Deciding an Interval Logic with Accumulated Durations[⋆]

Martin Fränzle[1] and Michael R. Hansen[2,⋆⋆]

[1] Dpt. Informatik, C. v. Ossietzky Universität Oldenburg, Germany
fraenzle@informatik.uni-oldenburg.de
[2] Informatics and Math. Modelling, Technical University of Denmark
mrh@imm.dtu.dk

Abstract. A decidability result and a model-checking procedure for a rich subset of Duration Calculus (DC) [19] is obtained through reductions to first-order logic over the real-closed field and to Multi-Priced Timed Automata (MPTA) [13]. In contrast to other reductions of fragments of DC to reachability problems in timed automata, the reductions do also cover constraints on positive linear combinations of accumulated durations. By being able to handle accumulated durations under chop as well as in arbitrary positive Boolean contexts, the procedures extend the results of Zhou et al. [22] on decidability of linear duration invariants to a much wider fragment of DC.

Keywords: Real-time systems, metric-time temporal logic, decidability, model-checking, multi-priced timed automata.

1 Introduction

The Duration Calculi (DC) are a family of metric-time temporal logics facilitating reasoning about embedded real-time systems at a high level of abstraction from operational detail [21,19]. Its major ingredients permitting such abstractness are, on one hand, that it is an interval-based [10] rather than a situation-based temporal logic [14] and, on the other hand, the notion of an accumulated duration of a predicate being true over some observation interval. While the former permits a less state-based style of specification, the latter supports abstraction from the fine-granular distribution of interesting or critical situations along the time line. An example is the accumulated runtime of some task in a multitasking environment, where the time instants where the task actually is run are of minor importance, provided the accumulated duration of running it before its deadline is sufficient for its completion.

[⋆] This work has been supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, www.avacs.org) and by Velux Fonden, Søborg, Denmark, through the Velux Visiting Professors Programme.

[⋆⋆] This work has been partially funded by The Danish Council for Strategic Research under project **MoDES**.

© Springer-Verlag Berlin Heidelberg 2007

While the abstractness supported by DC is desirable for system specification and analysis, it proved to be a burden for automatic verification support. Both the satisfiability problem and the model-checking problem wrt. timed automata of most non-trivial fragments of DC are known to be undecidable [20,19]. In the dense-time setting with finitely variable models as interpretation, decidability has in general only been obtained by either dropping metric time altogether [20] or by dropping accumulated durations and, furthermore, seriously restricting the use of negation or chop (DC's only modality) [5,12,8,6]. The only notable exception is [22], where a conjunction of *linear duration invariants* is automatically checked on the possible runs of a timed transition table, where transition occurrences are constrained by upper and lower bounds on the residence time in the source state. Linear duration invariants are, however, an extremely small fragment of DC: They are formulae $c_0 \leq \ell \Rightarrow \sum_{i=1}^{n} c_i \int P_i \leq c_{n+1}$ expressing that the weighted sum $\sum_{i=1}^{n} c_i \int P_i$ of the accumulated durations $\int P_i$ of some mutually exclusive state properties P_i is always less than c_{n+1}, provided the length of the observation interval exceeds c_0. Furthermore, the automaton model dealt with is very restrictive: by only featuring timing bounds on the residence time in the source state of a transition, it is considerably less expressive than timed transition systems with clocks [1,4]. In particular, it is not closed under, e.g., parallel composition.

Within this paper, we do complement the aforementioned decidability results by procedures that are able to

1. check satisfiability of formulae featuring multiple different accumulated durations within subformulae which, furthermore, may occur under arbitrarily nested chop and within complex Boolean contexts, provided the chop modalities occur in positive context, and to
2. check whether every run of a timed automaton A satisfies $\neg\phi$, where ϕ is a formula as described under point (1). This model-checking problem is usually written $A \models \neg\phi$, and in this special form ϕ is a specification of an undesired situation, and $A \models \neg\phi$ asserts that no run of A exist which exhibits the undesired situation. This idea is, for example, pursued in [15,11], where ϕ can have the restricted form of a *DC implementable* [16], thus abandoning accumulated durations and replacing chop by more restricted, operationally inspired operators. We extend their work by allowing formulae featuring accumulated durations and arbitrary positive chop.

For the decidability results concerning satisfiability of formulae, our construction builds on a small model property permitting the reduction of model construction for DC to satisfiability of first-order logic over the reals with addition $FOL(\mathbb{R}, +, \leq)$. The model-checking results are obtained through a reduction to Multi-Priced Timed Automata (MPTA) [13], where weighted sums of accumulated durations are encoded by prices. The syntactic structure of the formula to be checked reflects in the structure of the MPTA generated, where conjunction and disjunction map to the corresponding operations on automata, while the chop modality yields concatenation.

Structure of the paper: In Sect. 2, we introduce Duration Calculus and the relevant notions of satisfiability and satisfiability over length-bounded models. Section 3 provides the decidability result concerning satisfiability, while Sect. 5 provides the corresponding result for the model-checking problem. In between, Sect. 4 reviews multi-priced timed automata, as defined by Rasmussen and Larsen in [13]. Section 6, finally, discusses how close these results are to the decidability borderline.

2 Duration Calculus

Duration Calculus (abbreviated DC in the remainder) is a real-time logic that is developed for reasoning about durational constraints on time-dependent Boolean-valued states. Since its introduction in [21], many variants of DC have been defined [19]. In this paper we aim at a subset involving durational constraints, which can be supported by automated reasoning.

2.1 Syntax

The syntax of DC used in this paper is defined below. We shall define two syntax categories: *state expressions*, ranged over by $S, S_1, S_2, \ldots$, and *formulae*, ranged over by $\phi, \phi_1, \psi, \psi_1, \ldots$. State expressions are Boolean combinations of state variables, and they describe combined states of a system at a given point in time. Formulae can be considered as truth-valued functions on time intervals, i.e. for a given time interval, a formula is either true or false.

The abstract syntax for state expressions and formulae is defined by:

$$\begin{aligned} S &::= 0 \mid 1 \mid P \mid \neg S \mid S_1 \vee S_2 \\ \phi &::= \ell \bowtie k \mid \llbracket S \rrbracket \mid \textstyle\sum_{i=1}^{m} c_i \smallint S_i \bowtie k \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \frown \psi \ , \end{aligned}$$

where ℓ is a special symbol denoting the interval length, P ranges over state variables, $k, m, c_i \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$.

In the remainder, we will call any formula built according to the above syntax a *DC formula.* The subset of DC formulae where the chop modality "$\frown$" do only occur under a positive number of negations is denoted $\mathrm{DC}_{\mathrm{pos}}$. $\mathrm{DC}_{\setminus\neg}$ will name the set of all negation-free (at formula level) DC formulae. Finally, $\mathrm{DC}_{\mathrm{ub}}$ contains all $\mathrm{DC}_{\setminus\neg}$ formulae which contain only upper bound constraints on durations, i.e. where $\bowtie \in \{<, \leq\}$, and where exactly one duration constraint is a strict inequality.

