \frac{i_2 - i_1}{2}$, i.e. the midpoint of the interval $[t + i_1, t + i_2]$. Again, A must continuously hold during this jump. As such, A holds continuously from $t + i_2 - \frac{i_2 - i_1}{2}$ to $t + i_2$, which is the latter half of the interval $[t + i_1, t + i_2]$.

As a result, if the conjunction of (5.1) and (5.2) holds for a model M at time point t, then A holds in M over (at least) the interval $[t + i_1, t + i_2]$, i.e. $M, t \models \boxplus_{[i_1, i_2]} A$. Crucially, the endpoints we use are minimal and maximal in the following precise sense: any earlier start would shift the $\mathcal{U}$-jump too far left and fail to reach the midpoint, and any later start would be outside the allowed interval for the witness. Likewise, any later end for the $\mathcal{S}$-jump would fail to reach back to the midpoint. Therefore no strictly larger interval than $[t + i_1,\ t + i_2]$ is enforced by the construction.

Suppose that $M, t \models \boxplus_{[i_1, i_2]} A$. By the semantics of the $\boxplus$ operator, A holds at every time point within the interval $[t + i_1, t + i_2]$. Furthermore, since $\top$ is valid at all time points, it trivially holds that $M, t + i_2 \models \top$. Observe that $t + i_2$ lies within the interval $[t + i_2, t + i_2 + \kappa]$, and thus the condition $A \mathcal{U}_{[i_2 - i_1, i_2 - i_1 + \kappa]} \top$ is satisfied

at time $t + i_1$. Moreover, since $t + i_1$ belongs to the interval $[t + \frac{3i_1 - i_2}{2}, t + i_1]$, it follows that formula (5.1) holds at time t.

We now turn our attention to Equation (5.2). Note that $t + i_1$ lies within the interval $[t+i_1-\lambda, t+i_1]$, which can be equivalently expressed as $[(t+i_2)-(i_2-i_1+\lambda), (t+i_2)-(i_2-i_1)]$. Given that A holds throughout $[t+i_1, t+i_2]$ and $\top$ holds at $t+i_2$, it follows that $M, t+i_2 \models A\mathcal{S}_{[i_2-i_1, i_2-i_1+\lambda]}\top$, and consequently formula (5.2) holds in M at time t. Since $t + i_2$ lies within the interval $[t + i_2, t + \frac{3i_2-i_1}{2}]$, we conclude that there exists a time point t^+ in the interval $[t+i_2, t+\frac{3i_2-i_1}{2}]$ for which $A\mathcal{S}_{[i_2-i_1, i_2-i_1+\lambda]}\top$ holds in M at time t^+. As a result, formula (5.2) holds in M at time point t.

$M, t \models \boxplus_{[i_1,i_2]}A$ thus induces that both formulae (5.2) and (5.1) hold in M at time t, and therefore their conjunction as well.

□

Analogously, $\boxminus_{[i_1,i_2]}A$ can be rewritten into a similar expression.

Theorem 5.5. *Suppose $t \in T$, $I = [i_1, i_2]$ a non-negative interval, κ, λ positive time points and A a formula. Then*

$$\boxminus_{[i_1,i_2]}A \equiv \Big(\diamondminus_{[\frac{3i_1-i_2}{2}, i_1]} \left(A\mathcal{S}_{[i_2-i_1, i_2-i_1+\lambda]}\top\right)\Big) \wedge \Big(\diamondminus_{[i_2, \frac{3i_2-i_1}{2}]} \left(A\mathcal{U}_{[i_2-i_1, i_2-i_1+\kappa]}\top\right)\Big).$$

Proof. The proof of Theorem 5.5 is analogous to that of Theorem 5.4 . □

It follows from Theorems 5.4 and 5.5 that any formula with bounded interval subscripts can be rewritten into formulae containing only the temporal operators $\diamondplus$, $\diamondminus$, $\mathcal{S}$ and $\mathcal{U}$. Combined with Proposition 5.1, this shows that negation-free BMTL and Bounded MITL admit equivalent formulations based solely on $\mathcal{S}$ and $\mathcal{U}$.

5.5 Discussion

The results above show show that, within the negation-free bounded fragments considered in this chapter, the universal temporal operators can be rewritten using other temporal operators. While classical transformations of "always" operators into "once" and negation are well-known in temporal logic, our findings reveal that similar reductions remain possible within the negation-free bounded fragments studied here. This illustrates "always" statements remain expressible without negation in the bounded fragments considered. Given the influence of MTL on recent temporal reasoning frameworks, the obtained results can be carried over to MTL-based frameworks such as MEL and DatalogMTL.

