TECHNISCHE UNIVERSITÄT WIEN

DISSERTATION

Fixed Point Semantics for Stream Reasoning

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften unter der Leitung von

Ao.Univ.Prof. Dr.phil. Matthias Baaz
E104 – Institut für Diskrete Mathematik und Geometrie, TU Wien

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation

von

Dipl.-Ing. Christian Antić
Matrikelnummer: 0525482

Diese Dissertation haben begutachtet:

1. **Prof. Dr. Stefan Woltran**
Institut für Logik und Computation, TU Wien
2. **Prof. Dr. Pascal Hitzler**
The Center for Artificial Intelligence and Data Science, Kansas State University
3. **Prof. Dr. Matthias Baaz**
Institut für Diskrete Mathematik und Geometrie, TU Wien

Wien, am 12. August 2021

Kurzfassung

Menschen sind im Alltag mit Datenströmen (engl. "streams") aus ihrer Umgebung konfrontiert und das Schließen (engl. "reasoning") aus solchen Datenströmen ist für die menschliche Intelligenz von zentraler Bedeutung. Moderne digitale Dienste, wie zum Beispiel Google und Facebook, generieren in kürzester Zeit zahlreiche Daten die es maschinell zu verarbeiten gilt. Im letzten Jahrzehnt hat sich innerhalb der Künstlichen Intelligenz eine Forschungsrichtung dafür als besonders relevant hervorgehoben—das sogenannte *Stream Reasoning*. Vor kurzem wurde der regel-basierte Formalismus *LARS* für das nicht-monotone daten-basierte Schließen unter Anwendung der Antwortmengensemantik entwickelt. Syntaktisch sind LARS-Programme nichts anderes als Logikprogramme mit Negation, die zusätzlich Operatoren zum Ausdrücken zeitlicher Zusammenhänge erlauben, wobei der *Fenster-Operator* (engl. "window operator") von besonderem Interesse ist—dieser erlaubt es, relevante Zeitpunkte auszuwählen. Da LARS *fixe* Zeitintervalle für die Evaluierung von Programmen voraussetzt, ist der Formalismus in der aktuellen Form nicht flexibel genug, um *konstruktiv* mit sich rasch verändernden Daten umzugehen. Außerdem hat sich gezeigt, dass die von LARS verwendete und auf FLP-Redukten basierende Erweiterung der Antwortmengensemantik zirkuläre Schlüsse zulässt, wie sie auch von anderen Erweiterungen der klassischen Antwortmengensemantik bereits bekannt sind. Diese Doktorarbeit behebt alle erwähnten Schwächen von LARS und leistet einen Beitrag zu den Grundlagen des Stream Reasonings indem sie eine operationale Fixpunktsemantik für eine flexible Variante von LARS entwickelt die korrekt und konstruktiv in dem Sinne ist, dass Antwortmengen durch iterierte Anwendung eines Fixpunktoperators erzeugt werden und dadurch frei von zirkulären Schlüssen sind.

Abstract

Reasoning over streams of input data is an essential part of human intelligence. During the last decade *stream reasoning* has emerged as a research area within the AI-community with many potential applications. In fact, the increased availability of streaming data via services like Google and Facebook has raised the need for reasoning engines coping with data that changes at high rate. Recently, the rule-based formalism *LARS* for non-monotonic stream reasoning under the answer set semantics has been introduced. Syntactically, LARS programs are logic programs with negation incorporating operators for temporal reasoning, most notably *window operators* for selecting relevant time points. Unfortunately, by preselecting *fixed* intervals for the semantic evaluation of programs, the rigid semantics of LARS programs is not flexible enough to *constructively* cope with rapidly changing data dependencies. Moreover, we show that defining the answer set semantics of LARS in terms of FLP reducts leads to undesirable circular justifications similar to other ASP extensions. This thesis fixes all of the aforementioned shortcomings of LARS. More precisely, we contribute to the foundations of stream reasoning by providing an operational fixed point semantics for a fully flexible variant of LARS and we show that our semantics is sound and constructive in the sense that answer sets are derivable bottom-up and free of circular justifications.

Preface

This thesis is a synthesis of logic programming, temporal logic, and fixed point theory, all of which are important areas of theoretical computer science.

A fixed point theorem is a result proving that a function f will have at least one fixed point x with $f(x) = x$ given some general properties of f. Fixed point theorems are ubiquitous in mathematics and computer science. In topology, for instance, Brouwer's famous fixed point theorem, stating that every continuous function mapping a compact convex set to itself has a fixed point, has wide applications across numerous fields of mathematics. In theoretical computer science, on the other hand, the idea of bringing fixed points into the domain of programming language semantics sprang from the fertile brains of Christopher Strachey and Dana Scott in the late 1960s and early 1970s. More precisely, Dana Scott introduced *domain theory* as a way to give semantics to the lambda calculus, which later evolved into *denotational semantics* of programming languages, where the meaning of computer programs is provided in terms of functions mapping input into output. The mathematical basis is provided by the following well-known theorem due to Bronisław Knaster and Alfred Tarksi.

Theorem 1 (Knaster-Tarski)**.** *A monotone operator on a complete lattice has a least fixed point which is also its least prefixed point.*

Logic programming evolved from automated theorem proving in the late 1960s and early 1970s when Robert Kowalski developed SLD-resolution as a variant of John Alan Robinson's resolution rule and later collaborated with Alain Colmerauer in the development of PROLOG, a declarative programming language based on Horn clause logic with appealing computational properties for knowledge representation and reasoning in artificial intelligence. A characteristic feature of logic programming is its top-down or goal-oriented procedural interpretation in terms of SLD-resolution on the syntactic side and its constructive bottom-up semantic evaluation in terms of least fixed point computations on the semantic side. More precisely, it was the idea of Robert Kowalski and Maarten H. van Emden in the 1970s to use the Knaster-Tarksi theorem to give operational least model semantics to logic programs in terms of least fixed points of the immediate consequence operator and it is a key result in the theory of logic programming that SLD-resolution and least models capture the same meaning of programs.

Classical first-order logic was initially introduced as a mathematical tool for precisely formalizing and analyzing mathematical theories which are monotone in their nature in the sense that given a set of assumptions or axioms, the set of derivable conclusions monotonically increases when adding new assumptions. In contrast, commonsense reasoning is an inherently non-monotone form of logical reasoning for which classical first-order logic, in its original form, is not an adequate formalism. In the last three decades much effort has been investigated to capture essential parts of commonsense reasoning, with default logic, autoepistemic logic, and logic programs with negation being the most prominent formalisms today, the latter arguably being the most successful non-monotonic reasoning and knowledge representation formalism with many practical applications. Logic programming got more interesting and more complicated when an unary operator "∼" denoting *negation as failure* (or *default negation*) has been added to the syntax of logic programs. SLDNF-resolution is an extension of SLD-resolution which deals with negation as failure. In the late 1980s another interpretation of negation as failure was given by Michael Gelfond and Vladimir Lifschitz via a translation to autoepistemic logic which led to the fundamental notion of *stable models* or *answer sets*. Today, answer set programming is a well-established logic programming dialect with many applications in AI-related subfields such as, for example, planning, diagnosis, and abductive reasoning. Unfortunately, the Knaster-Tarski theorem, which is at the core of the fixed point semantics of negation-free logic programming, is not applicable to answer set programming as for this class of programs the immediate consequence operator is in general non-monotonic. It was Melvin Fitting who in the late 1990s and early 2000s extended the Knaster-Tarski theorem to the non-monotonic case. More precisely, Fitting realized that for answer set programming, the non-monotone 2-valued immediate consequence operator, which is defined on the lattice of 2-valued interpretations, had to be replaced by a *monotone* 4-valued operator defined on the bilattice of 4-valued interpretations or by a monotone 3-valued operator defined on consistent pairs of interpretations. This idea was later reformulated within an abstract algebraic framework called *approximation fixed point theory* by Marc Denecker, Victor Marek, and Mirosław Truszczyński. It is the 3-valued Fitting operator which we adapt in this thesis to the class of stream logic programs.

Temporal logic deals with sequences of events and it was the idea of Amir Pnueli in the late 1970s to use linear-time temporal logic, in which time is modeled as a linear ordering, for formal verification of computer programs.[1] Temporal logic is a special kind of modal logic giving semantics to expressions with tense like, for example, the sentence "I'm happy" which may be true at some particular point in time. Temporal operators like "always" and "eventually" allow the formal expression of temporal modalities. In stream reasoning, the main topic of this thesis, the "window" operator enables the selection of relevant time points for efficient reasoning from rapidly changing data.

In this thesis, I use tools and ideas from the aforementioned areas to provide a fixed point semantics to a class of answer set programs geared towards non-monotonic temporal reasoning about data streams. The idea came to me at the beginning of 2015 when I stumbled upon the first conference paper of Harald Beck, Thomas Eiter, and Minh Dao-Tran at TU Vienna on stream reasoning. I was unsatisfied with their semantics relying on fixed inter-

[1]For his seminal work introducing temporal logic into computing science, Pnueli was awarded the the 1996 Turing Award.

vals for the evaluation of programs as the philosophy behind stream reasoning is to cope with rapidly changing data. To my own surprise, using dynamic intervals in the semantic evaluation of programs turned out to be highly non-trivial. I've essentially finished the paper on stream reasoning in the first half of 2015 and then forgot about it for the next four years. In the years that followed I was mainly interested in analogical reasoning which should have been the topic of my thesis. However, in 2019 I decided to submit the paper on stream reasoning to the prestigious *Artificial Intelligence* journal and after its acceptance, my advisor Professor Matthias Baaz encouraged me—since I was doing the research on the paper on my own—to translate the paper into this short 27 pages thesis. I must thank Matthias Baaz for his financial and moral support during the years 2015-2020. In times where it is customary for students in the computer science department to finish their PhD studies with as many as 10–20 research papers (including numerous repetitions and "infinitely" many collaborators), you supported my independent research without exploiting me. In all those years at TU Vienna, you were the only person to use the words "creative freedom"—without you I would have never dared to do a PhD at TU Vienna. Thank you!

Finally, I want to acknowledge support of the Austrian Science Fund (FWF) under grant P31063-N35.

Vienna, Austria
August 12, 2021

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 12. August 2021

Christian Antić

Contents

1 Introduction

Reasoning over streams of input data is an essential part of human intelligence. During the last decade, *stream reasoning* has emerged as a research area within the AI-community with many potential applications, e.g., web of things, smart cities, and social media analysis (cf. [9, 25, 1]). In fact, the increased availability of streaming data via services like Google and Facebook has raised the need for reasoning engines coping with data that changes at high rate.

Logic programs are rule-based systems with the rules and facts being written in a sublanguage of predicate logic extended by a unary operator "$\sim$" denoting *negation-as-failure* (or *default negation*) [8]. While each monotone (i.e., negation-free) logic program has a unique least Herbrand model (with the least model semantics [31] being the accepted semantics for this class of programs), for general logic programs a large number of different purely declarative semantics exist. Many of it have been introduced some 30 years ago, among them the *answer set semantics* [20] and the *well-founded semantics* [32]. The well-founded semantics, because of its nice computational properties (computing the unique well-founded model is tractable), plays an important role in database theory. However, with the emergence of efficient solvers such as DLV [22], Smodels [29], Cmodels [21], and Clasp [19], programming under answer set semantics led to a predominant declarative problem solving paradigm, called *answer set programming* (or *ASP*) [24, 23]. Answer set programming has a wide range of applications and has been successfully applied to various AI-related subfields such as planning and diagnosis (for a survey see [7, 13, 4]). Driven by this practical needs, a large number of extensions of classical answer set programs have been proposed, e.g. *aggregates* (cf. [16, 17, 27]), *choice rules* [26], *dl-atoms* [14], and general *external atoms* [15]. For excellent introductions to the field of answer set programming we refer the reader to [7, 4, 13].

Beck et al. [5] introduced LARS, a Logic-based framework for Analytic Reasoning over Streams, where the semantics of LARS has been defined in terms of FLP-style answer sets [17]. Syntactically, LARS programs are logic programs with negation as failure incorporating operators for temporal reasoning, most notably *window operators* for selecting relevant time points. Unfortunately, by preselecting *fixed* intervals for the semantic evaluation of programs, the rigid semantics of LARS programs is not flexible enough to constructively cope with rapidly changing data dependencies. For example, sentences of the form "a holds at time point t if b holds at every *relevant* time point" are not expressible within LARS (cf. Example 3), as the interval of "relevant" time points changes dynamically, whereas LARS preselects a static interval. Our first step therefore is to refine and simplify Beck et al's [5] semantics in Section 2.4 by employing *dynamic* time intervals.