2.2 Semantics

An interpretation $\mathcal{I}$ associates a function $P_{\mathcal{I}} : \mathbb{R}_{\geq 0} \to \{0, 1\}$ with every state variable P, where $\mathbb{R}_{\geq 0}$ models the dense time line such that interpretations yield time-dependent, Boolean-valued valuations of state variables. We impose the *finite variability restriction* that $P_{\mathcal{I}}$ has at most a finite number of discontinuity points in any interval $[a, b]$.

The semantics of a state expression S, given an interpretation $\mathcal{I}$, is a function: $\mathcal{I}[\![S]\!] : \mathbb{R}_{\geq 0} \to \{0,1\}$, which is defined inductively as follows:

$$\begin{array}{llll}
\mathcal{I}[\![0]\!](t) &= 0 & \mathcal{I}[\![\neg S]\!](t) &= \begin{cases} 0 \text{ if } \mathcal{I}[\![S]\!](t) = 1 \\ 1 \text{ if } \mathcal{I}[\![S]\!](t) = 0 \end{cases} \\
\mathcal{I}[\![1]\!](t) &= 1 & & \\
\mathcal{I}[\![P]\!](t) &= P_{\mathcal{I}}(t) & \mathcal{I}[\![S_1 \vee S_2]\!](t) &= \begin{cases} 0 \text{ if } \mathcal{I}[\![S_1]\!](t) = \mathcal{I}[\![S_2]\!](t) = 0 \\ 1 \text{ otherwise.} \end{cases}
\end{array}$$

We shall use the abbreviation $S_{\mathcal{I}} \overset{\text{df}}{=} \mathcal{I}[\![S]\!]$.

Satisfaction of formulae ϕ is defined over pairs $(\mathcal{I}, [a,b])$ of an interpretation $\mathcal{I}$ and a time interval $[a,b]$ with $a, b \in \mathbb{R}_{\geq 0}$. Such a pair $(\mathcal{I}, [a,b])$ is called an *observation.* The satisfaction relation $\mathcal{I}, [a,b] \models \phi$ is defined recursively, where $\mathcal{I}$ is an interpretation, $[a,b]$ is an interval, and ϕ is a formula:

$$\begin{array}{lll}
\mathcal{I}, [a,b] \models \ell \bowtie k & \text{iff} & b - a \bowtie k \\
\mathcal{I}, [a,b] \models [\![S]\!] & \text{iff} & a < b \text{ and } \int_a^b S_{\mathcal{I}}(t)dt = b - a \\
\mathcal{I}, [a,b] \models \sum_{i=1}^m c_i \int S_i \bowtie k & \text{iff} & \sum_{i=1}^m c_i \int_a^b S_{i\mathcal{I}}(t)dt \bowtie k \\
\mathcal{I}, [a,b] \models \neg\phi & \text{iff} & \mathcal{I}, [a,b] \not\models \phi \\
\mathcal{I}, [a,b] \models \phi \vee \psi & \text{iff} & \mathcal{I}, [a,b] \models \phi \text{ or } \mathcal{I}, [a,b] \models \psi \\
\mathcal{I}, [a,b] \models \phi \wedge \psi & \text{iff} & \mathcal{I}, [a,b] \models \phi \text{ and } \mathcal{I}, [a,b] \models \psi \\
\mathcal{I}, [a,b] \models \phi \frown \psi & \text{iff} & \mathcal{I}, [a,m] \models \phi \text{ and } \mathcal{I}, [m,b] \models \psi, \\
& & \text{for some } m \in [a,b].
\end{array}$$

Whenever $\mathcal{I}, [a,b] \models \phi$ holds we say that ϕ is *true* in $[a,b]$ wrt. $\mathcal{I}$. A formula ϕ is said to be *valid* (written $\models \phi$) if $\mathcal{I}, [a,b] \models \phi$ holds for all interpretations $\mathcal{I}$ and all intervals $[a,b]$. Furthermore, a formula ϕ is *satisfiable* if $\mathcal{I}, [a,b] \models \phi$ for some observation $(\mathcal{I}, [a,b])$. Given $k \in \mathbb{N}$, we say that ϕ is *k-bounded satisfiable* if there is an interpretation $\mathcal{I}$ with at most k discontinuity points[1] and an interval $[a,b]$ such that $\mathcal{I}, [a,b] \models \phi$. In this case, we say that observation $(\mathcal{I}, [a,b])$ is a *k-bounded model* of ϕ.

Since every occurrence of a state variable is within the scope of an integral, we can form equivalence classes of interpretations, where no formula can distinguish between interpretations belonging to the same class. This leads to the following definition and lemma:

Definition 1. *Two interpretations $\mathcal{I}$,$\mathcal{I}'$ are called* equivalent in $[a,b]$, *written $\mathcal{I} \approx_{[a,b]} \mathcal{I}'$, if $P_{\mathcal{I}}$ and $P_{\mathcal{I}'}$ disagree on at most a finite number of points in $[a,b]$, for every state variable P.*

Lemma 1. *For any formula ϕ, interpretations $\mathcal{I}$,$\mathcal{I}'$ and interval $[a,b]$:*

$$\textit{If } \mathcal{I} \approx_{[a,b]} \mathcal{I}', \textit{ then } \mathcal{I}, [a,b] \models \phi \textit{ iff } \mathcal{I}', [a,b] \models \phi.$$

[1] Formally speaking, $\mathcal{I}$ is a vector of functions $P_{\mathcal{I}}$ and has no discontinuity points itself. By the discontinuity points of $\mathcal{I}$ we mean the set $\{t \in \mathbb{R} \mid P \text{ is state variable}, P_{\mathcal{I}} \text{ has a discontinuity point in } t\}$ of all discontinuity points of the individual $P_{\mathcal{I}}$.

3 Decidability of the Satisfiability Problem

It has been observed previously, e.g. by Guelev (personal communication, 1997) and by Hoenicke [11], that for fixed $k \in \mathbb{N}$, the k-bounded satisfiability problem for Duration Calculus (as defined in Sect. 2) is decidable via a reduction to first-order logic over the reals with addition $FOL(\mathbb{R}, +, <)$, whose decidability is classical [18].

