Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

DatalogMTL with Negation under Stable Models Semantics

Przemysław A. Wałęga[1] , **David J. Tena Cucala**[1] , **Egor V. Kostylev**[2] , **Bernardo Cuenca Grau**[1]
[1]Department of Computer Science, University of Oxford, UK
[2]Department of Informatics, University of Oslo, Norway
{przemyslaw.walega, david.tena.cucala,bernardo.cuenca.grau}@cs.ox.ac.uk, egork@ifi.uio.no

Abstract

We introduce negation under stable models semantics in DatalogMTL—a temporal extension of Datalog with metric operators. As a result, we obtain a rule language which combines the power of answer set programming with the temporal dimension provided by metric operators. We show that, in this setting, reasoning becomes undecidable over the rationals and decidable in EXPSPACE in data complexity over the integers. We also show that, if we restrict our attention to forward-propagating programs (where rules propagate information in a single temporal direction), reasoning over integers becomes PSPACE-complete in data complexity and hence no harder than over positive programs; however, reasoning over the rationals in this fragment remains undecidable.

1 Introduction

DatalogMTL (Brandt et al. 2018) extends positive Datalog with operators from metric temporal logic (MTL) (Koymans 1990), interpreted over the rational numbers or the integers. DatalogMTL is a powerful language for temporal reasoning, which can capture many other formalisms such as Datalog_{1S} (Chomicki and Imieliński 1988; 1989) and Templog (Abadi and Manna 1989), and which has found applications in areas such as ontology-based data access (Brandt et al. 2018), stream reasoning (Wałęga, Cuenca Grau, and Kaminski 2019), and logic programming (Brzoska 1998).

An important limitation of DatalogMTL in practice is that negation is disallowed in rules—indeed, motivated by a range of applications, there has recently been growing interest in logics that combine non-monotonic negation with temporal constructs (Cabalar and Vega 2007; Aguado et al. 2013; Cabalar et al. 2018; 2020; Beck, Dao-Tran, and Eiter 2018; Zaniolo 2012). In the context of DatalogMTL, Tena Cucala et al. (2021) recently proposed an extension of the language with stratified negation as failure and showed that the additional expressive power does not increase complexity of reasoning regardless of whether we consider the rational or the integer timeline (Wałęga et al. 2020).

In this paper we take a next step in this direction and consider DatalogMTL equipped with non-stratifiable negation interpreted under the stable models semantics (Gelfond and Lifschitz 1988; Brooks et al. 2007; Nogueira et al. 2001). This extension paves the way for the use of DatalogMTL in applications where derived information can be retracted in light of new evidence, minimality of models is required, or temporal inertia rules need to be formalised. For instance, consider a dental practice with the policy that patients with an appointment on a given time t are automatically booked for a check-up appointment one year later (i.e., at time $t+1$, represented by metric operator $\boxplus_1$), but this appointment must be cancelled if the patient makes another appointment in between—that is, within the interval $(t+0, t+1)$ represented by $\boxplus_{(0,1)}$. This policy can be written using the rule

$$\boxplus_1 \mathit{Appoint}(x) \leftarrow \mathit{Appoint}(x) \wedge \mathsf{not}\, \boxplus_{(0,1)} \mathit{Appoint}(x),$$

which is not stratifiable as it involves recursion via negation.

Our setting is closely related to recent research on combining answer set programming (ASP) with temporal logics. For example, the logic TEL (Cabalar and Vega 2007; Aguado et al. 2013; Cabalar et al. 2018) combines ASP with linear temporal logic, and the LARS language combines ASP with window-based temporal constructs for stream reasoning (Beck, Dao-Tran, and Eiter 2018). The logic recently proposed by Cabalar et al. (2020) is maybe the closest to our work, as it combines stable models semantics with propositional MTL interpreted over the natural numbers; this logic, however, is rather different from DatalogMTL, where the use of logical connectives and MTL operators is restricted in the spirit of Datalog to disallow disjunction and "existential" MTL operators (such as diamond, since, or until operators) in rule heads. As we show in our paper, considering a language with such restrictions can lead to favourable computational behaviour.

Our contributions are as follows. In Section 3 we present our extension DatalogMTL$^\neg$ of DatalogMTL with negation under stable models semantics, which we define, similarly to other temporal ASP formalisms, in terms of interpretations for the *here-and-there* intuitionistic logic (Heyting 1930; Pearce 1996). The resulting language extends both DatalogMTL with stratified negation and Datalog with stable negation. The main reasoning problem we consider is existence of a stable model for a program and dataset. We show in Section 4 that, in this setting, reasoning over the rational timeline is undecidable. The proof uses an involved reduction from the halting problem and applies even to propositional forward-propagating programs, where rules cannot propagate information towards the past, and data containing only bounded intervals. To regain decid-

ability, in Section 5, we focus on the integer timeline. We show in Section 5.1 that discreteness of the timeline has a crucial influence on the logic's computational behaviour, as reasoning becomes decidable and in EXPSPACE in data complexity; this is shown by exploiting Büchi automata and their complements to find candidate stable models and verify their minimality. Then, in Section 5.2 we show that, when restricted to forward-propagating (or equivalently backwards-propagating) programs and bounded datasets, reasoning becomes PSPACE-complete and so, no harder than for negation-free DatalogMTL. This is in stark contrast with the undecidability of the same fragment over the rationals.

2 Preliminaries

A *timeline* $\mathbb{T}$ is either the set $\mathbb{Q}$ of rationals or the set $\mathbb{Z}$ of integers. A $\mathbb{T}$-*time point* is an element of $\mathbb{T}$. A $\mathbb{T}$-*interval* ϱ is a subset of $\mathbb{T}$ satisfying two properties: first, for all $t_1, t_2 \in \varrho$ and $t \in \mathbb{T}$ such that $t_1 < t < t_2$, it is the case that $t \in \varrho$; second, the greatest lower bound ϱ^- and the least upper bound ϱ^+ of ϱ both belong to $\mathbb{T} \cup \{-\infty, \infty\}$. The bounds ϱ^- and ϱ^+ are called the *left* and *right endpoints* of ϱ, respectively. A $\mathbb{T}$-interval is *punctual* if it contains exactly one number, it is *positive* if it contains no negative numbers, it is *bounded* if both its left and right endpoints are in $\mathbb{T}$, and it is *closed* if it includes its both endpoints. In these and similar notions, we often omit the reference to $\mathbb{T}$ if it is clear from the context. We consider binary representations of integers, and fractional representations of rational numbers with an integer numerator and a positive integer denominator. We use standard representations of the form $\langle \varrho^-, \varrho^+ \rangle$ for a non-empty interval ϱ, where the *left bracket* $\langle$ is either [or (, the *right bracket* $\rangle$ is either] or), and, if numeric, the endpoints ϱ^- and ϱ^+ are represented as explained above. Brackets [and] indicate that the corresponding endpoints are included in the interval, whereas (and) indicate that they are not included; note that, by this convention, [and] cannot be used with endpoints $-\infty$ and ∞. We will often abbreviate a punctual interval $[t, t]$ as t. For a given timeline, no two intervals have the same representation; so, if it is clear from the context, we will abuse notation and identify each interval representation with the interval it represents.

Assume a function-free first-order vocabulary and a timeline $\mathbb{T}$. A *relational atom* is an expression of the form $P(\mathbf{s})$, where P is a predicate and $\mathbf{s}$ is a tuple of constants and variables of the same arity as P. A *metric atom* is an expression given by the following grammar, where $P(\mathbf{s})$ ranges over relational atoms and ϱ over positive intervals:

$$M ::= \top \mid \bot \mid P(\mathbf{s}) \mid \Diamond_\varrho M \mid \oplus_\varrho M \mid \boxminus_\varrho M \mid \boxplus_\varrho M \mid M \mathcal{S}_\varrho M \mid M \mathcal{U}_\varrho M.$$

A metric atom is *ground* if it mentions no variables. A *metric fact* is an expression $M@\varrho$, with M a ground metric atom and ϱ a non-empty interval; it is *relational* if so is M. A *dataset* is a finite set of relational facts; it is *bounded* if so are all intervals it mentions. An *interpretation* $\mathfrak{I}$ specifies, for each ground relational atom $P(\mathbf{c})$ and each time point $t \in \mathbb{T}$, whether $P(\mathbf{c})$ is *satisfied* at t, in which case we write $\mathfrak{I}, t \models_\mathbb{T} P(\mathbf{c})$. This notion extends to other ground metric atoms as given in Table 1. Interpretation $\mathfrak{I}$ is a *model* of a metric fact $M@\varrho$, written $\mathfrak{I} \models_\mathbb{T} M@\varrho$, if $\mathfrak{I}, t \models M$ for all $t \in \varrho \cap \mathbb{T}$; it is a *model* of a set $\mathcal{M}$ of metric facts (e.g., a dataset) if it is a model of all facts in $\mathcal{M}$. Set $\mathcal{M}$ *entails* a set $\mathcal{M}'$ of metric facts, written $\mathcal{M} \models \mathcal{M}'$, if every model of $\mathcal{M}$ is a model of $\mathcal{M}'$. Interpretation $\mathfrak{I}$ *contains* interpretation $\mathfrak{I}'$, written $\mathfrak{I}' \subseteq \mathfrak{I}$, if for each ground relational atom $P(\mathbf{c})$ and time point $t \in \mathbb{T}$, having $\mathfrak{I}', t \models_\mathbb{T} P(\mathbf{c})$ implies $\mathfrak{I}, t \models_\mathbb{T} P(\mathbf{c})$. Furthermore, $\mathfrak{I}$ is the *least* interpretation in a set X of interpretations, if $\mathfrak{I} \subseteq \mathfrak{I}'$ for every $\mathfrak{I}' \in X$.

