Compiling Metric Temporal Answer Set Programming*

A. Becker[1], P. Cabalar[2], M. Diéguez[3], J. Romero[1], S. Hahn[1], and T. Schaub[1]

[1]University of Potsdam, Germany
[2]University of Corunna, Spain
[3]LERIA, University of Angers, France

Abstract

We develop a computational approach to Metric Answer Set Programming (ASP) to allow for expressing quantitative temporal constrains, like durations and deadlines. A central challenge is to maintain scalability when dealing with fine-grained timing constraints, which can significantly exacerbate ASP's grounding bottleneck. To address this issue, we leverage extensions of ASP with difference constraints, a simplified form of linear constraints, to handle time-related aspects externally. Our approach effectively decouples metric ASP from the granularity of time, resulting in a solution that is unaffected by time precision.

1 Introduction

Metric temporal logics [12] allow for expressing quantitative temporal constrains, like durations and deadlines. As an example, consider the dentist scenario [19]: *"Ram is at his office and has a dentist appointment in one hour. For the appointment, he needs his insurance card which is at home and cash to pay the doctor. He can get cash from the nearby Atm. Table 1 shows the time in minutes needed to travel between locations: Dentist, Home, Office and Atm. For example, the time needed to travel between Ram's office to the Atm is 20 minutes. The available actions are: moving from one location to another and picking items such as cash or insurance. The goal is to find a plan which takes Ram to the doctor on time."* This example combines planning and scheduling, and nicely illustrates the necessity to combine qualitative and quantitative temporal constraints.

	D	*H*	*O*	*A*
D	0	20	30	40
H	20	0	15	15
O	30	15	0	20
A	40	15	20	0

Table 1: Durations between locations

Extensions to the Logic of Here-and-There and Equilibrium Logic [20] were developed in [9, 6] to semantically ground the incorporation of metric

*Under consideration for publication in International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2024) constraints into Answer Set Programming (ASP; [18]). Building upon these semantic foundations, we develop a computational approach to metric ASP. A central challenge in this is to maintain scalability when dealing with fine-grained timing constraints, which can significantly exacerbate ASP's grounding bottleneck. To address this issue, we leverage extensions of ASP with difference constraints [15], a simplified form of linear constraints, to handle time-related aspects externally. This approach effectively decouples metric ASP from the granularity of time, resulting in a solution that is unaffected by time precision. In detail, we (i) propose translations of metric logic programs into regular logic programs and their extension with difference constraints, (ii) prove the completeness and correctness of both translations in terms of equilibrium logic and its extensions with metric and difference constraints, and (iii) describe an implementation of both translations in terms of meta encodings and give some indicative experimental results. Conversely, we may consider metric logic programs as a high-level modeling language for logic programs with difference constraints in the context of temporal domains. We consider a simple yet expressive fragment in which the next operator can be supplied with a time frame expressed as an interval; the full paper shows how this extends to the entire language.

Related work. Pioneering work on extending ASP with linear integer constraints to express quantitative temporal relations was done in [4, 19]. This work ultimately inspired the development of hybrid ASP systems such as *clingcon* [3] and *clingo*[DL] [15]. Metric equilibrium logic was defined in [9, 6] by building on Linear temporal equilibrium logic [1]. The latter provides the logical foundations of the temporal ASP system *telingo* [8]. Metric concepts are also present in stream reasoning, notably, the approach of *lars* [5]. Metric extensions are also considered in Datalog [22], where they led to the *meteor* [23] and (temporal) *vadalog* [7] systems. Actions with durations in ASP are explored in [21]. Finally, [14] considers reductions of (monotonic) Metric temporal logic to Linear temporal logic. A comprehensive comparative account of metric approaches in logic programming is given in [6].

2 Background

To ease the formal elaboration of our translations from metric ASP to regular logic programs and their extension with difference constraints, we put ourselves into the context of the logical framework of Here-and-There and Equilibrium Logic.

The Logic of Here-and-There [20]. A formula over an alphabet $\mathcal{A}$ is defined as

$$\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \quad \text{where } a \in \mathcal{A} .$$

We define the derived operators $\neg\varphi = \varphi \to \bot$ and $\top = \neg\bot$. Elements a of $\mathcal{A}$ are called (Boolean) *atoms*. A *literal* is an atom or an atom preceded by negation, viz. a or $\neg a$. A *theory* is a set of formulas. We sometimes write $\varphi \leftarrow \psi$ instead of $\psi \to \varphi$ to follow logic programming conventions. A *program* is a set of implications of the form $\varphi \leftarrow \psi$ where φ is a disjunction of literals and ψ is a conjunction of literals.

We represent an *interpretation* T as a set of atoms $T \subseteq \mathcal{A}$. An HT-*interpretation* is a pair $\langle H, T\rangle$ of interpretations such that $H \subseteq T$; it is said to be *total* if $H = T$. An HT-interpretation $\langle H, T\rangle$ *satisfies* a formula φ, written $\langle H, T\rangle \models \varphi$, if

1. $\langle H,T\rangle \models a$ if $a \in H$
2. $\langle H,T\rangle \models \varphi \wedge \psi$ if $\langle H,T\rangle \models \varphi$ and $\langle H,T\rangle \models \psi$
3. $\langle H,T\rangle \models \varphi \vee \psi$ if $\langle H,T\rangle \models \varphi$ or $\langle H,T\rangle \models \psi$
4. $\langle H,T\rangle \models \varphi \rightarrow \psi$ if $\langle H',T\rangle \not\models \varphi$ or $\langle H',T\rangle \models \psi$ for each $H' \in \{H,T\}$

An HT-interpretation $\langle H,T\rangle$ is an *HT-model* of a theory Γ if $\langle H,T\rangle \models \varphi$ for each $\varphi \in \Gamma$. A total model $\langle T,T\rangle$ of a theory Γ is an *equilibrium model* if there is no other model $\langle H,T\rangle$ of Γ with $H \subset T$; T is also called a *stable model* of Γ.