Lemma 2. *Let $k \in \mathbb{N}$ and ϕ a DC formula.*

1. *It is decidable whether ϕ is k-bounded satisfiable.*
2. *If ϕ is k-bounded satisfiable then ϕ is satisfiable.*
3. *If ϕ is satisfiable then there exists $l \in \mathbb{N}$ such that ϕ is l-bounded satisfiable.*
4. *There is no algorithm which, given a satisfiable formula ϕ, computes the bound l from item 3.*

Proof. A proof of (1) can be found in [11, p.24ff]. (2) and (3) are obvious from the definitions. (4) is a consequence of the general undecidability results of Duration Calculus (e.g., [20,19]) and the decidability result stated in (1). □

Item 4 of Lemma 2 shows that k-bounded satisfiability is much more limited than satisfiability in general and that, consequently, the corresponding decidability results are of limited value. For full DC, they do only provide a semi-decision procedure for (unbounded) satisfiability, based on testing increasing k in Lemma 2 (1) and exploiting the correspondences from Lemma 2 (2 and 3).

We shall show below that formulae of $\mathrm{DC}_{\mathrm{pos}}$ have a small-model property permitting effective computation of a bound on the length of minimal models of satisfiable formulae. According to Lemma 2 (1), this implies decidability of the satisfiability problem. The main idea behind this result is that the truth value of a formula $\phi \in \mathrm{DC}_{\mathrm{pos}}$ for an observation $(\mathcal{I}, [a, b])$ is invariant to reshuffling of certain segments of $\mathcal{I}$ in $[a, b]$.

To explain this, let $(\mathcal{I}_1, [a_1, b_1])$ and $(\mathcal{I}_2, [a_2, b_2])$ be observations. Then *observation concatenation* $a : (\mathcal{I}_1, [a_1, b_1]) \frown (\mathcal{I}_2, [a_2, b_2])$ denotes the (set of)[2] observations $(\mathcal{I}', [a, a + b_1 - a_1 + b_2 - a_2])$ with $\mathcal{I}'$ for all state variables P satisfying $\forall t \in [0, b_1 - a_1).P_{\mathcal{I}_1}(a_1 + t) = P_{\mathcal{I}'}(a + t)$ and $\forall t \in (0, b_2 - a_2].P_{\mathcal{I}_2}(a_2 + t) = P_{\mathcal{I}'}(a + b_1 - a_1 + t)$. We shall omit repeated a : in repeated concatenations $a : (a : \mathcal{I}_1 \frown \mathcal{I}_2) \frown \mathcal{I}_3$.

Lemma 3. *Let ϕ be a chop-free formula and $(\mathcal{I}, [a, b]) = a : \mathcal{O}_1 \frown \mathcal{O}_2 \frown \cdots \frown \mathcal{O}_k$ be a concatenation of observations $\mathcal{O}_i$. Then*

$$a : (\mathcal{O}_1 \frown \mathcal{O}_2 \frown \cdots \frown \mathcal{O}_k), [a, b] \models \phi \textit{ iff } a : (\mathcal{O}_{i_1} \frown \mathcal{O}_{i_2} \frown \cdots \frown \mathcal{O}_{i_k}), [a, b] \models \phi \ ,$$

for any permutation $i_1, i_2, \ldots i_k$ of $1, 2, \ldots, k$.

[2] Note that interpretation outside the observation interval is irrelevant to the semantics of DC such that the fact that concatenation actually yields a set is irrelevant in practice.

Proof. The proof is by induction on the structure of ϕ. The base case $\ell \sim k$ is simple, since the truth value depends on the interval $[a, b]$ only. The other two base cases: $\llbracket S \rrbracket$ and $\sum_{i=1}^{m} c_i \int S_i \bowtie k$, are simple since their truth values are defined in terms of integrals of state expressions, and such integrals are invariant to the reshuffling. The inductive steps for the propositional connectives are straightforward. $\square$

This lemma provides a small-model property for any chop-free formula ϕ. Suppose that ϕ contains n state variables, and that $(\mathcal{I}, [a, b])$ is a model of ϕ. There are 2^n different truth assignments to n Boolean variables, and the above lemma allows us to reshuffle the segments of $\mathcal{I}$ in $[a, b]$ to arrive at a 2^n-bounded model of ϕ.

Corollary 1. *If a chop-free formula ϕ is satisfiable then it has a 2^n-bounded model, where n is the number of state variables occurring in ϕ*

To show the small model property for $\mathrm{DC}_{\mathrm{pos}}$, we first introduce another operator to DC: In a *timed chop* $\phi \frown_c \psi$, where $c \in \mathbb{R}_{\geq 0}$, the chop point is confined to occur exactly at time c:

$$\mathcal{I}, [a, b] \models \phi \frown_c \psi \;\text{ iff }\; a \leq c \leq b \text{ and } \mathcal{I}, [a, c] \models \phi \text{ and } \mathcal{I}, [c, b] \models \psi.$$

It is obvious that a DC formula $\phi \in \mathrm{DC}_{\mathrm{pos}}$ is satisfiable iff there is some satisfiable formula ψ which is syntactically equal to ψ except that all chops have been replaced by timed chops. For such a ψ, we can now show that ψ, if satisfiable, has a model of length linear in the number of (timed) chops in ψ.

Lemma 4. *If ψ does not contain an untimed chop and is satisfiable then ψ is $2^n(m + 1)$-bounded satisfiable, where m is the number of (timed) chops in ψ and n is the number of state variables occurring in ψ.*

Proof. sketch: Between chop points —which are now fixed to constant occurrence times and thus cannot permute—, one can reshuffle the segments in $\mathcal{I}$ arbitrarily, thus ending up with at most 2^n segments between each two chops according to Corollary 1. Since there are m chop points, there are $m + 1$ such segments. $\square$

As chop is a relaxation of timed chop, all models of ψ are also models of ϕ. Therefore, the above result generalizes to DC formulae with untimed chop:

Corollary 2. *If a formula $\phi \in DC_{pos}$ is satisfiable then it has a $2^n(m + 1)$-bounded model, where m is the number of chops in ϕ and n is the number of state variables occurring in ϕ.*

Proof. As $\mathrm{DC}_{\mathrm{pos}}$ contains the duals of all operators except chop,[3] we can rewrite ϕ to negation-free form $\phi' \in \mathrm{DC}_{\setminus\neg}$. If ϕ' is satisfiable then it has at least one satisfiable counterpart ψ containing only timed chops. According to the previous Lemma, ψ has a $2^n(m + 1)$-bounded model. As satisfaction of timed chop implies satisfaction of chop, and due to monotonicity of all other operators in the negation-free fragment $\mathrm{DC}_{\setminus\neg}$, this model is also a model of ϕ' and thus ϕ. $\square$

[3] For $\lceil S \rceil$, we have the duality $\lceil S \rceil = \neg(\ell = 0 \vee \int \neg S > 0)$. All other dualities are the classical ones from predicate logics.