$\mathfrak{I}, t \models_\mathbb{T} \top$	for each $t \in \mathbb{T}$
$\mathfrak{I}, t \models_\mathbb{T} \bot$	for no $t \in \mathbb{T}$
$\mathfrak{I}, t \models_\mathbb{T} \Diamond_\varrho M$	iff $\mathfrak{I}, t' \models_\mathbb{T} M$ for some t' with $t - t' \in \varrho$
$\mathfrak{I}, t \models_\mathbb{T} \oplus_\varrho M$	iff $\mathfrak{I}, t' \models_\mathbb{T} M$ for some t' with $t' - t \in \varrho$
$\mathfrak{I}, t \models_\mathbb{T} \boxminus_\varrho M$	iff $\mathfrak{I}, t' \models_\mathbb{T} M$ for all t' with $t - t' \in \varrho$
$\mathfrak{I}, t \models_\mathbb{T} \boxplus_\varrho M$	iff $\mathfrak{I}, t' \models_\mathbb{T} M$ for all t' with $t' - t \in \varrho$
$\mathfrak{I}, t \models_\mathbb{T} M_1 \mathcal{S}_\varrho M_2$	iff $\mathfrak{I}, t' \models_\mathbb{T} M_2$ for some t' with $t - t' \in \varrho$ and $\mathfrak{I}, t'' \models_\mathbb{T} M_1$ for all $t'' \in (t', t)$
$\mathfrak{I}, t \models_\mathbb{T} M_1 \mathcal{U}_\varrho M_2$	iff $\mathfrak{I}, t' \models_\mathbb{T} M_2$ for some t' with $t' - t \in \varrho$ and $\mathfrak{I}, t'' \models_\mathbb{T} M_1$ for all $t'' \in (t, t')$

Table 1: Semantics of ground metric atoms

3 DatalogMTL with Negation

In this section we propose DatalogMTL$^\neg$, which extends DatalogMTL with stratified negation as defined in (Tena Cucala et al. 2021) to support unstratified use of negation in rules interpreted under stable model semantics.

The syntax of DatalogMTL$^\neg$ is the natural extension of the positive case: rule bodies are conjunctions of atoms and negated atoms, whereas rule heads are defined as in DatalogMTL. Forward-propagating DatalogMTL$^\neg$ programs are also defined analogously to the positive case (Wałęga, Cuenca Grau, and Kaminski 2019), by requiring that rule bodies and heads do not mention metric operators referring to the future and to the past, respectively.

Definition 1. *A* rule r *is an expression of the form*

$$M \leftarrow M_1 \wedge \cdots \wedge M_k \wedge \mathsf{not}\, M_{k+1} \wedge \cdots \wedge \mathsf{not}\, M_m, \quad (1)$$

where $m \geq k \geq 0$, each M_i is a metric atom, and M is a metric atom specified by the following grammar, where $P(\mathbf{s})$ ranges over relational atoms and ϱ over positive intervals:

$$M ::= \top \mid \bot \mid P(\mathbf{s}) \mid \boxminus_\varrho M \mid \boxplus_\varrho M.$$

The head *of r is the consequent M and the* body *is the conjunction in the antecedent, where $M_1, \ldots, M_k$ are its* positive *body atoms, and $M_{k+1}, \ldots, M_m$ are its* negated *body atoms. Rule r is* safe *if each variable it mentions occurs in some positive body atom, it is* ground *if it has no variables, it is* positive *if it has no negated body atoms. A* (DatalogMTL$^\neg$) program *is a finite set of safe rules; it is* ground *or* positive *if all its rules are.*

DatalogMTL$^\neg_{\mathsf{FP}}$ *is the fragment of* DatalogMTL$^\neg$ *consisting of* forward-propagating *programs, where operators $\oplus$,*

$\boxplus$, and $\mathcal{U}$ are disallowed in rule bodies, and operator $\boxminus$ is disallowed in the head.

The definition of stable models for Datalog with negation relies on the reduct construction by Gelfond and Lifschitz (1988), which has been adapted to various extensions of ASP (Faber, Leone, and Pfeifer 2004). These reduct constructions, however, do not have a natural equivalent in DatalogMTL$^\neg$, where interpretations may satisfy a fact at some, but not all points of the infinite timeline, and it is thus unclear which rules or atoms should be included in the reduct. Instead, following the approach of Cabalar and Vega; Cabalar et al. (2007; 2020), we define stable models for DatalogMTL$^\neg$ analogously to the models of *equilibrium logic* (Pearce 1996), which are defined in terms of interpretations for the *here-and-there* intuitionistic logic (Heyting 1930). The semantics of this logic is based on here-and-there interpretations, so let us start by generalising such interpretations to the context of DatalogMTL$^\neg$.

For the remainder of this section, we fix timeline $\mathbb{T}$ which will be implicit in all our defintions and technical results.

Definition 2. *An* HT-interpretation *is a pair $(\mathfrak{H}, \mathfrak{T})$ of interpretations such that $\mathfrak{H} \subseteq \mathfrak{T}$. It is an* HT-model *of a dataset $\mathcal{D}$ if $\mathfrak{H}$ is a model of $\mathcal{D}$. Furthermore, $(\mathfrak{H}, \mathfrak{T})$ is an* HT-model *of a rule r of Form* (1) *if, for each assignment ν of constants to variables making r ground and for each $t \in \mathbb{T}$, both of the following conditions hold:*

1. *if $\mathfrak{H}, t \models_{\mathbb{T}} \nu(M_i)$ for all $i \in \{1, \dots, k\}$ and $\mathfrak{T}, t \not\models_{\mathbb{T}} \nu(M_j)$ for all $j \in \{k+1, \dots, m\}$, then $\mathfrak{H}, t \models_{\mathbb{T}} \nu(M)$; and*
2. *if $\mathfrak{T}, t \models_{\mathbb{T}} \nu(M_i)$ for all $i \in \{1, \dots, k\}$ and $\mathfrak{T}, t \not\models_{\mathbb{T}} \nu(M_j)$ for all $j \in \{k+1, \dots, m\}$, then $\mathfrak{T}, t \models_{\mathbb{T}} \nu(M)$.*

Finally, $(\mathfrak{H}, \mathfrak{T})$ is an HT-model *of a program if it is an HT-model of all its rules.*

An HT-interpretation is therefore a pair $(\mathfrak{H}, \mathfrak{T})$ of standard interpretations. Interpretation $\mathfrak{H}$ is contained in $\mathfrak{T}$, and determines whether a dataset is satisfied. Although both interpretations are used to evaluate rules, it is $\mathfrak{T}$ which evaluates negative body atoms and thus determines when a rule with negation can be "triggered". When this happens, the rule ensures that if any of $\mathfrak{H}$ or $\mathfrak{T}$ satisfies the positive body atoms, then it will also satisfy the head.

Observe that if $(\mathfrak{H}, \mathfrak{T})$ is an HT-interpretation then, by $\mathfrak{H} \subseteq \mathfrak{T}$, all relational facts entailed by $\mathfrak{H}$ are also entailed by $\mathfrak{T}$. We show next that this property can be generalised to arbitrary metric facts.

Proposition 3. *For every HT-interpretation $(\mathfrak{H}, \mathfrak{T})$, a metric atom M, and a time point t, if $\mathfrak{H}, t \models M$ then $\mathfrak{T}, t \models M$.*

Proof sketch. We proceed by induction on the structure of M. If M is $\top$ or $\bot$, the claim holds trivially, and if M is a relational atom, then the claim holds by $\mathfrak{H} \subseteq \mathfrak{T}$. The inductive step can be shown for each operator by directly applying their semantics on a case-by-case basis. □

Although the converse statement does not always hold, we can nonetheless prove the following result, which will underpin our definition of stable models.

Theorem 4. *Let $(\mathfrak{T}, \mathfrak{T})$ be an HT-model of a program Π and a dataset $\mathcal{D}$. Then, the set of interpretations $\mathfrak{H}$ such that $(\mathfrak{H}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$ has a least interpretation.*

Proof sketch. We construct the least interpretation using transfinite induction. Let $\mathfrak{H}_0$ be the least model of $\mathcal{D}$. Then, for each successor ordinal α, we let $\mathfrak{H}_\alpha$ be the least interpretation such that, for each ground rule of Form (1) obtained from a rule in Π by assigning constants to variables, and each t, if $\mathfrak{H}_{\alpha-1}, t \models M_i$ for each $1 \leq i \leq k$ and $\mathfrak{T}, t \not\models M_j$ for each $k+1 \leq j \leq m$, then $\mathfrak{H}_\alpha, t \models M$. Note that $\mathfrak{H}_\alpha$ is well-defined since M cannot mention $\bot$ as otherwise the body of the ground rule would hold in $\mathfrak{T}$, and so, $(\mathfrak{T}, \mathfrak{T})$ would not be an HT-model as we assumed. Next, for each limit ordinal α, we let $\mathfrak{H}_\alpha$ be $\bigcup_{\beta<\alpha} \mathfrak{H}_\beta$. Using transfinite induction and the fact that $(\mathfrak{T}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$, we can easily show that $\mathfrak{H}_\alpha \subseteq \mathfrak{T}$ for every ordinal α.

Let $\mathfrak{H} = \mathfrak{H}_{\omega_1}$, where ω_1 is the first uncountable ordinal. We have $\mathfrak{H} \subseteq \mathfrak{T}$, so $(\mathfrak{H}, \mathfrak{T})$ is an HT-interpretation. Furthermore, by construction, $(\mathfrak{H}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. Finally, notice that the construction of $\mathfrak{H}$ extends $\mathfrak{H}_0$ only with those facts which are necessary to make $(\mathfrak{H}, \mathfrak{T})$ an HT-model of Π and $\mathcal{D}$, and so $\mathfrak{H}$ is the unique least interpretation satisfying the property in the theorem. □

In what follows, given a program Π, a dataset $\mathcal{D}$, and an interpretation $\mathfrak{T}$ such that $(\mathfrak{T}, \mathfrak{T})$ is an HT-model of Π, we let $\mathfrak{H}^{\mathfrak{T}}_{\Pi,\mathcal{D}}$ denote the least interpretation whose existence is guaranteed by Theorem 4.