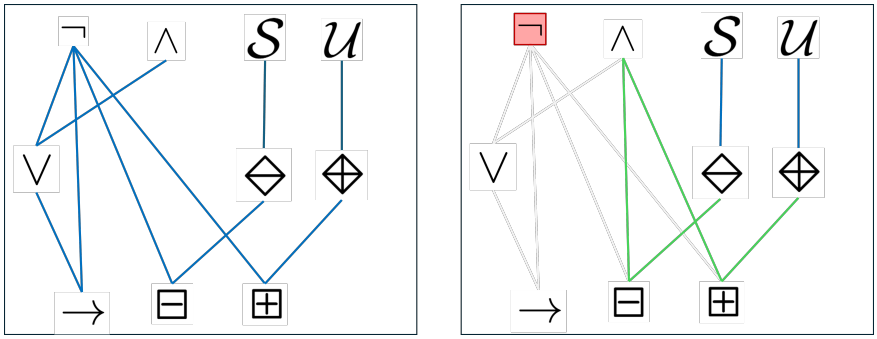


Figure 5.2: Expressibility dependencies between temporal operators under closed-world (left) and open-world (right) assumptions. Blue edges denote previously known dependencies, while green edges denote newly established dependencies.

The expressivity consequences of the results obtained in this chapter are visually captured in Figure 5.2. Traditional closed-world reasoning environments can support the full collection of temporal operators because negation allows universal modalities to be expressed in terms of existential ones, as shown on the left-hand side,. In contrast, in open-world settings, the absence of negation appears to remove access to the universal modalities $\boxplus$ and $\boxminus$. The results of this chapter show that this view is too restrictive for the bounded fragments considered. By showing that $\boxplus$ and $\boxminus$ can be expressed using only $\mathcal{S}$ and $\mathcal{U}$, we establish that these operators remain available in negation-free MTL despite the absence of negation. Consequently, the negation-free bounded fragments studied in this chapter remain capable of expressing both existential and certain universal temporal patterns using only $\mathcal{S}$ and $\mathcal{U}$. The practical implication is that reasoning frameworks intended for open and distributed environments, such as the Semantic Web, IoT ecosystems, and data spaces, do not need to sacrifice the ability to express persistence and invariance conditions merely because they avoid closed-world assumptions.

The constructions following Theorems 5.2 and 5.3 rely on punctual intervals, i.e. singleton bounds of the form $[i, i]$. While these are admissible in BMTL, their use is well-known to introduce subtle semantic and computational issues. In dense time domains, punctual requirements make formula satisfaction fragile: an event must occur at exactly one time point, so even infinitesimal timestamp perturbations may invalidate the formula. This lack of robustness has been repeatedly highlighted in the MTL literature. More importantly, as shown by Alur, Feder, and Henzinger [2] and by Ouaknine and Worrell [28], allowing punctual constraints renders satisfiability and model checking for MTL undecidable. This motivated the development of MITL, which explicitly excludes singleton intervals in order to recover decidability. Consequently, while Theorems 5.2–5.3 establish that $\boxplus$ and $\boxminus$ can be eliminated within negation-free BMTL using punctual constraints, these transformations are not applicable to MITL

or its well-behaved subfragments. This observation motivates the alternative equivalences presented in Theorems 5.4–5.5. These results show that, even under the stricter MITL syntax, equivalent formulations of the universal operators can be constructed, albeit via more intricate transformations. This extends the operator-reduction results to the well-studied decidable fragments of MTL.

Looking forward, the syntactical simplifications lead to several avenues for future research. From a theoretical perspective, it is an open question whether analogous syntactic reductions can be achieved in temporal reasoning frameworks with different temporal operators or semantics, for example LARS. This would further prompt us to reconsider the theoretical minimal requirements of temporal logic frameworks for open-world infrastructures such as IoT and the SW. These theoretical results help us better understand the strengths and weaknesses of SR systems, which in turn allows us to better target research efforts towards more efficient, scalable, and predictable reasoning systems for dynamic and data-intensive environments.

Approached from a more practical angle, the expressions obtained in this paper can be leveraged in MTL-based frameworks; via the obtained expressions, the more complex temporal operators $\boxplus, \boxminus, \diamondplus, \diamondminus$ can be implemented as "built-ins" on top of the existing theory, to improve readability (and thereby user convenience) without modifying the underlying theory. Finally, an interesting direction for future work is to explore whether similar syntactic simplifications can be extended to unbounded intervals (i.e., $[0, +\infty[$). In the present work, the key idea relies on reconstructing a bounded interval by covering it from both endpoints. This approach fundamentally presupposes boundedness, since there is no "right endpoint" in intervals such as $[0, +\infty[$. Extending the method would therefore require qualitatively different techniques, most likely ones that reason about asymptotic behavior or employ limit-style constraints rather than interval reconstruction. Exploring such alternative approaches opens promising possibilities for broadening the applicability of the results presented here.

5.6 Conclusion

In this work, we have shown that negation-free bounded fragments of MTL admit substantial syntactic simplifications while preserving temporal expressivity within those fragments. Specifically, we propose equivalent rewritings of the temporal operators $\boxplus$ and $\boxminus$ in terms of combinations of $\diamondplus$, $\diamondminus$, $\mathcal{U}$, and $\mathcal{S}$. This is the case for negation-free BMTL, as follows from Theorems 5.2-5.3, but also for the more restrictive Bounded MITL, as illustrated by Theorems 5.4-5.5. Furthermore, using known expressibility relationships between existential modalities and the operators $\mathcal{S}$ and $\mathcal{U}$, we showed that the considered fragments admit formulations based solely on the $\mathcal{S}$ and $\mathcal{U}$ operators.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN: 0-201-53771-0.