As a consequence, we obtain decidability of the satisfiability problem of DC:

Theorem 1. *It is decidable whether a formula $\phi \in DC_{pos}$ is satisfiable.*

Proof. According to Corollary 2, in order to check for satisfiability of ϕ it suffices to check whether ϕ has a $2^n(m+1)$-bounded model, where m is the number of chops in ϕ and n is the number of state variables occurring in ϕ. Lemma 2 (1) shows decidability of $2^n(m+1)$-bounded satisfiability. $\square$

As –after rewriting to negation-free form $\mathrm{DC}_{\setminus\neg}$— there are no negations in our fragment of DC, the $FOL(\mathbb{R},+,<)$ formula constructed turns out to be in the existential fragment of $FOL(\mathbb{R},+,<)$[4]. Its size is linear in $|\phi|$ and in the bound $k = 2^n(m+1)$ of model construction. For a fixed number n of state variables, it is thus worst-case quadratic in $|\phi|$. As deciding the existential fragment of $FOL(\mathbb{R},+,<)$ is NP-complete, this implies that satisfiability of DC formulae with a fixed number of state variables is in NP. Without a bound on the number of variables, it obviously is singly exponential.

4 Priced Timed Automata

In this section, we review the definition of *Linearly Multi-Priced Timed Automata* (*MPTA*) together with the theorems that we shall use in order to establish our decidability result for DC. The presentation of MPTA is based on [13]. MPTA are an extension of timed automata [1,4], where *prices* are associated with edges and locations. The cost of taking an edge is the price of that edge, and the cost of staying in a location is given by the product of the *cost-rate* for that location and the time spent in the location.

Let $\mathbb{C}$ be a finite set of clocks. An *atomic constraint* is a formula of the form: $x \bowtie n$, where $x \in \mathbb{C}$, $\bowtie \in \{\leq, =, \geq, <, >\}$, and $n \in \mathbb{N}$. A *clock constraint* over $\mathbb{C}$ is a conjunction of atomic constraints. Let $B(\mathbb{C})$ denote the set of clock constraints over $\mathbb{C}$ and let $B(\mathbb{C})^*$ denote the set of clock constraints over $\mathbb{C}$ involving only upper bounds, i.e. $\leq$ or $<$. Furthermore, let $2^{\mathbb{C}}$ denote the power set of $\mathbb{C}$.

A *clock valuation* $v : \mathbb{C} \to R_{\geq 0}$ is a function assigning a non-negative real number with each clock. The valuation v *satisfies* a clock constraint $g \in B(\mathbb{C})$, if each conjunct of g is true in v. In this case we write $v \in g$. Let $\mathbb{R}^{\mathbb{C}}_{\geq 0}$ denote the set of all clock valuations.

Definition 2 (cf. [13]). *A* multi-priced timed automaton *A over clocks $\mathbb{C}$ is a tuple (L, l_0, E, I, P), where L is a finite set of locations, l_0 is the initial location, $E \subseteq L \times B(\mathbb{C}) \times 2^{\mathbb{C}} \times L$ is the set of edges, where an edge contains a source, a guard, a set of clocks to be reset, and a target. $I : L \to B(\mathbb{C})^*$ assigns invariants to locations, and $P : (L \cup E) \to \mathbb{N}^m$ assigns a vector of prices to both locations and edges. In the case of $(l, g, r, l') \in E$, we write $l \xrightarrow{g,r} l'$.*

In order to give semantics to linearly multi-priced timed automata, the notion of a multi-priced transition system is introduced. A *multi-priced transition system*

[4] Also known as "LinSAT", featuring powerful tool support, e.g. [3,9].

is a structure $T = (S, s_0, \Sigma, \longrightarrow)$, where S is a, possibly infinite, set of states, $s_0 \in S$ is the initial state, Σ is a finite set of labels, and $\longrightarrow$ is a partial function from $S \times \Sigma \times S$ to $\mathbb{R}^m_{\geq 0}$, defining the possible transitions and their associated costs. The notation $s \xrightarrow{a}_{\boldsymbol{p}} s'$ means that $\longrightarrow (s, a, s')$ is defined and equal to $\boldsymbol{p}$. An *execution* of T is a finite sequence $\alpha = s_0 \xrightarrow{a_1}_{\boldsymbol{p}_1} s_1 \xrightarrow{a_2}_{\boldsymbol{p}_2} s_2 \cdots s_{n-1} \xrightarrow{a_n}_{\boldsymbol{p}_n} s_n$, and the associated *cost* of α is $\text{cost}(\alpha) = \sum_{i=1}^n \boldsymbol{p}_i$.

For a given state s and a vector $\boldsymbol{u} = (u_1, \ldots, u_{m-1}) \in \mathbb{R}^{m-1}_{\geq 0}$, let $\text{mincost}_{T,\boldsymbol{u}}(s)$ denote the *minimum cost* wrt. the last component of the price vector of reaching s while respecting the upper bound constraints to the other prices which are given by $\boldsymbol{u}$. This is defined as the infimum of the cost of all executions ending in s and respecting price constraint $\boldsymbol{u}$, i.e.

$$\text{mincost}_{T,\boldsymbol{u}}(s) = \inf \left\{ \text{cost}(\alpha)_m \,\middle|\, \begin{array}{l} \alpha \text{ an execution of } T \text{ ending in } s, \\ \forall i \in \mathbb{N}_{<m}.\text{cost}(\alpha)_i \leq u_i \end{array} \right\} .$$

Furthermore, for a set of states $G \subseteq S$, let $\text{mincost}_{T,\boldsymbol{u}}(G)$ denote the minimal cost of reaching some state in G while respecting the upper price bounds $\boldsymbol{u}$.

The semantics of a linearly multi-priced timed automaton $A = (L, l_0, E, I, P)$ is a *multi-priced transition system* $T_A = (S, s_0, \Sigma, \longrightarrow)$, where

- $S = L \times \mathbb{R}^{\mathbb{C}}_{\geq 0}$,
- $s_0 = (l_0, v_0)$, where v_0 is the (clock) valuation assigning 0 to every clock,
- $\Sigma = E \cup \{\delta\}$, where δ indicates a delay and $e \in E$ the edge taken, and
- the partial transition function $\longrightarrow$ is defined as follows:
  - $(l, v) \xrightarrow{\delta}_{\boldsymbol{p}} (l, v + d)$ if $\forall e. 0 \leq e \leq d : v + e \in I(l)$, and $\boldsymbol{p} = d \cdot P(l)$,
  - $(l, v) \xrightarrow{e}_{\boldsymbol{p}} (l', v')$ if $(l, g, r, l') \in E, v \in g, v' = v[r \mapsto 0]$ and $\boldsymbol{p} = P(e)$,

  where $v + d$ means the clock valuation where the value of x is $v(x) + d$, for $x \in \mathbb{C}, d \in R_{\geq 0}$, and $v[r \mapsto 0]$ is the valuation which is as v except that clocks in r are mapped to 0.

In case T_A performs a δ step $(l, v) \xrightarrow{\delta}_{\boldsymbol{p}} (l, v + d)$, we say that the *duration of the step* is d. All other steps, i.e. those labelled $e \in E$, have duration 0.