In equilibrium logic, a model of a program is a standard interpretation satisfying the program and such that there exists no here-and-there model of the program with the same second component but a strictly smaller first component. This ensures that equilibrium logic implements a kind of minimal model reasoning. We next generalise this notion to DatalogMTL$^\neg$ by building on our previous definition of the least interpretation $\mathfrak{H}^{\mathfrak{T}}_{\Pi,\mathcal{D}}$.

Definition 5. *An interpretation $\mathfrak{T}$ is a* stable model *of a program Π and a dataset $\mathcal{D}$ if and only if $(\mathfrak{T}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$, and $\mathfrak{H}^{\mathfrak{T}}_{\Pi,\mathcal{D}} = \mathfrak{T}$.*

We next show that our semantics for DatalogMTL$^\neg$ also generalises the semantics of (positive) DatalogMTL programs. If a DatalogMTL program Π and dataset $\mathcal{D}$ have a model, they also admit a least model (Brandt et al. 2017). This can be equivalently reformulated by stating that if the set of interpretations $\mathfrak{I}$ such that $(\mathfrak{I}, \mathfrak{I})$ is an HT-model of Π and $\mathcal{D}$ is not empty, then it has a least interpretation.

Theorem 6. *Let Π be a positive program and let $\mathcal{D}$ be a dataset. For every interpretation $\mathfrak{I}$, we have that $\mathfrak{I}$ is a stable model of Π and $\mathcal{D}$ if and only if $\mathfrak{I}$ is their least model.*

Proof sketch. The theorem is a consequence of the following statement. Assume that Π and $\mathcal{D}$ admit an HT-model; then, there exists an interpretation $\mathfrak{I}_{\min}$ such that (i) $\mathfrak{I}_{\min} = \mathfrak{H}^{\mathfrak{T}}_{\Pi,\mathcal{D}}$, where $\mathfrak{T}$ is the second component of any HT-model of Π and $\mathcal{D}$, and (ii) $\mathfrak{I}_{\min}$ is the least interpretation among interpretations $\mathfrak{T}$ such that $(\mathfrak{T}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. This statement can be shown using the

construction in the proof of Theorem 4 and the fact that Π is positive, and the theorem follows easily from it. Indeed, for the forward direction assume that $\mathfrak{I}$ is a stable model of Π and $\mathcal{D}$, so $\mathfrak{I} = \mathfrak{H}^{\mathfrak{I}}_{\Pi,\mathcal{D}}$. Then, by Condition (i) we have $\mathfrak{H}^{\mathfrak{I}}_{\Pi,\mathcal{D}} = \mathfrak{I}_{\min}$, and so, $\mathfrak{I} = \mathfrak{I}_{\min}$. Thus, by Condition (ii), $\mathfrak{I}$ is the least model of Π and $\mathcal{D}$. For the reverse direction, observe that by Condition (i), $\mathfrak{I}_{\min} = \mathfrak{H}^{\mathfrak{I}_{\min}}_{\Pi,\mathcal{D}}$, and so $\mathfrak{I}_{\min}$ is a stable model of Π and $\mathcal{D}$. □

It follows that, if a positive program and a dataset have a model, then they have a stable model. Note, however, that this is not the case for many other temporal logics with stable models semantics (Cabalar and Demri 2011; Bozzelli and Pearce 2015), and the reason why this property holds in our setting is given by Theorem 4.

Finally, we can easily extend our results further and show that our semantics also generalises that of stratifiable DatalogMTL$^\neg$ programs (Tena Cucala et al. 2021), where rules do not have cyclic dependencies via negation and can be organised in a set of strata. Each stratifiable, $\bot$-free program Π and dataset $\mathcal{D}$ have a single least model constructed by applying to $\mathcal{D}$ rules of Π stratum by stratum in a minimal way. Then, as in the case of positive programs, we can show that this least model corresponds to the single stable model of Π and $\mathcal{D}$. Hence, positive and stratifiable programs cannot have multiple stable models, as is the case in plain Datalog. Arbitrary DatalogMTL$^\neg$ programs, however, can have any number of stable models. For example, a program with rules $R \leftarrow \mathsf{not}\, P$ and $P \leftarrow \mathsf{not}\, R$ has infinitely many stable models, as, for every $t \in \mathbb{T}$, an interpretation that satisfies P at t and R at every other time point is a stable model.

In the rest of the paper we study decidability and complexity of *reasoning*, which is (in the context of this paper) the problem of checking if a DatalogMTL$^\neg$ program Π and a dataset $\mathcal{D}$ have a stable model. We focus on *data complexity*—that is, assume that Π is fixed and only $\mathcal{D}$ forms the input—which is relevant to data intensive applications.

4 Undecidability over the Rationals

In this section we focus on the rational timeline, so let us fix $\mathbb{T} = \mathbb{Q}$. In this setting, standard reasoning problems are PSPACE-complete in data complexity for positive programs (Wałęga et al. 2019), as well as for programs with stratified negation (Tena Cucala et al. 2021).

We next show that, in stark contrast, reasoning in DatalogMTL$^\neg$ (under the stable models semantics) is undecidable. Furthermore, undecidability holds even if programs are considered fixed and forward-propagating, and even if the input datasets are bounded.

Theorem 7. *Reasoning in propositional* DatalogMTL$^\neg_{\mathsf{FP}}$ *over* $\mathbb{Q}$ *and bounded datasets is undecidable in data complexity.*

Proof sketch. Let $\mathcal{M} = (\Sigma, \mathcal{Q}, \delta, q_{init}, q_{halt})$ be a deterministic Turing machine with Σ the input alphabet, $\mathcal{Q}$ the set of states, $\delta : \Sigma_\sqcup \times (\mathcal{Q} \setminus \{q_{halt}\}) \longrightarrow \Sigma_\sqcup \times \mathcal{Q} \times \{\mathsf{L}, \mathsf{R}\}$ the transition function for $\Sigma_\sqcup = \Sigma \cup \{\sqcup\}$ and blank symbol $\sqcup$, L and R the symbols indicating the head movements, and

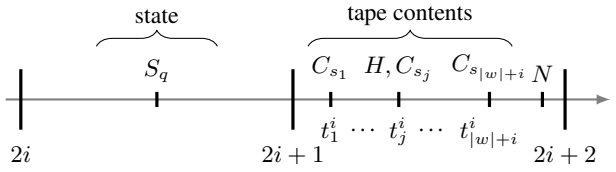

Figure 1: Encoding of the ith configuration

$q_{init}, q_{halt} \in \mathcal{Q}$ the initial and halting states. Without loss of generality, we assume that $\mathcal{M}$ never tries to move to the left of the left-most cell of the tape.

We construct a propositional DatalogMTL$^\neg_{\mathsf{FP}}$ program $\Pi_\mathcal{M}$ and then reduce (in AC^0) every input word w to a dataset $\mathcal{D}_w$ with only bounded intervals so that $\mathcal{M}$ halts on w if and only if $\Pi_\mathcal{M}$ and $\mathcal{D}_w$ do not have a stable model.

We represent, for each $i \geq 1$, the i-th configuration in the run of $\mathcal{M}$ on input w within the interval $[2i, 2i+2)$, as illustrated in Figure 1, where we assume that in the configuration the state is q, the head is over cell j, and the contents of the first $|w|+i$ cells of the tape are symbols $s_1, \ldots, s_{|w|+i}$ (in this configuration, at most $|w|+i$ cells are non-empty). The state is encoded within the first half $[2i, 2i+1]$ of the interval: a proposition S_q holds at some time point within $[2i, 2i+1]$. The contents of the tape and the head position are encoded within the second half $(2i+1, 2i+2)$ of the interval; in particular, $|w|+i$ time points $t^i_1 < \cdots < t^i_{|w|+i}$ in $(2i+1, 2i+2)$ are used so that, for each j, a proposition C_{s_j} holding at t^i_j means that cell j contains symbol s_j, and a proposition H at t^i_j means that the head is over cell j. We also use additional propositions: S holding in intervals $[2i, 2i+1]$ responsible for representing states, N holding at a single new time point in $(2i+1, 2i+2)$ beyond $t^i_{|w|+i}$, which will allow us to increase the number of time points encoding cells, $\overline{N}$ and $\overline{H}$ simulating negations of N and H, respectively, $\overline{C}$ marking points not used to encode the tape contents, and L used for encoding left-moving transitions.

The first block of rules in $\Pi_\mathcal{M}$ consists of the following rules, for each $X \in \{N, H\}$ and $s \in \Sigma_\sqcup$:

$$\overline{X} \leftarrow \mathsf{not}\, X, \qquad X \leftarrow \mathsf{not}\, \overline{X}, \qquad \bot \leftarrow X \wedge \overline{X},$$
$$\bot \leftarrow X \wedge \diamondsuit_{(0,1)} X, \quad \overline{X} \leftarrow S, \quad \overline{N} \leftarrow C_s, \quad \overline{H} \leftarrow \overline{C},$$
$$\bot \leftarrow S \wedge (\overline{C} \wedge \overline{N}) \mathcal{S}_{(0,1)} C_s.$$

The first three rules state that, at each time point, either X or $\overline{X}$ holds. The fourth rule states that X cannot hold twice in an open interval of length 1. By the fifth rule, X and S cannot hold at the same time point. The sixth rule states that $\overline{N}$ holds in all time points encoding cells. The second to last rule states that H does not hold in time points that do not encode cells. The last rule ensures that after time point $t^i_{|w|+i}$ encoding the last relevant cell in the i-th configuration, there is another time point within $(2i+1, 2i+2)$ where N holds. (Note that the last rule uses conjunction within a metric operator, which is not syntactically allowed, but can be easily simulated by replacing $\overline{C} \wedge \overline{N}$ with a fresh proposition P and adding a rule $P \leftarrow \overline{C} \wedge \overline{N}$.)

