Reasoning about Action and Change in Timed Domains

Yuping Shen and **Guangrui Dang** and **Xishun Zhao**
Institution of Logic and Cognition
Department of Philosophy
Sun Yat-sen University, P.R. China 510275
{shyping,danggr,hsszxs}@mail.sysu.edu.cn

Abstract

Reasoning about Action and Change (RAC) has been an important topic since the early days of Artificial Intelligence research, numerous logical formalisms for RAC were proposed. Recently, two formalisms named $\mathcal{A}_T$ and AL^2_{TC} have extended RAC to so-called *timed domains*, i.e., domains where actions are required to be performed within a certain amount of time or after a certain amount of time has elapsed. However, we argue that $\mathcal{A}_T$ has a semantic defect in modeling timed domains and reasoning in AL^2_{TC} is relatively difficult due to its higher complexity. In this paper, we propose an extension of language $\mathcal{E}$ called *timed action language* $\mathcal{TE}$ for RAC in timed domains, which avoids the semantic defect of $\mathcal{A}_T$, in addition, a polynomial translation from $\mathcal{TE}$ to *Satisfiability Modulo Theory*(SMT) is given, s.t. $\mathcal{TE}$ can be easily implemented on top of SMT solvers. Moreover, it is surprising that the complexity for checking satisfiability and entailment in $\mathcal{TE}$ remain the same with $\mathcal{E}$, i.e., we have extended $\mathcal{E}$ to handle timed domains without increasing its complexity.

Introduction

Reasoning about Action and Change (RAC)(Y.Shoham 1987; Reiter 2001; van Harmelen, Lifschitz, and Porter 2007) has been an important topic since the early days of Artificial Intelligence research. Numerous logical formalisms and their variants have been proposed to address RAC problems, e.g., Situation Calculus(Mccarthy and Hayes 1969), Event Calculus(Kowalski and Sergot 1986), Nonmonotonic Causal Logic(McCain and Turner 1995), etc. There are also many efforts devoted to *action languages*, which are considered as high-level, natural language-based formalisms for RAC, e.g., action languages $\mathcal{A}$(Gelfond and Lifschitz 1993), $\mathcal{E}$(Kakas and Miller 1997), etc.

Recently, the so-called *timed action language* $\mathcal{A}_T$(Simon, Mallya, and Gupta 2005) and action logic AL^2_{TC}(Cabalar, Otero, and Pose 2000) extend the study of RAC to handle *timed domains*, i.e., domains where actions are required to be performed within a certain amount of time or after a certain amount of time has elapsed. E.g., a timed Yale Shooting domain requires action shoot to occur within some time after loading the gun, otherwise the scared bird will fly away and cannot be shot. Such domains are not only considered as toy problems in commonsense reasoning, but also widely studied as real-world applications, e.g., Assembly Plant(Clarke, Grumberg, and Peled 2000) and Rail Road Crossing Control(Alur 1999).

To express timed domains, the concepts of *clocks* and *clock constraints*(Alur 1999) are introduced in $\mathcal{A}_T$. E.g., saying that a flying bird can only be shot within 5 seconds can be expressed in $\mathcal{A}_T$ by an effect proposition: "$Shoot$ **causes** $\neg Alive$ **when** $x < 5$" where x is a clock and $x < 5$ a clock constraint. According to the semantics of $\mathcal{A}_T$, one can infer that the value proposition: "$\neg Alive$ **after** $Shoot$ **at** $x < 3$" is *true*, which means the bird is not alive if we shoot within 3 seconds after the scenario begins. However, since $\mathcal{A}_T$ is not a *narrative-based* logic, the exact time at which an action occurs or a fluent holds is *not* known, in other words, it lacks an absolute flow of time. E.g., in the above effect proposition, "$Shoot$ **at** $x < 3$" does not specify the exact time at which action $Shoot$ occurs, and even if we know "$\neg Alive$ **after** $Shoot$ **at** $x < 3$" is true, we still do not know the exact time when $\neg Alive$ holds. This leads to difficulties to handle real-world applications which require an absolute time structure to define synchronized clocks[1] for component interacting. It can be illustrated that $\mathcal{A}_T$ may give counter-natural results for such applications, as we shall see in Section 6.

Another work concerning RAC in timed domains is narrative-based action logic AL^2_{TC}(Cabalar, Otero, and Pose 2000), it has an absolute time structure thus the above defect can be avoided. However, AL^2_{TC} does not introduce explicit clocks, which are required by most timed applications, furthermore, reasoning in AL^2_{TC} is relatively difficult, e.g., deciding the satisfiability of a theory concerning the frame problem lies in Σ^P_2, which is a class widely believed higher than NP in the polynomial hierarchy. In fact, the author of (Cabalar, Otero, and Pose 2000) did not present methods for realize AL^2_{TC}.

In this work, we propose a formalism called timed action language $\mathcal{TE}$, which is an extension of the narrative-based language $\mathcal{E}$. It provides an absolute time structure thus avoids the semantic defect of $\mathcal{A}_T$, and adopts explicit clocks to suit time applications. Besides, a polynomial translation

[1] Clocks are said to be synchronized if they advance with the same rate.

from $\mathcal{TE}$ to *Satisfiability Modulo Theory*(SMT)(Nieuwenhuis 2009; Clark Barrett and Tinelli 2009) is given, s.t. $\mathcal{TE}$ can be easily implemented on top of efficiently developed, state-of-the-art SMT solvers like MathSAT(Bozzano et al. 2005), TSAT++(Armando et al. 2004). In addition, it is surprising to see that the complexity for checking satisfiability and entailment in $\mathcal{TE}$ remains the same with $\mathcal{E}$, i.e., NP-complete and coNP-complete respectively. In other words, the language $\mathcal{E}$ has been extended to handle timed domains without increasing its complexity.