The main results that we shall exploit concerning linearly multi-priced timed automata is that the minimum cost of reaching some target location is computable for any (set of) target location(s) and any upper bound on the remaining prices: Given an MPTA $A = (L, l_0, E, I, P)$, a target $G \subset L$, and some cost constraint $\boldsymbol{u} \in \mathbb{R}^{m-1}_{\geq 0}$, we define the *minimum cost* $\text{mincost}_{A,\boldsymbol{u}}(G)$ to be $\text{mincost}_{T_A,\boldsymbol{u}}(G \times \mathbb{R}^{\mathbb{C}}_{\geq 0})$.

Theorem 2 ([13]). *For any MPTA $A = (L, l_0, E, I, P)$, any set $G \subset L$, and any cost constraint $\boldsymbol{u} \in \mathbb{R}^{m-1}_{\geq 0}$, the minimum cost $\text{mincost}_{A,\boldsymbol{u}}(G)$ is computable.*

5 Encoding of DC$_{\text{ub}}$ Formulae by MPTA

Within this section, we will provide an encoding of DC$_{\text{ub}}$ formulae ϕ by MPTA representing their models. The encoding will be such that each model of ϕ

corresponds to a run of the corresponding MPTA with the associated costs representing and satisfying the duration constraints in ϕ. In detail, we shall represent each formula ϕ by a tuple $(L, s, E, I, P, f, \Lambda)$ denoted A_ϕ, where (L, s, E, I, P) is a multi-priced timed automaton, f is a special *final location* to be reached, and Λ is a function associating a DC state-expression S with every location. The construction will be such that the automaton will not be allowed to spend positive time in the final location, and the intuition is that the satisfying observations of ϕ are represented by the set of executions of A_ϕ ending in f. Subformulae of the form $\sum_{i=1}^{m} c_i \int S_i \bowtie k$ will, however, receive a special treatment. The intuition about the automaton for such a formula is that its executions ending in f can generate all possible interpretations to the state variables and that the value of the expression $\sum_{i=1}^{m} c_i \int S_i$ is the cost of the execution, and a bounding of the cost or an analysis of the minimal cost of executions can be used to decide satisfaction of $\sum_{i=1}^{m} c_i \int S_i \bowtie k$.

5.1 The Construction

In the construction we shall use the following conventions:

- the cost of an edge is always 0,
- the cost-rate of a location is 0 unless otherwise stated,
- the invariant of a location is true unless otherwise stated,
- the mark of a location l is the state expression 1, i.e. $\Lambda(l) = 1$, unless otherwise stated.

In the following we assume that the formula ϕ under consideration contains n distinct state variables $P_1, \ldots, P_n$ and m subformulae $\sum_{i=1}^{m_j} c_{i,j} \int S_{i,j} \bowtie_j k_j$, where $\bowtie_m = <$ and $\bowtie_j = \leq$ for every $j < m$. We shall give a recursive construction of an automaton which follows the structure of the formula. The base cases are $\ell \bowtie k$, $\llbracket S \rrbracket$ and $\sum_{i=1}^{m_j} c_i \int S_i \bowtie_j k_j$.

The case $\phi = \ell \bowtie k$. Let $A_\phi = (L, s, E, I, P, f, \Lambda)$, where

- $L = \{s, f\}$,
- $E = \{(s, x \bowtie k, \{x\}, f)\}$, and
- $I(f) = x \leq 0$.

This automaton is depicted in Fig. 1(a).

The case $\phi = \llbracket S \rrbracket$. Let $A_\phi = (L, s, E, I, P, f, \Lambda)$, where

- $L = \{s, l_1, f\}$,
- $E = \{e_1, e_2, e_3\}$, where $e_1 = (s, \text{true}, \{\}, l_1)$, $e_2 = (l_1, y > 0, \{y\}, s)$, and $e_3 = (l_1, x > 0, \{x\}, f)$,
- $I(s) = y \leq 0$ and $I(f) = x \leq 0$, and
- $\Lambda(l_1) = S$.

This automaton is depicted in Fig. 1(b).

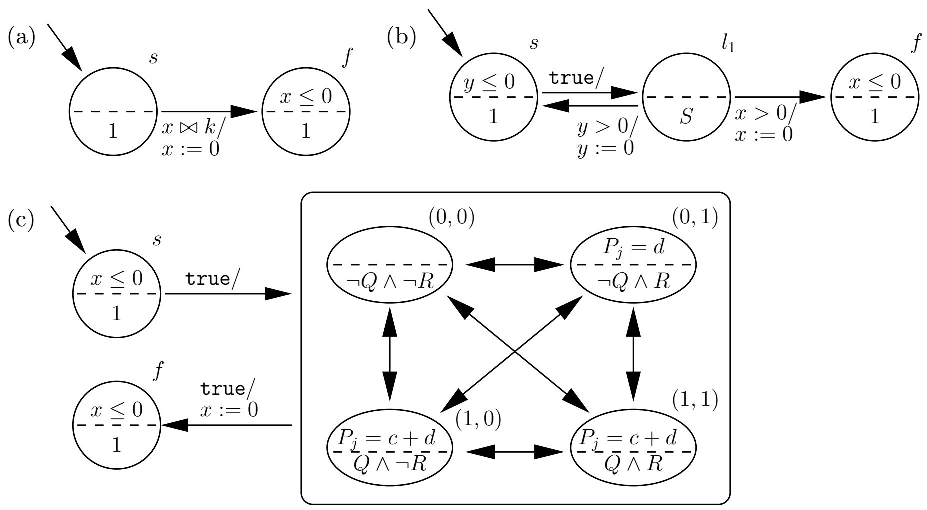

Fig. 1. MPTA encoding of atomic formulae: (a) $\ell \bowtie k$, (b) $\lceil S \rceil$, (c) $c\int Q + d\int Q \vee R \bowtie k$. State decorations above the dashed line denote invariants and cost assignments (both omitted if trivial), while those below the dashed line denote the labeling function Λ.

The case $\phi = \sum_{i=1}^{m_j} c_{i,j} \int S_{i,j} \bowtie_j k_j$. Let $A_\phi = (L, s, E, I, P, f, \Lambda)$, where $L = \{s, f\} \cup \{0,1\}^n$ and E, I, P and Λ are defined below. Each n-tuple in $\{0,1\}^n$ is a bit-vector $\boldsymbol{b} = (b_1, \ldots, b_n)$ and the idea with this is that $b_i = 1$ iff the value of P_i is 1 in that state.