The second block consists of the following rules, propagating propositions to the interval encoding the consequent configuration, for every $s \in \Sigma_{\sqcup}$:

$$\boxplus_2 S \leftarrow S, \qquad \boxplus_2 \overline{C} \leftarrow \overline{C} \wedge \overline{N},$$
$$\boxplus_2 C_{\sqcup} \leftarrow N \wedge \diamondminus_{(0,\infty)} S_{q_{init}}, \qquad \boxplus_2 C_s \leftarrow C_s \wedge \overline{H}.$$

The first rule states that S is always propagated into the future from time point t to $t+2$. The second rule states that, if t does not encode a cell and $\overline{N}$ holds therein, then $t+2$ does not encode a cell either. By the third rule, if N holds at t then $t+2$ encodes an empty cell. The last rule states that, if t encodes a cell with contents s and the head is not on this cell, then $t+2$ also encodes a cell with contents s.

Next, we encode the left-moving transitions. Proposition L is used to mark a time point encoding a cell such that the head was on it in the previous computation step and then moved to the left. Program $\Pi_{\mathcal{M}}$ has the following rules for every $s \in \Sigma$ and $q \in \mathcal{Q}$ with transition $\delta(s,q) = (s', q', \mathsf{L})$, and every $s^* \in \Sigma$:

$$\boxplus_2 L \wedge \boxplus_1 S_{q'} \wedge \boxplus_2 C_{s'} \leftarrow H \wedge C_s \wedge \diamondminus_{(0,2)} S_q,$$
$$\bot \leftarrow L \wedge \boxminus_{(0,1)} \overline{H}, \quad \bot \leftarrow L \wedge \diamondminus_{(0,1)} (C_{s^*} \wedge \diamondminus_{(0,1)} H).$$

Here, the first rule simulates the transition: L holds as intended, the state is changed from q to q', and the contents of a cell under the head changed from s to s' (conjunction in the head is used for brevity and can be simulated by several rules). The last two rules encode the position of the head by stating that H holds at the first time point encoding a cell to the left of the time point with L.

Similarly, for each transition $\delta(s,q) = (s', q', \mathsf{R})$ moving the head to the right and any $s^* \in \Sigma$, program $\Pi_{\mathcal{M}}$ has the rules

$$\boxplus_1 S_{q'} \wedge \boxplus_2 C_{s'} \leftarrow H \wedge C_s \wedge \diamondminus_{(0,2)} S_q,$$
$$\boxplus_2 H \leftarrow C_{s^*} \wedge \overline{C} \mathcal{S}_{(0,1)} (H \wedge C_s) \wedge \diamondminus_{(0,2)} S_q,$$
$$\boxplus_2 H \leftarrow N \wedge \overline{C} \mathcal{S}_{(0,1)} (H \wedge C_s) \wedge \diamondminus_{(0,2)} S_q.$$

Here, the first rule encodes the change of the state and the contents of the cell above which the head is. The last two rules simulate the change of the position of the head.

Finally, $\Pi_{\mathcal{M}}$ contains the rule $\bot \leftarrow S_{q_{halt}}$, ensuring that reaching the halting state yields an inconsistency.

With the construction of $\Pi_{\mathcal{M}}$ complete, we next reduce an input word $w = s_1 \dots s_{|w|}$ to a dataset $\mathcal{D}_w$. Assuming for brevity that w is non-empty, we let $t_k = 1 + \frac{k}{|w|+1}$ for each $k \in \{1, \dots, |w|\}$ (it is only important here that $1 < t_1 < \dots < t_{|w|} < 2$) and then $\mathcal{D}_w$ contains the facts:

$$S@[0,1], \quad S_{q_{init}}@1, \quad H@t_1, \quad C_{s_1}@t_1, \ \dots, \ C_{s_{|w|}}@t_{|w|},$$
$$\overline{C}@(1,t_1), \ \overline{C}@(t_1,t_2), \ \dots, \ \overline{C}@(t_{|w|},2),$$
$$\overline{N}@[0,1], \qquad \overline{H}@[0,1].$$

Intuitively, $\mathcal{D}_w$ describes the initial configuration of $\mathcal{M}$ on w within $[0,2)$; the initial state is encoded in 1 and $t_1, \dots, t_{|w|}$ encode the first $|w|$ cells of $\mathcal{M}$. Moreover, $\overline{C}$ holds in all other time points in $(1,2)$, whereas $\overline{N}$ and $\overline{H}$ hold in $[0,1]$.

We can show that $\Pi_{\mathcal{M}}$ and $\mathcal{D}_w$ have a stable model if and only if $\mathcal{M}$ does not halt on w. Indeed, if $\mathfrak{T}$ is a stable model of $\Pi_{\mathcal{M}}$ and $\mathcal{D}_w$, then the location of the propositions holding in $\mathfrak{T}$ in an interval $[2i, 2i+2)$ can be treated as an encoding of configuration i of the run of $\mathcal{M}$ on w. Interpretation $\mathfrak{T}$ satisfies $\bot \leftarrow S_{q_{halt}}$, and so we can show that none of these configurations mentions a halting state—that is, the run does not halt. If $\mathcal{M}$ does not halt on w, then we can construct a stable model of $\Pi_{\mathcal{M}}$ and $\mathcal{D}_w$, where for each $i \in \mathbb{N}$ the interval $[2i, 2i+2)$ represents configuration i of the run. □

5 Decidability over the Integers

In this section we consider the integer timeline and thus fix $\mathbb{T} = \mathbb{Z}$. In particular, we show that in this case reasoning is decidable and in EXPSPACE in data complexity; furthermore, complexity drops to PSPACE if we restrict our attention to forward-propagating programs and datasets mentioning only bounded intervals—a setting well-suited for stream reasoning (Wałęga, Cuenca Grau, and Kaminski 2019; Ronca et al. 2018); note that in this setting, the additional expressive power provided by stable models comes at no computational cost since reasoning in the corresponding positive fragment is already PSPACE-complete (Wałęga et al. 2020; 2019).

In prior work on positive and stratifiable programs, upper bounds for reasoning have been established by constructing generalised Büchi automata that accept (words describing) models of a given program and dataset (Wałęga et al. 2020; Tena Cucala et al. 2021). Checking existence of a stable model is more demanding, as we additionally need to ensure model minimality as in Definition 5; this requirement is non-trivial, and we will handle it differently for the cases of arbitrary and forward-propagating programs.

In the general case (Section 5.1), we construct automata that allow us to check existence of a model and automata for checking existence of a smaller model. Then, a word accepted by the first automata but not by the latter represents a stable model. This construction is conceptually similar to that of Cabalar and Demri (2011) for a logic with linear temporal operators, and involves complementing nondeterministic automata which leads to an exponential blowup. Consequently, we obtain an EXPSPACE upper bound and thus an exponential gap in data complexity with respect to the positive programs (Wałęga et al. 2020). In the case of forward-propagating programs (Section 5.2) we propose a different construction, which exploits the fact that rules propagate information in a single temporal direction. This allows us to build automata that guarantee model minimality without the need of complementation. As a result, we can establish a tight PSPACE bound in data complexity.

5.1 General Programs

It will be convenient for our presentation to assume that programs are in a normal form analogous to that by Tena Cucala et al. (2021) for stratifiable programs. In each normalised rule the head is a relational atom or $\bot$, there is neither nesting of metric operators nor occurrences of $\diamondminus$ or $\boxminus$ in rule bodies, and the only unbounded interval allowed is $[0, \infty)$.

Proposition 8. *Each program Π can be transformed in polynomial time into a program Π' in normal form such that, for each dataset $\mathcal{D}$, program Π and dataset $\mathcal{D}$ have a stable model if and only if so do Π' and $\mathcal{D}$.*

For the remainder of this section we assume that Π is a program in normal form and $\mathcal{D}$ is a dataset. We also use the following notation. Let $\mathsf{ground}(\Pi, \mathcal{D})$ be the set of all ground rules that can be obtained by replacing variables in Π with constants from Π and $\mathcal{D}$. Then, $\mathsf{at}(\Pi, \mathcal{D})$ is the set of all relational atoms in $\mathcal{D}$, all metric atoms in rules of $\mathsf{ground}(\Pi, \mathcal{D})$, and all metric atoms of the form $\boxminus_{[0,\infty)} M$ and $\boxplus_{[0,\infty)} M$, with M a relational atom mentioned in $\mathsf{ground}(\Pi, \mathcal{D})$.

Next we define the notion of a *window*—a fragment of an HT-interpretation over a particular interval; such windows will serve as states of our automata.

Definition 9. *A* window *is a tuple (ϱ, H, T, b), where ϱ is a closed (and hence bounded) interval, H and T are sets of metric facts $M@t$ with $M \in \mathsf{at}(\Pi, \mathcal{D})$ and $t \in \varrho$ such that $H \subseteq T$, and there are interpretations $\mathfrak{H}$ and $\mathfrak{T}$ satisfying*

- *$M@t \in H$ if and only if $\mathfrak{H} \models M@t$, and*
- *$M@t \in T$ if and only if $\mathfrak{T} \models M@t$,*

for each $M \in \mathsf{at}(\Pi, \mathcal{D})$ and $t \in \varrho$, whereas $b \in \{0, 1\}$. The length *of the window is the length of ϱ. The window is* initial *if $H = T$ and $b = 0$, or $H \neq T$ and $b = 1$.*

Intuitively, a window (ϱ, H, T, b) is a fragment of an HT-interpretation $(\mathfrak{H}, \mathfrak{T})$ restricted to ϱ, where H and T describe facts holding within ϱ in, respectively, $\mathfrak{H}$ and $\mathfrak{T}$. Windows will serve as states of automata that recognise word representations of specific HT-interpretations, and in this process the flag b is used to distinguish between stable and non-stable models; in particular, it is set to 1 if $H \neq T$ in the current state of the considered run, or if such an equality holds in some previous window in this run.

By definition, a window can be extended to an HT-interpretation. Moreover, this HT-interpretation can be an HT-model of Π only if the window locally satisfies Π, which we define next.