The rest of the paper is organized as follows. Section 2 recalls some background knowledge about $\mathcal{E}$ and in Section 3 the syntax and semantics of $\mathcal{TE}$ are presented. Section 4 contains the polynomial time translation from $\mathcal{TE}$ theory to SMT formula and Section 5 provides a brief complexity analysis of $\mathcal{TE}$. We give some analysis of related work $\mathcal{A}_T$ and AL^2_{TC} in Section 6. Conclusion and future work are presented in Section 7.

Action Language $\mathcal{E}$

The Syntax of $\mathcal{E}$

An action language $\mathcal{E}$ (Kakas and Miller 1997) is a 3-tuple $\langle \mathbb{N}, \Delta, \Phi \rangle$, in which $\mathbb{N}$ is the set of natural numbers called *time points*, Δ is a non-empty set of *action symbols*, Φ is a non-empty set of *fluent symbols*. A *fluent literal* is a fluent symbol possibly preceded by $\neg$. There are three kinds of propositions in $\mathcal{E}$:

- C-proposition of the form

$$A \textbf{ initiates } F \textbf{ if } C, \text{ or } A \textbf{ terminates } F \textbf{ if } C$$

  in which $A \in \Delta$, $F \in \Phi$ and C a set of fluent literals;
- H-proposition of the form A **happens-at** T in which $A \in \Delta, T \in \mathbb{N}$;
- T-proposition of the form L **holds-at** T in which L is a fluent literal and $T \in \mathbb{N}$.

A *domain description* (or *theory*) D in $\mathcal{E}$ is a finite set of c-propositions, h-propositions and t-propositions. We illustrate the syntax of $\mathcal{E}$ by giving a classic example below.

Example: Yale Shooting in $\mathcal{E}$

Let $\mathcal{E} = \langle \mathbb{N}, \{Shoot, Load\}, \{Loaded, Alive\} \rangle$, a theory D_{YS} of Yale Shooting consists of the following propositions:

$$Load \textbf{ initiates } Loaded$$

$$Shoot \textbf{ terminates } Alive \textbf{ if } \{Loaded\}$$

$$Alive \textbf{ holds-at } 0,\ \neg Loaded \textbf{ holds-at } 0$$

$$Load \textbf{ happens-at } 1,\ Shoot \textbf{ happens-at } 2$$

The semantics of $\mathcal{E}$

To give the meaning of a theory in $\mathcal{E}$, we shall introduce the semantics of $\mathcal{E}$. Let C be a set of fluent literals and $T \in \mathbb{N}$, an *interpretation* of a theory in $\mathcal{E}$ is a mapping $H : \Phi \times \mathbb{N} \mapsto \{True, False\}$, we say H *satisfies* C at T, denoted by $H \models_T C$, if for each fluent symbol $F \in C$, $H(F,T) = True$ and for each $\neg F \in C$, $H(F,T) = False$. Let D be a theory in $\mathcal{E}$, $F \in \Phi$, a time point $T \in \mathbb{N}$ is said to be an *initiation-point* (resp. *termination-point*) for F in H w.r.t. D, if there is an h-proposition A **happens-at** T and a c-proposition A **initiates**(resp.**terminates**) F **if** C in D, s.t., $H \models_T C$.

Definition 1 (Model of $\mathcal{E}$)**.** *A* model *of a theory D in $\mathcal{E}$ is an interpretation H s.t. for every $F \in \Phi$ and for every time points $T_1 < T_3$, the following conditions hold:*

1. *For each t-proposition in D of the form "F (resp. $\neg F$) **holds-at** T", $H(F,T) = True$ (resp. $False$).*
2. *If there is no initiation or termination point T_2 for F in H w.r.t. D s.t. $T_1 \leq T_2 < T_3$, then $H(F,T_1) = H(F,T_3)$, i.e., fluent values remain unchanged if not affected by any action.*
3. *If T_1 is an initiation (resp. termination) point for F in H w.r.t. D and there is no termination (resp. initiation) point T_2 for F in H w.r.t. D s.t. $T_1 < T_2 < T_3$, then $H(F,T_3) = True$ (resp. $False$), i.e., initiating (resp. terminating) a fluent sets its value to $True$ (resp. $False$).*

A theory D in $\mathcal{E}$ is *satisfiable* if it has a model, a t-proposition L **holds-at** T is called an *entailment* of D, written as $D \models L$ **holds-at** T, if it holds under every model of D. The semantics of $\mathcal{E}$ succinctly solves the *frame problem*, i.e., to describe what is not changed during the evolution of a world. Consider the theory D_{YS}, it is not hard to see it is satisfiable, in all its models (only a unique one indeed), $Alive$ and $\neg Loaded$ hold at time point 1 by condition 2 in Definition 1. At time point 2 $Loaded$ holds since it is initiated by $Load$ at time point 1 by condition 3, similarly, $Alive$ does not hold at time point 3 since it is terminated by $Shoot$. In fact, $D_{YS} \models \neg Alive$ **holds-at** T for any $T \geq 3$ and $D_{YS} \models Loaded$ **holds-at** T for any $T \geq 2$.

Timed Action Language $\mathcal{TE}$

Now we present the main contribution of this paper, i.e., an extension of $\mathcal{E}$ called timed action language $\mathcal{TE}$ to handle timed domains. The notions of clocks and clock constraints are central to express such domains, which originated from the theory of *timed automata*(Alur 1999), and later widely accepted in modeling timed systems. The language $\mathcal{A}_T$ also adopted these notions and we follow the line to invent $\mathcal{TE}$.