[2] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. "The Benefits of Relaxing Punctuality". In: *Journal of the ACM (JACM)* 43.1 (1996), pp. 116–146.

[3] Teodoro Baldazzi, Luigi Bellomarini, and Emanuel Sallinger. "Reasoning over Financial Scenarios with the Vadalog System". In: vol. 26. OpenProceedings. org. Konstanz, Germany, 2023, pp. 782–791.

[4] David Basin et al. "Monitoring Metric First-Order Temporal Properties". In: *Journal of the ACM (JACM)* 62.2 (2015), pp. 1–45.

[5] Harald Beck, Minh Dao-Tran, and Thomas Eiter. "LARS: A logic-based framework for analytic reasoning over streams". In: *Artificial Intelligence* 261 (2018), pp. 16–70.

[6] Luigi Bellomarini et al. "The Temporal Vadalog System: Temporal Datalog-Based Reasoning". In: *Theory and Practice of Logic Programming* 25.2 (2025), pp. 168–196.

[7] Pieter Bonte et al. "Grounding Stream Reasoning Research". In: *Transactions on Graph Data and Knowledge* 2 (2024), pp. 1–47.

[8] Pieter Bonte et al. "Languages and systems for RDF stream processing, a survey". In: *The VLDB Journal* 34.4 (2025), p. 50.

[9] Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. "On the Expressiveness of TPTL and MTL". In: *Information and Computation* 208.2 (2010), pp. 97–116.

[10] Sebastian Brandt et al. "Querying log data with metric temporal logic". In: *Journal of Artificial Intelligence Research* 62 (2018), pp. 829–877.

[11] Pedro Cabalar et al. "Metric temporal answer set programming over timed traces". In: *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer. 2022, pp. 117–130.

[12] David J. Tena Cucala et al. "Stratified negation in Datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.7 (2021), pp. 6488–6495.

[13] Mathias De Brouwer et al. "Context-Aware and Privacy-Preserving Homecare Monitoring through Adaptive Query Derivation for IoT Data Streams with DIVIDE". In: *Semantic Web Journal* (2019).

[14] Amirhoshang Hoseinpour Dehkordi, Majid Alizadeh, and Ali Movaghar. *Linear Temporal Public Announcement Logic: A New Perspective for Reasoning the Knowledge of Agents*. 2020. arXiv: 2009.03793 [cs.LO].

[15] Emanuele Della Valle et al. "It's a streaming world! Reasoning upon rapidly changing information". In: *IEEE Intelligent Systems* 24.6 (2009), pp. 83–89.

[16] Klaus Denecke. "Partial clones". In: *Asian-European Journal of Mathematics* 13.08 (2020), p. 2050161.

[17] Patrick Doherty et al. "Hdrc3: A Distributed Hybrid Deliberative/Reactive Architecture for Unmanned Aircraft Systems". In: *Handbook of Unmanned Aerial Vehicles* (2015), pp. 849–952.

[18] Marcelo Finger and Dov Gabbay. "Adding a temporal dimension to a logic system". In: *Journal of Logic, Language and Information* 1.3 (1992), pp. 203–233.

[19] Victor Gutierrez Basulto, Jean Christoph Jung, and Ana Ozaki. "On Metric Temporal Description Logics". In: *ECAI 2016: 22nd European Conference on Artificial Intelligence*. Vol. 285. IOS Press. 2016, pp. 837–845.

[20] Pat Hayes. *Re: Why must the Web be monotonic? from Pat Hayes on 2001-07-24 (www-rdf-logic@w3.org from July 2001)*. https://lists.w3.org/Archives/Public/www-rdf-logic/2001Jul/0067.html.

[21] Patrick Hochstenbach et al. *RDF Surfaces: Enabling Classical Negation on the Semantic Web*. 2024. arXiv: 2406.10659 [cs.AI].

[22] Bas Ketsman and Christoph Koch. "Datalog with Negation and Monotonicity". In: *23rd International Conference on Database Theory (ICDT 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik. Wadern, Germany, 2020.

[23] Ron Koymans. "Specifying real-time properties with metric temporal logic". In: *Real-time systems* 2.4 (1990), pp. 255–299.

[24] B. S. Kulpeshov and S. V. Sudoplatov. "Algebras of binary formulas for weakly circularly minimal theories with trivial definable closure". In: *Siberian Mathematical Journal* 65.2 (2024), pp. 316–327.

[25] Thodsaporn Kumduang, Songpon Sriwongsa, and Khwancheewa Wattanatripop. "Combinatorial results to measure the complexity of quantifier free formulas in algebraic systems". In: *Asian-European Journal of Mathematics* (2025), p. 2550126.