The set of edges $E = E_1 \cup E_2 \cup E_3$ is defined as the union of three sets, where

- $E_1 = \{(s, \text{true}, \emptyset, \boldsymbol{b}) \mid \boldsymbol{b} \in \{0,1\}^n\}$,
- $E_2 = \{(\boldsymbol{b}, \text{true}, \emptyset, \boldsymbol{b'}) \mid \boldsymbol{b}, \boldsymbol{b'} \in \{0,1\}^n \wedge \boldsymbol{b} \neq \boldsymbol{b'}\}$, and
- $E_3 = \{(\boldsymbol{b}, \text{true}, \{x\}, f) \mid \boldsymbol{b} \in \{0,1\}^n\}$.

For $\boldsymbol{b} \in \{0,1\}^n$, we define two sets: $\boldsymbol{b}^+ = \{l \in \mathbb{N} \mid 1 \leq l \leq n \wedge b_l = 1\}$ and $\boldsymbol{b}^- = \{l \in \mathbb{N} \mid 1 \leq l \leq n \wedge b_l = 0\}$. Let $F(\boldsymbol{b})$ denote the state expression:

$$\bigwedge_{l \in \boldsymbol{b}^-} \neg P_l \wedge \bigwedge_{l \in \boldsymbol{b}^+} P_l \,.$$

For each state expression $S_{i,j}$ occurring in the summation $\sum_{i=1}^{m_j} c_{i,j} \int S_{i,j}$, we define the cost rate as follows:

$$C(\boldsymbol{b})(S_{i,j}) = \begin{cases} c_{i,j}, & \text{if } F(\boldsymbol{b}) \Rightarrow S_{i,j}, \\ C(\boldsymbol{b})(S_{i,j}) = 0 & \text{otherwise.} \end{cases}$$

The invariants of locations are as follows: $I(s) = x \leq 0, I(f) = x \leq 0$, and for all other locations the invariant is true.

The cost assignment $P : L \cup E \rightarrow \mathbb{N}^m$ is defined as follows:

$$P(l)_k = \begin{cases} 0 & \text{if } l = s \text{ or } l = f \text{ or } k \neq j \text{ or } l \in E \\ \sum_{i=1}^{m_j} C(l)(S_{i,j}) & \text{otherwise.} \end{cases}$$

The definition of the labelling function Λ is $\Lambda(l) = 1$ iff $l = s$ or $l = f$ and $F(l)$ otherwise. An example of this automaton construction is shown in Fig. 1(c).

We now consider the recursive cases: $\phi \vee \psi$, $\phi \wedge \psi$ and $\phi^\frown\psi$. In these cases, we will assume that the automata $A_\phi = (L_1, s_1, E_1, I_1, P_1, f_1, \Lambda_1)$ and $A_\psi = (L_2, s_2, E_2, I_2, P_2, f_2, \Lambda_2)$, have disjoint sets of locations and clocks, respectively.

The case $\phi \vee \psi$. Assume that s and f are two new locations and that x is a new clock. Let $A_{\phi\vee\psi} = (L, s, E, I, P, f, \Lambda)$, where

- $L = \{s, f\} \cup L_1 \cup L_2$,
- $E = \{e_1, e_2, e_3, e_4\} \cup E_1 \cup E_2$, where $e_1 = (s, \text{true}, \{\}, s_1)$, $e_2 = (s, \text{true}, \{\}, s_2)$, $e_3 = (f_1, \text{true}, \{x\}, f)$, and $e_4 = (f_2, \text{true}, \{x\}, f)$.
- $I(s) = I(f) = x \le 0$, $I(l) = I_1(l)$, for $l \in L_1$, and $I(l) = I_2(l)$, for $l \in L_2$,
- $P(l) = P_1(l)$, for $l \in L_1$, and $P(l) = P_2(l)$, for $l \in L_2$, and
- $\Lambda(l) = \Lambda_1(l)$, for $l \in L_1$, and $\Lambda(l) = \Lambda_2(l)$, for $l \in L_2$.

The case: $\phi \wedge \psi$. Let $A_{\phi\wedge\psi} = (L, (s_1, s_2), E, I, P, (f_1, f_2), \Lambda)$, where

- $L = \{(l_1, l_2) \in L_1 \times L_2 \mid \Lambda_1(l_1) \wedge \Lambda_2(l_2) \text{ is satisfiable}\}$,
- $$E = \left\{ ((l_1, l_2), g_1 \wedge g_2, r_1 \cup r_2, (l_1', l_2')) \;\middle|\; \begin{array}{c} (l_1, l_2), (l_1', l_2') \in L \\ \wedge\, (l_1, g_1, r_1, l_1') \in E_1 \\ \wedge\, (l_2, g_2, r_2, l_2') \in E_2 \end{array} \right\} \cup$$
  $$\{((l_1, l_2), g_1, r_1, (l_1', l_2)) \mid (l_1, l_2), (l_1', l_2) \in L \wedge (l_1, g_1, r_1, l_1') \in E_1\} \cup$$
  $$\{((l_1, l_2), g_1, r_1, (l_1, l_2')) \mid (l_1, l_2), (l_1, l_2') \in L \wedge (l_2, g_2, r_2, l_2') \in E_2\}$$
- $I(l_1, l_2) = I_1(l_1) \wedge I_2(l_2)$, for $(l_1, l_2) \in L$,
- $P(l_1, l_2)_k = P_1(l_1)_k + P_2(l_2)_k$, for $(l_1, l_2) \in L$ and $1 \le k \le m$ and
- $\Lambda(l_1, l_2) = \Lambda_1(l_1) \wedge \Lambda_2(l_2)$, for $(l_1, l_2) \in L$.

The case: $\phi \frown \psi$. Let $A_{\phi\frown\psi} = (L_1 \cup L_2, s_1, E, I, P, f_2, \Lambda)$, where

- $E = \{(f_1, \text{true}, C_2, s_2)\} \cup E_1 \cup E_2$, where C_2 is the set of clocks used by A_ψ,
- $I(l) = I_1(l)$, for $l \in L_1$, and $I(l) = I_2(l)$, for $l \in L_2$,
- $P(l) = P_1(l)$, for $l \in L_1$, and $P(l) = P_2(l)$, for $l \in L_2$.
- $\Lambda(l) = \Lambda_1(l)$, for $l \in L_1$, and $\Lambda(l) = \Lambda_2(l)$, for $l \in L_2$.

Note that the transition from f_1 to s_2 has to be taken immediately when f_1 is reached, as the clock constraints imposed in $I_1(f_1)$ does not permit durational stays in f_1.

5.2 Correspondence Between Interpretations of Formulae and Runs of Corresponding MPTA

The above construction yields a correspondence between satisfiability of the encoded DC formula and existence of runs in A_ϕ featuring adequate prices. In order to show this, we shall first establish a connection between DC observations and the runs of automata.

