Under consideration for publication in Formal Aspects of Computing

Model-Checking Dense-Time Duration Calculus[1]

Martin Fränzle
Department of Computer Science, Carl-von-Ossietzky Universität Oldenburg, Germany

Abstract. Since the seminal work of Zhou Chaochen, M. R. Hansen, and P. Sestoft on decidability of dense-time Duration Calculus [ZHS93] it is well-known that decidable fragments of Duration Calculus can only be obtained through withdrawal of much of the interesting vocabulary of this logic. While this was formerly taken as an indication that key-press verification of implementations with respect to elaborate Duration Calculus specifications were also impossible, we show that the model property is well decidable for *realistic* designs which feature natural constraints on their switching dynamics.

The key issue is that the classical undecidability results rely on a notion of validity of a formula that refers to a class of models which is considerably richer than the possible behaviours of actual embedded real-time systems: that of finitely variable trajectories. By analysing two suitably sparser model classes we obtain model-checking procedures for rich subsets of Duration Calculus. Together with undecidability results also obtained, this sheds light upon the exact borderline between decidability and undecidability of Duration Calculi and related logics.

Keywords: Dense-time Duration Calculus; Decidability; Model-Checking

1. Introduction

Duration Calculus, as introduced in [ZHR91] and thoroughly analysed in [HZ97], is a logic for reasoning about embedded real-time systems at a high level of abstraction from operational detail. While the abstractness of the vocabulary of Duration Calculus is desirable for system specification and analysis, it proved to be a burden for automatic verification support. Until publication of the author's Ph.D. thesis [Frä97], model-checking any interesting metric-time subset of dense-time Duration Calculus was thought to be impossible, unless the use of temporal operators or of negation is seriously restricted, as in [ZZYL94, BLR95, Lak96]. However, the corresponding undecidability results rely on a notion of validity of a formula that resorts to

[1] This report reflects work that has been partially funded by the Deutsche Forschungsgemeinschaft under contract DFG La 426/13-2 while the author was with the Computer Science Department of the Christian-Albrechts-Universität Kiel, Germany, and under contract DFG Da 206/5-2 at the author's current affiliation.
Correspondence and offprint requests to: Martin Fränzle, Department of Computer Science, Carl-von-Ossietzky Universität Oldenburg, P.O. Box 2503, D-26111 Oldenburg, Germany.
Email: `Martin.Fraenzle@Informatik.Uni-Oldenburg.de`

a very rich class of models: that of systems exhibiting *finitely variable trajectories*, i.e. trajectories that can only exhibit a finite, yet *unbounded*, number of state changes within finite time. Some of the undecidability proofs fail immediately if the number of possible state changes over finite intervals of time is finitely bounded.

On the other hand, Duration Calculus is intended to be a design instrument for embedded real-time systems, which are, like any technical system, band-limited. The possibility to have arbitrarily short transients, which is offered by finitely variable models, is thus unimportant for describing such systems. Undecidability results which depend *crucially* on finite variability do, therefore, not provide a direct indication as to whether model-checking real embedded system designs against Duration Calculus specifications is possible or not. This insight has led us to investigate decidability of those subsets of Duration Calculus, where the undecidability results crucially depend on finite variability, over more restricted model classes. The findings obtained with respect to the feasibility of model-checking Duration Calculus are reviewed in this article. The detailed constructions underlying these results can be found in [Frä97].

The structure of this article is as follows: we start by introducing Duration Calculus in Section 2 and then proceed in Section 3 by providing an interpretation of timed automata [AD94] and timed regular expression [ACM97] that is consistent with the trajectory-based standard interpretation of Duration Calculus. Sections 4 to 6 reveal model-checking procedures for specific device classes. These device classes range from dynamically unconstrained timed automata, where the model property turns out to be undecidable for any subset of Duration Calculus featuring metrics, negation, and chop, over band-limited timed automata modelling asynchronous circuits, to synchronously clocked systems. Section 9 discusses the results and their relevance to DC-based system design. Finally, Appendix A reveals an effective construction for recognising the time-wise discrete counterexamples, i.e. those corresponding to synchronously clocked behaviour, of Duration-Calculus formulae.