Definition 10. *A window (ϱ, H, T, b)* locally satisfies *Π if, for each grounding of Form* (1) *of a rule in Π and each $t \in \varrho$,*

- *$M@t \in H$ whenever $M_i@t \in H$ for each $i \in \{1, \dots, k\}$ and $M_j@t \notin T$ for each $j \in \{k+1, \dots, m\}$, and*
- *$M@t \in T$ whenever $M_i@t \in T$ for each $i \in \{1, \dots, k\}$ and $M_j@t \notin T$ for each $j \in \{k+1, \dots, m\}$.*

Next, given an initial window $\mathcal{W}_0$, we define automata $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$, which will allow us to recognise HT-models of Π that extend $\mathcal{W}_0$. In particular, if $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ accept words w_1 and w_2 respectively, then we can construct an HT-model extending $\mathcal{W}_0$, whose part located to the left of $\mathcal{W}_0$ is described by w_1, and the part to the right of $\mathcal{W}_0$ by w_2.

Definition 11. *Let $\mathcal{W}_0 = (\varrho_0, H_0, T_0, b_0)$ be an initial window locally satisfying Π. Then, $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$ is the generalised nondeterministic Büchi automaton with the following components:*

1. *the set $\mathcal{Q}$ of states is the set of all windows that locally satisfy Π and have the same length as $\mathcal{W}_0$;*
2. *the alphabet Σ is the set of all $\sigma \subseteq \mathsf{at}(\Pi, \mathcal{D})$;*
3. *the transition function $\delta : \mathcal{Q} \times \Sigma \to 2^{\mathcal{Q}}$ is such that $(\varrho', H', T', b') \in \delta\big((\varrho, H, T, b), \sigma\big)$ if*
   - *$\varrho' = [\varrho^- + 1, \varrho^+ + 1]$,*
   - *$M@t \in H'$ if and only if $M@t \in H$, for every $M \in \mathsf{at}(\Pi, \mathcal{D})$ and $t \in \varrho' \cap \varrho$,*
   - *$T' = \{M@t' \in T \mid t' \in \varrho'\} \cup \{M@\varrho^+ + 1 \mid M \in \sigma\}$, and*
   - *$b' = 1$ whenever $b = 1$ or $H' \neq T'$, otherwise $b' = 0$;*
4. *the initial state q_0 is $\mathcal{W}_0$;*
5. *the accepting condition $\mathcal{F}$ is the family of sets of states containing, for each $\boxplus_{[0,\infty)} M \in \mathsf{at}(\Pi, \mathcal{D})$, the sets*

$$\{(\varrho, H, T, b) \in \mathcal{Q} \mid \text{there is } t \in \varrho \text{ such that } \boxplus_{[0,\infty)}\, M@t \in H \text{ or } M@t \notin H\},$$

$$\{(\varrho, H, T, b) \in \mathcal{Q} \mid \text{there is } t \in \varrho \text{ such that } \boxplus_{[0,\infty)}\, M@t \in T \text{ or } M@t \notin T\},$$

and, for each $M_1 \mathcal{U}_{[0,\infty)} M_2 \in \mathsf{at}(\Pi, \mathcal{D})$, the sets

$$\{(\varrho, H, T, b) \in \mathcal{Q} \mid \text{there is } t \in \varrho \text{ such that } M_1 \mathcal{U}_{[0,\infty)} M_2@t \notin H \text{ or } M_2@t \in H\},$$

$$\{(\varrho, H, T, b) \in \mathcal{Q} \mid \text{there is } t \in \varrho \text{ such that } M_1 \mathcal{U}_{[0,\infty)} M_2@t \notin T \text{ or } M_2@t \in T\}.$$

The automaton $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ is defined similarly, except that, in the definition of ϱ', we set $\varrho' = [\varrho^- - 1, \varrho^+ - 1]$, in the definition of T', we replace $\varrho^+ + 1$ with $\varrho^- - 1$, and, in the definition of $\mathcal{F}$, we use $\boxminus$ and $\mathcal{S}$ instead of $\boxplus$ and $\mathcal{U}$, respectively.

To capture the semantics of metric operators and to ensure that the dataset is satisfied, the length of windows in the automata cannot be too short. In particular, we will use windows of the length of $\varrho_{(\Pi,\mathcal{D})} = [t_{\min}, t_{\max} + t_\Pi]$, for $t_{\min}$ and $t_{\max}$ the smallest and largest numbers mentioned in $\mathcal{D}$ and t_Π the largest number in Π (if $\mathcal{D}$ mentions no numbers then we set $t_{\min} = t_{\max} = 0$, and if Π mentions no numbers then we set $t_\Pi = 1$). Then, we can show that Π and $\mathcal{D}$ have an HT-model if and only if there is an initial window $\mathcal{W}_0 = (\varrho_0, H_0, T_0, b_0)$ locally satisfying Π such that $\varrho_0 = \varrho_{(\Pi,\mathcal{D})}$, $H_0 \models \mathcal{D}$, and there are words w_1 and w_2 accepted by $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$, respectively.

To check existence of a stable model, however, we need automata that recognise HT-models $(\mathfrak{H}, \mathfrak{T})$ with $\mathfrak{H} = \mathfrak{T}$ and automata that recognise HT-models $(\mathfrak{H}, \mathfrak{T})$ with $\mathfrak{H} \subsetneq \mathfrak{T}$. The intersection of the former with the complements of the latter allows us to recognise stable models—that is, essentially, HT-models $(\mathfrak{T}, \mathfrak{T})$ for which there are no models $(\mathfrak{H}, \mathfrak{T})$ with $\mathfrak{H} \subsetneq \mathfrak{T}$. These automata are defined as follows.

Definition 12. *Let $\mathcal{W}_0 = (\varrho_0, H_0, T_0, b_0)$ be an initial window locally satisfying Π. We define non-deterministic generalised Büchi automata $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$, $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ and $\mathcal{C}^{\leftarrow}_{\mathcal{W}_0}$, $\mathcal{C}^{\rightarrow}_{\mathcal{W}_0}$ as follows:*

- *assuming $H_0 = T_0$ and $b_0 = 0$, $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ are defined as $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$, respectively, except that for a window (ϱ, H, T, b) to be a state we additionally require $H = T$ (and hence $b = 0$),*
- *$\mathcal{C}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{C}^{\rightarrow}_{\mathcal{W}_0}$ are defined as $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$, respectively, except that we add to the accepting condition $\mathcal{F}$ the set $\{(\varrho, H, T, b) \in \mathcal{Q} \mid b = 1\}$.*

Intuitively, $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ recognise interpretations $\mathfrak{T}$ such that $(\mathfrak{T}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. Furthermore, interpretations $\mathfrak{T}$ accepted by $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{C}^{\rightarrow}_{\mathcal{W}_0}$, or by $\mathcal{C}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ are such that $(\mathfrak{H}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$, for some $\mathfrak{H} \subsetneq \mathfrak{T}$. Hence, as we show in the next lemma, we can use these automata to recognise stable models.

Lemma 13. *Program Π and dataset $\mathcal{D}$ have a stable model if and only if there is an initial window $\mathcal{W}_0 = (\varrho_0, T_0, T_0, 0)$ locally satisfying Π with $\varrho_0 = \varrho_{(\Pi,\mathcal{D})}$, $T_0 \models \mathcal{D}$, and words w_1 and w_2 over $2^{\mathtt{at}(\Pi,\mathcal{D})}$ such that*

1. *w_1 and w_2 are accepted by $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$, respectively,*
2. *there is no initial window $\mathcal{W}'_0 = (\varrho_0, H_0, T_0, b_0)$ locally satisfying Π such that $H_0 \models \mathcal{D}$, and w_1 and w_2 are accepted either by $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$, respectively, or by $\mathcal{A}^{\leftarrow}_{\mathcal{W}'_0}$ and $\mathcal{C}^{\rightarrow}_{\mathcal{W}'_0}$, respectively.*

Proof sketch. Assume first that there exist $\mathcal{W}_0$, w_1, and w_2 satisfying Conditions 1 and 2. By Condition 1, there is an accepting run $\mathcal{W}_0, \mathcal{W}_{-1}, \dots$ of $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ on w_1, and an accepting run $\mathcal{W}_0, \mathcal{W}_1, \dots$ of $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ on w_2, where $\mathcal{W}_i = (\varrho_i, T_i, T_i, 0)$. We argue that the least model $\mathfrak{T}$ of relational facts in $\bigcup_{i\in\mathbb{Z}} T_i$ is a stable model of Π and $\mathcal{D}$. By construction, $(\mathfrak{T}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. Suppose for contradiction that $\mathfrak{T}$ is not stable, so there exists $\mathfrak{H} \subsetneq \mathfrak{T}$ such that $(\mathfrak{H}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. To this end, we *decompose* $(\mathfrak{H}, \mathfrak{T})$ into windows of the length of $\varrho_{(\Pi,\mathcal{D})}$—that is, windows $\mathcal{W}'_i = (\varrho'_i, H'_i, T'_i, b'_i)$ locally satisfying Π where, for each $i \in \mathbb{Z}$, (i) $\varrho'_i = [\varrho^-_{(\Pi,\mathcal{D})} + i, \varrho^+_{(\Pi,\mathcal{D})} + i]$, (ii) H'_i is a set of facts $M@t$ such that $M \in \mathtt{at}(\Pi, \mathcal{D})$, $t \in \varrho'_i$, and $\mathfrak{H} \models M@t$, (iii) T'_i is a set of facts $M@t$ such that $M \in \mathtt{at}(\Pi, \mathcal{D})$, $t \in \varrho'_i$, and $\mathfrak{T} \models M@t$, (iv) $b'_i = 0$ if $H'_i = T'_i$, and otherwise $b'_i = 1$. So $\mathcal{W}'_0, \mathcal{W}'_{-1}, \dots$ is an accepting run of $\mathcal{A}^{\leftarrow}_{\mathcal{W}'_0}$ on w_1, and $\mathcal{W}'_0, \mathcal{W}'_1, \dots$ is an accepting run of $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$ on w_2. Moreover, since $\mathfrak{H} \subsetneq \mathfrak{T}$, there is $i \in \mathbb{Z}$ such that $H'_i \neq T'_i$, and so $b'_i = 1$. If $i \leq 0$, then $b'_j = 1$ for all $j \leq i$, and so $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ accepts w_1; analogously if $i \geq 0$, then $\mathcal{C}^{\rightarrow}_{\mathcal{W}'_0}$ accepts w_2. Thus, Condition 2 does not hold, leading to a contradiction.