Extensions of the answer set semantics adhere to minimal models or, even more restricting, to models free of unfoundedness. However, FLP-answer sets of stream logic programs may permit undesirable circular justifications similar to other ASP extensions (cf. [28, 2]). Fixed point semantics of logic programs (cf. [18]), on the other hand, are constructive by nature, which suggests to define a fixed point semantics for stream logic programs targeted for foundedness, by recasting suitable operators in such a way that the FLP semantics can be reconstructed or refined, in the sense that a subset of the respective answer sets are selected (sound "approximation"). The benefit is twofold: by coinciding semantics, we get operable fixed point constructions, and by refined semantics, we obtain a sound approximation that is constructive. For this we recast two well-known fixed point operators from ordinary to stream logic programs, namely the van Emden-Kowalski operator [31] and the Fitting operator [18]. This task turns out to be non-trivial due to the intricate properties of windows [3, 5] and other modal operators occurring in rule heads. We show that the so obtained operators inherit the following characteristic properties: models of a program are characterized by the prefixed points of its associated van Emden-Kowalski operator, and the Fitting operator is monotone with respect to a suitable ordering which guarantees the existence of certain least fixed points, namely the so obtained constructive answer sets. We then show the constructiveness of our fixed point semantics in terms of level mappings [28]. Specifically, we prove that our semantics captures those answer sets which possess a level mapping or, equivalently, which are free of circular justifications, which is regarded as a positive feature.

The rest of the thesis is structured as follows. In Section 2, we define the syntax and semantics of stream logic programs first in the vein of [5] (Section 2.3) followed by our refined semantics in Section 2.4. Sections 3 and 4 constitute the main part of the thesis. More precisely, in Section 3.1, we define a novel (partial) model operator for the evaluation of rule heads, and, in Sections 3.2 and 3.3, we recast the well-known van Emden-Kowalski operator $\mathrm{T}_{\mathcal{P}}$ and the Fitting operator $\Phi_{\mathcal{P}}$ from ordinary to stream logic programs and prove some non-trivial properties. In Section 4, we then define a fixed point semantics for stream logic programs in terms of the (extended) Fitting operator and prove in our Main Theorem 16 the soundness of our approach. Afterwards, in Section 5, we characterize our semantics in terms of level mappings and conclude that our semantics is sound, constructive, and free of circular justifications.

2 Stream Logic Programs

We denote the set $\mathbb{N} \cup \{\infty\}$ by $\mathbb{N}^\infty$. A *partially ordered set* (or *poset*) is a pair $\langle L, \leq \rangle$ where L is a set and $\leq$ is a reflexive, antisymmetric, and transitive binary relation on L. A *lattice* is a poset $\langle L, \leq \rangle$ where every pair of elements $x, y \in L$ has a unique greatest lower bound and least upper bound in L. We call $\langle L, \leq \rangle$ *complete* if every subset has a greatest lower bound and a least upper bound. For any two elements $x, y \in L$, we define the *interval* $[x, y] = \{z \in L \mid x \leq z \leq y\}$. Given a mapping $f : L \to L$, we call $x \in L$ a *prefixed point* of f if $f(x) \leq x$, and we call x a *fixed point* of f if $f(x) = x$. Moreover, we call f *monotone* if $x \leq y$ implies $f(x) \leq f(y)$, for all $x, y \in L$. In case f has a *least fixed point*, we denote it by $\mathrm{lfp}\, f$. Moreover, for a mapping $g : L \times L \to L$ we denote by $g(-, y)$ the function mapping every $x \in L$ to $g(x, y) \in L$.

2.1 Streams and Windows

In the rest of the thesis, Σ will denote a finite nonempty set of propositional atoms containing the special symbol $\top$.

A *formula* (over Σ) is defined by the grammar

$$\alpha ::= a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \to \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \mid \boxplus_{[\ell,r]}\alpha$$

where $a \in \Sigma$, $t \geq 1$, and $\ell, r \in \mathbb{N}^\infty$, $\ell \leq r$. We call α (i) $\Box$*-free* if it does not contain $\Box$; (ii) *monotone* if it does not contain $\neg, \to, \Box$; and (iii) *normal* if it does not contain $\neg, \vee, \to, \Diamond$.

A *stream* (over Σ) is an infinite sequence $\mathbf{I} = I_1 I_2 \ldots$ of subsets of Σ, i.e., $I_t \subseteq \Sigma$ for all *time points* $t \geq 1$. We call a stream $\mathbf{J} = J_1 J_2 \ldots$ a *substream* of $\mathbf{I}$, in symbols $\mathbf{J} \subseteq \mathbf{I}$, if $J_t \subseteq I_t$ for all $t \geq 1$. In the sequel, we omit empty sets in a sequence and write, e.g., $I_1 I_3$ instead of $I_1 \emptyset I_3 \emptyset\emptyset \ldots$, and we denote the empty sequence $\emptyset\emptyset\ldots$ simply by $\emptyset$. We define the *support* of $\mathbf{I}$, in symbols $\operatorname{supp} \mathbf{I}$, to be the tightest interval $[t_1, t_2]$ containing $\{t \geq 1 \mid I_t \neq \emptyset\}$; formally, $t_1 = \min\{t \geq 1 \mid I_t \neq \emptyset\}$ and $t_2 = \max\{t \geq 1 \mid I_t \neq \emptyset\}$ in case $\mathbf{I} \neq \emptyset$, and $\operatorname{supp} \emptyset = \emptyset$.

A *window*[1] is a function $[-]$ mapping every stream $\mathbf{I} = I_1 I_2 \ldots$ to the substream $\mathbf{I}[\ell, r; t] = I_{\max\{0,t-\ell\}} \ldots I_{t+r}$ of $\mathbf{I}$, where $\ell, r \in \mathbb{N}^\infty$, $\ell \leq r$. Note that $[-]$ and supp are monotone

[1]Beck et al. [5] employed more sophisticated windows and called them *window functions*; for simplicity, we consider here only the windows defined above and note that our results are independent of the particular choice of windows.

functions, that is, $\mathbf{I} \subseteq \mathbf{J}$ implies $\mathbf{I}[\ell, r; t] \subseteq \mathbf{J}[\ell, r; t]$ and $\operatorname{supp} \mathbf{I} \subseteq \operatorname{supp} \mathbf{J}$, for all $\ell, r \in \mathbb{N}^\infty$ and $t \geq 1$.

2.2 Syntax

A (*stream logic*) *program* $\mathcal{P}$ is a finite nonempty set of *rules* of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_j, {\sim}\beta_{j+1}, \ldots, {\sim}\beta_k, \quad k \geq j \geq 1, \tag{2.1}$$

where α is a normal t-formula, $\beta_1, \ldots, \beta_k$ are formulas, and $\sim$ denotes *negation-as-failure* [8]. We will often write $\overset{r}{\leftarrow}$ in expressions of the form (2.1) to make the name of the rule explicit. For convenience, we define for a rule r of the form (2.1), $\mathrm{H}(r) = \alpha$, and $\mathrm{B}(r) = \beta_1 \wedge \ldots \wedge \beta_j \wedge \neg\beta_{j+1} \wedge \ldots \wedge \neg\beta_k$. As is customary in logic programming, we will interpret every finite set A of formulas as the conjunction $\bigwedge A$ over all formulas in A. We call a rule r a *fact* if $\mathrm{B}(r) = \top$, and we call r *ordinary* if $\alpha, \beta_1, \ldots, \beta_k \in \Sigma$. Moreover, we define $\mathrm{H}(\mathcal{P})$ to be the conjunction of all rule heads occurring in $\mathcal{P}$, that is, $\mathrm{H}(\mathcal{P}) = \bigwedge_{r \in \mathcal{P}} \mathrm{H}(r)$.

2.3 Semantics of Beck et al. (2018)

We now recall the FLP-style answer set semantics [17] as defined in [5] and we show that their semantics yields counter-intuitive answer sets (cf. Example 2).

Let T be an interval in $\mathbb{N}$ and let $B \subseteq \Sigma$ be a finite set, called the *background data*. We define the *entailment relation* $\models_B$, with respect to B, for all streams $\mathbf{I}$, $a \in \Sigma - \{\top\}$, formulas α, β, and all time points $t \in T$:

1. $\mathbf{I}, T, t \models_B \top$;
2. $\mathbf{I}, T, t \models_B a$ if $a \in I_t \cup B$;
3. $\mathbf{I}, T, t \models_B \neg\alpha$ if $\mathbf{I}, T, t \not\models_B \alpha$;
4. $\mathbf{I}, T, t \models_B \alpha \wedge \beta$ if $\mathbf{I}, T, t \models_B \alpha$ and $\mathbf{I}, T, t \models_B \beta$;
5. $\mathbf{I}, T, t \models_B \alpha \vee \beta$ if $\mathbf{I}, T, t \models_B \alpha$ or $\mathbf{I}, T, t \models_B \beta$;
6. $\mathbf{I}, T, t \models_B \alpha \rightarrow \beta$ if $\mathbf{I}, T, t \not\models_B \alpha$ or $\mathbf{I}, T, t \models_B \beta$;
7. $\mathbf{I}, T, t \models_B \Diamond\alpha$ if $\mathbf{I}, T, t' \models_B \alpha$, for some $t' \in T$;
8. $\mathbf{I}, T, t \models_B \Box\alpha$ if $\mathbf{I}, T, t' \models_B \alpha$, for all $t' \in T$;
9. $\mathbf{I}, T, t \models_B @_{t'}\alpha$ if $\mathbf{I}, T, t' \models_B \alpha$, and $t' \in T$;

10. $\mathbf{I}, T, t \models_B \boxplus_{[\ell,r]}\alpha$ if $\mathbf{I}[\ell, r; t], T, t \models_B \alpha$.

In case $\mathbf{I}, T, t \models_B \alpha$, we call $\mathbf{I}$ a (t, T)-*model* of α.

We wish to evaluate $\mathcal{P}$ with respect to some fixed stream $\mathbf{D}$, called the *data stream*. We call a stream $\mathbf{I}$ an *interpretation stream* for $\mathbf{D}$ if $\mathbf{D} \subseteq \mathbf{I}$, and we say that such an interpretation stream $\mathbf{I}$ is a (t, T)-*model* of $\mathcal{P}$ if $\mathbf{I}, T, t \models_B \mathrm{B}(r) \to \mathrm{H}(r)$, for all rules $r \in \mathcal{P}$. The *reduct* of $\mathcal{P}$ with respect to $\mathbf{I}$ and T at time point t is given by

$$\mathcal{P}^{\mathbf{I},T,t} = \{r \in \mathcal{P} \mid \mathbf{I}, T, t \models_B \mathrm{B}(r)\}.$$

Definition 1. Let T be a closed interval in $\mathbb{N}$ and let $t \in T$. An interpretation stream $\mathbf{I}$ for $\mathbf{D}$ is a (t, T)-*answer stream* of $\mathcal{P}$ (for $\mathbf{D}$) if $\mathbf{I}$ is a (t, T)-model of $\mathcal{P}^{\mathbf{I},T,t}$ and there is no (t, T)-model $\mathbf{J}$ of $\mathcal{P}^{\mathbf{I},T,t}$ (for $\mathbf{D}$) with $\mathbf{J} \subsetneq \mathbf{I}$.

Note that the minimality condition in Definition 1 is given with respect to the same interval T, which is crucial. In fact, the following example shows that as a consequence of Definition 1, trivial programs may have infinitely many answer streams which is counter-intuitive from an answer set programming perspective.

Example 2. The ordinary program P consisting of a single fact a has the single answer set $\{a\}$. Given some arbitrary time point $t \geq 1$ for the evaluation of P within the LARS context defined above, we therefore expect P to have the single answer stream $\{a\}_t$. Unfortunately, under Beck et al's [5] semantics, P has *infinitely* many answer streams: the $(t, [t, t])$-answer stream $\{a\}_t$, the $(t, [t, t+1])$-answer stream $\{a\}_t\emptyset_{t+1}$, the $(t, [t, t+2])$-answer stream $\{a\}_t\emptyset_{t+1}\emptyset_{t+2}$ and so on.