Let $A = (L, s, E, I, P, f, \Lambda)$ and $\alpha = s_0 \xrightarrow{a_1}_{\boldsymbol{p}_1} s_1 \xrightarrow{a_2}_{\boldsymbol{p}_2} s_2 \cdots s_{n-1} \xrightarrow{a_n}_{\boldsymbol{p}_n} s_n$ be a run of (L, s, E, I, P). The *duration* of α, written $\Delta(\alpha)$, is the sum of all the durations of steps in α. We shall below define the set of DC observations generated by run α as a set of interpretations observed over the interval $[0, \Delta(\alpha)]$. We first define anchored concatenation $(\mathcal{I}_1, [0, e_1]) \frown (\mathcal{I}_2, [0, e_2])$ of observations $(\mathcal{I}_1, [0, e_1])$ and $(\mathcal{I}_2, [0, e_2])$ as the set of observations $0 : (\mathcal{I}_1, [0, e_1]) \frown (\mathcal{I}_2, [0, e_2])$, as defined on page 205. This definition extends to sets of observations: $S_1 \frown S_2 = \bigcup_{\mathcal{O}_1 \in S_1, \mathcal{O}_2 \in S_2} \mathcal{O}_1 \frown \mathcal{O}_2$.

Based on this, we will now define $\mathit{Intp}(\alpha)$ in two steps: First, we define $\mathit{Intp}(s_i)$ for each step in $\alpha = s_0 \xrightarrow{a_1}_{\boldsymbol{p}_1} s_1 \ldots$. Then, we concatenate these observations. For each step s_i in α, the set $\mathit{Intp}(s_i)$ of interpretations over that state is defined by:

$$\begin{aligned} \mathit{Intp}(s_i) &= \{(\mathcal{I}, [0, 0]) \mid \mathcal{I} \text{ an arbitary interpretation}\} \\ &\quad \text{if } i = 0 \text{ or if } s_i \text{ is reached via an edge } e \in E \text{ in } \alpha, \\ \mathit{Intp}(s_i) &= \{(\mathcal{I}, [0, d]) \mid \mathcal{I}, [0, d] \models [\![\Lambda(l_i)]\!]\} \\ &\quad \text{if } s_i \text{ is reached by a delay transition of duration } d \text{ in } \alpha. \end{aligned}$$

The set of observations $\mathit{Intp}(\alpha)$ corresponding to α is then defined as the concatenation of the individual $\mathit{Intp}(s_i)$:

$$\mathit{Intp}(\alpha) = \mathit{Intp}(s_0) \frown \mathit{Intp}(s_1) \frown \cdots \frown \mathit{Intp}(s_n) \ .$$

With the above correspondence between runs and interpretations, we can now formalize the correspondence between DC formulae and the corresponding multi-priced timed automata.

Lemma 5. *Let ϕ be a $DC_{\setminus\neg}$ formula and $A_\phi = (L, s, E, I, P, f, \Lambda)$ be the corresponding multi-priced timed automaton. Then $\mathcal{I}, [0, e] \models \phi$ iff there exists a run α of A_ϕ with $(\mathcal{I}, [0, e]) \in \mathit{Intp}(\alpha)$ and $\mathrm{cost}(\alpha)_j \bowtie_j k_j$ for $1 \leq j \leq m$.*

Proof. By induction over the structure of ϕ. □

As a consequence, we obtain a correspondence between satisfiability of the encoded DC formula and existence of runs in A_ϕ featuring adequate prices.

Theorem 3. *Let ϕ be a formula in DC_{ub}, let $A_\phi = (L, s, E, I, P, f, \Lambda)$ be the corresponding multi-priced timed automaton, and let $\boldsymbol{u} = (k_1, \ldots, k_{m-1})$. Then $\mathrm{mincost}_{(L,s,E,I,P),\boldsymbol{u}}(f) < k_m$ iff ϕ is satisfiable.*

Proof. By the previous lemma, $\mathcal{I}, [0, b] \models \phi$ iff there is a run α of A_ϕ such that $(\mathcal{I}, [0, b]) \in \mathit{Intp}(\alpha)$ and $\mathrm{cost}(\alpha)_j \bowtie_j k_j$ for $1 \leq j \leq m$. As Intp is a total function, this implies that ϕ is satisfiable iff A_ϕ has run α with $\mathrm{cost}(\alpha)_j \bowtie_j k_j$. By $\bowtie_j = \leq$, for $1 \leq j < m$, and $\bowtie_m = <$, this is the case iff $\mathrm{mincost}_{T,\boldsymbol{u}}(s) < k_m$. □

The above construction can also be used for model-checking timed automata wrt. negations of DC formulae. The cornerstone is to exploit an appropriate automata product between timed automata and priced timed automata to establish an automata-based verification procedure. The *model-checking problem* considered here has the form $A \models \neg\phi$, where $A = (L_1, s_1, E_1, I_1, \Lambda_1)$ is an arbitrary timed automaton (L_1, s_1, E_1, I_1), extended by a labeling $\Lambda_1 : L_1 \to S$ of locations with state expressions. We say that $A \models \neg\phi$ holds iff for each run α of A, the set[5] of all corresponding DC interpretations $\mathit{Intp}(\alpha)$ satisfies $\neg\phi$.

Theorem 4. *Let $A = (L_1, s_1, E_1, I_1, \Lambda_1)$ be a timed automaton (L_1, s_1, E_1, I_1) extended by a location labelling $\Lambda_1 : L_1 \to S$, let ϕ be a DC_{ub} formula, let $A_\phi = (L_2, s_2, E_2, I_2, P_2, f_2, \Lambda_2)$, and let $\boldsymbol{u} = (k_1, \ldots, k_{m-1})$. Then $A \models \neg\phi$ iff* $\mathrm{mincost}_{(L,s,E,I,P),\boldsymbol{u}}(f \times L_1) \geq k_m$*, where*

- *$B = (L_1, s_1, E_1, I_1, P_0, s, \Lambda_1)$ is A converted to an MPTA by extension with the trivial cost function $P_0 \equiv \mathbf{0}$ and an irrelevant terminal state $s \in L_1$,*
- *$(L, s, E, I, P, f, P) = A_\phi \otimes B$ is the multi-priced automaton product from case $\phi \wedge \psi$.*

Proof. Similar to the previous theorem it can be shown that for each run α of A and each model $(\mathcal{I}, [0, b])$ of ϕ with $(\mathcal{I}, [0, b]) \in \mathit{Intp}(\alpha)$ it is the case that α is a run of $A_\phi \otimes B$ with $\mathrm{cost}(\alpha)_j \bowtie_j k_j$ for $1 \leq j \leq m$. I.e., A has a run α with $\mathit{Intp}(\alpha) \models \phi$ iff $\mathrm{mincost}_{(L,s,E,I,P),\boldsymbol{u}}(f \times L_1) < k_m$. Consequently, all runs of A satisfy $\neg\phi$ iff $\mathrm{mincost}_{(L,s,E,I,P),\boldsymbol{u}}(f \times L_1) \geq k_m$. $\square$

Model-checking timed automata against $\mathrm{DC}_{\mathrm{ub}}$ formulae is thus possible.