The Syntax of $\mathcal{TE}$

A *timed action language* $\mathcal{TE}$ is a 5-tuple $\langle \mathbb{N}, \Delta, \Phi, \Theta, \mathcal{B}(\Theta) \rangle$, where $\mathbb{N}$ is the set of natural numbers called *time points*, Δ is a non-empty set of *action symbols*, Φ is a non-empty set of *fluent symbols*, Θ is a non-empty set of variables over $\mathbb{N}$ called *clocks*, $\mathcal{B}(\Theta)$ is the set of all *clock constraints* built on Θ, each clock constraint $\psi \in \mathcal{B}(\Theta)$ is an expression of the form $x \bowtie n$ or $x - y \bowtie n$, where $x, y \in \Theta$, $\bowtie \in \{\leq, <, =, >, \geq\}$ and $n \in \mathbb{N}$. The propositions in $\mathcal{TE}$ are defined as follows:

- C-proposition is either of the form

$$A \textbf{ initiates } F \textbf{ resets } \lambda \textbf{ if } C \textbf{ when } \Psi$$

  or of the form

$$A \textbf{ terminates } F \textbf{ resets } \lambda \textbf{ if } C \textbf{ when } \Psi$$

where $A \in \Delta$, $F \in \Phi$, $\lambda \subseteq \Theta$ a set of clocks, C a set of fluent literals and $\Psi \subseteq \mathcal{B}(\Theta)$ a set of clock constraints;

- H-proposition of the form A **happens-at** T in which $A \in \Delta, T \in \mathbb{N}$;
- T-proposition of the form L **holds-at** T in which L is a fluent literal and $T \in \mathbb{N}$.

Intuitively, the c-propositions in $\mathcal{TE}$ introduce a new kind of action effect: clock resetting, together with a new kind of action precondition: clock constraint. Simply speaking, resetting a clock is to start counting its ticks, and executing an action requires satisfying related clock constraints in addition to fluent preconditions. Please note that the definitions of h- and t-propositions remain the same w.r.t. $\mathcal{E}$. Similarly, a domain description or theory in $\mathcal{TE}$ is a finite set of c-, h-, t-propositions, we give an example below.

Example: Timed Yale Shooting in $\mathcal{TE}$

Let $\mathcal{TE} = \langle \mathbb{N}, \{Load, Shoot\}, \{Loaded, Alive\}, \{x\}, \mathcal{B}(\{x\})\rangle$, a $\mathcal{TE}$ theory D_{TYS} of timed Yale Shooting(Simon, Mallya, and Gupta 2005; Cabalar, Otero, and Pose 2000) consists of the following propositions, in which keywords **resets** λ, **if** C, **when** Ψ are omitted if λ, C, Ψ are empty:

$$Shoot \textbf{ terminates } Alive \textbf{ if } \{Loaded\} \textbf{ when } \{x < 5\}$$

$$Load \textbf{ initiates } Loaded \textbf{ resets } \{x\}$$

$$Alive \textbf{ holds-at } 0, \neg Loaded \textbf{ holds-at } 0$$

$$Load \textbf{ happens-at } 1, Shoot \textbf{ happens-at } 3$$

This example can be considered as an extension of the classic Yale Shooting, in the sense that a bird can only be shot within 5 seconds after loading the gun.

The Semantics of $\mathcal{TE}$

To define proper semantics for $\mathcal{TE}$, we shall first give meanings for clocks and clock constraints, which are originated from (Alur 1999). The readings of clocks are given by a *clock assignment* ν, which maps every $x \in \Theta$ to a value in $\mathbb{N}$. We say that ν satisfies a clock constraint ψ, written as $\nu \models_c \psi$, if ψ holds under ν according to the standard arithmetic semantics. If there exists such a ν, then ψ is also called *satisfiable* and ν is a *solution* to ψ.

For modeling clock advancing and resetting, we introduce the following definitions. Given $\delta \in \mathbb{N}$, let $\nu + \delta$ be the clock assignment that maps every $x \in \Theta$ to $\nu(x) + \delta$, and for $\lambda \subseteq \Theta$, let $\nu[\lambda := 0]$ be the clock assignment that maps each $x_i \in \lambda$ to 0 and every clock $x_j \in \Theta \backslash \lambda$ remains unchanged w.r.t. ν. In particular, ν_0 is a clock assignment that maps every clock to 0. Given a set of clock constraints Ψ, we say $\nu \models_c \Psi$ if for each $\psi \in \Psi$, $\nu \models_c \psi$. If there exists such a ν, we say Ψ is satisfiable. E.g., let $\Theta = \{x, y\}$, $\nu' = \{\langle x, 5\rangle \langle y, 1\rangle\}$ satisfies the set of clock constraints $\{x \geq 3, y < 2\}$, $\nu' + 2 = \{\langle x, 7\rangle, \langle y, 3\rangle\}$, $\nu'[\{x\} := 0] = \{\langle x, 0\rangle, \langle y, 1\rangle\}$ and $\nu_0 = \{\langle x, 0\rangle, \langle y, 0\rangle\}$.

Essentially, a clock assignment gives clock values at one time point, this is not enough to describe the evolution of clocks on an absolute time structure, which normally includes infinite time points. So we introduce *clock interpretation* for a $\mathcal{TE}$ theory, which is a mapping K that maps each natural number to a clock assignment and $K(0) = \nu_0$. Intuitively speaking, K gives clock values along with a time line $\mathbb{N}$, and at the beginning(i.e., time point 0) all clock readings are 0.

Having defined clock interpretation, we follow $\mathcal{E}$ to introduce *fluent interpretation*, which is a mapping $H : \Phi \times \mathbb{N} \mapsto \{True, False\}$, moreover, a set of fluent literals C is said to be satisfied by H at $T \in \mathbb{N}$, denoted by $H \models_T C$, if for each fluent symbol $F \in C$, $H(F, T) = True$ and for each $\neg F \in C$, $H(F, T) = False$. Now an *interpretation* of a $\mathcal{TE}$ theory is defined as a pair $\langle H, K\rangle$, where H is a fluent interpretation and K is a clock interpretation.