The reason for the existence of the infinitely many answer streams for the trivial program in Example 2 is the preselection of the *fixed* interval T in Definition 1 for the semantic evaluation of programs. As a negative consequence of this choice, which appears to be an artificial simplification of the semantics of programs, is that some specifications which occur in practice cannot be expressed within the LARS language as is demonstrated by the following example.

Example 3. Let T be some interval and let $t \in T$ be some time point. According to Definition 1, the statement "a holds at t if b holds at every time point in T" is formalized by the single rule $a \leftarrow \Box b$ evaluated at time point t. Now consider the slightly different statement "a holds at t if b holds at every relevant time point in the support[2] of the input data." As natural as this statement seems, it is *not* expressible within the LARS language. The intuitive reason is that the support function is flexible and depends on the

[2]Recall from Section 2.2 that the support of a stream is given by the tightest interval containing all non-empty (i.e., relevant) time points.

data, whereas the preselected interval T is fixed by the programmer and therefore does not depend on the data. In a sense, preselecting fixed intervals for the semantic evaluation of programs contradicts the very idea of stream reasoning which aims at coping with data that changes at a high rate by incorporating window operators on a *syntactic* level for selecting relevant time points. Arguably, it is therefore more natural to formalize the first statement by the rule $a \leftarrow \boxplus_T \Box b$ thus syntactically encoding the restricted interval T, and interpreting $a \leftarrow \Box b$ as a formalization of the second statement (cf. Example 8).

2.4 Refined Semantics

We refine the FLP-style semantics of [5] (cf. Definition 1) by employing *dynamic* intervals. For this we first refine the entailment relation by using the support function in the definition of $\Box$ and $\Diamond$ for dynamically computing intervals instead of the fixed interval T used by [5]:

1. $\mathbf{I}, t \models_B \Diamond\alpha$ if $\mathbf{I}, t' \models_B \alpha$, for some $t' \in \operatorname{supp} \mathbf{I}$;
2. $\mathbf{I}, t \models_B \Box\alpha$ if $\mathbf{I}, t' \models_B \alpha$, for all $t' \in \operatorname{supp} \mathbf{I}$;
3. $\mathbf{I}, t \models_B @_{t'}\alpha$ if $\mathbf{I}, t' \models_B \alpha$, for $t' \geq 1$.

In case $\mathbf{I}, t \models_B \alpha$, we call $\mathbf{I}$ a *t-model* of α, and we call α *t-consistent* (resp., *t-inconsistent*) if α has at least one (resp., no) t-model. For convenience, we call α a *t-formula* if α is t-consistent.

Example 4. The formula $\boxplus_{[0,0]} @_2 a$ is 1-inconsistent since $@_2$ is a reference to time point 2 which is outside the scope of the window $\boxplus_{[0,0]}$ evaluated at time point 1. More precisely, let $\mathbf{I} = I_1 I_2 \ldots$ be an arbitrary stream and compute $\mathbf{I}[0, 0; 1] = I_1$ which implies $I_1, 2 \not\models_B a$—so $\mathbf{I}$ is not a 1-model of α.

Remark 5. Note that t-inconsistency of normal formulas can be easily verified by a syntactic check as in the example above and in the rest of the thesis we assume that (normal) formulas occurring in rule heads are t-consistent, for all relevant t.

We can now refine and simplify the definition of answer streams by omitting the reference to interval T which gives a more natural minimality condition.

Definition 6. An interpretation stream $\mathbf{I}$ for $\mathbf{D}$ is a *t-answer stream* of $\mathcal{P}$ (for $\mathbf{D}$) if $\mathbf{I}$ is a substream minimal t-model of $\mathcal{P}^{\mathbf{I},t}$.

Example 7. The ordinary program P of Example 2 consisting of the single fact a has the single t-answer stream $\{a\}_t$ as expected.

Example 8. The second statement in Example 3 is formalized according to Definition 6 by the rule $a \leftarrow \Box b$ as desired. For the first statement in Example 3, we need to distinguish two cases: (i) in case T is a subinterval of $\operatorname{supp}\mathbf{D}$, we can formalize the first statement via $a \leftarrow \boxplus_T \Box b$ as desired; however, (ii) if T is not a subinterval of $\operatorname{supp}\mathbf{D}$, then we need to add auxiliary symbols to $\mathbf{D}$ as in Proposition 2 below.

We now illustrate the above concepts in more detail with the following running example.

Example 9. Consider the program $\mathcal{P}$ consisting of the following rules:

$$@_2 a \xleftarrow{r_1} {\sim}@_7 c \qquad\qquad \boxplus_{[1,\infty]}\Box c \xleftarrow{r_3} {\sim}@_2 a$$

$$\boxplus_{[\infty,0]}\Box a \xleftarrow{r_2} {\sim}c \qquad\qquad \boxplus_{[2,3]}\Box(a \wedge b) \xleftarrow{r_4} \boxplus_{[0,1]}\Diamond c, \Box d.$$

Let the background data B consist of the single proposition d, and let the data stream $\mathbf{D}$ be given by

$$\mathbf{D} = \{a\}_1\{a,b\}_5\{c\}_{10}.$$

That is, the propositions a and b hold at time point 5 and so on. Then, the 5-answer streams of $\mathcal{P}$ (for $\mathbf{D}$) are given by:

$$\mathbf{I} = \{a\}_1\{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10};$$
$$\mathbf{J} = \{a\}_1\{a\}_2\{a\}_3\{a\}_4\{a,b\}_5\{c\}_{10}.$$

For instance, we verify that $\mathbf{I}$ is indeed a 5-answer stream of $\mathcal{P}$. First of all, note that $\mathbf{I}$ is a 5-model of $\mathcal{P}$ (for $\mathbf{D}$): (i) as c holds in $\mathbf{I}$ at time points 5 and 7, we have $\mathbf{I}, 5 \not\models_B {\sim}@_7 c$ and $\mathbf{I}, 5 \not\models_B {\sim}c$ which implies $\mathbf{I}, 5 \models_B r_1$ and $\mathbf{I}, 5 \models_B r_2$; (ii) as c holds at every time point in the interval $[4, 10]$, we have $\mathbf{I}, 5 \models_B \boxplus_{[1,\infty]}\Box c$ which implies $\mathbf{I}, 5 \models_B r_3$; and (iii) as a and b hold at every time point in $[3, 8]$, we have $\mathbf{I}, 5 \models_B \boxplus_{[2,3]}\Box(a \wedge b)$ which implies $\mathbf{I}, 5 \models_B r_4$.

Now we argue that $\mathbf{I}$ is a *minimal* 5-model of $\mathcal{P}^{\mathbf{I},5} = \{r_3, r_4\}$. To this end, suppose $\mathbf{I}' = \mathbf{I}'_1\mathbf{I}'_2 \ldots$ is a 5-model of $\mathcal{P}^{\mathbf{I},5}$ for $\mathbf{D}$ with $\mathbf{D} \subseteq \mathbf{I}' \subseteq \mathbf{I}$. Then, since $\mathbf{I}', 5 \models_B \mathrm{B}(r_3)$ and $\mathbf{I}', 5 \models_B \mathrm{B}(r_4)$, for $\mathbf{I}'$ to be a 5-model of $\mathcal{P}$ we must have $\mathbf{I}', 5 \models_B \mathrm{H}(r_3)$ and $\mathbf{I}', 5 \models_B \mathrm{H}(r_4)$; but this is equivalent to $\mathbf{I}'[1,\infty;5], t \models_B c$, for all $t \in [4, 10]$, and $\mathbf{I}'[2,3;5], t' \models_B a \wedge b$, for all $t' \in [3, 8]$ where $\mathbf{I}'[1,\infty;5] = I'_4 \ldots I'_{10}$ and $\mathbf{I}'[2,3;5] = I'_3 \ldots I'_8$, respectively. That is, we have $c \in I'_t$, for all $t \in [4, 10]$, and $a, b \in I'_{t'}$, for all $t' \in [3, 8]$—but this, together with $\mathbf{D} \subseteq \mathbf{I}' \subseteq \mathbf{I}$, immediately implies $\mathbf{I}' = \mathbf{I}$ which shows that $\mathbf{I}$ is indeed a minimal 5-model of $\mathcal{P}^{\mathbf{I},5}$ and therefore a 5-answer stream of $\mathcal{P}$.

It is important to emphasize that we can capture Beck et al's [5] semantics as follows.

Proposition 2. *Let $T = [t_1, t_2]$ be an interval and let $t \in T$ be some time point. An interpretation stream $\mathbf{I}$ for $\mathbf{D}$ is a (t, T)-answer stream of P if, and only if, $\mathbf{I} \cup \{\#\}_{t_1} \ldots \{\#\}_{t_2}$ is a t-answer stream of*

$$\boxplus_T P \cup \{@_t\# \mid t \in T\},$$

where # is a special symbol not occurring in Σ *and* $\boxplus_T P$ *consists of all rules of the form*

$$\boxplus_T r = \boxplus_T \alpha \leftarrow \boxplus_T \beta_1, \ldots, \boxplus_T \beta_j, \sim \boxplus_T \beta_{j+1}, \ldots, \sim \boxplus_T \beta_k, \quad r \in P.$$

At this point, we have successfully extended the FLP-style answer set semantics from ordinary to stream logic programs by refining Beck et al's [5] semantics. Unfortunately, as for other program extensions (cf. [28, 2]), our FLP-style semantics may permit circular justifications as is demonstrated by the following example.

Example 10. Consider the program $\mathcal{R}$ consisting of the following two rules:

$$a \xleftarrow{r_1} \Box b$$
$$b \xleftarrow{r_2} \Box a.$$

We argue that the t-model $\{a, b\}_t$ of $\mathcal{R}$ is a t-answer stream of $\mathcal{R}$, for every $t \geq 1$ (and $\mathbf{D} = B = \emptyset$): (i) The empty stream $\emptyset$ is not a t-model of $\mathcal{R}^{\{a,b\}_t,t} = \mathcal{R}$ since both rules fire in $\emptyset$; (ii) the stream $\{a\}_t$ is not a t-model of $\mathcal{R}$ since r_2 fires; (iii) the stream $\{b\}_t$ is not a t-model of $\mathcal{R}$ since r_1 fires. This shows that $\{a, b\}_t$ is indeed a subset minimal t-model of $\mathcal{R}^{\{a,b\}_t,t}$ and, hence, a t-answer stream of $\mathcal{R}$.

In the next two sections, we will develop the tools for formalizing the reasoning in Example 9 in an operational setting (cf. Example 18), while avoiding circular justifications.

3 Fixed Point Operators

In this section, we recast the following well-known fixed point operators from ordinary to stream logic programs: (i) the van Emden-Kowalski operator $\mathrm{T}_{\mathcal{P}}$ [31], and (ii) the Fitting operator $\Phi_{\mathcal{P}}$ [18]. This task turns out to be non-trivial due to the intricate properties of windows and other modal operators occurring in rule heads.

In the rest of the thesis, let $\mathbf{I}$ be a stream, let $\mathbf{D}$ be some data stream, let B be some background data, and let $t \geq 1$ be some fixed time point.

3.1 The Model Operator

In this subsection, we define an operator for the evaluation of rule heads. Specifically, given a normal t-formula α, we wish to construct a t-model of α which is in some sense minimal with respect to a given stream $\mathbf{I}$ (cf. Theorem 9).

Definition 11. For normal t-formulas α and β, and for $a \in \Sigma$, we define the *partial model operator* $\mathrm{M}_{\mathbf{I},t}$ at time point t and with respect to $\mathbf{I}$, inductively as follows:

$$\mathrm{M}_{\mathbf{I},t}(a) = \begin{cases} \{a\}_t & \text{if } a \notin B, \\ \emptyset & \text{if } a \in B; \end{cases}$$
$$\mathrm{M}_{\mathbf{I},t}(\alpha \wedge \beta) = \mathrm{M}_{\mathbf{I},t}(\alpha) \cup \mathrm{M}_{\mathbf{I},t}(\beta);$$
$$\mathrm{M}_{\mathbf{I},t}(\Box\alpha) = \bigcup_{t' \in \operatorname{supp}\mathbf{I}} \mathrm{M}_{\mathbf{I},t'}(\alpha);$$
$$\mathrm{M}_{\mathbf{I},t}(@_{t'}\alpha) = \mathrm{M}_{\mathbf{I},t'}(\alpha);$$
$$\mathrm{M}_{\mathbf{I},t}(\boxplus_{[\ell,r]}\alpha) = \mathrm{M}_{\mathbf{I}[\ell,r;t],t}(\alpha).$$

Finally, define the *model operator* $\mathrm{MM}_{\mathbf{I},t}$ to be the twofold application of $\mathrm{M}_{\mathbf{I},t}$, that is,

$$\mathrm{MM}_{\mathbf{I},t}(\alpha) = \mathrm{M}_{\mathrm{M}_{\mathbf{I},t}(\alpha),t}(\alpha).$$

One can easily derive the following computation rules for the model operator:

$$\mathrm{MM}_{\mathbf{I},t}(a) = \mathrm{M}_{\mathbf{I},t}(a);$$
$$\mathrm{MM}_{\mathbf{I},t}(@_{t'}\alpha) = \mathrm{MM}_{\mathbf{I},t'}(\alpha);$$
$$\mathrm{MM}_{\mathbf{I},t}(\boxplus_{[\ell,r]}\alpha) = \mathrm{MM}_{\mathbf{I}[\ell,r;t],t}(\alpha).$$

In case α is $\Box$-free, we will often write $\mathrm{M}_t(\alpha)$ instead of $\mathrm{M}_{\mathbf{I},t}(\alpha)$ to indicate that the evaluation of $\mathrm{M}_{\mathbf{I},t}$ does not depend on $\mathbf{I}$.