The Logic of Here-and-There with Constraints [10]. The syntax of HT$_c$ relies on a signature $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, similar to constraint satisfaction problems, where elements of set $\mathcal{X}$ represent variables and elements of $\mathcal{D}$ are domain values (usually identified with their respective constants). The constraint atoms in $\mathcal{C}$ provide an abstract way to relate values of variables and constants according to the atom's semantics. For instance, *difference constraint atoms* are expressions of the form '$x - y \leq d$', containing variables $x, y \in \mathcal{X}$ and the domain value $d \in \mathcal{D}$. A constraint formula φ over $\mathcal{C}$ is defined as

$$\varphi ::= \bot \mid c \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \quad \text{where } c \in \mathcal{C}$$

Concepts like defined operators, programs, theories, etc. are analogous to HT. Variables can be assigned some value from $\mathcal{D}$ or left *undefined*. For the latter, we use the special symbol $\boldsymbol{u} \notin \mathcal{D}$ and the extended domain $\mathcal{D}_u = \mathcal{D} \cup \{\boldsymbol{u}\}$. A *valuation* v is a function $v : \mathcal{X} \rightarrow \mathcal{D}_u$. We let $\mathbb{V}$ stand for the set of all valuations. We sometimes represent a valuation v as a set $\{(x, v(x)) \mid x \in \mathcal{X}, v(x) \in \mathcal{D}\}$ of pairs, so that $(x, \boldsymbol{u})$ is never formed. This allows us to use standard set inclusion, $v \subseteq v'$, for comparing $v, v' \in \mathbb{V}$.

The semantics of constraint atoms is defined in HT$_c$ via *denotations*, that is, functions $[\![\cdot]\!] : \mathcal{C} \rightarrow 2^{\mathbb{V}}$ mapping each constraint atom to a set of valuations. An *HT$_c$-interpretation* is a pair $\langle h,t\rangle$ of valuations such that $h \subseteq t$; it is *total* if $h = t$. Given a denotation $[\![\cdot]\!]$, an HT$_c$-interpretation $\langle h,t\rangle$ *satisfies* a constraint formula φ, written $\langle h,t\rangle \models \varphi$, if

1. $\langle h,t\rangle \models c$ if $h \in [\![c]\!]$
2. $\langle h,t\rangle \models \varphi \wedge \psi$ if $\langle h,t\rangle \models \varphi$ and $\langle h,t\rangle \models \psi$
3. $\langle h,t\rangle \models \varphi \vee \psi$ if $\langle h,t\rangle \models \varphi$ or $\langle h,t\rangle \models \psi$
4. $\langle h,t\rangle \models \varphi \rightarrow \psi$ if $\langle v,t\rangle \not\models \varphi$ or $\langle v,t\rangle \models \psi$ for each $v \in \{h,t\}$

An HT$_c$-interpretation $\langle h,t\rangle$ is an *HT$_c$-model* of a theory Γ if $\langle h,t\rangle \models \varphi$ for every $\varphi \in \Gamma$. A total model $\langle t,t\rangle$ of a theory Γ is a *constraint equilibrium model* if there is no other model $\langle h,t\rangle$ of Γ with $h \subset t$.

The Metric Temporal Logic of Here-and-There [9, 6]. Given an alphabet $\mathcal{A}$ and $\mathbb{I} = \{[m..n) \mid m \in \mathbb{N}, n \in \mathbb{N} \cup \{\omega\}\}$, a metric temporal formula is defined as[1]

$$\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \mathbf{I} \mid \circ_I \varphi \mid \Box_I \varphi \mid \Diamond_I \varphi \quad \text{where } a \in \mathcal{A}, I \in \mathbb{I}$$

[1]More general formulas, including until, release and past operators, are presented in [6].

The last three cases deal with metric temporal operators, which are indexed by some interval I. In words, $\circ_I$, $\Box_I$, and $\Diamond_I$ are called *next*, *always*, and *eventually*. $\mathbf{I}$ simply refers to the initial state. We write $\circ$, $\Box$, $\Diamond$ for $\circ_{[0..\omega)}$, $\Box_{[0..\omega)}$, $\Diamond_{[0..\omega)}$, respectively. In addition to the aforedefined Boolean operators, we define $\mathbf{F} = \neg\circ\top$, which allow us to refer to the final state. Concepts like programs, theories, etc. are analogous to HT.

The semantics of temporal formulas is defined via *traces*, being sequences $\mathbf{T} = (T_i)_{i\in[0..\lambda)}$ of interpretations $T_i \subseteq \mathcal{A}$; λ is the length of $\mathbf{T}$. Here, we consider only traces of finite length. We define the ordering $\mathbf{H} \leq \mathbf{T}$ between traces of the same length λ as $H_i \subseteq T_i$ for each $i \in [0..\lambda)$, and $\mathbf{H} < \mathbf{T}$ as both $\mathbf{H} \leq \mathbf{T}$ and $\mathbf{H} \neq \mathbf{T}$. An *HT-trace* over $\mathcal{A}$ of length λ is a sequence of pairs $(\langle H_i, T_i\rangle)_{i\in[0..\lambda)}$ with $H_i \subseteq T_i$ for any $0 \leq i < \lambda$; it is *total* if $H_i = T_i$. For convenience, we represent it as the pair $\langle \mathbf{H}, \mathbf{T}\rangle$ of traces $\mathbf{H} = (H_i)_{i\in[0..\lambda)}$ and $\mathbf{T} = (T_i)_{i\in[0..\lambda)}$.

Metric information is captured by timing functions. Given $\lambda \in \mathbb{N}$, we say that $\tau : [0..\lambda) \to \mathbb{N}$ is a (strict) *timing function* wrt λ if $\tau(0) = 0$ and $\tau(k) < \tau(k+1)$ for $0 \leq k < \lambda - 1$. A *timed* HT-trace $(\langle \mathbf{H}, \mathbf{T}\rangle, \tau)$ over $\mathcal{A}$ and $(\mathbb{N}, <)$ of length λ is a pair consisting of an HT-trace $\langle \mathbf{H}, \mathbf{T}\rangle$ over $\mathcal{A}$ of length λ and a timing function τ wrt λ. A timed HT-trace $\mathbf{M} = (\langle \mathbf{H}, \mathbf{T}\rangle, \tau)$ of length λ over alphabet $\mathcal{A}$ *satisfies* a metric formula φ at $k \in [0..\lambda)$, written $\mathbf{M}, k \models \varphi$, if