Similar to $\mathcal{E}$, language $\mathcal{TE}$ adopts the notions of initiation and termination point, furthermore, the concept of *resetting point* is invented. Let D be a $\mathcal{TE}$ theory, $\langle H, K\rangle$ be an interpretation of $\mathcal{TE}$ and $F \in \Phi$, a time point $T \in \mathbb{N}$ is said to be an *initiation point* (resp. *termination point*) of F in $\langle H, K\rangle$ w.r.t. D, if:

- there exist both an h-proposition A **happens-at** T, and a c-proposition A **initiates** F **resets** λ **if** C **when** Ψ (resp. A **terminates** F **resets** λ **if** C **when** Ψ) in D s.t.,
- $H \models_T C$, and $K(T) \models_c \Psi$. Furthermore, if the set of clocks λ is non-empty, then T is also called a *resetting point* for λ in $\langle H, K\rangle$ w.r.t. D.

Definition 2 (Model of $\mathcal{TE}$)**.** *Let D be a $\mathcal{TE}$ theory, an interpretation $\langle H, K\rangle$ is called a* model *for D, if for each $F \in \Phi$ and time points $T_1, T_2 \in \mathbb{N}$ with $T_1 < T_2$, the following conditions hold:*

1. *For each t-proposition in D of the form "F (resp. $\neg F$) **holds-at** T", $H(F, T) = True$ (resp. $False$).*
2. *If there is no initiation point or termination point $T \in \mathbb{N}$ for F in $\langle H, K\rangle$ w.r.t. D s.t. $T_1 \leq T < T_2$, then $H(F, T_2) = H(F, T_1)$.*
3. *If T_1 is an initiation-point (resp. termination-point) for F in $\langle H, K\rangle$ w.r.t. D, and there exists no termination-point (resp. initiation-point) T for F in $\langle H, K\rangle$ w.r.t. D s.t. $T_1 < T < T_2$, then $H(F, T_2) = True$(resp. $False$).*
4. *If there exists no resetting point T for any clocks s.t. $T_1 \leq T < T_2$, then $K(T_2) = K(T_1) + (T_2 - T_1)$.*
5. *If T_1 is a resetting point for sets of clocks $\lambda_1, \ldots, \lambda_n$ in $\langle H, K\rangle$ w.r.t. D, and there exists no resetting point T for any clocks s.t. $T_1 < T < T_2$, then $K(T_2) = K(T_1)[\lambda_1 \cup \ldots \cup \lambda_n := 0] + (T_2 - T_1)$.*

The concepts of *satisfiability* and *entailment* in $\mathcal{TE}$, are defined as the same in $\mathcal{E}$, i.e., a theory D in $\mathcal{TE}$ is *satisfiable* if it has a model, a t-proposition L **holds-at** T is called an *entailment* of D, written as $D \models L$ **holds-at** T, if it holds under every model of D.

It is not hard to see the major modification of the semantics of $\mathcal{TE}$ w.r.t. $\mathcal{E}$ is the characterization of the clock behavior. Simply speaking, all clocks start with 0 and run with the same rate. The reading of a clock is the elapsed time since the last time it was reset. We illustrate the mechanism in Figure 1. In the figure, clocks x and y start with 0 and run with the same rate, i.e, they are synchronized. At time point 3, x get reset and then at 4 its reading is 1, while clock y was not affected, it just keeps running.

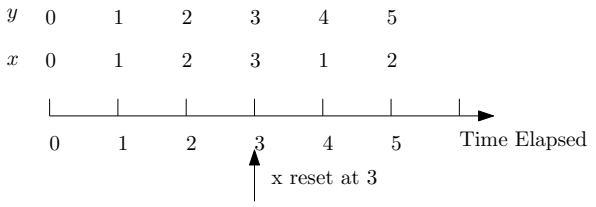

Figure 1: Clock Mechanism in $\mathcal{TE}$

Consider the timed Yale Shooting theory D_{TYS}, according to the semantics of $\mathcal{TE}$, it is easy to see it has a unique model, as shown in Figure 2. Observe that action $Load$ not only makes the gun $Loaded$, but also starts counting the elapsed time, $Shoot$ happens at time point 3 fulfils clock constraint $x < 5$, with $Loaded$ holds, the bird is then no longer $Alive$ from time point 4 forth.

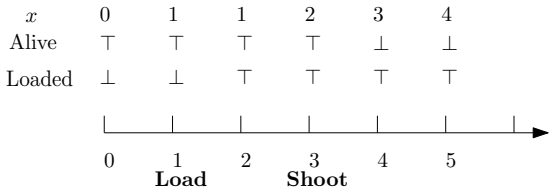

Figure 2: The model of timed Yale Shooting theory D_{TYS}

In next section we concern the implementation of $\mathcal{TE}$. The authors of (Dimopoulos, Kakas, and Michael 2004) proposed a polynomial translation from a theory in $\mathcal{E}$ to answer set programming(ASP), s.t. $\mathcal{E}$ can be implemented on top of answer set programming solvers. Similarly, we propose a polynomial translation from a $\mathcal{TE}$ theory to a SMT formula, s.t. $\mathcal{TE}$ can be implemented on top of SMT solvers.

A SMT Approach to Implement $\mathcal{TE}$

Roughly speaking, SMT concerns the integration of a background theory solver and a SAT engine, the former handles constraints from the background theory while the latter performs large scale search without knowing the semantics of the constraints. SMT has gained a great deal of interest in the last few years, and has been successfully applied to a number of optimization/verification/planning problems, e.g., (Audemard et al. 2002; Janhunen, Niemelä, and Sevalnev 2009). Our approach is to construct a polynomial time translation from a $\mathcal{TE}$ theory into a SMT formula, by computing the translated formula via SMT solvers, the original theory can be solved. Compared to implementations based on SAT or ASP, SMT solvers handle background constraints more efficiently and concisely. The background theory of SMT involved here is so-called *linear arithmetic*, which is quite well supported by SMT solvers like MathSAT and TSAT++.