[26] Daniel de Leng and Fredrik Heintz. "Qualitative spatio-temporal stream reasoning with unobservable intertemporal spatial relations using landmarks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (2016).

[27] Paul Marinier et al. *Maintaining time alignment with multiple uplink carriers*. US Patent 8,934,459. Jan. 2015.

[28] Joël Ouaknine and James Worrell. "Some Recent Results in Metric Temporal Logic". In: *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. Berlin/Heidelberg, Germany, 2008, pp. 1–13.

[29] Georgios M. Santipantakis et al. "A Stream Reasoning System for Maritime Monitoring". In: *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. Wadern, Germany, 2018, pp. 20–1.

[30] Mattias Tiger and Fredrik Heintz. "Stream Reasoning using Temporal Logic and Predictive Probabilistic State Models". In: *2016 23rd International Symposium on Temporal Representation and Reasoning (TIME)*. IEEE. New York, NY, USA, 2016, pp. 196–205.

[31] Doan Quang Tu et al. "IoT streaming data integration from multiple sources". In: *Computing* 102.10 (Oct. 1, 2020), pp. 2299–2329. ISSN: 1436-5057. DOI: 10.1007/s00607-020-00830-9.

[32] A. M. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* 2.1 (1937), pp. 230–265.

[33] Jacopo Urbani, Markus Krötzsch, and Thomas Eiter. "Chasing Streams with Existential Rules". In: *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR 2022)*. IJCAI Organization, 2022, pp. 415–419.

[34] Przemysław Andrzej Wałęga et al. "DatalogMTL with Negation Under Stable Models Semantics". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 18. 1. 2021, pp. 609–618.

6 Conclusions

This chapter concludes the dissertation by synthesising the results developed across the preceding chapters. Rather than focusing on individual technical contributions in isolation, the conclusion revisits the central research challenges and research questions introduced in Chapter 1 and evaluates them in light of the insights obtained throughout the dissertation. This retrospective perspective makes it possible to assess how the different contributions jointly address the problem of semantic fragmentation in Stream Reasoning (SR) and how they advance a principled foundation for semantic interoperability.

6.1 Consolidating the Contributions of this Dissertation

Across the preceding chapters, SR has been examined from a semantic perspective, with particular attention to how temporal abstractions, reasoning assumptions, and formal design choices shape the meaning of inference over evolving data. While these aspects were analysed in different theoretical settings, the present section consolidates the main results and clarifies how they relate to the overarching research questions formulated in Chapter 1.

Rather than introducing new formalisms or claims, the discussion that follows reconnects the contributions to the broader research context outlined in the introduction. It revisits the core problem of semantic fragmentation in SR, summarises the main insights gained, and reflects on their implications for interoperability. In doing so, the conclusion aims not only to recapitulate results, but to place them

within a coherent semantic narrative that highlights their role in advancing a principled foundation for interoperable SR.

6.1.1 Research challenge C1: Semantic Fragmentation in Stream Reasoning formalisms

Chapter 2 addresses Research Challenge C1 by systematically analysing the fragmented landscape of SR formalisms through the lens of Research Question RQ1. Rather than comparing approaches at the level of syntax or implementation, the chapter investigates how different formalisms commit to distinct semantic choices, and whether these choices can be captured in a principled and comparative manner. To this end, a broad range of representative SR approaches is examined along a set of recurring semantic and usage dimensions.

H1. The analysis presented in Chapter 2 lends clear support to Hypothesis H1, which proposed that the diversity of SR approaches can be captured using a finite and recurring set of semantic dimensions. Despite the apparent heterogeneity of the field, the surveyed formalisms consistently diverge along a small number of core dimensions, including temporal abstraction, stream selection mechanisms, treatment of absence and negation, and reasoning objectives. This finding shows that semantic diversity in SR is structured rather than arbitrary, and that systematic comparison becomes possible once these dimensions are made explicit.

H2. Evidence for Hypothesis H2 emerges from a closer inspection of how surface-level differences relate to underlying meaning. The survey reveals that formalisms with markedly different syntactic appearances may nevertheless share similar semantic commitments, while syntactically similar approaches can differ substantially in their interpretation of time, persistence, or entailment. This observation reinforces the conclusion that fragmentation in SR is driven primarily by semantic assumptions rather than by syntactic variation, and that interoperability challenges cannot be resolved through syntactic alignment alone.

H3. The findings of Chapter 2 also speak directly to Hypothesis H3, which considered the possibility that a single SR formalism might subsume the semantics of others across all identified dimensions. Rather than revealing such a universally dominating approach, the survey highlights a landscape shaped by trade-offs. Different formalisms prioritise different combinations of expressivity, robustness, and applicability, reflecting deliberate design choices informed by assumptions about data availability, temporal interpretation, and deployment context.