1. $\mathbf{M}, k \models a$ if $a \in H_k$
2. $\mathbf{M}, k \models \varphi \wedge \psi$ if $\mathbf{M}, k \models \varphi$ and $\mathbf{M}, k \models \psi$
3. $\mathbf{M}, k \models \varphi \vee \psi$ if $\mathbf{M}, k \models \varphi$ or $\mathbf{M}, k \models \psi$
4. $\mathbf{M}, k \models \varphi \to \psi$ if $\mathbf{M}', k \not\models \varphi$ or $\mathbf{M}', k \models \psi$,
   for both $\mathbf{M}' = \mathbf{M}$ and $\mathbf{M}' = (\langle \mathbf{T}, \mathbf{T}\rangle, \tau)$
5. $\mathbf{M}, k \models \mathbf{I}$ if $k = 0$
6. $\mathbf{M}, k \models \circ_I\, \varphi$ if $k + 1 < \lambda$ and $\mathbf{M}, k{+}1 \models \varphi$ and $\tau(k{+}1) - \tau(k) \in I$
7. $\mathbf{M}, k \models \Diamond_I\, \varphi$ if $\mathbf{M}, i \models \varphi$ for some $i \in [k..\lambda)$ with $\tau(i) - \tau(k) \in I$
8. $\mathbf{M}, k \models \Box_I\, \varphi$ if $\mathbf{M}, i \models \varphi$ for all $i \in [k..\lambda)$ with $\tau(i) - \tau(k) \in I$

A timed HT-trace $\mathbf{M}$ is an *MHT-model* of a metric theory Γ if $\mathbf{M}, 0 \models \varphi$ for all $\varphi \in \Gamma$. A total MHT-model $(\langle \mathbf{T}, \mathbf{T}\rangle, \tau)$ of a theory Γ is a *metric equilibrium model* if there is no other model $(\langle \mathbf{H}, \mathbf{T}\rangle, \tau)$ of Γ with $\mathbf{H} < \mathbf{T}$.

For illustration, let us consider the formalization of the dentist scenario in (1) to (9). We assume that variables L and L' are substituted by distinct locations *office*, *atm*, *dentist*, and *home*; and variable I by items *cash* and *icard*. We use $\delta\langle L, L'\rangle$ to refer to the distance between two locations from Table 1. As in [19], we assume that Ram is automatically picking up items when being at the same position.

$$\Box(at(ram, office) \leftarrow \mathbf{I}) \tag{1}$$
$$\Box(at(cash, atm) \leftarrow \mathbf{I}) \tag{2}$$
$$\Box(at(icard, home) \leftarrow \mathbf{I}) \tag{3}$$

$$\Box(\textstyle\bigvee_{L'\neq L} go(ram, L') \leftarrow at(ram, L) \wedge \neg \mathbf{F}) \tag{4}$$
$$\Box(has(ram, I) \leftarrow at(ram, L) \wedge at(I, L)) \tag{5}$$
$$\Box(at(I, L) \leftarrow at(ram, L) \wedge has(ram, I)) \tag{6}$$
$$\Box(\circ_{[\delta\langle L,L'\rangle..\delta\langle L,L'\rangle+1)} at(ram, L') \leftarrow at(ram, L) \wedge go(ram, L')) \tag{7}$$
$$\Box(\circ has(ram, I) \leftarrow has(ram, I) \wedge \neg \mathbf{F}) \tag{8}$$
$$\Box(\circ at(I, L) \leftarrow \neg has(ram, I) \wedge at(I, L) \wedge \neg \mathbf{F}) \tag{9}$$

In brief, Rules (1) to (3) give the initial situation. Rule (4) delineates possible actions. Rules (5) and (6) capture indirect effects. Rule (7) is the effect of moving from location L to L'; it uses the next operator restricted by the duration between locations. Rules (8) and (9) address inertia.

3 Metric Logic Programs

Metric logic programs, defined as metric theories composed of implications akin to logic programming rules, derive their semantics from their metric equilibrium models. Syntactically, a *metric logic program* over $\mathcal{A}$ is a set of *metric rules* of form

$$\Box(\alpha \leftarrow \beta) \quad \text{or} \quad \Box(\circ_I a \leftarrow \beta)$$

for $\alpha = a_1 \vee \cdots \vee a_m \vee \neg a_{m+1} \vee \cdots \vee \neg a_n$, $\beta = b_1 \wedge \cdots \wedge b_o \wedge \neg b_{o+1} \wedge \cdots \wedge \neg b_p$, and $m, n, o, p \in \mathbb{N}$ with $a, a_i \in \mathcal{A}$ for $1 \leq i \leq n$, and $b_i \in \mathcal{A} \cup \{\mathbf{I}, \mathbf{F}\}$ for $1 \leq i \leq p$.

While our considered language fragment excludes global temporal operators and disjunctive metric heads, it effectively captures state transitions and allows for imposing timing constraints upon them. A comprehensive treatment is provided in our full paper, but they are omitted here for simplicity as they require more elaborate translations.

Our two alternative translations share a common structure, each divided into three distinct parts. The first part maps a metric program into a regular one. This part captures the state transitions along an HT-trace specified by the metric program, and is common to both translations. The second and third part capture the timing function along with its interplay with the interval constraints imposed by the metric program, respectively. The two variants of these parts are described in Section 4 and 5 below.

The first part of our translation takes a metric program over $\mathcal{A}$ and yields rules over $\mathcal{A}^* = \bigcup_{k\in\mathbb{N}} \mathcal{A}_k$ for $\mathcal{A}_k = \{a_k \mid a \in \mathcal{A}\}$ and $k \in \mathbb{N}$. Atoms of form a_k in $\mathcal{A}^*$ represent the values taken by variable $a \in \mathcal{A}$ at different points k along a trace of length λ.

We begin by inductively defining the translation $(r)_k$ of a metric rule $\Box r$ at k for $0 \leq k < \lambda$ and $\lambda \in \mathbb{N}$ as follows:[2]

$$(a)_k = a_k \text{ if } a \in \mathcal{A} \qquad (\bot)_k = \bot$$
$$(\circ_I a)_k = \begin{cases} \bot & \text{if } k = \lambda - 1 \\ (a)_{k+1} & \text{otherwise} \end{cases} \qquad \begin{aligned}(\varphi_1 \wedge \varphi_2)_k &= (\varphi_1)_k \wedge (\varphi_2)_k \\ (\varphi_1 \vee \varphi_2)_k &= (\varphi_1)_k \vee (\varphi_2)_k\end{aligned}$$
$$(\mathbf{I})_k = \begin{cases} \top & \text{if } k = 0 \\ \bot & \text{otherwise} \end{cases} \qquad (\alpha \leftarrow \beta)_k = \{(\varphi_1)_k \leftarrow (\varphi_2)_k\}$$

[2]Note that $\top$, $\neg$ and $\mathbf{F}$ are defined operators.