Example 12. Let α be the $\Box$-free normal 1-formula $\boxplus_{[0,0]}@_1 a \wedge @_2 b$, and compute

$$\mathrm{M}_{\emptyset,1}(\boxplus_{[0,0]}@_1 a \wedge @_2 b) = \mathrm{M}_{\emptyset[0,0;1],1}(a) \cup \mathrm{M}_{\emptyset,2}(b) = \{a\}_1\{b\}_2$$

which is a 1-model of α. On the other hand, for the normal 1-formula $\beta = \Box a \wedge b$ containing $\Box$, we obtain

$$\mathrm{M}_{\emptyset,1}(\Box a \wedge b) = \mathrm{M}_{\emptyset,1}(\Box a) \cup \mathrm{M}_{\emptyset,1}(b) = \mathrm{M}_{\emptyset,1}(b) = \{b\}_1$$

which is *not* a 1-model of β; however, by applying $\mathrm{M}_{\emptyset,1}$ twice, we do obtain a 1-model of β:

$$\mathrm{MM}_{\emptyset,1}(\Box a \wedge b) = \mathrm{M}_{\{b\}_1,1}(\Box a \wedge b) = \mathrm{M}_{\{b\}_1,1}(a) \cup \mathrm{M}_{\{b\}_1,1}(b) = \{a,b\}_1.$$

Intuitively, to obtain a 1-model of β, we have to apply $\mathrm{M}_{\emptyset,1}$ twice as the subformula b induces an expansion of the support of the generated stream which has to be taken into account for the generation of a 1-model for $\Box a$ (note that conjunctions are treated separately by the partial model operator).

Example 12 indicates that $\Box$ requires a special treatment. In fact, if α is $\Box$-free then $\mathrm{M}_{\mathbf{I},t}$ does not depend on $\mathbf{I}$ and, consequently, in this case $\mathrm{M}_{\mathbf{I},t}(\alpha)$ and $\mathrm{MM}_{\mathbf{I},t}(\alpha)$ coincide which simplifies the matters significantly.

Proposition 3. *For every $\Box$-free normal t-formula α, $\mathrm{M}_{\mathbf{I},t}(\alpha) = \mathrm{M}_{\mathbf{J},t}(\alpha)$ holds for all streams $\mathbf{I}$ and $\mathbf{J}$; consequently, $\mathrm{M}_{\mathbf{I},t}(\alpha) = \mathrm{MM}_{\mathbf{I},t}(\alpha)$.*

Proof. By definition of the partial model operator, $\mathrm{M}_{\mathbf{I},t}(\alpha)$ depends on $\mathbf{I}$ only if α contains $\Box$. The second assertion follows from the first with $\mathbf{J} = \mathrm{M}_{\mathbf{I},t}(\alpha)$ and $\mathrm{MM}_{\mathbf{I},t}(\alpha) = \mathrm{M}_{\mathbf{J},t}(\alpha)$. $\Box$

In the next two propositions, we show some monotonicity properties of the (partial) model operator.

Proposition 4. *For every normal t-formula α and all streams $\mathbf{I}$ and $\mathbf{J}$, $\mathbf{I} \subseteq \mathbf{J}$ implies $\mathrm{M}_{\mathbf{I},t}(\alpha) \subseteq \mathrm{M}_{\mathbf{J},t}(\alpha)$ and $\mathrm{MM}_{\mathbf{I},t}(\alpha) \subseteq \mathrm{MM}_{\mathbf{J},t}(\alpha)$. Moreover, $A \subseteq B$ implies $\mathrm{M}_{\mathbf{I},t}(A) \subseteq \mathrm{M}_{\mathbf{I},t}(B)$ and $\mathrm{MM}_{\mathbf{I},t}(A) \subseteq \mathrm{MM}_{\mathbf{I},t}(B)$, for all finite sets A and B of normal t-formulas.*

Proof. The first inclusion can be shown by a straightforward structural induction on α, so we prove here only the case $\alpha = \Box\beta$, for some normal t-formula β:

$$\mathrm{M}_{\mathbf{I},t}(\Box\beta) = \bigcup_{t' \in \operatorname{supp} \mathbf{I}} \mathrm{M}_{\mathbf{I},t'}(\beta) \subseteq \bigcup_{t' \in \operatorname{supp} \mathbf{J}} \mathrm{M}_{\mathbf{I},t'}(\beta) \overset{\text{IH}}{\subseteq} \bigcup_{t' \in \operatorname{supp} \mathbf{J}} \mathrm{M}_{\mathbf{J},t'}(\beta) = \mathrm{M}_{\mathbf{J},t}(\Box\beta).$$

The second inclusion, $\mathrm{MM}_{\mathbf{I},t}(\alpha) \subseteq \mathrm{MM}_{\mathbf{J},t}(\alpha)$, is a direct consequence of the first. Finally, the second part of the proposition is an immediate consequence of the first part and the definition of the (partial) model operator. $\square$

Proposition 5. *For every normal t-formula α,* $\operatorname{supp} \mathrm{M}_{\mathbf{I},t}(\alpha) = \operatorname{supp} \mathrm{MM}_{\mathbf{I},t}(\alpha)$ *and*

$$\mathrm{M}_{\mathbf{I},t}(\alpha) \subseteq \mathrm{MM}_{\mathbf{I},t}(\alpha).$$

Proof. The first identity can be proved by a straightforward structural induction on α.

We prove the inclusion by structural induction on α. The only non-trivial case is $\alpha = \Box\beta$, for some normal t-formula β:

$$\begin{aligned}
\mathrm{M}_{\mathbf{I},t}(\Box\beta) &= \bigcup_{t' \in \operatorname{supp} \mathbf{I}} \mathrm{M}_{\mathbf{I},t'}(\beta) \\
&\overset{\text{IH}}{\subseteq} \bigcup_{t' \in \operatorname{supp} \mathbf{I}} \mathrm{MM}_{\mathbf{I},t'}(\beta) \\
&= \bigcup_{t' \in \operatorname{supp} \mathbf{I}} \mathrm{M}_{\mathrm{M}_{\mathbf{I},t'}(\beta),t'}(\beta) \\
&\subseteq \bigcup_{t' \in \operatorname{supp} \mathbf{I}} \mathrm{M}_{\mathrm{M}_{\mathbf{I},t}(\Box\beta),t'}(\beta) \\
&\subseteq \bigcup_{t' \in \operatorname{supp} \mathbf{I}} \mathrm{M}_{\mathrm{M}_{\mathbf{I},t}(\Box\beta),t'}(\Box\beta) \\
&= \bigcup_{t' \in \operatorname{supp} \mathbf{I}} \mathrm{MM}_{\mathbf{I},t'}(\Box\beta) \\
&= \mathrm{MM}_{\mathbf{I},t}(\Box\beta)
\end{aligned}$$

where the second and third inclusion follows from Proposition 4 together with

$$\mathrm{M}_{\mathbf{I},t'}(\beta) \subseteq \mathrm{M}_{\mathbf{I},t}(\Box\beta) \quad \text{for all } t' \in \operatorname{supp} \mathbf{I},$$

and the last equality holds since:

$$\mathrm{MM}_{\mathbf{I},t}(\Box\beta) = \mathrm{MM}_{\mathbf{I},t'}(\Box\beta) \quad \text{for all } t' \in \operatorname{supp} \mathbf{I}. \tag{3.1}$$

To prove (3.1), first note that, by definition, $\mathrm{M}_{\mathbf{I},t}(\Box\beta) = \mathrm{M}_{\mathbf{I},t'}(\Box\beta)$ holds for all $t' \in \operatorname{supp} \mathbf{I}$; consequently:

$$\begin{aligned}
\mathrm{MM}_{\mathbf{I},t}(\Box\beta) &= \mathrm{M}_{\mathrm{M}_{\mathbf{I},t}(\Box\beta),t}(\Box\beta) \\
&= \bigcup_{t'' \in \operatorname{supp} \mathrm{M}_{\mathbf{I},t}(\Box\beta)} \mathrm{M}_{\mathrm{M}_{\mathbf{I},t}(\Box\beta),t''}(\beta) \\
&= \bigcup_{t'' \in \operatorname{supp} \mathrm{M}_{\mathbf{I},t'}(\Box\beta)} \mathrm{M}_{\mathrm{M}_{\mathbf{I},t'}(\Box\beta),t''}(\beta) \\
&= \mathrm{M}_{\mathrm{M}_{\mathbf{I},t'}(\Box\beta),t'}(\Box\beta) \\
&= \mathrm{MM}_{\mathbf{I},t'}(\Box\beta).
\end{aligned}$$

$\square$

It will often be convenient to separate a proof into a $\Box$-free and a general case. Therefore we define the translation $\alpha_{\mathbf{I},t}$ of α with respect to $\mathbf{I}$ at time point t to be the homomorphic extension to all normal t-formulas of:

$$(\Box\alpha)_{\mathbf{I},t} = \bigwedge_{t' \in \operatorname{supp}\mathbf{I}} @_{t'}\alpha_{\mathbf{I},t'}$$
$$(\boxplus_{[\ell,r]}\alpha)_{\mathbf{I},t} = \boxplus_{[\ell,r]}\alpha_{\mathbf{I}[\ell,r;t],t}$$

where we interpret the empty conjunction in $(\Box\alpha)_{\emptyset,t}$ as $\top$. Intuitively, $._{\mathbf{I},t}$ eliminates every $\Box$ occurring in α while preserving the meaning of α in the following sense.

Proposition 6. *For every normal t-formula α, and for all streams $\mathbf{I}$ and $\mathbf{J}$ with* $\operatorname{supp}\mathbf{I} = \operatorname{supp}\mathbf{J}$, *we have $\mathbf{I}, t \models_B \alpha$ if, and only if, $\mathbf{I}, t \models_B \alpha_{\mathbf{J},t}$.*

Proof. A straightforward structural induction on α. $\square$

Interestingly, the next proposition shows that we can simulate the model operator by the partial model operator applied to an appropriate translation of the input formula.

Proposition 7. *For every normal t-formula α, $\mathrm{M}_{\mathbf{I},t}(\alpha) = \mathrm{M}_{\mathbf{I},t}(\alpha_{\mathbf{I},t})$ and, consequently, $\mathrm{MM}_{\mathbf{I},t}(\alpha) = \mathrm{M}_{\mathbf{I},t}(\alpha_{\mathbf{M},t})$ with $\mathbf{M} = \mathrm{M}_{\mathbf{I},t}(\alpha)$.*

Proof. The first identity can be proved by a straightforward structural induction on α, and the second identity follows from the first, i.e., $\mathrm{MM}_{\mathbf{I},t}(\alpha) = \mathrm{M}_{\mathbf{M},t}(\alpha) = \mathrm{M}_{\mathbf{I},t}(\alpha_{\mathbf{M},t})$. $\square$

Monotone formulas inherit their name from the following property.

Proposition 8. *For every monotone formula α, $\mathbf{I}, t \models_B \alpha$ implies $\mathbf{J}, t \models_B \alpha$, for all streams $\mathbf{I} \subseteq \mathbf{J}$.*

We are now ready to prove our first theorem which shows that $\mathrm{MM}_{\mathbf{I},t}(\alpha)$ is a t-model of α which is in some sense “minimal” (with respect to $\mathbf{I}$).