SMT with Linear Arithmetic Relations

The following linear arithmetic concepts come from (Jonsson and Bäckström 1998) and (Bozzano et al. 2005). Let V be a set of *boolean variables* and let X be a set of variables defined on $\mathbb{R}$. A *linear relation* is of the form $\alpha \otimes r$, where α is a linear expression (polynomial with degree 1) over X, $r \in \mathbb{R}$ and $\otimes \in \{<, \leq, =, \neq, \geq, >\}$. Notice that all clock constraints are linear relations, but not vice versa.[2]

An *SMT-LA-formula*(or *formula* for short) is a combination of boolean variables from V and linear relations on X through standard logic connectives $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$. E.g., $((3x_1 = 6) \rightarrow v_1) \wedge (v_2 \vee \neg(x_2 < 3))$ is a formula in which v_1, v_2 are boolean variables and x_1, x_2 are real variables. An *SMT-interpretation* $\mathcal{I}$ for a formula is a mapping that maps boolean and real variables to boolean and real values respectively and preserves constant values. $\mathcal{I}$ is called a model for a formula, if the formula holds under $\mathcal{I}$ according to the standard logical and arithmetic semantics. A formula is *satisfiable* if it has at least one model. E.g., the above formula is satisfiable, a SMT-interpretation $\mathcal{I}$ that maps v_1 to $\top$ and $x_2 = 4$ is a model for it.

Translating $\mathcal{TE}$ theory to SMT Formula

Before constructing the translation, it is necessary to mention that although a model of $\mathcal{TE}$ theory theoretically concerns infinite time points(i.e., over $\mathbb{N}$), it can be showed only finitely many time points are needed to represent the model, these time points are called *witness points*. Let TP_D be the set of time points occurring in a theory D, a set of witness points WP_D of D, is the set $\{0\} \cup TP_D \cup \{\kappa\}$, in which κ is a natural number s.t. $\kappa \neq 0$ and $\kappa > T$ for every $T \in TP_D$. The definition and proposition below formally state the above fact, which is crucial to the forthcoming translation.

Definition 3 (Succinct Representation)**.** *Let $\langle H, K\rangle$ be an interpretation of $\mathcal{TE}$ theory D, WP_D a set of witness points of D. Then $\langle H_{|WP_D}, K_{|WP_D}\rangle$ is called a* succinct representation *of $\langle H, K\rangle$, in which $H_{|WP_D}$ and $K_{|WP_D}$ is the restriction of H and K on WP_D respectively.*

Proposition 1. *Let $\langle H, K\rangle$ be a model of $\mathcal{TE}$ theory D, $\langle H_{|WP_D}, K_{|WP_D}\rangle$ a* succinct representation *of it. Then there exists a fast algorithm $Extr$, s.t. for any $T \in \mathbb{N}, F \in \Phi$, $H(F,T)$ and $K(T)$ can be returned by $Extr(H_{|WP_D}, K_{|WP_D}, T)$ in polynomial time.*

In other words, when referring to a model in $\mathcal{TE}$, it is enough to give only the fluent/clock values on WP_D. This allows us to describe $\mathcal{TE}$ models by SMT models.

Now we present a translation from a theory D of $\mathcal{TE} = \langle \mathbb{N}, \Delta, \Phi, \Theta, \mathcal{B}(\Theta)\rangle$ to a SMT formula MF_D. For simplicity, assume WP_D consists of a complete increasing sequence $\{0, 1, \ldots, n-1, n\}$.[3] We first construct the alphabet for MF_D:

- Boolean Variables (b.v.) . For each $A \in \Delta$, introduce b.v. $A_0, \ldots, A_{n-1}$, where each A_i stands for an action occurrence A at i. For each $F \in \Phi$, introduce b.v. $F_0, \ldots, F_n$

[2] If we rule out $\{<, >\}$ from $\mathcal{TE}$, then a restricted SMT instance called Difference Logic(Janhunen, Niemelä, and Sevalnev 2009) is sufficient for our translation.

[3] This will not affect the generality, since we can introduce some redundant t-propositions to D to fill the gaps in WP_D.

and $F'_0, \ldots, F'_n$, where each F_i means fluent F holds at i, and F'_i means F does not hold at i. In addition, introduce b.v. $INI_1^F, \ldots, INI_n^F$, $TMN_1^F, \ldots, TMN_n^F$, where INI_i^F(resp. TMN_i^F) means fluent F is "going to hold (resp. not hold)" at i. Moreover, for each clock $x \in \Theta$, introduce b.v. $RST_0^x, \ldots, RST_{n-1}^x$ where each RST_i^x stands for a reset of clock x at i.

- Real variables. For each clock $x \in \Theta$, introduce real variables $x_0, \ldots, x_n$, where each x_i will store the value of clock x at i.

The expected SMT formula MF_D is a conjunction of the following formulas:

- Formulas for specifying initial clock values and h-propositions.

$$\bigwedge_{x\in\Theta}(x_0 = 0) \qquad \bigwedge_{A \textbf{ happens-at } T\in D} A_T \tag{1}$$

$$\bigwedge_{A\in\Delta, A \textbf{ happens-at } T\notin D, T\in\{0,\ldots,n-1\}} \neg A_T \tag{2}$$

- Formulas for specifying t-propositions.

$$\bigwedge_{F \textbf{ holds-at } T\in D} F_T \qquad \bigwedge_{\neg F \textbf{ holds-at } T\in D} F'_T \tag{3}$$

- Formulas for describing executions of c-propositions.