For the other implication, assume that $\mathfrak{T}$ is a stable model of Π and $\mathcal{D}$. We show how to construct the required $\mathcal{W}_0$, w_1, and w_2. For this, we decompose $(\mathfrak{T}, \mathfrak{T})$ into windows $\mathcal{W}_i = (\varrho_i, T_i, T_i, 0)$ as above. By construction, $\varrho_0 = \varrho_{(\Pi,\mathcal{D})}$ and $T_0 \models \mathcal{D}$, so $\mathcal{W}_0$ satisfies the initial requirements from the lemma. Next, we construct words $w_1 = \sigma_{-1}\sigma_{-2}\cdots$ and $w_2 = \sigma_1\sigma_2\cdots$, where σ_k is $T_k \setminus T_{k+1}$ if $k < 0$, or $T_k \setminus T_{k-1}$ if $k > 0$. It remains to show that $\mathcal{W}_0$, w_1, and w_2 satisfy Conditions 1 and 2. For Condition 1, it suffices to observe that, by construction, $\mathcal{W}_0, \mathcal{W}_{-1}, \dots$ is an accepting run of $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ on w_1, and $\mathcal{W}_0, \mathcal{W}_1, \dots$ is an accepting run of $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ on w_2. To show Condition 2, suppose for contradiction that there exists $\mathcal{W}'_0$ from the lemma such that w_1 and w_2 are accepted by $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$, respectively (the other case is symmetric). Hence, $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ has an accepting run $\mathcal{W}'_0, \mathcal{W}'_{-1}, \dots$ on w_1 and $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$ has an accepting run $\mathcal{W}'_0, \mathcal{W}'_1, \dots$ on w_2, where $\mathcal{W}'_i = (\varrho_i, H_i, T_i, b_i)$. Then, $\mathfrak{T}$ is the least model of all relational facts in $\bigcup_{i\in\mathbb{Z}} T_i$, and we let $\mathfrak{H}$ be the least model of relational facts in $\bigcup_{i\in\mathbb{Z}} H_i$. We can show that $(\mathfrak{H}, \mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. However, the accepting condition of $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ guarantees that $b_i = 1$, for some $i \leq 0$, and so $H_i \subsetneq T_i$. Thus $\mathfrak{H} \subsetneq \mathfrak{T}$, so $\mathfrak{T}$ is not stable, rising a contradiction. □

Lemma 13 provides a reduction of existence of a stable model to checking specific properties of our automata. As we show next, the latter is feasible in ExpSpace.

Theorem 14. *Reasoning in* DatalogMTL$^{\neg}$ *over $\mathbb{Z}$ is in* ExpSpace *in data complexity.*

Proof sketch. It suffices to show that checking existence of $\mathcal{W}_0$, w_1, and w_2 from Lemma 13 is feasible in ExpSpace in the size of (the representation of) $\mathcal{D}$. First, we observe that the length of $\varrho_{(\Pi,\mathcal{D})}$ is exponential, and so is the size of windows over $\varrho_0 = \varrho_{(\Pi,\mathcal{D})}$. Thus, it is feasible in ExpSpace to guess $\mathcal{W}_0 = (\varrho_0, T_0, T_0, 0)$ and to verify that it is an initial window locally satisfying Π and that $T_0 \models \mathcal{D}$. Next, we need to check existence of w_1 and w_2 as in Lemma 13. The main obstacle is the size of states in the automata: $\mathcal{W}_0$ is exponentially big, and other states of the automata from Lemma 13 can be arbitrarily large since time points in windows are arbitrary integers. We will show, in two steps, how to restrict attention to automata with polynomially big states.

In the first step, we show that it suffices to consider windows of length $t_\Pi + 1$, where we recall that t_Π does not depend on $\mathcal{D}$. Indeed, we can show that for every $X \in \{\mathcal{A}, \mathcal{B}\}$ and every $\mathcal{W} = (\varrho, H, T, b)$ such that $X^{\leftarrow}_{\mathcal{W}}$ is well-defined, the automata $X^{\leftarrow}_{\mathcal{W}}$ and $X^{\leftarrow}_{\mathcal{W}_L}$ are equivalent, where $\mathcal{W}_L = (\varrho_L, H_L, T_L, b_L)$ is the left-most window of length $t_\Pi + 1$ contained in $\mathcal{W}$ (i.e., such that $\varrho_L = [\varrho^-, \varrho^- + t_\Pi]$, $H_L = \{M@t \in H \mid t \in \varrho_L\}$, $T_L = \{M@t \in T \mid t \in \varrho_L\}$, and $b_L = 1$ if and only if $H_L \neq T_L$). We can analogously define the right-most window $\mathcal{W}_R$ of length $t_\Pi + 1$ contained in $\mathcal{W}$, and show that $X^{\rightarrow}_{\mathcal{W}}$ and $X^{\rightarrow}_{\mathcal{W}_R}$ are equivalent. Moreover, if $H = T$, these equivalences hold also for $X = \mathcal{C}$, but if $H \neq T$, then $\mathcal{C}^{\leftarrow}_{\mathcal{W}}$ and $\mathcal{C}^{\rightarrow}_{\mathcal{W}}$ are equivalent to $\mathcal{A}^{\leftarrow}_{\mathcal{W}_L}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_R}$, respectively.

In the second step we observe that, rather than considering automata with states of unbounded size (each of length $t_\Pi + 1$), we can construct equivalent automata with states of polynomial size. In particular, for each $X^{\leftarrow}_{\mathcal{W}_L}$ with $X \in \{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$, we define an automaton $\widetilde{X}^{\leftarrow}_{\mathcal{W}_L}$: its states are the quotient set of the equivalence relation $\sim$ between states of $X^{\leftarrow}_{\mathcal{W}_l}$ such that $\mathcal{W} \sim \mathcal{W}'$ if by increasing all time points mentioned in $\mathcal{W}$ by some integer we obtain $\mathcal{W}'$, and the other components of $\widetilde{X}^{\leftarrow}_{\mathcal{W}_L}$ are defined accordingly. Thus $X^{\leftarrow}_{\mathcal{W}_L}$ and $\widetilde{X}^{\leftarrow}_{\mathcal{W}_L}$ are equivalent, but all states of $\widetilde{X}^{\leftarrow}_{\mathcal{W}_L}$ are of polynomial size. Similarly we construct $\widetilde{X}^{\rightarrow}_{\mathcal{W}_R}$ from $X^{\rightarrow}_{\mathcal{W}_R}$.

Now, to check the conditions from Lemma 13 we characterise each initial window $\mathcal{W}'_0 = (\varrho_0, H_0, T_0, b_0)$ occurring there by the triple consisting of two initial windows $(\mathcal{W}'_0)_L$, $(\mathcal{W}'_0)_R$, and flag $b \in \{0, 1\}$ such that $b = 1$ if and only if $H_0 \neq T_0$. Since $(\mathcal{W}'_0)_L$ and $(\mathcal{W}'_0)_R$ are polynomially big, there are exponentially many such triples. Hence, we can guess in ExpSpace the set $\mathcal{T}$ of all these triples and verify that each of them is a valid characterisation. Furthermore, we treat a pair of words $w_1 = \sigma_{-1}\sigma_{-2}\cdots$ and

$w_2 = \sigma_1\sigma_2\cdots$ as a single word $(\sigma_{-1},\sigma_1)(\sigma_{-2},\sigma_2)\cdots$ and combine pairs of corresponding automata so that they accept combined words; in particular, for all $X, Y \in \{\mathcal{A},\mathcal{B},\mathcal{C}\}$, combining $\widetilde{X}^{\leftarrow}_{\mathcal{W}_L}$ and $\widetilde{Y}^{\rightarrow}_{\mathcal{W}_R}$ gives rise to a (polynomially bigger) automaton $\widetilde{X}^{\leftarrow}_{\mathcal{W}_L}\widetilde{Y}^{\rightarrow}_{\mathcal{W}_R}$ simulating runs of $\widetilde{X}^{\leftarrow}_{\mathcal{W}_L}$ and $\widetilde{Y}^{\rightarrow}_{\mathcal{W}_R}$ in parallel. Thus, conditions from Lemma 13 reduce to checking non-emptiness of the language of an automaton obtained by intersecting $\widetilde{\mathcal{B}}^{\leftarrow}_{(\mathcal{W}_0)_L}\widetilde{\mathcal{B}}^{\rightarrow}_{(\mathcal{W}_0)_R}$ with the complements of all the automata corresponding to triples from $\mathcal{T}$, where a triple $(\mathcal{W}'_0)_L$, $(\mathcal{W}'_0)_R$, b corresponds to $\widetilde{\mathcal{A}}^{\leftarrow}_{(\mathcal{W}_0)_L}\widetilde{\mathcal{A}}^{\rightarrow}_{(\mathcal{W}_0)_R}$ if $b = 1$, and to two automata $\widetilde{\mathcal{C}}^{\leftarrow}_{(\mathcal{W}_0)_L}\widetilde{\mathcal{A}}^{\rightarrow}_{(\mathcal{W}_0)_R}$ and $\widetilde{\mathcal{A}}^{\leftarrow}_{(\mathcal{W}_0)_L}\widetilde{\mathcal{C}}^{\rightarrow}_{(\mathcal{W}_0)_R}$ otherwise. Since the automata are nondeterministic, their complements have states of exponential size and an intersection of exponentially many such complemented automata gives rise to an automaton with exponentially big states. Hence, checking non-emptiness of the final automaton is feasible in EXPSPACE using a standard on-the-fly approach (Baier and Katoen 2008). □

5.2 Forward-Propagating Programs

In this section, we consider reasoning with forward-propagating—that is DatalogMTL$^{\neg}_{\mathsf{FP}}$—programs (see Definition 1) and bounded datasets. This setting was already considered for positive programs and shown to be well-suited for applications such as stream reasoning (Wałęga, Cuenca Grau, and Kaminski 2019; Ronca et al. 2018).