Theorem 9. *For every normal t-formula α, $\mathrm{MM}_{\mathbf{I},t}(\alpha)$ is a t-model of α, that is,*

$$\mathrm{MM}_{\mathbf{I},t}(\alpha), t \models_B \alpha.$$

In case α is $\Box$-free, a single application of $\mathrm{M}_{\mathbf{I},t}$ suffices, that is, $\mathrm{M}_{\mathbf{I},t}(\alpha), t \models_B \alpha$. Moreover, if $\mathbf{I}$ is a t-model of α, then $\mathrm{MM}_{\mathbf{I},t}(\alpha) \subseteq \mathbf{I}$.

Proof. We start with the second assertion and prove by structural induction on α that in case α is $\Box$-free, $\mathrm{M}_{\mathbf{I},t}(\alpha), t \models_B \alpha$. The induction hypothesis $\alpha = a \in \Sigma$, and the case $\alpha = @_{t'}\beta$ are straightforward. In what follows, let β and γ denote normal t-formulas. For $\alpha = \beta \wedge \gamma$, we have $\mathrm{M}_{\mathbf{I},t}(\beta \wedge \gamma) = \mathrm{M}_{\mathbf{I},t}(\beta) \cup \mathrm{M}_{\mathbf{I},t}(\gamma)$ and, by induction hypothesis, $\mathrm{M}_{\mathbf{I},t}(\beta), t \models_B \beta$ and $\mathrm{M}_{\mathbf{I},t}(\gamma), t \models_B \gamma$. Since β and γ are $\Box$-free, we have $\mathrm{M}_{\mathbf{I},t}(\beta) \cup \mathrm{M}_{\mathbf{I},t}(\gamma), t \models_B \beta \wedge \gamma$ as a consequence of Proposition 8 (recall that $\Box$-free normal formulas are monotone). Finally, for $\alpha = \boxplus_{[\ell,r]}\beta$ we have $\mathrm{M}_{\mathbf{I},t}(\boxplus_{[\ell,r]}\beta) = \mathrm{M}_{\mathbf{I},t}(\beta)$ and, by induction hypothesis, $\mathrm{M}_{\mathbf{I},t}(\beta), t \models_B \beta$. Since $\boxplus_{[\ell,r]}\beta$ is t-consistent by assumption, $\mathrm{M}_{\mathbf{I},t}(\beta) = \mathrm{M}_{\mathbf{I},t}(\beta)[\ell, r; t]$ (cf. Remark 5 and 13) and, hence, $\mathrm{M}_{\mathbf{I},t}(\beta)[\ell, r; t], t \models_B \beta$ which is equivalent to $\mathrm{M}_{\mathbf{I},t}(\boxplus_{[\ell,r]}\beta), t \models_B \boxplus_{[\ell,r]}\beta$.

We now turn to the general case and prove that $\mathrm{MM}_{\mathbf{I},t}(\alpha), t \models_B \alpha$ holds for any normal t-formula α, by first translating α into a $\Box$-free formula, and then referring to the first part of the proof. Let $\mathbf{M} = \mathrm{M}_{\mathbf{I},t}(\alpha)$. Since $\alpha_{\mathbf{M},t}$ is $\Box$-free, we know from the first part of the proof that

$$\mathrm{M}_{\mathbf{I},t}(\alpha_{\mathbf{M},t}), t \models_B \alpha_{\mathbf{M},t}. \tag{3.2}$$

By Proposition 7,

$$\mathrm{MM}_{\mathbf{I},t}(\alpha) = \mathrm{M}_{\mathbf{I},t}(\alpha_{\mathbf{M},t}). \tag{3.3}$$

From (3.2) and (3.3) we infer

$$\mathrm{MM}_{\mathbf{I},t}(\alpha), t \models_B \alpha_{\mathbf{M},t}.$$

Now since $\operatorname{supp} \mathbf{M} = \operatorname{supp} \mathrm{MM}_{\mathbf{I},t}(\alpha)$ (cf. Proposition 5), Proposition 6 proves our claim.

Finally, a straightforward structural induction on α shows $\mathrm{MM}_{\mathbf{I},t}(\alpha) \subseteq \mathbf{I}$. $\Box$

Remark 13. We want to emphasize that the requirement in Theorem 9 of α being t-consistent is essential. For instance, reconsider the 1-inconsistent normal formula $\alpha = \boxplus_{[0,0]}@_2 a$ of Remark 5, and compute $\mathrm{MM}_{\mathbf{I},1}(\alpha) = \{a\}_2$ which is *not* a 1-model of α.

3.2 The van Emden-Kowalski Operator

We are now ready to extend the well-known van Emden-Kowalski operator to the class of stream logic programs.

Definition 14. We define the *van Emden-Kowalski operator* $\mathrm{T}_{\mathcal{P},\mathbf{D},t}$ of $\mathcal{P}$ (for $\mathbf{D}$ at time point t), for every stream $\mathbf{I}$, by

$$\mathrm{T}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) = \mathbf{D} \cup \mathrm{MM}_{\mathbf{I},t}(\{\mathrm{H}(r) \mid r \in \mathcal{P} : \mathbf{I}, t \models_B \mathrm{B}(r)\}).$$

As for ordinary logic programs [31], prefixed points of the van Emden-Kowalski operator $\mathrm{T}_{\mathcal{P},\mathbf{D},t}$ characterize the models of $\mathcal{P}$ (for $\mathbf{D}$ at time point t).

Theorem 10. *A stream* $\mathbf{I}$ *is a* t*-model of* $\mathcal{P}$ *if, and only if,* $\mathbf{I}$ *is a prefixed point of* $\mathrm{T}_{\mathcal{P},\mathbf{D},t}$.

Proof. Suppose $\mathbf{I}$ is a t-model of $\mathcal{P}$ and note that this is equivalent to

$$\mathbf{I}, t \models_B \mathrm{H}(\mathcal{P}^{\mathbf{I},t}). \tag{3.4}$$

Moreover, note that we can rewrite the van Emden-Kowalski operator more compactly as

$$\mathrm{T}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) = \mathbf{D} \cup \mathrm{MM}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})).$$

So we have to show $\mathrm{MM}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) \subseteq \mathbf{I}$ (recall that $\mathbf{D} \subseteq \mathbf{I}$ holds by assumption), but this follows directly from (3.4) together with the last part of Theorem 9.

For the other direction, we show that $\mathrm{MM}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) \subseteq \mathbf{I}$ implies $\mathbf{I}, t \models_B \mathrm{H}(\mathcal{P}^{\mathbf{I},t})$. Let $\mathbf{M} = \mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t}))$. By Proposition 5,

$$\mathbf{M} = \mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) \subseteq \mathrm{MM}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) = \mathrm{M}_{\mathbf{M},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) \subseteq \mathbf{I}. \tag{3.5}$$

On the other hand, $\mathbf{M} \subseteq \mathbf{I}$ and Proposition 4 imply

$$\mathrm{M}_{\mathbf{M},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) \subseteq \mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})). \tag{3.6}$$

Consequently, from (3.5) and (3.6) we infer

$$\mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) = \mathrm{MM}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})). \tag{3.7}$$

Intuitively, (3.7) means that in case $\mathrm{MM}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) \subseteq \mathbf{I}$, one application of $\mathrm{M}_{\mathbf{I},t}$ suffices (cf. Theorem 9 and Example 12). Moreover, Proposition 7 implies

$$\mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) = \mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})_{\mathbf{I},t}) \subseteq \mathbf{I}. \tag{3.8}$$

Since $\mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})_{\mathbf{I},t})$ is a t-model of $\mathrm{H}(\mathcal{P}^{\mathbf{I},t})_{\mathbf{I},t}$ (cf. Theorem 9),

$$\mathrm{M}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})_{\mathbf{I},t}) \subseteq \mathbf{I}$$

holds by (3.8), and $\mathrm{H}(\mathcal{P}^{\mathbf{I},t})_{\mathbf{I},t}$ is monotone, Proposition 8 and (3.8) imply

$$\mathbf{I}, t \models_B \mathrm{H}(\mathcal{P}^{\mathbf{I},t})_{\mathbf{I},t}. \tag{3.9}$$

Finally, Proposition 6 and (3.9) imply $\mathbf{I}, t \models_B \mathrm{H}(\mathcal{P}^{\mathbf{I},t})$. □

Example 15. Reconsider the program $\mathcal{P}$ of Example 9 consisting of the following rules:

$$@_2 a \xleftarrow{r_1} {\sim}@_7 c \qquad \boxplus_{[1,\infty]}\Box c \xleftarrow{r_3} {\sim}@_2 a$$
$$\boxplus_{[\infty,0]}\Box a \xleftarrow{r_2} {\sim}c \qquad \boxplus_{[2,3]}\Box(a \wedge b) \xleftarrow{r_4} \boxplus_{[0,1]}\Diamond c, \Box d.$$

We have argued in Example 9 that the interpretation stream

$$\mathbf{I} = \{a\}_1\{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10}$$

of $\mathcal{P}$ for $\mathbf{D} = \{a\}_1\{a,b\}_5\{c\}_{10}$ and $B = \{d\}$ is a 5-model of $\mathcal{P}$. Now we want to rigorously prove that $\mathbf{I}$ is a 5-model of $\mathcal{P}$ by showing that $\mathbf{I}$ is a prefixed point of $\mathrm{T}_{\mathcal{P},\mathbf{D},5}$. We compute:

$$\begin{aligned}
\mathbf{M} &= \mathrm{M}_{\mathbf{I},5}(\{\mathrm{H}(r_3),\mathrm{H}(r_4)\}) \\
&= \mathrm{M}_{\mathbf{I},5}(\boxplus_{[1,\infty]}\Box c \wedge \boxplus_{[2,3]}\Box(a\wedge b)) \\
&= \mathrm{M}_{\mathbf{I},5}(\boxplus_{[1,\infty]}\Box c) \cup \mathrm{M}_{\mathbf{I},5}(\boxplus_{[2,3]}\Box(a\wedge b)) \\
&= \bigcup_{t\in\operatorname{supp}\mathbf{I}[1,\infty;5]} \mathrm{M}_{\mathbf{I}[1,\infty;5],t}(c) \cup \bigcup_{t\in\operatorname{supp}\mathbf{I}[2,3;5]} \mathrm{M}_{\mathbf{I}[2,3;5],t}(a\wedge b) \\
&= \{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10}
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{T}_{\mathcal{P},\mathbf{D},5}(\mathbf{I}) &= \mathbf{D} \cup \mathrm{M}\mathrm{M}_{\mathbf{I},5}(\{\mathrm{H}(r_3),\mathrm{H}(r_4)\}) \\
&= \mathbf{D} \cup \mathrm{M}_{\mathbf{M},5}(\boxplus_{[1,\infty]}\Box c) \cup \mathrm{M}_{\mathbf{M},5}(\boxplus_{[2,3]}\Box(a\wedge b)) \\
&= \mathbf{D} \cup \bigcup_{t\in\operatorname{supp}\mathbf{M}[1,\infty;5]} \mathrm{M}_{\mathbf{M}[1,\infty;5],t}(c) \cup \bigcup_{t\in\operatorname{supp}\mathbf{M}[2,3;5]} \mathrm{M}_{\mathbf{M}[2,3;5],t}(a\wedge b) \\
&= \mathbf{D} \cup \mathbf{M} \\
&= \{a\}_1\{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10} \\
&= \mathbf{I}.
\end{aligned}$$

3.3 The Fitting Operator

In the presence of negation, the van Emden-Kowalski operator is non-monotonic and cannot be iterated bottom-up. We therefore extend the (3-valued) Fitting operator [18] from ordinary to stream logic programs as follows. Firstly, we define a *3-valued stream* to be a pair of streams $(\mathbf{I},\mathbf{J})$ with $\mathbf{I}\subseteq\mathbf{J}$ or, equivalently, a sequence of pairs $(I_1,J_1)(I_2,J_2)\ldots$ with $I_t\subseteq J_t$ for all $t\geq 1$, with the intuitive meaning that every $a\in I_t$ (resp., $a\notin J_t$) is *true* (resp., *false*) at time point t, whereas every $a\in J_t - I_t$ is *undefined* at t.