  For a c-proposition $p \in D$ of the form

  $$A \textbf{ initiates } F \textbf{ resets } \lambda \textbf{ if } C \textbf{ when } \Psi$$

  or

  $$A \textbf{ terminates } F \textbf{ resets } \lambda \textbf{ if } C \textbf{ when } \Psi$$

  let $exe(p, i), i \in \{0, \ldots, n-1\}$ denote the following formula:

  $$A_i \wedge \bigwedge C_i \wedge \bigwedge \Psi[x/x_i, \ldots, z/z_i] \text{ in which:}$$

  (i) C_i is the set of boolean variables obtained from C by replacing each negative literal $\neg F \in C$ by F'_i and each $F \in C$ by F_i;

  (ii) $\Psi[x/x_i, \ldots, z/z_i]$ denotes the set of clock constraints obtained from Ψ by replacing every occurrence of clocks $x, \ldots, z$ by real variables $x_i, \ldots, z_i$ respectively. In particular, for $i \geq 1$, clock constraints of the form $x = 0$ in Ψ is replaced by b.v. RST_i^x. Intuitively, $exe(p, i)$ describes an execution of a c-proposition p at stage i. Now let D_F^{INI}(resp. D_F^{TMN}) be the set of all c-propositions in D for initiating(resp. terminating) fluent F, and let D_x^{RST} be the set of all c-propositions in D for resetting clock x. Formulas for describing executions of c-propositions in D are then given below.

$$\bigwedge_{F\in\Phi}\Big(\bigwedge_{i\in\{0,\ldots,n-1\}}(INI_{i+1}^F \leftrightarrow \bigvee_{p\in D_F^{INI}} exe(p,i))\Big) \tag{4}$$

$$\bigwedge_{F\in\Phi}\Big(\bigwedge_{i\in\{0,\ldots,n-1\}}(TMN_{i+1}^F \leftrightarrow \bigvee_{p\in D_F^{TMN}} exe(p,i))\Big) \tag{5}$$

$$\bigwedge_{x\in\Theta}\Big(\bigwedge_{i\in\{0,\ldots,n-1\}}(RST_i^x \leftrightarrow \bigvee_{p\in D_x^{RST}} exe(p,i))\Big) \tag{6}$$

Formula (4)(resp. (5)) says that a fluent $F \in \Phi$ is initiated (resp. terminated) at stage i, if and only if at least one of the c-propositions for initiating (resp. terminating) F is executed at stage $i-1$. Similarly, formula (6) describes the clock resetting at each stage. Please note that if $D_F^{INI} = \emptyset$, then formula (4) degenerates to

$$\bigwedge_{F\in\Phi}\Big(\bigwedge_{i\in\{1,\ldots,n\}}\neg INI_i^F\Big)$$

which means if there exists no c-proposition for initiating F, then F will never be initiated. This also holds for the cases $D_F^{TMN} = \emptyset$ and $D_x^{RST} = \emptyset$.

- Formulas for generating fluent effects and describing persistence.

$$\bigwedge_{F\in\Phi}\Big(\bigwedge_{i\in\{1,\ldots,n\}}(F_i \leftrightarrow INT_i^F \vee (F_{i-1} \wedge \neg TMN_i^F))\Big) \tag{7}$$

$$\bigwedge_{F\in\Phi}\Big(\bigwedge_{i\in\{1,\ldots,n\}}(F'_i \leftrightarrow TMN_i^F \vee (F'_{i-1} \wedge \neg INI_i^F))\Big) \tag{8}$$

Formula (7) says that a fluent F holds at stage i, if and only if it is initiated at stage i or it holds at stage $i-1$ and was not terminated. Formula (8) is similar.

- Formulas for specifying clock values.

$$\bigwedge_{x\in\Theta}\Big(\bigwedge_{i\in\{1,\ldots,n\}}(RST_{i-1}^x \rightarrow (x_i = 1))\Big) \tag{9}$$

$$\bigwedge_{x\in\Theta}\Big(\bigwedge_{i\in\{1,\ldots,n\}}(\neg RST_{i-1}^x \rightarrow (x_i = x_{i-1} + 1))\Big) \tag{10}$$

$$\bigwedge_{x\in\Theta}\Big(\bigwedge_{i\in\{1,\ldots,n\}}(RST_{i-1}^x \vee \neg RST_{i-1}^x) \wedge \neg(RST_{i-1}^x \wedge \neg RST_{i-1}^x)\Big) \tag{11}$$

Formulas (9), (10) and (11) say that if a clock is rest then its value becomes 1 in the next time point, otherwise the value is increased by 1.

- Formulas for generating complete knowledge at the beginning.

$$\bigwedge_{F\in\Phi, F \textbf{ holds-at } 0\notin D, \neg F \textbf{ holds-at } 0\notin D} ((F'_0 \vee F_0) \wedge \neg(F'_0 \wedge F_0)) \tag{12}$$

- Formulas for eliminating inconsistent models.

$$\bigwedge_{F\in\Phi}\Big(\bigwedge_{i\in\{0,\ldots,n\}}(\neg(F'_i \wedge F_i))\Big) \tag{13}$$

The construction of MF_D is now complete, i.e., MF_D is the conjunction of the above numbered SMT formulas. Observe that each step of the translation/construction can be done in polynomial time, thus the whole construction is polynomial.