Note that we drop the always operator $\square$ preceding metric rules in the translation; it is captured by producing a rule instance for every $0 \leq k < \lambda$. Accordingly, for a metric program P over $\mathcal{A}$ and $\lambda \in \mathbb{N}$, we define

$$\Pi_\lambda(P) = \bigcup_{\square r \in P,\, 0 \leq k < \lambda} (r)_k \quad \text{over } \mathcal{A}^* \ .$$

For illustration, consider the instance of (7) for moving from *office* to *home*

$$\square(\circ_{[15..16)} \mathit{at}(\mathit{ram}, \mathit{home}) \leftarrow \mathit{at}(\mathit{ram}, \mathit{office}) \wedge \mathit{go}(\mathit{ram}, \mathit{home})) \ . \tag{10}$$

Our translation ignores $\square$ at first and yields:

$$\bot \leftarrow \mathit{at}(\mathit{ram}, \mathit{office})_k \wedge \mathit{go}(\mathit{ram}, \mathit{home})_k \quad \text{for } k = \lambda - 1 \tag{11}$$

$$\mathit{at}(\mathit{ram}, \mathit{home})_{k+1} \leftarrow \mathit{at}(\mathit{ram}, \mathit{office})_k \wedge \mathit{go}(\mathit{ram}, \mathit{home})_k \quad \text{otherwise} \tag{12}$$

When assembling $\Pi_\lambda(P)$ for $\lambda = 100$ and P being the rules in (1) to (9), we account for $\square$ by adding 99 instances of the rule in (12) and a single instance of (11).

This first part of our translation follows Kamp's translation [17] for Linear temporal logic. Of particular interest is the translation of $\circ_I a \leftarrow \beta$. The case analysis accounts for the actual state transition of the next operator, which is infeasible at the end of the trace. Thus, we either derive a_{k+1} or a contradiction. The metric aspect is captured by the translations in Section 4 and 5. Whenever all intervals in a metric programs P are of form $[0..\omega)$, we get a one-to-one correspondence between MHT-traces of length λ of P with an arbitrary yet fixed timing function and HT-interpretations of $\Pi_\lambda(P)$. Finally, it is worth noting that the size of the resulting program $\Pi_\lambda(P)$ grows with λ.

4 Translation of Metric Logic Programs to HT

We begin by formalizing timing functions τ via Boolean atoms in $\mathcal{T} = \{t_{k,d} \mid k, d \in \mathbb{N}\}$. An atom like $t_{k,d}$ represents that $\tau(k) = d$. To obtain finite theories, we furthermore impose an upper bound $\nu \in \mathbb{N}$ on the range of τ. Hence, together with the trace length λ, our formalization $\Delta_{\lambda,\nu}$ only captures timing functions τ satisfying $\tau(\lambda - 1) \leq \nu$.

Given $\lambda, \nu \in \mathbb{N}$, we let

$$\Delta_{\lambda,\nu} = \{t_{0,0}\} \cup \{\textstyle\bigvee_{d < d' \leq \nu} t_{k+1,d'} \leftarrow t_{k,d} \mid 0 \leq k < \lambda - 1, 0 \leq d \leq \nu\} \tag{13}$$

Starting from $\tau(0) = 0$, represented by $t_{0,0}$, the rule in (13) assigns strictly increasing time points to consecutive states, reflecting that $\tau(k) < \tau(k+1)$ for $0 \leq k < \lambda - 1$.

The last part of our formalization accounts for the interplay of the timing function with the interval conditions imposed in the program. Given $\lambda, \nu \in \mathbb{N}$ and a metric program P, we let

$$\Psi_{\lambda,\nu}(P) = \{\bot \leftarrow (\beta)_k \wedge t_{k,d} \wedge t_{k+1,d'} \mid 0 \leq k < \lambda - 1, 0 \leq d < d' \leq \nu, \quad d' - d < m, \square(\circ_{[m..n)} a \leftarrow \beta) \in P\} \cup \tag{14}$$

$$\{\bot \leftarrow (\beta)_k \wedge t_{k,d} \wedge t_{k+1,d'} \mid 0 \leq k < \lambda - 1, 0 \leq d < d' \leq \nu, \quad n \in \mathbb{N}, d' - d \geq n, \square(\circ_{[m..n)} a \leftarrow \beta) \in P\} \tag{15}$$

The integrity constraints ensure that for every metric rule $\square(\circ_{[m..n)} a \leftarrow \beta)$ the duration between the kth and $(k+1)$st state in a trace falls within interval $[m..n)$. With $\tau(k) = d$ and $\tau(k+1) = d'$, this amounts to checking whether $d + m \leq d' < d + n$, if n is finite; otherwise, the verification of the upper bound in (15) is dropped for $n = \omega$.

Note that the size of both $\Delta_{\lambda,\nu}$ and $\Psi_{\lambda,\nu}(P)$ is proportional to $\mathcal{O}(\lambda \cdot \nu^2)$. Hence, long traces and even more severely fine-grained timing functions lead to a significant blow up when translating metric programs into regular ones with the above formalization.

For the rule in (10), we get

$$\bot \leftarrow \mathit{at}(\mathit{ram}, \mathit{office})_k \wedge \mathit{go}(\mathit{ram}, \mathit{home})_k \wedge t_{k,d} \wedge t_{k+1,d'} \quad \text{for } d' - d < 15 \quad (16)$$

$$\bot \leftarrow \mathit{at}(\mathit{ram}, \mathit{office})_k \wedge \mathit{go}(\mathit{ram}, \mathit{home})_k \wedge t_{k,d} \wedge t_{k+1,d'} \quad \text{for } d' - d \geq 16 \quad (17)$$

and $0 \leq k < \lambda - 1$, $0 \leq d < d' \leq \nu$. For $\lambda = 100$ and $\nu = 1000$, this then amounts to roughly 10^8 instances for each of the above constraints.

In what follows, we characterize the effect of our formalization in terms of HT-models, and ultimately show the completeness and correctness of our translation.

Definition 1. An HT interpretation $\langle H, T \rangle$ over $\mathcal{A}$ with $\mathcal{T} \subseteq \mathcal{A}$, is *timed* wrt $\lambda \in \mathbb{N}$, if there is a timing function τ wrt λ such that for all $0 \leq k < \lambda$, $d \in \mathbb{N}$, we have

$$\langle H, T \rangle \models t_{k,d} \text{ iff } \tau(k) = d \quad \text{and} \quad \langle T, T \rangle \models t_{k,d} \text{ iff } \tau(k) = d$$

We also call τ the timing function induced by $\langle H, T \rangle$.

Proposition 1. *Let $\lambda, \nu \in \mathbb{N}$.*
If $\langle T, T \rangle$ is an equilibrium model of $\Delta_{\lambda,\nu}$ then $\langle T, T \rangle$ is timed wrt λ.