We then define the *precision ordering*[1] $\subseteq_p$ on the set of all 3-valued streams by

$$(\mathbf{I},\mathbf{J}) \subseteq_p (\mathbf{I}',\mathbf{J}') \quad\Longleftrightarrow\quad \mathbf{I}\subseteq\mathbf{I}' \text{ and } \mathbf{J}'\subseteq\mathbf{J}.$$

Intuitively, $(\mathbf{I},\mathbf{J}) \subseteq_p (\mathbf{I}',\mathbf{J}')$ means that $(\mathbf{I}',\mathbf{J}')$ is a "tighter" interval inside $(\mathbf{I},\mathbf{J})$. The maximal elements with respect to $\subseteq_p$ are exactly the (2-valued) streams where we identify each stream $\mathbf{I}$ with $(\mathbf{I},\mathbf{I})$. Note that since distinct streams have no upper bound with respect to the precision ordering, the set of all 3-valued streams is not a lattice.

We extend the entailment relation to 3-valued streams as follows.

[1]The precision ordering corresponds to the *knowledge ordering* $\leq_k$ in [18]; cf. [12].

Definition 16. For every 3-valued stream $(\mathbf{I}, \mathbf{J})$ and formula α,

$$(\mathbf{I}, \mathbf{J}), t \models_B \alpha \quad \Longleftrightarrow \quad \mathbf{K}, t \models_B \alpha \text{ for every } \mathbf{K} \in [\mathbf{I}, \mathbf{J}].$$

The intuition behind Definition 16 is as follows. Recall that a formula α containing $\neg, \rightarrow$, or $\Box$ may be non-monotone in the sense of Proposition 8, and in this case we have to take all possible extensions $\mathbf{K} \in [\mathbf{I}, \mathbf{J}]$ of $\mathbf{I}$ into account.

Now define the *Fitting operator* $\Phi_{\mathcal{P},\mathbf{D},t}$ of $\mathcal{P}$ for $\mathbf{D}$ at time point t, for every 3-valued stream $(\mathbf{I}, \mathbf{J})$, by

$$\Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I}, \mathbf{J}) = \mathbf{D} \cup \mathrm{MM}_{\mathbf{I},t}(\{\mathrm{H}(r) \mid r \in \mathcal{P} : (\mathbf{I}, \mathbf{J}), t \models_B \mathrm{B}(r)\}).$$

The only difference between $\Phi_{\mathcal{P},\mathbf{D},t}$ and $\mathrm{T}_{\mathcal{P},\mathbf{D},t}$ is that $\Phi_{\mathcal{P},\mathbf{D},t}$ evaluates the body of a rule in a 3-valued stream, which guarantees the monotonicity of $\Phi_{\mathcal{P},\mathbf{D},t}$ with respect to the precision ordering (cf. Proposition 12). As for ordinary logic programs, the Fitting operator encapsulates the van Emden-Kowalski operator.

Proposition 11. *For every stream* $\mathbf{I}$*,* $\Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I}, \mathbf{I}) = \mathrm{T}_{\mathcal{P},\mathbf{D},t}(\mathbf{I})$*.*

4 Fixed Point Semantics

In this section, we define a fixed point semantics for the class of stream logic programs in terms of the Fitting operator defined above. More precisely, we first show that the Fitting operator is monotone with respect to the precision ordering, and conclude that certain least fixed points, the so-called $\Phi_{\mathcal{P},\mathbf{D},t}$-answer streams, exist (cf. Definition 17). Then, we compare our constructive semantics to the FLP-style semantics of [5] (cf. Theorem 16 and Theorem 17).

Proposition 12. *The Fitting operator $\Phi_{\mathcal{P},\mathbf{D},t}$ is monotone with respect to the precision ordering.*

Proof. Let $(\mathbf{I},\mathbf{J})$ and $(\mathbf{I}',\mathbf{J}')$ be 3-valued streams with $(\mathbf{I},\mathbf{J}) \subseteq_p (\mathbf{I}',\mathbf{J}')$. For an arbitrary rule $r \in \mathcal{P}$, $(\mathbf{I},\mathbf{J}), t \models_B \mathrm{B}(r)$ implies $(\mathbf{I}',\mathbf{J}'), t \models_B \mathrm{B}(r)$ as a direct consequence of Definition 16. Finally, Proposition 4 implies $\Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I},\mathbf{J}) \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I}',\mathbf{J}')$. □

A consequence of Proposition 12 is that in case $\mathbf{I}$ is a t-model of $\mathcal{P}$, $\Phi_{\mathcal{P},\mathbf{D},t}(-,\mathbf{I})$ is a monotone operator on the *complete* lattice $[\emptyset,\mathbf{I}]$, since for every $\mathbf{K} \in [\emptyset,\mathbf{I}]$,

$$\Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{K},\mathbf{I}) \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I},\mathbf{I}) = \mathrm{T}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) \subseteq \mathbf{I}$$

holds by Proposition 11 and Theorem 10.

Define the operator $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}$ on the set of all t-models of $\mathcal{P}$ by

$$\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) = lfp\, \Phi_{\mathcal{P},\mathbf{D},t}(-,\mathbf{I}).$$

The soundness of $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}$ is justified by the well-known Knaster-Tarski theorem which guarantees the existence of least fixed points of monotone operators on complete lattices.

We are now ready to formulate our fixed point semantics.

Definition 17. We call every t-model $\mathbf{I}$ of $\mathcal{P}$ (for $\mathbf{D}$) a $\Phi_{\mathcal{P},\mathbf{D},t}$-*answer stream* if $\mathbf{I}$ is a fixed point of $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}$.

For readers not familiar with the fixed point theory of logic programming, we briefly recall the basic intuitions behind Definition 17 in the setting of ordinary logic programs. For the moment, let $\mathcal{P}$ be an ordinary program, and let I be an interpretation of $\mathcal{P}$. The *Gelfond-Lifschitz reduct* of $\mathcal{P}$ with respect to I is defined by[1]

$$\mathcal{P}^I = \left\{ \mathrm{H}(r) \leftarrow \mathrm{B}^+(r) \mid r \in \mathcal{P} : I \cap \mathrm{B}^-(r) = \emptyset \right\}.$$

We call I an *answer set* [20] of $\mathcal{P}$ if I is the least model of $\mathcal{P}^I$, which coincides with $\mathit{lfp}\,\mathrm{T}_{\mathcal{P}^I}$. So the computation of answer sets according to [20] is a two-step process, and [18] showed how these two steps can be emulated by a single (monotone) operator, namely the Fitting operator $\Phi_\mathcal{P}$. Specifically, the identity $\Phi_\mathcal{P}(-, I) = \mathrm{T}_{\mathcal{P}^I}$ implies that I is an answer set of $\mathcal{P}$ if and only if I is the least fixed point of $\Phi_\mathcal{P}(-, I)$ or, equivalently, if I is a fixed point of $\Phi^\dagger_\mathcal{P}$. It is now clear that Definition 17 is an extension of the ordinary answer set semantics to stream logic programs.

Proposition 13. *For an ordinary program $\mathcal{P}$, I is an answer set of $\mathcal{P}$ if, and only if, $\mathbf{I} = I_t = I$ is a t-answer stream of $\mathcal{P}$, for every $t \geq 1$.*

We illustrate our fixed point semantics with the following example.

Example 18. Reconsider the program $\mathcal{P}$ of Example 9 consisting of the following rules:

$$@_2 a \xleftarrow{r_1} {\sim}@_7 c \qquad \boxplus_{[1,\infty]}\Box c \xleftarrow{r_3} {\sim}@_2 a$$

$$\boxplus_{[\infty,0]}\Box a \xleftarrow{r_2} {\sim}c \qquad \boxplus_{[2,3]}\Box(a \wedge b) \xleftarrow{r_4} \boxplus_{[0,1]}\Diamond c, \Box d.$$

We have argued in Example 9 that the interpretation stream

$$\mathbf{I} = \{a\}_1\{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10}$$

of $\mathcal{P}$ for $\mathbf{D} = \{a\}_1\{a,b\}_5\{c\}_{10}$ and $B = \{d\}$ is a 5-answer stream of $\mathcal{P}$. We now want to apply our tools from above to rigorously prove that $\mathbf{I}$ is a $\Phi_{\mathcal{P},\mathbf{D},5}$-answer stream by computing $\Phi^\dagger_{\mathcal{P},\mathbf{D},5}(\mathbf{I})$ bottom-up as follows. We start the computation with $\mathbf{I}_0 = \emptyset$:

$$\Phi_{\mathcal{P},\mathbf{D},5}(\emptyset, \mathbf{I}) = \mathbf{D} \cup \mathrm{MM}_{\emptyset,5}(\mathrm{H}(r_3)) = \mathbf{D} \cup \mathrm{MM}_{\emptyset,5}(\boxplus_{[1,\infty]}\Box c) = \mathbf{D} \cup \mathrm{MM}_{\emptyset[1,\infty;5],5}(\Box c) = \mathbf{D}$$

where the last equality follows from $\emptyset[1,\infty;5] = \emptyset$ and $\mathrm{MM}_{\emptyset,5}(\Box c) = \emptyset$. Then we continue the computation with $\mathbf{I}_1 = \mathbf{D}$:

$$\begin{aligned}
\Phi_{\mathcal{P},\mathbf{D},5}(\mathbf{I}_1, \mathbf{I}) &= \mathbf{D} \cup \mathrm{MM}_{\mathbf{I}_1,5}(\mathrm{H}(r_3)) \\
&= \mathbf{D} \cup \mathrm{MM}_{\mathbf{I}_1,5}(\boxplus_{[1,\infty]}\Box c) \\
&= \mathbf{D} \cup \mathrm{MM}_{\mathbf{I}_1[1,\infty;5],5}(\Box c) \\
&= \mathbf{D} \cup \mathrm{M}_{\mathrm{M}_{\mathbf{I}_1[1,\infty;5],5}(\Box c),5}(\Box c) \\
&= \mathbf{D} \cup \mathrm{M}_{\{c\}_4 \dots \{c\}_{10},5}(\Box c) \\
&= \mathbf{D} \cup \{c\}_4 \dots \{c\}_{10} \\
&= \{a\}_1\{c\}_4\{a,b,c\}_5\{c\}_6\{c\}_7\{c\}_8\{c\}_9\{c\}_{10} \\
&= \mathbf{I}_2.
\end{aligned}$$

[1]Here $\mathrm{B}^-(r)$ denotes the negated atoms in the body of r, and $\mathrm{B}^+(r)$ denotes $\mathrm{B}(r) - \mathrm{B}^-(r)$.

For the third iteration, we first compute

$$\begin{aligned}\mathbf{M} &= \mathrm{M}_{\mathbf{I}_2,5}(\{\mathrm{H}(r_3), \mathrm{H}(r_4)\})\\ &= \mathrm{M}_{\mathbf{I}_2,5}(\boxplus_{[1,\infty]}\Box c \wedge \boxplus_{[2,3]}\Box(a \wedge b))\\ &= \mathrm{M}_{\mathbf{I}_2,5}(\boxplus_{[1,\infty]}\Box c) \cup \mathrm{M}_{\mathbf{I}_2,5}(\boxplus_{[2,3]}\Box(a \wedge b))\\ &= \mathrm{M}_{\mathbf{I}_2[1,\infty;5],5}(\Box c) \cup \mathrm{M}_{\mathbf{I}_2[2,3;5],5}(\Box(a \wedge b))\\ &= \{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10}\end{aligned}$$

and then:

$$\begin{aligned}\Phi_{\mathcal{P},\mathbf{D},5}(\mathbf{I}_2,\mathbf{I}) &= \mathbf{D} \cup \mathrm{MM}_{\mathbf{I}_2,5}(\{\mathrm{H}(r_3), \mathrm{H}(r_4)\})\\ &= \mathbf{D} \cup \mathrm{MM}_{\mathbf{I}_2,5}(\boxplus_{[1,\infty]}\Box c \wedge \boxplus_{[2,3]}\Box(a \wedge b))\\ &= \mathbf{D} \cup \mathrm{M}_{\mathbf{M},5}(\boxplus_{[1,\infty]}\Box c) \cup \mathrm{M}_{\mathbf{M},5}(\boxplus_{[2,3]}\Box(a \wedge b))\\ &= \mathbf{D} \cup \mathrm{M}_{\mathbf{M}[1,\infty;5]}(\Box c) \cup \mathrm{M}_{\mathbf{M}[2,3;5],5}(\Box(a \wedge b))\\ &= \mathbf{D} \cup \mathbf{M}\\ &= \{a\}_1\{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10}\\ &= \mathbf{I}_3.\end{aligned}$$

Finally, we verify that $\mathbf{I} = \mathbf{I}_3$ is a fixed point of (cf. Example 15 and Proposition 11):

$$\Phi_{\mathcal{P},\mathbf{D},5}(\mathbf{I},\mathbf{I}) = \mathrm{T}_{\mathcal{P},\mathbf{D},5}(\mathbf{I}) = \mathbf{I}.$$

In summary, the above computations show that $\mathbf{I}$ is a fixed point of $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},5}$ or, equivalently, a $\Phi_{\mathcal{P},\mathbf{D},5}$-answer stream.