Proposition 2. *The presented translation from a $\mathcal{TE}$ theory to a SMT formula MF_D is polynomial.*

Furthermore, we have the following theorem:

Theorem 4. *Let D be a $\mathcal{TE}$ theory and MF_D be its translated SMT formula over witness points WP_D. Then $\langle H_{|WP_D}, K_{|WP_D}\rangle$ is a succinct representation of a model $\langle H, K\rangle$ of D, if and only if MF_D has a SMT model $\mathcal{I}$ s.t. for every $F \in \Phi$, every $T \in WP_D$ and every $x \in \Theta$,*

1. *$H(F,T) = True$ (resp. $False$) if and only if $\mathcal{I}(F_T) = \top$ (resp. $\mathcal{I}(F'_T) = \top$),*
2. *$K(T)(x) = \mathcal{I}(x_T)$.*

The proof of Theorem 4 can be done by induction on the size of WP_D and show by cases. The theorem gives completeness and soundness of the polynomial translation, and models of $\mathcal{TE}$ theory can be easily decoded from models returned by SMT solvers.

Computational Complexity Analysis

In this section we shall briefly show that the reasoning complexity of $\mathcal{TE}$ remains the same with $\mathcal{E}$, this is surprising since we have extended $\mathcal{E}$ to $\mathcal{TE}$ to handle timed domains. We first present a complexity result of SMT satisfiability checking.

Theorem 5 (SMT-LA SAT)**.** *Checking if a given SMT-LA formula is satisfiable is NP-complete.*

The NP-hardness of SMT-LA SAT is obvious. For the NP membership, according to the main result of (Jonsson and Bäckström 1998), i.e., checking satisfiability for conjunction of linear expressions is in P, a non-deterministic polynomial time procedure for deciding SMT-LA-SAT can be constructed as follows: for a SMT-LA formula γ, guess a pure boolean interpretation $\mathcal{I}$, which regards linear expressions in γ also as boolean literals. Quickly check if $\mathcal{I}$ satisfies γ, if so, collect linear expressions that were assigned true under $\mathcal{I}$, form them into a conjunction and apply the polynomial algorithm in (Jonsson and Bäckström 1998) to check if the conjunction is satisfiable, if so the γ is satisfiable.

Since SMT-LA SAT is in NP, together with Proposition 2, Theorem 4, 5, it implies that checking satisfiability of a $\mathcal{TE}$ theory is also in NP. Furthermore, since $\mathcal{TE}$ is a strict extension of $\mathcal{E}$ and it has been prove that checking satisfiability in $\mathcal{E}$ is generally NP-hard(Dimopoulos, Kakas, and Michael 2004), it follows that:

Theorem 6 ($\mathcal{TE}$ SAT)**.** *Deciding the satisfiability of a $\mathcal{TE}$ theory D is NP-complete.*

This is in some sense surprising since $\mathcal{TE}$ is a strict extension of $\mathcal{E}$, however the complexity for satisfiability checking remains the same. It is worth to point out that the result strongly depend on the fact that all clock values are given at the beginning, i.e., set to 0, otherwise we do not know if the problem still in NP. The complexity for checking entailment is closely related to checking satisfiability, since F **holds-at** T is an entailment of D if and only if $D \cup \{\neg F$ **holds-at** $T\}$ is unsatisfiable, thus we have:

Corollary 1. *Deciding whether a t-proposition is entailed by a $\mathcal{TE}$ theory D is coNP-complete.*

It can also be proved in $\mathcal{TE}$ a similar result to $\mathcal{E}$:

Proposition 3. *A knowledge complete $\mathcal{TE}$ theory D has at most one model and checking satisfiability for D is in P.*

In the proposition, a theory D is called *knowledge complete*, if for each $F \in \phi$, either F **holds-at** 0 or $\neg F$ **holds-at** 0 in D.

Related Work and Discussion

Besides the timed logics we mentioned in this paper, there exist variants of Situation Calculus and Event Calculus, which could be considered as formalisms for reasoning about time, e.g., Pinto's Situation Calculus(Pinto and Reiter 1995). They are first-order logics and thus generally more expressive than propositional ones, however, to the best of our knowledge, they have not been tailored for timed domains(Alur 1999) so far. Due to space limitations, we mainly restrict our discussion on $\mathcal{A}_T$ and AL^2_{TC}.

Timed Action Language $\mathcal{A}_T$

As we mentioned before, the defect of $\mathcal{A}_T$(Simon, Mallya, and Gupta 2005) due to the lack of an absolute flow of time. Simply speaking, without an absolute time structure, $\mathcal{A}_T$ is unable to specify clocks values thus $\mathcal{A}_T$ may provide unnatural and unexpected results for timed domains that require synchronized clocks. We informally illustrate this via a variant of timed Yale Shooting domain, which considers the shooting scenario in a room equipped with an automated door. The door can only be opened from outside, and it automatically closes in 10 seconds if someone opens it. At the beginning the bird is in the room and the gunman is out. Also, shooting has to be performed within 5 seconds after loading, otherwise the scared bird flies away through the window. The theory in $\mathcal{A}_T$ is given below, which uses two clocks x and y:

$$Shoot \textbf{ causes } \neg Alive \textbf{ if } Loaded, In \textbf{ when } x < 5$$
$$Load \textbf{ causes } Loaded \textbf{ resets } x \textbf{ if } \neg Loaded, In$$
$$GetIn \textbf{ causes } In \textbf{ resets } y \textbf{ if } \neg In$$
$$GetOut \textbf{ causes } \neg In \textbf{ if } In \textbf{ when } y \leq 10$$
$$\textbf{Initially } Alive, \neg In, \neg Loaded$$

In order to kill the bird without being trapped in the room, the gunman has to perform a series of actions which consistent with the clock constraints: $GetIn$, $Load$, $Shoot$ and $GetOut$. The following value proposition describes such a case:

$$\neg Alive, \neg In \textbf{ after}$$
$$GetIn; Load \textbf{ at } y < 2; Shoot \textbf{ at } x < 3; GetOut \textbf{ at } y < 8$$

So, under well-defined semantics, the value proposition should be *true* w.r.t. the above theory in $\mathcal{A}_T$. However, this is NOT the case. The value proposition is assigned truth value *unknown*, and the theory(augmented with the value proposition) has no model at all. An intuitive explanation is that $\mathcal{A}_T$ does not know the exact running rates of the clocks and just simply rejects the value proposition. Consequently, $\mathcal{A}_T$ cannot describe many real-world timed domains correctly, while $\mathcal{TE}$ gives nice solutions to such applications.