Taken together, the contributions of Chapter 2 demonstrate that semantic fragmentation in SR is both systematic and irreducible. By providing a structured semantic framework for comparison, the chapter resolves Challenge C1 not by advocating a single unifying formalism, but by enabling explicit reasoning about how and why different approaches diverge. This establishes a necessary foundation for the subsequent challenges addressed in this dissertation, where specific dimensions of this fragmentation are analysed, related, and, where possible, bridged.

6.1.2 Research challenge C2: Foundational Choices in Temporal Semantics, Negation and Their Consequences

Research Challenge C2 concerns the semantic consequences of foundational choices in temporal reasoning, in particular the interpretation of time and the treatment of absence and negation. This challenge is addressed through Research Questions RQ2 and RQ3, which examine these two dimensions separately, and through Hypotheses H4 and H5, which formulate expectations about their semantic alignment and expressive adequacy.

H4. The temporal abstraction dimension of Challenge C2 is examined through RQ2 and Hypothesis H4, which are addressed in Chapter 3. Rather than treating metric-based and window-based approaches as incompatible paradigms, the chapter investigates the precise conditions under which their semantics align. By developing semantics-preserving translations between carefully delineated fragments of DatalogMTL and Logic-based framework for Analytic Reasoning over Streams (LARS), the analysis demonstrates that alignment is possible without collapsing the paradigms into a single model. These results confirm Hypothesis H4 by showing that well-defined subfragments admit mutual translation while retaining sufficient expressive power, and by making the restrictions that govern such correspondence explicit.

H5. A different perspective on Challenge C2 is taken in Chapter 5, which addresses RQ3 and Hypothesis H5 by focusing on the role of negation under open-world assumptions. Instead of comparing alternative temporal abstractions, this chapter examines how temporal expressivity is affected when negation is excluded altogether. Through a detailed study of negation-free bounded fragments of Metric Temporal Logic (MTL), the analysis characterises expressivity-preserving relationships between temporal operators and identifies reduced operator sets sufficient for the fragments considered. This substantiates Hypothesis H5 and clarifies how temporal expressivity is affected by the absence of negation in open and distributed environments, where assumptions about completeness cannot be maintained.

Taken together, the contributions of Chapters 3 and 5 resolve Research Challenge C2 by demonstrating that both temporal abstraction and negation can be

treated as explicit semantic dimensions rather than implicit design choices. By clarifying how different temporal paradigms relate and by isolating expressive cores suitable for open-world reasoning, the dissertation provides a more precise account of temporal semantics in SR. For the broader SR field, these results offer formal guidance on when reasoning outcomes obtained under different assumptions can be aligned and when semantic divergence is unavoidable, thereby directly supporting principled interoperability.

6.1.3 Research challenge C3: Bridging Theoretical Expressivity and Practical Reasoning Infrastructure

Research Challenge C3 concerns the gap between the expressive power of SR semantics and the requirements imposed by practical, large-scale reasoning infrastructures. While earlier challenges addressed semantic fragmentation and foundational choices in temporal reasoning, Challenge C3 focuses on whether expressive temporal semantics can be realised without relying on specialised or prototype execution engines. This challenge is addressed through Research Question RQ4, which is examined from both a theoretical and a practical perspective via Hypotheses H6 and H7.

H6. The theoretical aspect of Challenge C3 is addressed in Chapter 4 through Hypothesis H6, which considers whether fragments of expressive SR semantics admit semantics-preserving mappings to non-temporal reasoning formalisms. By developing a formal embedding of DatalogMTL into plain Datalog, the chapter demonstrates that metric temporal reasoning can be represented within a non-temporal setting while preserving temporal entailment and minimal-model semantics. This result shows that expressive temporal semantics need not be intrinsically tied to dedicated temporal engines, but can instead be realised through semantic transformation, thereby confirming Hypothesis H6.

H7. Beyond theoretical feasibility, Chapter 4 also evaluates the practical relevance of this embedding through Hypothesis H7. The analysis demonstrates that the identified fragment retains sufficient expressivity to capture temporally rich reasoning tasks that arise in realistic monitoring scenarios. Moreover, by implementing the embedding on top of existing, mature Datalog reasoning engines, the chapter illustrates that expressive SR can be deployed within trusted and widely adopted infrastructures. This confirms that the semantic fragment identified in Chapter 4 is not only theoretically well-founded, but also suitable for practical use.

Taken together, the contributions of Chapter 4 resolve Research Challenge C3 by showing that the deployment barrier associated with expressive SR is not a consequence of semantic complexity per se, but of how semantics are realised computationally. By decoupling temporal meaning from specialised execution

mechanisms, the dissertation demonstrates a viable path toward scalable, maintainable, and interoperable SR systems. For the broader SR field, this result shifts the focus from the development of bespoke engines toward semantics-preserving integrations with existing reasoning ecosystems, thereby supporting wider adoption in safety-critical and industrial settings.