We now wish to relate our fixed point semantics from above to the FLP-style semantics of [5] presented in Section 2. Firstly, we prove some auxiliary lemmas.

Lemma 14. *Let* $(\mathbf{I},\mathbf{J})$ *be a 3-valued stream, and let* $\mathbf{K} \in [\mathbf{I},\mathbf{J}]$. *Then,* $\Phi_{\mathcal{P}^{\mathbf{K},t},\mathbf{D},t}(\mathbf{I},\mathbf{J}) = \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I},\mathbf{J})$.

Proof. Define $\mathcal{P}^{(\mathbf{I},\mathbf{J}),t} = \{r \in \mathcal{P} \mid (\mathbf{I},\mathbf{J}), t \models_B \mathrm{B}(r)\}$. As a direct consequence of 3-valued entailment (cf. Definition 16), we have the following inclusions:

$$\mathcal{P}^{(\mathbf{I},\mathbf{J}),t} \subseteq \mathcal{P}^{\mathbf{K},t} \subseteq \mathcal{P}. \tag{4.1}$$

By the monotonicity of $\mathrm{MM}_{\mathbf{I},t}$ (cf. Proposition 4), $\Phi_{\mathcal{R},\mathbf{D},t}(\mathbf{I},\mathbf{J}) \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I},\mathbf{J})$ holds whenever $\mathcal{R} \subseteq \mathcal{P}$, for all programs $\mathcal{P}$ and $\mathcal{R}$. Therefore, we can conclude from (4.1):

$$\Phi_{\mathcal{P}^{(\mathbf{I},\mathbf{J}),t},\mathbf{D},t}(\mathbf{I},\mathbf{J}) \subseteq \Phi_{\mathcal{P}^{\mathbf{K},t},\mathbf{D},t}(\mathbf{I},\mathbf{J}) \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I},\mathbf{J}). \tag{4.2}$$

Note that by definition, we have $\Phi_{\mathcal{P}^{(\mathbf{I},\mathbf{J}),t},\mathbf{D},t}(\mathbf{I},\mathbf{J}) = \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I},\mathbf{J})$ which together with (4.2) entails $\Phi_{\mathcal{P}^{\mathbf{K},t},\mathbf{D},t}(\mathbf{I},\mathbf{J}) = \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I},\mathbf{J})$. □

Lemma 15. *For every prefixed point* $\mathbf{K} \subseteq \mathbf{I}$ *of* $\mathrm{T}_{\mathcal{P},\mathbf{D},t}$*,* $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) \subseteq \mathbf{K}$*.*

Proof. We compute $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}(\mathbf{I})$ bottom-up. Clearly, $\mathbf{K}_0 = \emptyset \subseteq \mathbf{I}$. Since $\Phi_{\mathcal{P},\mathbf{D},t}(-,\mathbf{I})$ is monotone, we have

$$\mathbf{K}_1 = \Phi_{\mathcal{P},\mathbf{D},t}(\emptyset,\mathbf{I}) \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{K},\mathbf{K}) = \mathrm{T}_{\mathcal{P},\mathbf{D},t}(\mathbf{K}) \subseteq \mathbf{K}.$$

Similarly, we can compute $\mathbf{K}_2 = \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{K}_1,\mathbf{I}) \subseteq \mathbf{K}$ and so on, which shows that the limit $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}(\mathbf{I})$ is contained in $\mathbf{K}$, i.e., $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) \subseteq \mathbf{K}$. □

We are now ready to prove the main result of this thesis.

Theorem 16. *Every* $\Phi_{\mathcal{P},\mathbf{D},t}$*-answer stream is a* t*-answer stream of* $\mathcal{P}$*.*

Proof. By assumption, we have $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) = \mathbf{I}$ which implies

$$\Phi^{\dagger}_{\mathcal{P}^{\mathbf{I},t},\mathbf{D},t}(\mathbf{I}) = \mathbf{I} \tag{4.3}$$

by Lemma 14, that is, $\mathbf{I}$ is a $\Phi_{\mathcal{P}^{\mathbf{I},t},\mathbf{D},t}$-answer stream. Since every $\Phi_{\mathcal{P}^{\mathbf{I},t},\mathbf{D},t}$-answer stream is a t-model of $\mathcal{P}^{\mathbf{I},t}$, it remains to show that $\mathbf{I}$ is a *minimal* t-model of $\mathcal{P}^{\mathbf{I},t}$. For this suppose there exists some stream $\mathbf{K}$ with $\mathbf{K} \subsetneq \mathbf{I}$ such that $\mathbf{K}$ is a t-model of $\mathcal{P}^{\mathbf{I},t}$. Then, by Theorem 10, we have $\mathrm{T}_{\mathcal{P}^{\mathbf{I},t},\mathbf{D},t}(\mathbf{K}) \subseteq \mathbf{K} \subsetneq \mathbf{I}$ which implies

$$\Phi^{\dagger}_{\mathcal{P}^{\mathbf{I},t},\mathbf{D},t}(\mathbf{I}) \subseteq \mathbf{K} \subsetneq \mathbf{I}$$

by Lemma 15—a contradiction to (4.3). □

Theorem 16 shows that our fixed point semantics is sound with respect to our FLP-style semantics. However, the next example shows that the converse of Theorem 16 fails in general.

Example 19. Reconsider the program $\mathcal{R}$ of Example 10 consisting of the rules

$$a \xleftarrow{r_1} \Box b$$
$$b \xleftarrow{r_2} \Box a.$$

In Example 10 we have seen that $\{a,b\}_t$ is a t-answer stream of $\mathcal{R}$, for every $t \geq 1$ (and $\mathbf{D} = B = \emptyset$). On the other hand, we have $\Phi_{\mathcal{R},t}(\emptyset,\{a,b\}_t) = \emptyset$ which shows that $\{a,b\}_t$ is *not* a $\Phi_{\mathcal{R},t}$-answer stream.

5 Level Mappings

In this section, we define level mappings for stream logic programs in the vein of [28], and prove in Theorem 17 that $\Phi_{\mathcal{P},\mathbf{D},t}$-answer streams characterize those t-models which posses a level mapping or, equivalently, which are free of circular justifications.

Firstly, we recast the notion of a partitioning to stream logic programs.

Definition 20. A *partitioning* of a stream $\mathbf{I}$ is a sequence of streams $\mathbf{S} = (\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_m)$, $m \geq 1$, where $\mathbf{S}_0 = \emptyset$, $\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_m = \mathbf{I}$, $\mathbf{S}_i \neq \emptyset$ for every $i \geq 1$, and $\mathbf{S}_i \cap \mathbf{S}_j = \emptyset$ for every $i \neq j \neq 0$.

We now define level mappings over such partitionings.

Definition 21. A *t-level mapping* of a stream $\mathbf{I}$ with respect to $\mathcal{P}$ (for $\mathbf{D}$) is a partitioning $\mathbf{S} = (\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_m)$ of $\mathbf{I}$ such that for all $1 \leq i \leq m$,

$$\mathbf{S}_i \subseteq \mathbf{D} \cup \mathrm{MM}_{\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{i-1},t}(\{\mathrm{H}(r) \mid r \in \mathcal{P} : (\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{i-1}, \mathbf{I}), t \models_B \mathrm{B}(r)\}). \tag{5.1}$$

We call $\mathbf{S}$ a *total* t-level mapping of $\mathbf{I}$ if in addition $\mathbf{I} = \mathbf{S}_0 \cup \ldots \cup \mathbf{S}_m$ is a t-model of $\mathcal{P}$.

The intuition behind Definition 21 is as follows. A partitioning $\mathbf{S} = (\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_m)$ with $\mathbf{S}_i = S_{i,1} S_{i,2} \ldots$, $1 \leq i \leq m$, is a t-level mapping of $\mathbf{I}$ if each proposition $a \in S_{i,t_i}$ (i.e., a holds in level i at time point t_i) is non-circularly justified by the rules in $\mathcal{P}$, i.e., either $a \in D_{t_i}$ or there exists a rule r in $\mathcal{P}$ justifying a at time point t_i, that is, a occurs in the head of r and the body of r "fires" in a level smaller than i. For $\mathbf{S}$ to be called total, we additionally require $\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_m = \mathbf{I}$ to contain every proposition occurring in a rule head which is derivable from $\mathbf{I}$, i.e., $\mathbf{D} \cup \mathrm{MM}_{\mathbf{I},t}(\mathrm{H}(\mathcal{P}^{\mathbf{I},t})) \subseteq \mathbf{I}$ which, by Theorem 10, is equivalent to $\mathbf{I}$ being a t-model of $\mathcal{P}$. Clearly, a stream $\mathbf{I}$ possessing a t-level mapping is free of circular justifications.

Note that we can rewrite (5.1) more compactly as

$$\mathbf{S}_i \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{i-1}, \mathbf{I}) \tag{5.2}$$

which shows the direct relationship between t-level mappings and the Fitting operator.

Example 22. Once again, reconsider the program $\mathcal{P}$ of Example 9. In Example 18 we have seen that

$$\mathbf{I} = \{a\}_1\{a,b\}_3\{a,b,c\}_4\{a,b,c\}_5\{a,b,c\}_6\{a,b,c\}_7\{a,b,c\}_8\{c\}_9\{c\}_{10}$$

is a $\Phi_{\mathcal{P},\mathbf{D},5}$-answer stream for $\mathbf{D} = \{a\}_1\{a,b\}_5\{c\}_{10}$. We construct the total 5-level mapping $\mathbf{S} = (\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3)$ of $\mathbf{I}$ for $\mathcal{P}$ as follows:[1]

$$\begin{aligned}
\mathbf{S}_0 &= \mathbf{I}_0 = \emptyset \\
\mathbf{S}_1 &= \mathbf{I}_1 - \mathbf{I}_0 = \mathbf{D} = \{a\}_1\{a,b\}_5\{c\}_{10} \\
\mathbf{S}_2 &= \mathbf{I}_2 - \mathbf{I}_1 = \{c\}_4\{c\}_5\{c\}_6\{c\}_7\{c\}_8\{c\}_9 \\
\mathbf{S}_3 &= \mathbf{I}_3 - \mathbf{I}_2 = \{a,b\}_3\{a,b\}_4\{a,b\}_6\{a,b\}_7\{a,b\}_8
\end{aligned}$$

where $\mathbf{I}_0 = \emptyset, \mathbf{I}_1 = \mathbf{D}, \mathbf{I}_2$, and $\mathbf{I}_3 = \mathbf{I}$ are the intermediate results in the bottom-up computation of $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},5}(\mathbf{I})$ (cf. Example 18).

We can characterize the $\Phi_{\mathcal{P},\mathbf{D},t}$-answer streams in terms of t-level mappings as follows.

Theorem 17. *A stream $\mathbf{I}$ is a $\Phi_{\mathcal{P},\mathbf{D},t}$-answer stream if, and only if, there is a total t-level mapping $\mathbf{S}$ of $\mathbf{I}$ with respect to $\mathcal{P}$.*

Proof. For the direction from left to right, we construct the total t-level mapping $\mathbf{S}$ of the $\Phi_{\mathcal{P},\mathbf{D},t}$-answer stream $\mathbf{I}$ as in Example 22. Let $\mathbf{I}_0 = \emptyset, \mathbf{I}_1, \ldots, \mathbf{I}_m = \mathbf{I}$ be the intermediate results of the bottom-up computation of $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) = \mathbf{I}$, i.e.,

$$\Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I}_{i-1}, \mathbf{I}) = \mathbf{I}_i, \quad 1 \le i \le m,$$

and define $\mathbf{S}_0 = \emptyset$ and $\mathbf{S}_i = \mathbf{I}_i - \mathbf{I}_{i-1}$, for all $1 \le i \le m$. By construction, we have $\mathbf{I}_i = \mathbf{S}_1 \cup \ldots \cup \mathbf{S}_i$, for all $1 \le i \le m$, which directly yields the inclusion in (5.2); moreover, since $\mathbf{I}$ is a t-model of $\mathcal{P}$, $\mathbf{S}$ is a total t-level mapping of $\mathbf{I}$ with respect to $\mathcal{P}$.