Proposition 2. *Let $\langle H, T \rangle$ be an HT interpretation and $\lambda \in \mathbb{N}$.*
If $\langle H, T \rangle$ is timed wrt λ and induces τ then $\langle H, T \rangle \models \Delta_{\lambda,\nu}$ for $\nu = \tau(\lambda - 1)$.

Clearly, the last proposition extends to equilibrium models.

Given a timed HT-trace $\mathbf{M} = (\langle H_k, T_k \rangle_{k \in [0..\lambda)}, \tau)$ of length λ over $\mathcal{A}$, we define $\theta(\mathbf{M})$ as HT interpretation $\langle H \cup X, T \cup X \rangle$ where

$$H = \{a_k \in \mathcal{A}_k \mid 0 \leq k < \lambda, a \in H_k\} \qquad T = \{a_k \in \mathcal{A}_k \mid 0 \leq k < \lambda, a \in T_k\}$$
$$X = \{t_{k,d} \mid \tau(k) = d, 0 \leq k < \lambda, d \in \mathbb{N}\}$$

Note that $\theta(\mathbf{M})$ is an HT interpretation timed wrt λ.

Conversely, given an HT interpretation $\langle H, T \rangle$ timed wrt λ over $\mathcal{A}^* \cup \mathcal{T}$ and its induced timing function τ, we define $\sigma(\langle H, T \rangle)$ as the timed HT-trace

$$(\langle \{a \in \mathcal{A} \mid a_k \in H\}, \{a \in \mathcal{A} \mid a_k \in T\} \rangle_{k \in [0..\lambda)}, \tau)$$

In fact, both functions σ and θ are invertibles, and we get a one-to-one correspondence between HT interpretations timed wrt λ and timed HT-traces of length λ.

Finally, we have the following completeness and correctness result.

Theorem 1 (Completeness)**.** *Let P be a metric logic program and $\mathbf{M} = (\langle \mathbf{T}, \mathbf{T} \rangle, \tau)$ a total timed HT-trace of length λ.*

If $\mathbf{M}$ is an metric equilibrium model of P, then $\theta(\mathbf{M})$ is an equilibrium model of $\Pi_\lambda(P) \cup \Delta_{\lambda,\nu} \cup \Psi_{\lambda,\nu}(P)$ with $\nu = \tau(\lambda - 1)$.

Theorem 2 (Correctness). *Let P be a metric logic program, and $\lambda, \nu \in \mathbb{N}$.*
If $\langle T, T\rangle$ is an equilibrium model of $\Pi_\lambda(P) \cup \Delta_{\lambda,\nu} \cup \Psi_{\lambda,\nu}(P)$, then $\sigma(\langle T, T\rangle)$ is a metric equilibrium model of P.

5 Translation of Metric Logic Programs to HT$_c$

We now present an alternative, refined formalization of the second and third parts, utilizing integer variables and difference constraints to capture the timing function more effectively. To this end, we use the logic of HT$_c$ to combine the Boolean nature of ASP with constraints on integer variables.

Given base alphabet $\mathcal{A}$ and $\lambda \in \mathbb{N}$, we consider the HT$_c$ signature $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$ where[3]

$$\begin{aligned}
&\mathcal{X} = \mathcal{A}^* \cup \{t_k \mid 0 \leq k < \lambda\}\\
&\mathcal{D} = \{\boldsymbol{t}\} \cup \mathbb{N}\\
&\mathcal{C} = \{a = \boldsymbol{t} \mid a \in \mathcal{A}^*\} \cup \{x = d \mid x \in \mathcal{X} \setminus \mathcal{A}^*, d \in \mathbb{N}\} \cup\\
&\qquad \{x - y \leq d \mid x, y \in \mathcal{X} \setminus \mathcal{A}^*, d \in \mathbb{N}\}
\end{aligned}$$

Rather than using Boolean variables, this signature represents timing functions τ directly by integer variables t_k, capturing that $\tau(k) = t_k$ for $0 \leq k < \lambda$. This is enforced by the integer constraints in $\mathcal{C}$, whose meaning is defined by the following denotations:

$$\begin{aligned}
[\![\, a = \boldsymbol{t} \,]\!] &= \{v \in \mathbb{V} \mid v(a) = \boldsymbol{t}\} \qquad\qquad \text{for all } a \in \mathcal{A}^*\\
[\![\, x = d \,]\!] &= \{v \in \mathbb{V} \mid v(x), d \in \mathbb{N},\ v(x) = d\}\\
[\![\, x - y \leq d \,]\!] &= \{v \in \mathbb{V} \mid v(x), v(y), d \in \mathbb{N},\ v(x) - v(y) \leq d\}
\end{aligned}$$

This leads us to the the following counterpart of $\Delta_{\lambda,\nu}$ in (13). Given $\lambda \in \mathbb{N}$, we define

$$\Delta^c_\lambda = \{t_0 = 0\} \cup \{t_k - t_{k+1} \leq -1 \mid 0 \leq k < \lambda - 1\} \tag{18}$$

Starting from $t_0 = 0$, the difference constraints in (18) enforce that $t_k < t_{k+1}$ reflecting that $\tau(0) = 0$ and $\tau(k) < \tau(k+1)$ for $0 \leq k < \lambda - 1$. Moreover, Δ^c_λ is unbound and thus imposes no restriction on timing functions. And no variable t_k is ever undefined:

Proposition 3. *Let $\langle h, t\rangle$ be an HT$_c$ interpretation and $\lambda \in \mathbb{N}$*
If $\langle h, t\rangle \models \Delta^c_\lambda$, then $h(t_k) \in \mathbb{N}$ for all $0 \leq k < \lambda$.

We also have $t(t_k) \in \mathbb{N}$ by definition of HT$_c$ interpretations, that is, since $h \subseteq t$.

Our variant of the third part of our translation re-expresses the ones in (14/15) in terms of integer variables and difference constraints. Given $\lambda \in \mathbb{N}$ and a metric logic program P, we define

$$\begin{aligned}
\Psi^c_\lambda(P) = \{\bot \leftarrow (\beta)_k \wedge \neg(t_k - t_{k+1} \leq -m) \mid 0 \leq k < \lambda - 1,& \qquad (19)\\
\Box(\circ_{[m..n)} a \leftarrow \beta) \in P\} \cup&\\
\{\bot \leftarrow (\beta)_k \wedge \neg(t_{k+1} - t_k \leq n - 1) \mid 0 \leq k < \lambda - 1,& \qquad (20)
\end{aligned}$$

[3]In HT$_c$ [10], Boolean variables are already captured by truth values $\boldsymbol{t}$ and $\boldsymbol{u}$ (rather than $\boldsymbol{f}$[alse]).

$$n \in \mathbb{N}, \Box(\circ_{[m..n)} a \leftarrow \beta) \in P\}$$

In fact, both Δ^c_λ and $\Psi^c_\lambda(P)$ drop the upper bound on the range of a timing function, as required in their Boolean counterparts. Hence, their size is only proportional to $\mathcal{O}(\lambda)$, and thus considerably smaller than their purely Boolean counterparts.