6.1.4 Overarching insights

The three research challenges addressed in this dissertation are connected by a common theme: interoperability in SR is fundamentally a semantic problem. Across different formalisms, divergences arise not merely from implementation choices or surface syntax, but from deeper assumptions about how time is represented, how relevance is determined, and how inference behaves in the presence of incomplete and evolving information. When these assumptions remain implicit, conclusions that appear comparable at an informal level may differ substantially in meaning, undermining principled exchange and reuse across systems.

Interoperability by means of semantic alignment. A first overarching insight is that interoperability cannot be reduced to syntactic standardisation or technical integration. Throughout the dissertation, differences between SR approaches are shown to reflect distinct semantic commitments rather than accidental design choices. Whether reasoning is window-based or metric-based, whether absence is treated as falsity or inconclusiveness, and whether temporal semantics are realised directly or embedded into non-temporal formalisms all have direct consequences for what conclusions mean.

The contributions of this dissertation demonstrate that meaningful interoperability is attainable when such assumptions are made explicit and related formally. Rather than requiring uniformity, interoperability emerges as conditional: formalisms can interoperate when their semantic foundations are compatible or can be connected through well-defined transformations. This shifts the focus from enforcing a single language or execution model to understanding the conditions under which reasoning outcomes can be aligned.

Managing semantic diversity without eliminating it. A second insight concerns the role of diversity in the evolution of SR. The survey results and subsequent formal analyses show that the field has developed through multiple, largely application-driven trajectories, each grounded in different assumptions about time, streams, and inference. This diversity is not a shortcoming to be corrected, but a reflection of the varied requirements placed on SR systems.

Rather than advocating a single dominant formalism, the dissertation contributes a set of semantic lenses through which existing approaches can be compared, related, and simplified. By identifying recurring semantic dimensions, analysing the

consequences of foundational design choices, and isolating expressive cores, the work demonstrates how semantic diversity can be managed systematically. In this view, interoperability is achieved not by eliminating differences, but by distinguishing between essential semantic divergence, conventional modelling choices, and aspects that can be bridged under explicit conditions.

Positioning semantic analysis within the SR research landscape. Taken together, these insights position the dissertation within the broader evolution of SR research. Much prior work has focused on extending expressivity or improving performance for particular use cases, often resulting in parallel developments grounded in different semantic assumptions. The present work complements these efforts by providing a unifying semantic perspective that clarifies how such developments relate to one another.

By analysing temporal abstraction, operator expressivity, and reasoning infrastructure through a common semantic framework, the dissertation contributes formal reference points that support cumulative progress in the field. Results obtained in one formal setting can be interpreted, compared, or transferred to others when their underlying assumptions are understood, rather than rediscovered or re-engineered in isolation. In this way, semantic clarity emerges not as a secondary concern, but as a prerequisite for principled, interoperable reasoning over streams in distributed and evolving environments.

6.2 Implications for Stream Reasoning Research and Practice

The results presented in this dissertation are primarily foundational in nature, aiming to clarify semantic relationships rather than to propose new execution architectures or systems. Nevertheless, making semantic assumptions explicit and formally relating existing approaches has consequences for how future work in SR may be framed, evaluated, and integrated. This section therefore discusses several implications that follow from the research results, while remaining mindful of their theoretical scope and the boundaries of what has been established.

6.2.1 Implications for the Design of Stream Reasoning Formalisms

One implication of the results is that the design space of SR formalisms can be approached with greater semantic awareness. The translations developed between metric and window-based reasoning illustrate that many differences between formalisms arise from modelling decisions rather than from fundamentally distinct notions of

reasoning over streams. This suggests that future formalisms may benefit from explicitly stating which temporal abstractions they adopt and how constructs, such as windows or time references, affect entailment.

The analysis of negation-free bounded temporal reasoning further indicates that expressive richness does not necessarily require a large or complex operator set within the fragments studied in this dissertation. In particular, when reasoning is performed under open or distributed assumptions, it may be possible to favour smaller, semantically transparent operator bases while retaining the expressivity required within a particular reasoning setting. This does not imply that richer operator sets are unnecessary in general, but it highlights the value of understanding which operators contribute essential semantic distinctions in a given reasoning context.

More generally, the results encourage a design perspective in which semantic assumptions, for example about persistence, absence of information, or evaluation context, are treated as first-class aspects of a formalism. Putting these assumptions on the forefront can help avoid accidental incompatibilities between formalisms that otherwise appear closely related, and can support more principled comparison and reuse of existing work. .

6.2.2 Implications for SR Engines and Implementations

Although this dissertation does not focus on implementation techniques, the semantic results have indirect implications for reasoning engines and system development. The demonstrated embedding of temporal reasoning into non-temporal Datalog shows that temporal semantics need not be intrinsically bound to specialised temporal engines. From a conceptual standpoint, this suggests that some distinctions between reasoning infrastructures reflect representational choices rather than hard semantic boundaries.

For practitioners, this observation may open opportunities to reuse existing reasoning engines, tooling, and optimisation strategies when supporting temporal reasoning, provided that the semantic conditions of the embedding are respected. At the same time, this dissertation does not claim that such embeddings are universally optimal or appropriate in all settings, as practical considerations such as performance, expressivity, and incremental evaluation remain highly context-dependent.