For the opposite direction, let $\mathbf{S} = (\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_m)$, $m \ge 1$, be a total t-level mapping of $\mathbf{I}$ with respect to $\mathcal{P}$. We need to show that $\mathbf{I} = \bigcup \mathbf{S}$, with $\bigcup \mathbf{S} = \mathbf{S}_1 \cup \ldots \cup \mathbf{S}_m$, is a fixed point of $\Phi^{\dagger}_{\mathcal{P},\mathbf{D},t}$. Since $\mathbf{S}$ is total, $\mathbf{I}$ is a t-model of $\mathcal{P}$, so we have by (5.2), the monotonicity of $\Phi_{\mathcal{P},\mathbf{D},t}$ (cf. Proposition 12), Proposition 11, and Theorem 10:

$$\begin{aligned}
\mathbf{I} = \bigcup \mathbf{S} &\subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{m-1}, \mathbf{I}) \\
&\subseteq \Phi_{\mathcal{P},\mathbf{D},t}\left(\bigcup \mathbf{S}, \mathbf{I}\right) \\
&= \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{I}, \mathbf{I}) = \mathrm{T}_{\mathcal{P},\mathbf{D},t}(\mathbf{I}) \subseteq \mathbf{I}.
\end{aligned}$$

So $\mathbf{I}$ is a fixed point of $\Phi_{\mathcal{P},\mathbf{D},t}(-, \mathbf{I})$ and it remains to show that there is no fixed point $\mathbf{K} \subsetneq \mathbf{I}$ of $\Phi_{\mathcal{P},\mathbf{D},t}(-, \mathbf{I})$. Suppose, towards a contradiction, that for some $\mathbf{K} \subsetneq \mathbf{I}$,

$$\Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{K}, \mathbf{I}) = \mathbf{K}. \tag{5.3}$$

[1] By "$-$" we mean here the point-wise relative complement, e.g., $\{a\}_1\{b\}_2 - \{b\}_2 = \{a\}_1$.

Since $\mathbf{K} \subsetneq \mathbf{I}$ there is some i, $1 \leq i \leq m$, such that $\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{i-1} \subseteq \mathbf{K} \subseteq \mathbf{S}_1 \cup \ldots \cup \mathbf{S}_i$. So by (5.2) and Proposition 12 we have

$$\mathbf{S}_i \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{i-1}, \mathbf{I}) \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{K}, \mathbf{I}) = \mathbf{K}.$$

Consequently,

$$\begin{aligned}\mathbf{K} &= \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{K}, \mathbf{I}) \\ &\subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_i, \mathbf{I}) \\ &\subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{i-1} \cup \mathbf{K}, \mathbf{I}) \\ &= \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{K}, \mathbf{I}) \\ &= \mathbf{K},\end{aligned}$$

which implies

$$\Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_i, \mathbf{I}) = \mathbf{K}. \tag{5.4}$$

From (5.2), (5.4), and the monotonicity of $\Phi_{\mathcal{P},\mathbf{D},t}$ (cf. Proposition 12) we infer

$$\begin{aligned}&\mathbf{S}_{i-1} \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_{i-2}, \mathbf{I}) \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_i, \mathbf{I}) = \mathbf{K}; \\ &\mathbf{S}_{i+1} \subseteq \Phi_{\mathcal{P},\mathbf{D},t}(\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_i, \mathbf{I}) = \mathbf{K}.\end{aligned}$$

Hence, $\mathbf{S}_j \subseteq \mathbf{K}$ for all $1 \leq j \leq m$, and so $\bigcup \mathbf{S} \subseteq \mathbf{K} \subsetneq \mathbf{I}$—a contradiction to $\bigcup \mathbf{S} = \mathbf{I}$. $\square$

Example 23. Reconsider the program $\mathcal{R}$ of Example 10 consisting of the following two rules:

$$\begin{aligned}a &\leftarrow \Box b \\ b &\leftarrow \Box a.\end{aligned}$$

In Example 10 we have seen that for every $t \geq 1$, $\mathbf{I} = \{a, b\}_t$ is a t-answer stream of $\mathcal{R}$. Note that a and b are circularly justified in $\mathcal{R}$. As $\mathbf{I}$ is *not* a $\Phi_{\mathcal{P},\mathbf{D},t}$-answer stream (cf. Example 19), there is no total t-level mapping of $\mathbf{I}$ by Theorem 17.

Note that Theorem 17 together with Theorem 16 (and Example 19) characterize our semantics as the strict constructive subclass of our FLP-style semantics.

6 Conclusion

This thesis contributed to the foundations of stream reasoning [9, 25, 1] by providing a sound and constructive extension of the answer set semantics from ordinary to stream logic programs. For this we refined the FLP-style semantics of [5]. Moreover, we extended the van Emden-Kowalski and Fitting operators from ordinary to stream logic programs. As a result of our investigations, we obtained constructive semantics of stream logic programs with nice properties. More precisely, it turned out that our fixed point semantics can be characterized in terms of level mappings or, equivalently, is free of circular justifications, which is regarded as a positive feature. Moreover, the algebraic nature of our fixed point semantics yields computational proofs which are satisfactory from a mathematical point of view.

As our fixed point semantics hinges on the (extended) Fitting operator, it can be reformulated within the algebraic framework of Approximation Fixed Point Theory (AFT) [12, 10], which is grounded in the work of Fitting on bilattices in logic programming (cf. [18]), and which captures a number of related (non-monotonic) formalisms (e.g., [11] and [2]). In the future, we wish to apply the full framework of AFT to LARS, which provides a well-founded semantics [32], a notion of strong and uniform equivalence [30], a bottom-up semantics for disjunctive programs [2], and a recently introduced algebraic notion of groundedness [6].

Bibliography

[1] D. D. Aglio, E. D. Valle, F. van Harmelen, and A. Bernstein. Stream reasoning: a survey and outlook. *Data Science*, 1(1-2):59–83, 2017.

[2] C. Antić, T. Eiter, and M. Fink. HEX semantics via approximation fixpoint theory. In P. Cabalar and T. C. Son, editors, *Proc. 12th International Conference on Logic Programming and Nonmonotonic Reasoning*, LNCS 8148, pages 102–115, 2013.

[3] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. *VLBD Journal*, 15(2):121–142, 2006.

[4] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.

[5] H. Beck, M. Dao-Tran, and T. Eiter. LARS: A logic-based framework for analytic reasoning over streams. *Artificial Intelligence*, 261:16–70, 2018.

[6] B. Bogaerts, J. Vennekens, and M. Denecker. Grounded fixpoints and their application in knowledge representation. *Artificial Intelligence*, 224:51–71, 2015.

[7] G. Brewka, T. Eiter, and M. Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, Dec. 2011.

[8] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[9] E. Della Valle, S. Ceri, F. Van Harmelen, and D. Fensel. It's a streaming world! Reasoning upon rapidly changing information. *IEEE Intelligent Systems*, 24(6):83–89, 2009.

[10] M. Denecker, M. Bruynooghe, and J. Vennekens. Approximation fixpoint theory and the semantics of logic and answer set programs. In E. Erdem, J. Lee, Y. Lierler, and D. Pearce, editors, *Correct Reasoning*, volume 7265 of *LNCS*, pages 178–194, Heidelberg, 2012. Springer-Verlag.

[11] M. Denecker, V. Marek, and M. Truszczyński. Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence*, 143(1):79–122, 2003.

[12] M. Denecker, V. Marek, and M. Truszczyński. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation*, 192(1):84–121, 2004.

[13] T. Eiter, G. Ianni, and T. Krennwallner. Answer set programming: a primer. In *Reasoning Web. Semantic Technologies for Information Systems,* volume 5689 of *Lecture Notes in Computer Science*, pages 40–110. Springer, Heidelberg, 2009.

[14] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, 172(12-13):1495–1539, Aug. 2008.

[15] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In L. P. Kaelbling and A. Saffiotti, editors, *Proc. 19th International Joint Conference on Artificial Intelligence*, pages 90–96, 2005.

[16] W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: semantics and complexity. In J. Alferes and J. Leite, editors, *JELIA 2004*, LNCS 3229, pages 200–212. Springer, Berlin, 2004.

[17] W. Faber, G. Pfeifer, and N. Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, 2011.

[18] M. Fitting. Fixpoint semantics for logic programming—a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.

[19] M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: from theory to practice. *Artificial Intelligence*, 187-188(C):52–89, 2012.

[20] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):365–385, 1991.

[21] E. Giunchiglia, Y. Lierler, and M. Maratea. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning*, 36(4):345–377, 2006.

[22] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[23] V. Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54, 2002.

[24] V. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In K. R. Apt, V. Marek, M. Truszczyński, and D. S. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer, Berlin,

1999.

[25] A. Mileo, M. Dao-Tran, T. Eiter, and M. Fink. Stream reasoning. In *Encyclopedia of Database Systems*, page 7. Springer Science+Business Media, 2 edition, 2017.

[26] I. Niemelä, P. Simons, and T. Soininen. Stable model semantics of weight constraint rules. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 1999),* volume 1730 of *Lecture Notes in Computer Science*, pages 317–331. Springer-Verlag, Berlin, 1999.

[27] N. Pelov. *Semantics of Logic Programs with Aggregates.* PhD thesis, Katholieke Universiteit Leuven, Leuven, Mar. 2004.

[28] Y.-D. Shen, K. Wang, T. Eiter, M. Fink, C. Redl, T. Krennwallner, and J. Deng. FLP answer set semantics without circular justifications for general logic programs. *Artificial Intelligence*, 213:1–41, 2014.

[29] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.

[30] M. Truszczyński. Strong and uniform equivalence of nonmonotonic theories–an algebraic approach. *Annals of Mathematics and Artificial Intelligence*, 48(3-4):245–265, 2006.

[31] M. H. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[32] A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):619–649, 1991.

Curriculum Vitae

Christian Antić
christian.antic@icloud.com
Vienna University of Technology
Wiedner Hauptstraße 8–10
1040 Vienna, Austria

Personal Information

Born on July 17th, 1986, in Baden Austria

Austrian citizen with Serbian background

Education at TU Vienna

2005–2010 Bachelor Program in Software and Information Engineering

2007–2010 Bachelor Program in Technical Mathematics (enough courses completed to be permitted to the doctoral program)

2010–2012 Master Program in Computational Intelligence (with distinction)

2014–2021 Doctoral Program in Technical Mathematics

since 2020 Doctoral Program in Computer Science

Professional Experience at TU Vienna

2008–2010 Tutor and assistant for various logic in computer science courses

1.6.2013–31.10.2013 Project assistant at the Research Unit of Knowledge Based Systems

1.4.2014–28.02.2015 Project assistant at the Research Unit of Knowledge Based Systems

1.4.2015–30.09.2015 Project assistant at the Institute for Discrete Mathematics and Geometry

1.8.2016–31.12.2016 Project assistant at the Institute for Discrete Mathematics and Geometry

1.1.2017–31.12.2017 (marginal employment) Project assistant at the Institute for Discrete Mathematics and Geometry

3.4.2018–30.06.2018 Project assistant at the Institute for Discrete Mathematics and Geometry

5.11.2018–31.01.2019 (marginal employment) Project assistant at the Institute for Discrete Mathematics and Geometry

Awards

[a1] Best Student Paper Award at ICLP 2014 for the paper [j1].

Publications and Preprints

[i4] Antić, C., 2021, Sequential composition of answer set programs, `https://arxiv.org/pdf/2104.12156.pdf`.

[i3] Antić, C., 2021, Analogical proportions, `https://arxiv.org/pdf/2006.02854.pdf`.

[i2] Antić, C., 2021, Sequential composition of propositional Horn theories, `https://arxiv.org/pdf/2009.05774.pdf`.

[i1] Antić, C., 2020, Logic-based analogical reasoning and learning, `https://arxiv.org/pdf/1809.09938.pdf`.

[j2] Antić, C., 2020, Fixed point semantics for stream reasoning, Artificial Intelligence, vol. 288.

[j1] Antić, C., 2014, On cascade products of answer set programs, Theory and Practice of Logic Programming, vol. 14, no. 4-5, pp. 711–723.

[c1] Antić, C., Eiter, T., Fink, M., 2013, HEX semantics via approximation fixpoint theory,

LPNMR 2013, pp 102–115.

Wien, am August 12, 2021

Christian Antić