6 Conclusion

Within this paper, two new decision procedures for rich subsets of Duration Calculus have been devised:

1. We have shown that satisfiability of DC formulas with linear combinations of accumulated durations, yet chop confined to occur in positive context only, is decidable.
2. A model-checking procedure for timed automata against DC formula with only upper bound duration constraints and only a single, outermost negation has been established based on a reduction to multi-priced timed automata.

Both procedures do considerably extend the scope of automatic procedures for DC beyond the previous state of the art: These procedures are the first to combine reasoning over accumulated durations and over chop within automated decision procedures. Furthermore, (2.) extends model-checking procedures for timed transition systems against accumulated duration properties, as pioneered in [22], from timed transition tables with per-transition delays to timed automata with clocks.

For the first of the two procedures, it is clear that the positive decidability results marks the frontier to undecidability, as admitting chop in negative context leads to undecidability [20]. The correspondence of DC without accumulated durations to timed regular expressions [2] shown in [8], together with the lacking closure of timed regular languages under negation [1], shows that decidability is

[5] Note that the labeling Λ_1 may permit multiple different valuations within a single location $l \in L_1$.

even lost without nesting of chop under different polarity; negative chop itself leads to undecidability. Accordingly, the encodings of two-counter machines by DC formulas used in [20] or of stop-watch automata used in [7, App. A] to demonstrate undecidability of DC do only use negative chop.

With respect to the model-checking result, the exact borderline to undecidability is open. While one might well expect that lower bounds on accumulated durations should also be decidable, e.g. through replacing minimum price reachability in priced timed automata by maximum price reachability, the current notion of maximum price reachability in priced timed automata does not permit an adequate reduction. Being inspired by scheduling problems, the theory of priced timed automata does define the maximum price to be infinite as soon as path length in the automaton is unbounded. This does interfere with the notion of accumulated duration, as an accumulated duration may well be bounded even though the number of state changes in the run is not a priori bounded, as can be seen from the formula $\phi = (\ell < 2 \wedge \textstyle\int P > 2)$. This formula is unsatisfiable, yet the automaton construction from Sect. 5 yields an automaton with unbounded path length (cf. Fig. 1(c)) such that maximum cost reachability would consider the cost P to be infinite, suggesting $\textstyle\int P > 2$ to hold.

Another open question is whether the more restricted notion of chop used in Interval Duration Logic (IDL) [17] facilitates model-checking of larger formula classes. It is obvious that all the procedures detailed in this paper do also work on IDL with the appropriate minor modifications.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Comput. Sci.*, 126(2):183–235, 1994.
2. E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In G. Winskel, editor, *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*. IEEE Computer Society Press, 1997.
3. G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowics, and R. Sebastiani. A SAT-based approach for solving formulas over boolean and linear mathematical propositions. In A. Voronkov, editor, *Automated Deduction — CADE-18*, volume 2392 of *Lecture Notes in Computer Science*, pages 193–208. Springer-Verlag, 2002.
4. J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi. Uppaal – a tool suite for automatic verification of real-time systems. In R. Alur, T. Henzinger, and E. Sonntag, editors, *Hybrid Systems III – Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 232–243. Springer-Verlag, 1997.
5. A. Bouajjani, Y. Lakhnech, and R. Robbana. From duration calculus to linear hybrid automata. In P. Wolper, editor, *Computer Aided Verification (CAV '95)*, volume 939 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
6. H. Dierks. Synthesizing controllers from real-time specifications. In *Tenth International Symposium on System Synthesis (ISSS '97)*, pages 126–133. IEEE Computer Society Press, 1997.
7. M. Fränzle. *Controller Design from Temporal Logic: Undecidability need not matter.* Dissertation, Technische Fakultät der Chr.-Albrechts-Universität Kiel, Germany, 1997.

8. M. Fränzle. Model-checking dense-time duration calculus. *Formal Aspects of Computing*, 16(2):121–139, 2004.
9. M. Fränzle and C. Herde. Efficient proof engines for bounded model checking of hybrid systems. In J. Bicarregui, A. Butterfield, and A. Arenas, editors, *Proceedings Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 04)*, volume 133 of *Electronic Notes in Theoretical Computer Science*, pages 119–137. Elsevier Science B.V., 2005.
10. J. Halpern, B. Moszkowski, and Z. Manna. A hardware semantics based on temporal intervals. In J. Diaz, editor, *International Colloquium on Automata, Languages, and Programming (ICALP'83)*, volume 154 of *Lecture Notes in Computer Science*, pages 278–291. Springer-Verlag, 1983.
11. J. Hoenicke. *Combination of Processes, Data and Time.* Dissertation, Carl von Ossietzky Universität, Oldenburg, Germany, 2006.
12. Y. Lakhnech. *Specification and Verification of Hybrid and Real-Time Systems.* Dissertation, Technische Fakultät der Chr.-Albrechts-Universität Kiel, Germany, 1996.
13. K. G. Larsen and J. I. Rasmussen. Optimal conditional reachability for multi-priced timed automata. In V. Sassone, editor, *Foundations of Software Science and Computation Structures (FOSSACS '05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 230–244. Springer-Verlag, 2005.
14. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*, volume 1. Springer-Verlag, 1992.
15. E.-R. Olderog and H. Dierks. Decomposing real-time specifications. In H. Langmaack, W. de Roever, and A. Pnueli, editors, *Compositionality: The Significant Difference*, Lecture Notes in Computer Science. Springer-Verlag, 1998.
16. A. P. Ravn. *Design of Embedded Real-Time Computing Systems.* Doctoral dissertation, Department of Computer Science, Danish Technical University, Lyngby, DK, Oct. 1995. Available as technical report ID-TR: 1995-170.
17. P. Sharma, P. K. Pandya, and S. Chakraborty. Bounded validity checking of interval duration logic. In *TACAS 2005*, volume 3440 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
18. A. Tarski. A decision method for elementary algebra and geometry. RAND Corporation, Santa Monica, Calif., 1948.
19. Zhou Chaochen and M. R. Hansen. *Duration Calculus — A Formal Approach to Real-Time Systems.* EATCS monographs on theoretical computer science. Springer-Verlag, 2004.
20. Zhou Chaochen, M. R. Hansen, and P. Sestoft. Decidability and undecidability results for duration calculus. In P. Enjalbert, A. Finkel, and K. W. Wagner, editors, *Symposium on Theoretical Aspects of Computer Science (STACS 93)*, volume 665 of *Lecture Notes in Computer Science*, pages 58–68. Springer-Verlag, 1993.
21. Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.
22. Zhou Chaochen, Zhang Jingzhong, Yang Lu, and Li Xiaoshan. Linear duration invariants. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT '94)*, volume 863 of *Lecture Notes in Computer Science*, pages 86–109. Springer-Verlag, 1994.