More broadly, the results underline the importance of semantic correctness when implementing translations or integrations between reasoning systems. Implementation-level interoperability that ignores semantic alignment risks producing systems that exchange information but not meaning. The formal relationships established in this dissertation can therefore serve as reference points when assessing whether an implementation preserves intended entailment behaviour across system boundaries.

6.2.3 Implications for Interoperability and Standardisation

The results of this dissertation have broader implications for how interoperability is approached in SR environments beyond individual systems or formalisms. In particular, they suggest that interoperability efforts should prioritise semantic coordination over full unification, and should explicitly account for heterogeneity in temporal interpretation, reasoning assumptions, and deployment contexts.

Rather than assuming that interoperability can be achieved through the adoption of a single language or execution model, the findings point toward approaches that make semantic assumptions explicit and relate them through well-defined transformations. These considerations highlight that interoperability in modern SR settings hinges on the ability to relate independently designed reasoning systems through explicit semantic guarantees, rather than on enforcing uniform formalisms or infrastructures.

Semantic coordination in data spaces and decentralised Web architectures. A setting in which this need for semantic coordination becomes particularly explicit is that of decentralised data spaces, including initiatives such as the European Healthcare Data Space (EHDS) and SOLID-based architectures on the decentralised Web, both of which were introduced in Chapter 1. In these settings, interoperability is expected to emerge across independently governed systems that retain control over their data, processing pipelines, and local reasoning mechanisms. Full standardisation of SR formalisms in such environments is neither realistic nor aligned with their design principles.

Within the EHDS, healthcare providers, research institutions, and public authorities are required to exchange data and derived insights across organisational and national boundaries, while complying with strict governance, privacy, and accountability constraints [4]. Similarly, SOLID-based architectures promote decentralisation by allowing data holders to independently select applications and reasoning components that operate over their personal or organisational data [5, 6]. In both cases, reasoning components are expected to interoperate without assuming shared temporal abstractions, closed-world reasoning, or uniform execution infrastructure.

The semantic results of this dissertation provide a foundation for such coordination. By identifying semantic fragments, translation principles, and conditions under which reasoning outcomes can be preserved across heterogeneous systems, the work clarifies when interoperability is meaningful and when semantic divergence must be respected. This enables data space participants to exchange reasoning results with a clear understanding of their semantic scope, rather than relying on implicit or undocumented assumptions.

Importantly, these findings do not prescribe a specific standard or architecture for EHDS- or SOLID-style deployments. Instead, they support an approach in which interoperability mechanisms are evaluated with respect to explicit semantic guarantees, allowing decentralised systems to coordinate without sacrificing autonomy. This perspective aligns with the core goals of data spaces and decentralised Web initiatives, where trust and interoperability are achieved through transparency and formal clarity rather than enforced uniformity.

6.3 Limitations, Open Questions, and Future Directions

The results presented in this dissertation clarify several semantic relationships between prominent SR approaches, but they do not aim to provide a complete account of all aspects of SR. As with any foundational investigation, the scope and methodology of the research impose limitations that are important to acknowledge. At the same time, these limitations reveal open questions and point toward future research directions that build naturally on the results obtained.

6.3.1 Scope Limitations and Design Choices

This subsection summarises the main boundaries of the dissertation and highlights the deliberate design choices that shape the scope of the results.

Focus on declarative, logic-based SR. A first and deliberate limitation of this research is its focus on declarative, logic-based SR formalisms with well-defined model-theoretic semantics. This excludes procedural, probabilistic, or purely data-driven approaches to stream processing, as well as hybrid systems in which learning and reasoning are tightly interwoven. While such approaches are increasingly relevant in practice, addressing them would require fundamentally different semantic and methodological tools.

Limited coverage of SR dimensions beyond temporal semantics. The dissertation also concentrates on formalisms that reason over time using explicit temporal abstractions, such as metric time or window-based views. Other dimensions of SR heterogeneity, i.e. uncertainty, inconsistency handling, or multi-agent interaction, are only touched upon indirectly, if at all. In particular, issues of probabilistic reasoning, soft constraints, or confidence propagation fall outside the scope of the present work.

Structural restrictions for semantic clarity. Finally, several results rely on structural restrictions that are chosen to ensure semantic clarity. Notably, the analysis of temporal operator expressivity is carried out in negation-free settings, motivated by open-world and distributed reasoning assumptions. These restrictions are not claims

about how SR should always be performed, but rather about what can be established formally under clearly stated conditions.

6.3.2 Open Theoretical Questions

Beyond the limitations outlined above, the dissertation gives rise to several open theoretical questions that invite further investigation.

Negation under partial and delayed observability. A number of theoretical questions remain open as a consequence of the scope choices outlined above. One prominent example concerns the role of negation in SR under partial or delayed observability. While this dissertation motivates negation-free reasoning as a robust baseline in open settings, it does not provide a general account of how various forms of negation should be interpreted in evolving streams. In particular, more recent approaches focused on stratified negation introduce interesting solutions for negation in open-world settings [3, 7]. The interaction between stratified negation and the syntactic simplification presented in this work provides an interesting theoretical avenue to explore [2].