First, we observe that the normalisation of a DatalogMTL$^{\neg}_{\mathsf{FP}}$ program, as defined in Section 5.1, results also in a DatalogMTL$^{\neg}_{\mathsf{FP}}$ program; thus, for the remainder of this section, let Π be an arbitrary fixed DatalogMTL$^{\neg}_{\mathsf{FP}}$ program in normal form, let $\mathcal{D}$ be a bounded dataset, and let t_Π, $t_{\min}$, and $t_{\max}$ be the integers defined for Π and $\mathcal{D}$ as in Section 5.1. We will show that this setting allows us to simplify the procedure used in Section 5.1 to check existence of a stable model of Π and $\mathcal{D}$. First, we will guess $\mathcal{W}_0$ over an interval located to the left of all intervals in $\mathcal{D}$. Then, on the one hand, we will use the fact that Π is forward-propagating to show that checking existence of a word w_1 accepted by the relevant automata, as stated in Lemma 13, can be done independently of $\mathcal{D}$. On the other hand, to check existence of the second word w_2 from Lemma 13 we define a new family of automata $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$, which can be used without resorting to complementation and thus avoiding the exponential blowup. As a result, we will show that the procedure is feasible in PSPACE.

The family of automata $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ refines automata $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ by ensuring that every state locally satisfies $\mathcal{D}$. Furthermore, we impose an additional restriction on states of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ to guarantee their minimality.

Definition 15. *A window (ϱ, H, T, b)* locally satisfies *$\mathcal{D}$ if $M@t \in H$ for each $M \in \mathsf{at}(\Pi,\mathcal{D})$ and $t \in \varrho$ such that $\mathcal{D} \models M@t$. Let $\mathcal{W}_0 = (\varrho_0, T_0, T_0, 0)$ be an initial window locally satisfying Π and $\mathcal{D}$. The generalised Büchi automaton $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ is the same as $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ in Definition 12 except that all its states locally satisfy $\mathcal{D}$, and for each state of the form $(\varrho, T, T, 0)$ there exists no window $(\varrho, H, T, 1)$ with $H \subsetneq T$ locally satisfying Π and $\mathcal{D}$.*

Note that $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ is essentially deterministic since $\delta(\mathcal{W},\sigma)$ contains at most one window for every state $\mathcal{W}$ and $\sigma \in \Sigma$.

The following lemma provides a result analogous to Lemma 13 for the setting considered in this section.

Lemma 16. *Program Π and dataset $\mathcal{D}$ admit a stable model if and only if there exists an initial window $\mathcal{W}_0 = (\varrho_0, T_0, T_0, 0)$ locally satisfying Π and $\mathcal{D}$, and mentioning only constants and predicates from Π, such that $\varrho_0 = [t_{\min} - (t_\Pi + 1), t_{\min} - 1]$ and there are words w_1 and w_2 over $2^{\mathsf{at}(\Pi,\mathcal{D})}$ satisfying the following:*

1. *w_1 and w_2 are accepted by $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$, respectively,*
2. *there is no initial window $\mathcal{W}'_0 = (\varrho_0, H_0, T_0, b_0)$ locally satisfying Π and $\mathcal{D}$ such that w_1 is accepted by $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$,*
3. *w_1 mentions only constants and predicates from Π.*

Proof sketch. Assume first that $\mathcal{W}_0$, w_1, and w_2 satisfy Conditions 1–3. Hence, $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ has an accepting run $\mathcal{W}_0, \mathcal{W}_{-1}, \dots$ on w_2 and $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ has an accepting run $\mathcal{W}_0, \mathcal{W}_1, \dots$ on w_2, where we let $\mathcal{W}_i = (\varrho_i, T_i, T_i, 0)$. We will show that the least model $\mathfrak{T}$ of relational facts in $\bigcup_{i\in\mathbb{Z}} T_i$ is a stable model of Π and $\mathcal{D}$. By construction, $(\mathfrak{T},\mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. Now, suppose towards a contradiction that $\mathfrak{T}$ is not a stable model, so there is an interpretation $\mathfrak{H} \subsetneq \mathfrak{T}$ such that $(\mathfrak{H},\mathfrak{T})$ is an HT-model of Π and $\mathcal{D}$. Then, as in Lemma 13, we decompose $(\mathfrak{H},\mathfrak{T})$ into windows $\mathcal{W}'_i = (\varrho_i, H_i, T_i, b_i)$. Since $\mathfrak{H} \subsetneq \mathfrak{T}$, there is $i \in \mathbb{Z}$ such that $H_i \neq T_i$. If $i \leq 0$, then $b_j = 1$ for all $j \leq i$, and so $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ accepts w_1 contradicting Condition 2. On the other hand, if $i > 0$, the existence of $\mathcal{W}'_i$ implies, by Definition 15, that $\mathcal{W}_i$ is not a state of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$, contradicting the assumption.

For the other implication, let $\mathfrak{T}$ be a stable model of Π and $\mathcal{D}$. We decompose $(\mathfrak{T},\mathfrak{T})$ into windows $\mathcal{W}_i = (\varrho_i, T_i, T_i, 0)$, where $\varrho_0 = [t_{\min} - (t_\Pi + 1), t_{\min} - 1]$, and construct words $w_1 = \sigma_{-1}\sigma_{-2}\cdots$ and $w_2 = \sigma_1\sigma_2\cdots$, where σ_k is $T_k \setminus T_{k+1}$ if $k < 0$, or $T_k \setminus T_{k-1}$ if $k > 0$. Using the fact that $\mathfrak{T}$ is a stable model of Π and $\mathcal{D}$ we can easily show that $\mathcal{W}_i$ locally satisfies $\mathcal{D}$, for each $i \in \mathbb{Z}$. Next, we will show that $\mathcal{W}_0$, w_1, and w_2 satisfy Conditions 1–3.

To show Condition 1 we observe that, by construction, $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ accepts w_1 and $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ accepts w_2. Now, suppose towards a contradiction that $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ does not accept w_2. Since $\mathcal{B}^{\rightarrow}_{\mathcal{W}_0}$ accepts w_2 and $\mathcal{W}_0, \mathcal{W}_1, \dots$ all locally satisfy $\mathcal{D}$, but $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ does not accept w_2, there is $i > 0$ such that $\mathcal{W}_0, \dots, \mathcal{W}_{i-1}$ is a run of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ on $\sigma_1\cdots\sigma_{i-1}$, but $\mathcal{W}_i$ is not a state of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$. Thus, by Definition 15, there is a window $\mathcal{W}'_i = (\varrho_i, H_i, T_i, 1)$ locally satisfying $\mathcal{D}$ and Π with $H_i \subsetneq T_i$. We will show that $\mathcal{W}_0, \dots, \mathcal{W}_{i-1}, \mathcal{W}'_i$ is a run of $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ on $\sigma_1\cdots\sigma_i$. Note that the transition function of $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ extends the one of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$, so it suffices to show that $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ has a transition from $\mathcal{W}_{i-1}$ to $\mathcal{W}'_i$ on σ_i, that is all four conditions from Item 3 in Definition 11 hold. All these conditions, except the second one, hold by the form of $\mathcal{W}_{i-1}$ and $\mathcal{W}'_i$, so it remains to show the second condition, stating that $M@t \in H_i$ if and only if $M@t \in T_{i-1}$, for each $M \in \mathsf{at}(\Pi,\mathcal{D})$ and $t \in \varrho_i \cap \varrho_{i-1}$. If this was not the case, we

could construct $H_{i-1} \subsetneq T_{i-1}$ such that $(\varrho_i, H_{i-1}, T_{i-1}, 1)$ is a window locally satisfying Π and $\mathcal{D}$ which, in turn, means that $\mathcal{W}_{i-1}$ is not a state of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ and raises a contradiction. Hence, $\mathcal{W}_0, \dots, \mathcal{W}_{i-1}, \mathcal{W}'_i$ is indeed an accepting run of $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ on $\sigma_1 \cdots \sigma_i$. Since Π is DatalogMTL$^{\neg}_{\mathsf{FP}}$, we can complete this run to an accepting run of $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ on w_2 where all windows locally satisfy $\mathcal{D}$. By combining this accepting run with the accepting run of $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ on w_1, as in the proof of Lemma 13, we can construct an HT-model $(\mathfrak{H}, \mathfrak{T})$ of Π and $\mathcal{D}$, with $\mathfrak{H} \subsetneq \mathfrak{T}$. This, however, means that $\mathfrak{T}$ is not a stable model of Π and $\mathcal{D}$, which raises a contradiction.

Now, suppose towards a contradiction that Condition 2 does not hold, so there is $\mathcal{W}'_0 = (\varrho_0, H_0, T_0, b_0)$ such that $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ accepts w_1. Then, using the fact that Π is forward-propagating, we can construct an accepting run $\mathcal{W}'_0, \mathcal{W}'_1, \dots$ of $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$ on w_2, where $\mathcal{W}'_i = (\varrho_i, H_i, T_i, b_i)$. In particular, given $\mathcal{W}_i$ we construct $\mathcal{W}_{i+1}$, where we let H_{i+1} be the minimal set such that $(\varrho_{i+1}, H_{i+1}, T_{i+1}, b_{i+1})$ is a window satisfying Π and $\mathcal{D}$ to which $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$ has a transition from $\mathcal{W}_i$. We can use the fact that Π is DatalogMTL$^{\neg}_{\mathsf{FP}}$ to show that such a windows exists for every $i \in \mathbb{N}$, and hence $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$ has an accepting run. However, since $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}'_0}$ have accepting runs, as in Lemma 13, we can use these runs to construct an HT-model $(\mathfrak{H}, \mathfrak{T})$ of Π and $\mathcal{D}$ with $\mathfrak{H} \subsetneq \mathfrak{T}$. Thus, $\mathfrak{T}$ is not stable, raising a contradiction.