For the rule in (10), we get

$$\bot \leftarrow \mathit{at}(\mathit{ram}, \mathit{office})_k \wedge \mathit{go}(\mathit{ram}, \mathit{home})_k \wedge \neg(t_k - t_{k+1} \leq -15) \tag{21}$$
$$\bot \leftarrow \mathit{at}(\mathit{ram}, \mathit{office})_k \wedge \mathit{go}(\mathit{ram}, \mathit{home})_k \wedge \neg(t_{k+1} - t_k \leq 15) \tag{22}$$

for $0 \leq k < \lambda - 1$. Given $\lambda = 100$, this only amounts to 10^2 instances.

Mirroring our approach in Section 4, we capture the meaning of Δ^c_λ using specialized HT$_c$-models. This leads to the completeness and correctness of our translation.

Definition 2. An HT$_c$ interpretation $\langle h, t \rangle$ over $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is *timed* wrt λ, if there is a timing function τ wrt λ such that $h(t_k) = \tau(k)$ and $t(t_k) = \tau(k)$ for all $0 \leq k < \lambda$.

As above, we call τ the timing function induced by $\langle h, t \rangle$.

Proposition 4. *Let $\langle h, t \rangle$ be an HT$_c$ interpretation and $\lambda \in \mathbb{N}$.*
If $\langle h, t \rangle \models \Delta^c_\lambda$ then $\langle h, t \rangle$ is timed wrt λ.

Proposition 5. *Let $\langle h, t \rangle$ be an HT$_c$ interpretation and $\lambda \in \mathbb{N}$.*
If $\langle h, t \rangle$ is timed wrt λ then $\langle h, t \rangle \models \Delta^c_\lambda$.

Unlike Proposition 1, the latter refrain from requiring HT$_c$-interpretations in equilibrium.

Given an HT-trace $\mathbf{M} = (\langle H_k, T_k \rangle_{k \in [0..\lambda)}, \tau)$ of length λ, we define $\theta^c(\mathbf{M})$ as the HT$_c$ interpretation $\langle h \cup x, t \cup x \rangle$ where h, t, x are valuations such that

$$h = \{(a_k, \boldsymbol{t}) \mid 0 \leq k < \lambda, a \in H_k\} \qquad t = \{(a_k, \boldsymbol{t}) \mid 0 \leq k < \lambda, a \in T_k\}$$
$$x = \{(t_k, d) \mid \tau(k) = d, 0 \leq k < \lambda, d \in \mathbb{N}\}$$

Similar to above, $\theta^c(\mathbf{M})$ is an HT$_c$ interpretation timed wrt λ.

Conversely, given an HT$_c$ interpretation $\langle h, t \rangle$ timed wrt λ and its induced timing function τ, we define $\sigma^c(\langle h, t \rangle)$ as the timed HT-trace

$$(\langle \{a \mid h(a_k) = \boldsymbol{t}\}, \{a \mid t(a_k) = \boldsymbol{t}\} \rangle_{k \in [0..\lambda)}, \tau)$$

As above, functions σ^c and θ^c are invertibles. Hence, we get a one-to-one correspondence between HT$_c$ interpretations timed wrt λ and timed HT-traces of length λ.

Finally, we have the following completeness and correctness result.

Theorem 3 (Completeness). *Let P be a metric logic program and $\mathbf{M} = (\langle \mathbf{T}, \mathbf{T} \rangle, \tau)$ a total timed HT-trace of length λ.*

If $\mathbf{M}$ is an metric equilibrium model of P, then $\theta^c(\mathbf{M})$ is a constraint equilibrium model of $\Pi_\lambda(P) \cup \Delta^c_\lambda \cup \Psi^c_\lambda(P)$.

Theorem 4 (Correctness). *Let P be a metric logic program, and $\lambda \in \mathbb{N}$.*

If $\langle t, t \rangle$ is a constraint equilibrium model of $\Pi_\lambda(P) \cup \Delta^c_\lambda \cup \Psi^c_\lambda(P)$, then $\sigma^c(\langle t, t \rangle)$ is a metric equilibrium model of P.

6 Implementation

In what follows, we rely on a basic acquaintance with the ASP system *clingo* [13]. We outline specialized concepts as they are introduced throughout the text. We show below how easily our approach is implemented via *clingo*'s meta encoding framework. This serves us as a blueprint for a more sophisticated future implementation. *Clingo* allows for reifying a ground logic program in terms of facts, which can then be (re)interpreted by a meta encoding. The result of another grounding is then channeled to the respective back-end, in our case a regular or hybrid ASP solver, respectively. Though, for brevity, we must refer to [16] for details, we mention that a reified ground program is represented by instances of predicates `atom_tuple`, `literal_tuple`, `rule`, `output`, etc. [4]

```
 1  time(0..lambda-1).

 3  conjunction(B,T) :- literal_tuple(B), time(T),
 4         hold(L,T) : literal_tuple(B, L), L > 0;
 5     not hold(L,T) : literal_tuple(B,-L), L > 0.

 7  body(normal(B),T) :- rule(_,normal(B)), conjunction(B,T).
 8  body(sum(B,G),T)  :- rule(_,sum(B,G)), time(T),
 9    #sum{W,L :     hold(L,T), weighted_literal_tuple(B, L,W), L>0;
10         W,L : not hold(L,T), weighted_literal_tuple(B,-L,W), L>0} >= G.

12    hold(A,T) : atom_tuple(H,A)   :- rule(disjunction(H),B), body(B,T).
13  { hold(A,T) : atom_tuple(H,A) } :- rule(     choice(H),B), body(B,T).
```

Listing 1: Timed meta encoding (`meta.lp`)