Action Logic AL^2_{TC}

The action logic AL^2_{TC}(Cabalar, Otero, and Pose 2000) is narrative-based thus avoids the above semantic defect of $\mathcal{A}_T$. However, reasoning in AL^2_{TC} is relatively hard, it can be proved that deciding the satisfiability of a theory concerning the frame problem in AL^2_{TC} at least lies in Σ^P_2, i.e., a class believed higher than NP in the polynomial hierarchy. Fairly speaking, this may not be seriously considered as a "defect" from a theoretical view, since AL^2_{TC} also naturally provides a solution to the ramification problem, which means it is more expressive than $\mathcal{TE}$ and $\mathcal{A}_T$. But from a practical point of view, solving Σ^P_2 problems is widely considered more difficult than problems in NP. So far, no polynomial translation from AL^2_{TC} to SMT is known, and no implementation method is mentioned in (Cabalar, Otero, and Pose 2000). Another issue is that AL^2_{TC} does not introduce explicit clocks, which makes it less flexible in describing timed domains. Informally speaking, explicit clocks allow a concise representation of clock constraints and provide natural and clear information of the application.

Conclusion and Future Work

In this paper, we have extended action language $\mathcal{E}$ to describe timed domains without increasing its complexity. The resulting language is called $\mathcal{TE}$. Compared with related work, $\mathcal{TE}$ overcomes a semantic defect of $\mathcal{A}_T$ and can be easily implemented on top of SMT solvers, while another similar logic, AL^2_{TC} is considered computationally harder and has no so-called explicit clocks. We expect that besides manage toy problems in commonsense reasoning, $\mathcal{TE}$ can play a more important role in real-world applications, e.g., serves as a verification/model checking tool, etc. The future work will cover the implementation of $\mathcal{TE}$, and extending it to deal with more complicated domains, e.g., timed domains involving ramification problem.

Acknowledgements

The authors would like to thank Nadia Creignou, Camilla Schwind and anonymous reviewers for their helpful comments. Particularly, the first author would like to thank Vladimir Lifschitz for his suggestions at KR Doctoral Consortium. This research has been partially supported by NSFC under Grant 60970040, 60970044, 60736020 and by Ministry of Eduction of China under Grant 05JJD72040122.

References

Alur, R. 1999. Timed automata. In Halbwachs, N., and Peled, D., eds., *CAV*, volume 1633 of *Lecture Notes in Computer Science*, 8–22. Springer.

Armando, A.; Castellini, C.; Giunchiglia, E.; and Maratea, M. 2004. A sat-based decision procedure for the boolean combination of difference constraints. In *SAT2004*.

Audemard, G.; Cimatti, A.; Kornilowicz, A.; and Sebastiani, R. 2002. Bounded model checking for timed systems. In *FORTE*, 243–259.

Bozzano, M.; Bruttomesso, R.; Cimatti, A.; Junttila, T. A.; van Rossum, P.; Schulz, S.; and Sebastiani, R. 2005. Mathsat: Tight integration of sat and mathematical decision procedures. *Journal of Automated Reasoning* 35(1-3):265–293.

Cabalar, P.; Otero, R. P.; and Pose, S. G. 2000. Temporal constraint networks in action. In Horn, W., ed., *ECAI*, 543–547. IOS Press.

Clark Barrett, Roberto Sebastiani, S. A. S., and Tinelli, C. 2009. Satisfiability modulo theories. In *Handbook of Satisfiability*. IOS Press.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 2000. *Model Checking*. MIT Press.

Dimopoulos, Y.; Kakas, A. C.; and Michael, L. 2004. Reasoning about actions and change in answer set programming. In *LPNMR*, 61–73.

Gelfond, M., and Lifschitz, V. 1993. Representing action and change by logic programs. *Journal of Logic Programming* 17(2/3/4):301–321.

Janhunen, T.; Niemelä, I.; and Sevalnev, M. 2009. Computing stable models via reductions to difference logic. In *LPNMR*, 142–154.

Jonsson, P., and Bäckström, C. 1998. A unifying approach to temporal constraint reasoning. *Artif. Intell.* 102(1):143–155.

Kakas, A. C., and Miller, R. 1997. A simple declarative language for describing narratives with actions. *Journal of Logic Programming.* 31(1-3):157–200.

Kowalski, R., and Sergot, M. 1986. A logic-based calculus of events. *New Gen. Comput.* 4(1):67–95.

McCain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In *IJCAI*, 1978–1984.

Mccarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, 463–502. Edinburgh University Press.

Nieuwenhuis, R. 2009. Sat modulo theories: Enhancing sat with special-purpose algorithms. In *SAT 2009,LNCS 5584*, 1–1. Springer-Verlag.

Pinto, J., and Reiter, R. 1995. Reasoning about time in the situation calculus. *Ann. Math. Artif. Intell.* 14(2-4):251–268.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Simon, L.; Mallya, A.; and Gupta, G. 2005. Design and implementation of $\mathcal{A}_T$: A real-time action description language. In *LOPSTR2005,LNCS3901*, 44–60. Springer-Verlag.

van Harmelen, F.; Lifschitz, V.; and Porter, B., eds. 2007. *Handbook of Knowledge Representation*. Elsevier Science.

Y.Shoham. 1987. *Reasoning about Action and Change*. MIT Press.