Finally, to prove Condition 3, we show that if $\mathfrak{T} \models M@t$ for some relational atom M and $t < t_{\min}$, then all constants and predicates in M occur in Π. Indeed, since $\mathfrak{T}$ is a stable model of Π and $\mathcal{D}$, we have $\mathfrak{T} = \mathfrak{H}_{\omega_1}$, for the sequence of interpretations $\mathfrak{H}_\alpha$ from Theorem 4. Then, we can show by a transfinite induction on ordinals α, that if $\mathfrak{H}_\alpha \models M@t$ for some relational atom M and $t < t_{\min}$, then all constants and predicates in M occur in Π. Hence, all T_i, for $i \leq 0$, mention only predicates and constants from Π, and hence so does w_1, as stated in Condition 3. □

Next, we use Lemma 16 to establish a tight PSPACE bound for reasoning in DatalogMTL$^{\neg}_{\mathsf{FP}}$.

Theorem 17. *Reasoning in* DatalogMTL$^{\neg}_{\mathsf{FP}}$ *over* $\mathbb{Z}$ *and bounded datasets is* PSPACE*-complete in data complexity.*

Proof sketch. For the lower bound we observe that Wałęga et al. (2020) showed PSPACE-hardness in data complexity of checking existence of models for a class of programs which is strictly smaller than the class of positive DatalogMTL$^{\neg}_{\mathsf{FP}}$ programs. Their reduction can be modified in a straightforward way so that the involved dataset is bounded. Then, Theorem 6 directly implies that the same lower bound holds for all (i.e., not necessarily positive) DatalogMTL$^{\neg}_{\mathsf{FP}}$ programs, as required.

For the upper bound, by Lemma 16, it suffices to show that checking existence of a window $\mathcal{W}_0$ for which there are words w_1 and w_2 satisfying Conditions 1–3 stated there is feasible in PSPACE. First, we observe that $\mathcal{W}_0$ is over $\varrho_0 = [t_{\min} - (t_\Pi + 1), t_{\min} - 1]$, so its length does not depend on $\mathcal{D}$. Hence, $\mathcal{W}_0$ is polynomially big (in the size of representation of $\mathcal{D}$), and so it can be guessed in PSPACE. Next, we show how to verify existence of appropriate words w_1 and w_2. To verify existence of a word w_1 accepted by $\mathcal{B}^{\leftarrow}_{\mathcal{W}_0}$ (first part of Condition 1) which is not accepted by any $\mathcal{C}^{\leftarrow}_{\mathcal{W}'_0}$ (Condition 2) we can use the approach from the proof of Theorem 14. We observe that $\mathcal{W}_0$ mentions only constants and predicates from Π, and so, the same holds for windows $\mathcal{W}'_0$ from Condition 2. Moreover, by Condition 3, w_1 also mentions only constants and predicates from Π, and so the above check can be performed independently of $\mathcal{D}$.

It remains to be shown that checking existence of a word w_2 accepted by $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ (second part of Condition 1) is feasible in PSPACE. To this end, we check existence of an accepting run $\mathcal{W}_0, \mathcal{W}_1, \dots$ of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ in two steps. First, we guess windows $\mathcal{W}_1, \dots, \mathcal{W}_j$ one by one, for $j = t_{\max} - t_{\min} + 2(t_\Pi + 1)$, and second, we check if $\mathcal{F}^{\rightarrow}_{\mathcal{W}_j}$ has an accepting run. We observe that each of the windows $\mathcal{W}_1, \dots, \mathcal{W}_j$ is of polynomial size, and so guessing them one by one, as well as checking that $\mathcal{F}^{\rightarrow}_{\mathcal{W}_0}$ has transitions between consecutive windows, is feasible in PSPACE. To check non-emptiness of the language of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_j}$, we construct for it an automaton $\widetilde{\mathcal{F}}^{\rightarrow}_{\mathcal{W}_j}$ in a similar way as we constructed $\widetilde{X}^{\rightarrow}_{\mathcal{W}_j}$ for $X^{\rightarrow}_{\mathcal{W}_j}$ in the proof of Theorem 14. The difference, however, is that the set of states of $\widetilde{\mathcal{F}}^{\rightarrow}_{\mathcal{W}_j}$ is the quotient set of $\sim$ between only those states of $\mathcal{F}^{\rightarrow}_{\mathcal{W}_j}$ which are over intervals located entirely to the right of $t_{\max} + t_\Pi$. Since such windows are located far enough from all facts in $\mathcal{D}$, all of them mention the same atoms to locally satisfy $\mathcal{D}$ independently of their positions. Then, we can show that $\mathcal{F}^{\rightarrow}_{\mathcal{W}_j}$ and $\widetilde{\mathcal{F}}^{\rightarrow}_{\mathcal{W}_j}$ are equivalent. Hence, it remains to check if the language of the latter automaton is non-empty, which is feasible in PSPACE using a standard on-the-fly approach (in particular, to check if a guessed window is a state, we use the above observation about local satisfiability of $\mathcal{D}$). □

Note that the assumption that $\mathcal{D}$ is bounded has been used to ensure existence of a time point $t_{\min}$ such that no fact of $\mathcal{D}$ holds to the left of it. Thus, our results can be extended to show that reasoning is still PSPACE in data complexity for datasets where intervals are only bounded on the left.

6 Conclusion and Future Work

We have extended DatalogMTL with negation-as-failure under stable model semantics and shown that reasoning in this language is undecidable over the rationals but EXPSPACE in data complexity over the integer timeline. We have also identified an interesting fragment of the language for which reasoning remains undecidable over the rationals but becomes PSPACE-complete in data complexity over the integers (and thus no harder than in the negation-free case).

We see many avenues for future work. The more immediate challenge is to provide tight data complexity bounds for reasoning in the full language over the integers, where we only know that it is in EXPSPACE and PSPACE-hard for data complexity. We also plan to consider combined complexity and identify fragments of the language where reasoning becomes decidable over the rationals.

Acknowledgments

This work was supported by the EPSRC projects OASIS (EP/S032347/1), AnaLOG (EP/P025943/1), and UK FIRES (EP/S019111/1), the AIDA project via the Alan Turing Institute (EP/N510129/1), the Norwegian Research Council via the SIRIUS Centre for Research Based Innovation (Grant Nr. 237898), Samsung Research UK, and Siemens AG.

References

Abadi, M., and Manna, Z. 1989. Temporal logic programming. *Journal of symbolic computation* 8(3):277–295.

Aguado, F.; Cabalar, P.; Diéguez, M.; Pérez, G.; and Vidal, C. 2013. Temporal equilibrium logic: a survey. *J. of Applied Non-Classical Logics* 23(1-2):2–24.

Baier, C., and Katoen, J. 2008. *Principles of model checking*. MIT press.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2018. LARS: A logic-based framework for analytic reasoning over streams. *Artif. Intell.* 261:16–70.

Bozzelli, L., and Pearce, D. 2015. On the complexity of temporal equilibrium logic. In *LICS*, 645–656.

Brandt, S.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2017. Ontology-based data access with a Horn fragment of metric temporal logic. In *AAAI*, 1070–1076.

Brandt, S.; Kalaycı, E. G.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2018. Querying log data with metric temporal logic. *J. Artif. Intell. Res.* 62:829–877.

Brooks, D. R.; Erdem, E.; Erdogan, S. T.; Minett, J. W.; and Ringe, D. 2007. Inferring phylogenetic trees using answer set programming. *J. Autom. Reason.* 39(4):471–511.

Brzoska, C. 1998. Programming in metric temporal logic. *Theor. Comput. Sci.* 202(1-2):55–125.

Cabalar, P., and Demri, S. 2011. Automata-based computation of temporal equilibrium models. In *LOPSTR*, 57–72.

Cabalar, P., and Vega, G. P. 2007. Temporal equilibrium logic: A first approach. In *EUROCAST*, 241–248.

Cabalar, P.; Kaminski, R.; Schaub, T.; and Schuhmann, A. 2018. Temporal answer set programming on finite traces. *Theory Pract. Log. Program.* 18(3-4):406–420.

Cabalar, P.; Dieguez, M.; Schaub, T.; and Schuhmann, A. 2020. Towards metric temporal answer set programming. *Theory and Practice of Logic Programming* 20(5):783–798.

Chomicki, J., and Imieliński, T. 1988. Temporal deductive databases and infinite objects. In *PODS*, 61–73.

Chomicki, J., and Imieliński, T. 1989. Relational specifications of infinite query answers. *ACM SIGMOD Record* 18(2):174–183.

Faber, W.; Leone, N.; and Pfeifer, G. 2004. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *JELIA*, 200–212.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP*, 1070–1080.

Heyting, A. 1930. Die formalen regeln der intuitionistischen logik. In *Sitzungsberichte der Preussischen Akademie der Wissenschaften, physikalisch-mathematische klass*, 42–56.

Koymans, R. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Syst.* 2(4):255–299.

Nogueira, M.; Balduccini, M.; Gelfond, M.; Watson, R.; and Barry, M. 2001. An A prolog decision support system for the space shuttle. In *ASPOCP*.

Pearce, D. 1996. A new logical characterisation of stable models and answer sets. In *NMELP*, 57–70.

Ronca, A.; Kaminski, M.; Cuenca Grau, B.; Motik, B.; and Horrocks, I. 2018. Stream reasoning in temporal Datalog. In *AAAI*, 1941–1948. Menlo Park, California: AAAI Press.

Tena Cucala, D.; Wałęga, P. A.; Cuenca Grau, B.; and Kostylev, E. V. 2021. Stratified negation in Datalog with metric temporal operators. In *AAAI*.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2019. DatalogMTL: Computational complexity and expressive power. In *IJCAI*, 1886–1892.

Wałęga, P.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2020. DatalogMTL over integer timeline. In *KR*, 526–541.

Wałęga, P.; Cuenca Grau, B.; and Kaminski, M. 2019. Reasoning over streaming data in metric temporal Datalog. In *AAAI*, 1941–1948.

Zaniolo, C. 2012. Logical foundations of continuous query languages for data streams. In *Datalog 2.0*, 177–189.