Listing 1 modifies the basic meta encoding in [16] by adding a variable `T` for time steps to all derived predicates. Their range is fixed in Line 1. In this way, an atom `hold(a,k)` stands for a_k in $\mathcal{A}^*$, where `a` is the numeric identifier of a in $\mathcal{A}$. While this encoding handles Boolean connectives, the metric ones are treated in Listing 2. The rules in Line 1 and 2 restore the symbolic representation of the numerically identified atoms, which allows us to analyze the inner structure of modalized propositions. Lines (4/5) and (7/8) deal with **I** and **F**, respectively. Lines 10 and 11 realize the metric next operator, $\circ_I a$, represented by term `next(I,a)`. Together Listing 1 and 2 account for $\Pi_\lambda(P)$.

```
 1  true(O,K) :- output(O,B), time(K), hold(L,K) : literal_tuple(B,L).
 2  hold(L,K) :- output(O,B), time(K), true(O,K),  literal_tuple(B,L).

 4  :- true(initially,K), time(K), K!=0.
 5  true(initially,0).

 7  :- true(finally,K), time(K), K!=lambda-1.
 8  true(finally,lambda-1).

10  :- true(next(_,_),K), time(K), K=lambda-1.
11  true(V,K+1) :- true(next(I,V),K).
```

Listing 2: Meta encoding for metric part of $\Pi_\lambda(P)$ (`bounded.lp`)

When expressing time via Boolean variables, the two previous listings are combined with Listing 3 and 4 below, which realize $\Delta_{\lambda,\nu}$ and $\Psi_{\lambda,\nu}(P)$, respectively. Atoms $t_{k,d}$ in $\mathcal{T}$ are represented by `t(k,d)`. The upper bound ν on the timing function's range is

[4]Below we draw upon the symbol table, captured by `output/2`, for extracting syntactic entities.

given by `v`, and ω is represented by `w`. The two encodings directly mirror the definitions of $\Delta_{\lambda,\nu}$ and $\Psi_{\lambda,\nu}(P)$, with one key difference: the rule bodies in (14) and (15) are replaced in Lines 1 and 2 of Listing 4 by auxiliary atoms of predicate `true/2`.

```
1  timepoint(0..v).

3  t(0,0).
4  t(K+1,D') : D<D', timepoint(D') :- t(K,D), time(K+1).
```

Listing 3: Meta encoding for $\Delta_{\lambda,\nu}$ (`time.lp`)

```
1  :- true(next((M,N),V),K), t(K,D), t(K+1,D'), D' - D < M.
2  :- true(next((M,N),V),K), t(K,D), t(K+1,D'), D' - D >= N , N!=w.
```

Listing 4: Meta encoding for $\Psi_{\lambda,\nu}(P)$ (`interval.lp`)

When expressing time in terms of integer variables, we rely on difference constraints for modeling timing functions. Such simplified linear constraints have the form '$x - y \leq d$' for $x, y \in \mathcal{X}$ and $d \in \mathbb{Z}$ and are supported by the *clingo* extensions *clingcon* [3] and *clingo*[DL] [15]. We use below *clingcon*'s syntax and represent them as '`&sum{`x `;` y`} <=` d'. A Boolean atom a can be seen as representing '$a = \boldsymbol{t}$'. In the case at hand, Listing 1 and 2 are now completed by Listing 5 and 6 below. As above, they faithfully replicate the definitions of Δ^c_λ and $\Psi^c_\lambda(P)$. Unlike above, however, the timing function is now captured by integer variables of form `t(k)` and its range restriction is now obsolete. The rules in Listing 5 mirror the two conditions on timing functions, namely, that `t(0)` equals zero and that the instances of `t(K+1)` receive a strictly greater integer than the ones of `t(K)` for `K` ranging from `0` to `lambda-1`. Similarly, given that `w` stands for ω, the two rules in Listing 6 correspond to the difference constraints in (19) and (20).

```
1  &sum{t(0)} = 0.
2  &sum{t(K) ; -t(K+1)} <= -1 :- time(K), time(K+1).
```

Listing 5: Meta encoding for Δ^c_λ (`time-c.lp`)

```
1  &sum{t(K); -t(K+1)} <= -M  :- true(next((M,N),V),K), time(K), time(K+1).
2  &sum{t(K+1); -t(K)} <= N-1 :- true(next((M,N),V),K), time(K), time(K+1), N!=w.
```

Listing 6: Meta encoding for $\Psi^c_\lambda(P)$ (`interval-c.lp`)

As in Listing 4, we use auxiliary atoms of predicate `true/2` rather than the corresponding rule bodies. Notably, we shifted in Listing 6 the difference constraints in (19) and (20) from the body to the head. This preserves (strong) equivalence in HT_c whenever all variables comprised in a constraint atom are defined, as guaranteed by Proposition 3.

```
1  item(icard). item(cash).
2  loc(dentist). loc(office). loc(atm). loc(home).

4  at(ram,office) :- initially.
5  at(cash,atm)   :- initially.
6  at(icard,home) :- initially.

8  go(ram,L') : loc(L'), L'!=L :- at(ram,L), not finally.

10 has(ram,I) :- at(ram,L), at(I,L), item(I).
11 at(ram,L)  :- at(ram,L), has(ram,I).
```

```
13 next((D,D+1),at(ram,L')) :- at(ram,L), go(ram,L'), distance(L,L',D).
15 next((0,w),has(ram,I)) :- has(ram,I), not finally.
16 next((0,w),at(I,L)) :- at(I,L), item(I), not has(_,I), not finally.
```

Listing 7: Metric logic program for dentist example (`dentist.lp`)

Listing 7 illustrates the above concepts using the dentist example (leaving out the representation of Table 1 in terms of `distance/3`). For simplicity, we assume that each rule is implicitly in the scope of an always operator $\Box$. Moreover, included temporal operators and their comprised atoms are exempt from simplifications during grounding.[5] We use predicate `next/2` for the metric next operator. As an example, consider the instance of Line 13 for moving from *office* to *home*, viz. the counterpart of (10).

```
1 next((15,16),at(ram,home)) :-
2              at(ram,office), go(ram,home), distance(office,home,15).
```

Note that `next((0,w),·)` in the head of the last two rules stands for $\circ$ aka $\circ_{[0..\omega)}$.

Note that the above encoding does not enforce that Ram is at the dentist in one hour with all necessary items. Ideally, this is represented with a global operator like $\Diamond_I \phi$. In our example, this amounts to adding the (informal) rules

```
1 :- initially, not ◇[0..61)goal
2 goal :- at(ram,dentist), has(ram,icard), has(ram,cash).
```

In the two approaches at hand, we may compensate the lack of global metric operators by replacing Line 1 by '`:- finally, not goal`' and enforcing the time limit either by setting ν to 60 in our HT-based approach or by extending Listing 5 with '`&sum{t(K)} <= 60 :- time(K), not time(K+1).`' in our HT$_c$-based approach. However, this is only effective when the goal is achieved at the final step.