Interacting temporal abstractions and broader SR paradigms. Another open question relates to the interaction between different temporal abstractions beyond the settings considered here. Although formal translations are established between metric and window-based reasoning under specific assumptions, it remains unclear how these relationships behave in more complex scenarios involving, for example, overlapping windows, adaptive window policies, or heterogeneous time granularities. Understanding how such features affect entailment and semantic preservation remains an open research challenge. Additionally, the field of SR encompasses more than only metric- and window-based reasoning. Among others, event-based reasoning formalisms also have reputable representatives within the field (e.g. RTEC [1]). Formal translation between a wider net of SR branches, in turn enabling wider interoperability, remains a topic for further exploration.

6.3.3 Directions for Future Research

The contributions of this dissertation naturally point toward several avenues for future research that build on, extend, or complement the results presented here.

Extending semantic results to richer logical fragments. One natural direction is the extension of the presented translation and embedding results to richer logical fragments, including controlled forms of negation or aggregation, for example stratified negation. Such work would help clarify which aspects of semantic alignment scale to more expressive settings and which require fundamentally new techniques.

Refining and generalising semantic alignments. Another direction for future research concerns the extension and consolidation of the formal relationships established in this dissertation. Several chapters identify precise conditions under which translations between formalisms preserve entailment, but they also expose boundaries where such preservation no longer holds. In the interest of maximally aligning metric- and window-based reasoning, the identified fragments could be honed further to maximise semantic exchange while minimizing restriction of the involved formalisms. Similarly, the syntactic simplification results for negation-free temporal reasoning invite investigation into how far these results extend once additional language constructs are introduced.

Optimising semantics-preserving realisations. Finally, the semantic embedding of temporal reasoning into non-temporal Datalog suggests studying which optimisations, restrictions, or alternative encodings preserve correctness while improving practical feasibility, thereby further clarifying the trade-offs between expressivity, semantic fidelity, and execution models.

Connecting semantic foundations to empirical evaluation. From a broader perspective, there is scope for connecting the foundational results of this dissertation more directly to empirical and systems-oriented research. While the dissertation intentionally avoids performance claims, its semantic analyses could inform experimental studies that evaluate how different semantic choices affect system behaviour in realistic deployments. Such studies would not only provide empirical evidence of the semantic results, in addition to the theoretical evidence presented, but could help assess their practical relevance and limitations in concrete application domains.

6.4 Final Outlook

The central message of this dissertation is that semantic clarity is a prerequisite for robust interoperability in SR. As reasoning systems continue to be deployed in open, heterogeneous, and distributed environments, implicit assumptions about time, observability, and entailment are increasingly likely to create friction within the environment. Addressing such friction does not require eliminating diversity, but managing it through precise semantic understanding.

The results presented here do not resolve all such challenges, nor are they intended to. Instead, they provide formal tools and perspectives that make semantic relationships explicit and analysable. These foundations will support both theoretical and practical future work that builds interoperable SR systems in a principled and transparent manner.

To advance the field, future SR research and system development should treat semantic commitments as first-class design choices rather than implicit side effects

of language or engine selection. Formalisms, systems, and evaluation benchmarks should make their underlying assumptions about time, relevance, and inference explicit, and should document the semantic conditions under which their results can be meaningfully interpreted or exchanged. Doing so will enable cumulative progress, reduce fragile integration practices, and support the deployment of reasoning systems that remain interpretable and trustworthy as they evolve alongside the environments in which they operate.

References

[1] Alexander Artikis, Marek Sergot, and Georgios Paliouras. "An event calculus for event recognition". In: *IEEE Transactions on Knowledge and Data Engineering* 27.4 (2014), pp. 895–908.

[2] Luís Cruz-Filipe, Graça Gaspar, and Isabel Nunes. "Reconciling communication delays and negation". In: (2022), pp. 151–169. DOI: 10.1007/978-3-031-17715-6_11.

[3] David J. Tena Cucala et al. "Stratified negation in Datalog with metric temporal operators". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.7 (2021), pp. 6488–6495.

[4] European Commission. *Proposal for a Regulation of the European Parliament and of the Council on the European Health Data Space*. COM(2022) 197 final. May 3, 2022.

[5] Ruben Verborgh. *Let's talk about pods*. Blog post. Dec. 2022. URL: https://ruben.verborgh.org/blog/2025/08/12/inside-the-insight-economy/.

[6] W3C Solid Community Group. *Solid Protocol*. Tech. rep. World Wide Web Consortium (W3C), 2021.

[7] Przemysław Andrzej Wałęga et al. "DatalogMTL with Negation Under Stable Models Semantics". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 18. 1. 2021, pp. 609–618.



A Sudoku puzzle depicting reasoning across interdependent constraints, reflecting the core principles of logical inference.