The example in Listing 7 is addressed with *clingo* in the following way.

```
clingo dentist.lp --output=reify |
clingo 0 - meta.lp bounded.lp time.lp interval.lp -c lambda=4 -c v=110
```

When using *clingcon* instead, it suffices to replace the second line with:

```
clingcon 0 - meta.lp bounded.lp time-c.lp interval-c.lp -c lambda=4
```

Our choices of `lambda` and `v` allow for all movement combinations within the 4 steps required to reach the goal; we obtain in each case 27 solutions. Once we include the query-oriented additions from above, we obtain a single model instead.

To gather some indicative results on the scalability of both approaches, we multiplied the durations in Table 1 (and limit `v` for *clingo*) with 1, 5, and 10 and summarize the results in Table 2. Despite the rather limited setting, we observe that the usage of integer variables leads to a complete independence of performance and time granularity.

7 Conclusion

We presented a computational approach to metric ASP that allows for fine-grained timing constraints. We developed two alternative translations from firm semantic foundations,

[5]In meta encodings, this is done by adding corresponding `#external` directives.

	clingo [16]			*clingcon* [3]			*clingo*[DL] [15]		
	solve	ground	#rules	solve	ground	#rules	solve	ground	#rules
1	0.01	0.281	297211	0.00	0.020	2165	0.00	0.018	2165
5	7.09	19.358	7527451	0.00	0.020	2165	0.00	0.018	2165
10	609.46s	143.083	30177751	0.00	0.020	2165	0.00	0.018	2165

Table 2: Indicative results for *clingo*, *clingcon*, *clingo*[DL] (times in seconds; no resource limits).

and proved their completeness and correctness. Our second translation has a clear edge over the first one, when it comes to a fine-grained resolution of time. This is achieved by outsourcing the treatment of time. Clearly, this is not for free either. *clingo*[DL] maps difference constraints into graphs, whose nodes are time variables and weighted edges reflect the actual constraints. This results in a quadratic space complexity. *clingcon* pursues a lazy approach to constraint solving that gradually unfolds an ASP encoding of linear constraints. In the worst case, this amounts to the space requirements of our first translation. As well, such constraints are hidden from the ASP solver and cannot be used for directing the search. Hence, despite our indicative observations, a detailed empirical analysis is needed to account for the subtleties of our translation and its target systems.

However, a prominent use case involves employing the identity (timing) function, where intervals reference only state indices within traces. This coarser notion of time reduces the discrepancy between our two translations. Further improvement is possible through more sophisticated Boolean encodings, such as an order encoding [11, 2].

References

[1] Aguado, F., Cabalar, P., Diéguez, M., Pérez, G., Schaub, T., Schuhmann, A., Vidal, C.: Linear-time temporal answer set programming. TPLP 23(1), 2-56 (2023).

[2] Banbara, M., Gebser, M., Inoue, K., Ostrowski, M., Peano, A., Schaub, T., Soh, T., Tamura, N., Weise, M.: aspartame: Solving constraint satisfaction problems with answer set programming. In: LPNMR. pp. 112-126. Springer (2015)

[3] Banbara, M., Kaufmann, B., Ostrowski, M., Schaub, T.: Clingcon: The next generation. TPLP 17(4), 408-461 (2017).

[4] Baselice, S., Bonatti, P., Gelfond, M.: Towards an integration of answer set and constraint solving. In: ICLP. pp. 52-66. Springer (2005)

[5] Beck, H., Dao-Tran, M., Eiter, T.: LARS: A logic-based framework for analytic reasoning over streams. AIJ 261, 16-70 (2018).

[6] Becker, A., Cabalar, P., Diéguez, M., Schaub, T., Schuhmann, A.: Metric temporal equilibrium logic over timed traces. CoRR abs/2304.14778 (2023)

[7] Bellomarini, L., Blasi, L., Nissl, M., Sallinger, E.: The temporal vadalog system. In: RuleML+RR. pp. 130-145. Springer (2022).

[8] Cabalar, P., Diéguez, M., Laferriere, F., Schaub, T.: Implementing dynamic answer set programming over finite traces. In: ECAI. pp. 656-663. IOS (2020).

[9] Cabalar, P., Diéguez, M., Schaub, T., Schuhmann, A.: Towards metric temporal answer set programming. TPLP 20(5), 783-798 (2020)

[10] Cabalar, P., Kaminski, R., Ostrowski, M., Schaub, T.: An ASP semantics for default reasoning with constraints. In: IJCAI. pp. 1015–1021. IJCAI/AAAI Press (2016).

[11] Crawford, J., Baker, A.: Experimental results on the application of satisfiability algorithms to scheduling problems. In: AAAI. pp. 1092-1097. AAAI (1994)

[12] Fisher, M., Gabbay, D., Vila, L. (eds.): Handbook of Temporal Reasoning in Artificial Intelligence, Elsevier (2005)

[13] Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T., Thiele, S.: Potassco User Guide. (2015), `http://potassco.org`

[14] Hustadt, U., Ozaki, A., Dixon, C.: Theorem proving for pointwise metric temporal logic over the naturals via translations. JAR 64(8), 1553-1610 (2020).

[15] Janhunen, T., Kaminski, R., Ostrowski, M., Schaub, T., Schellhorn, S., Wanko, P.: Clingo goes linear constraints over reals and integers. TPLP 17(5-6), 872-888 (2017).

[16] Kaminski, R., Romero, J., Schaub, T., Wanko, P.: How to build your own ASP-based system?! TPLP 23(1), 299-361 (2023).

[17] Kamp, J.: Tense Logic and the Theory of Linear Order. Ph.D. thesis, UCLA (1968)

[18] Lifschitz, V.: Answer Set Programming. Springer (2019).

[19] Mellarkod, V.: Integrating ASP and CLP systems: computing answer sets from partially ground programs. Ph.D. thesis, Texas Tech (2007)

[20] Pearce, D.: A new logical characterisation of stable models and answer sets. In: NMELP. pp. 57-70. Springer (1997).

[21] Son, T., Baral, C., Tuan, L.: Adding time and intervals to procedural and hierarchical control specifications. In: AAAI'04 pp. 92–97. AAAI Press (2004)

[22] Wałega, P., Cuenca Grau, B., Kaminski, M., Kostylev, E.: DatalogMTL: Computational complexity and expressive power. In: IJCAI. pp. 1886-1892. ijcai.org (2019)

[23] Wang, D., Hu, P., Wałega, P., Grau, B.: MeTeoR: Practical reasoning in Datalog with metric temporal operators. In: AAAI. pp. 5906-5913. AAAI (2022).