2. Duration Calculus

Duration Calculus (abbreviated DC in the remainder) is a metric-time temporal logic that is specially tailored towards reasoning about durational constraints on time-dependent Boolean-valued states. The syntax of the subsets of DC formulae that we will study is

$$\begin{array}{rcl}
\langle formula\rangle & ::= & \langle atomicform\rangle \mid \\
 & & \neg\langle formula\rangle \mid \\
 & & (\langle formula\rangle \wedge \langle formula\rangle) \mid \\
 & & (\langle formula\rangle \frown \langle formula\rangle) \\
k & ::\in & \mathbb{N} \\
\langle state\,assertion\rangle & ::= & \langle variable\rangle \mid \neg\langle state\,assertion\rangle \mid (\langle state\,assertion\rangle \wedge \langle state\,assertion\rangle) \\
\langle variable\rangle & ::\in & Varname \ ,
\end{array}$$

where *Varname* is a countable set of variable names.

Concerning the available atomic formulae, we distinguish different subsets of DC. In the first subset we investigate, often called the $\{\llbracket P\rrbracket, l = k\}$-fragment [ZHS93] or $RDC_1(1)$ [HZ97], atomic formulae take the form

$$\begin{array}{rcl}
\langle atomicform\rangle & ::= & \llbracket\langle state\,assertion\rangle\rrbracket \mid \\
 & & l = k \ ,
\end{array}$$

while in the so-called $\{\int P = k\}$-fragment,

$$\langle atomicform\rangle \quad ::= \quad \int \langle state\,assertion\rangle = k \ .$$

Furthermore, we will briefly encounter the $\{\llbracket P\rrbracket\}$-fragment (called RDC in [HZ97]), where atomic formulae are *only* of the form

$$\langle atomicform\rangle \quad ::= \quad \llbracket\langle state\,assertion\rangle\rrbracket \ ,$$

and the $\{\int P = \int Q\}$-fragment (called RDC_2 in [HZ97]), where atomic formulae take the form

$$\langle atomicform\rangle \quad ::= \quad \int\langle state\ assertion\rangle = \int\langle state\ assertion\rangle \ .$$

Table 1. Semantics of Duration Calculus

Satisfaction of a formula ϕ *by an observation* $(tr, [a,b])$, denoted $tr, [a,b] \models \phi$, is defined by

$$
\begin{array}{lcll}
tr, [a,b] & \models & \int P = k & \text{iff} \quad \int_a^b \chi \circ [\![P]\!] \circ tr(t)\,\mathrm{dt} = k \ , \\
tr, [a,b] & \models & \int P = \int Q & \text{iff} \quad \int_a^b \chi \circ [\![P]\!] \circ tr(t)\,\mathrm{dt} = \int_a^b \chi \circ [\![Q]\!] \circ tr(t)\,\mathrm{dt} \ , \\
tr, [a,b] & \models & \lceil P \rceil & \text{iff} \quad b > a \wedge \int_a^b \chi \circ [\![P]\!] \circ tr(t)\,\mathrm{dt} = b - a \ , \\
tr, [a,b] & \models & l = k & \text{iff} \quad b - a = k \ , \\
tr, [a,b] & \models & \neg\phi & \text{iff} \quad tr, [a,b] \not\models \phi \ , \\
tr, [a,b] & \models & (\phi \wedge \psi) & \text{iff} \quad tr, [a,b] \models \phi \ \text{ and } \ tr, [a,b] \models \psi \ , \\
tr, [a,b] & \models & (\phi \frown \psi) & \text{iff} \quad \exists\, m \in [a,b] \,.\, \begin{pmatrix} tr, [a,m] \models \phi \quad \text{and} \\ tr, [m,b] \models \psi \end{pmatrix} \ ,
\end{array}
$$

where $[\![P]\!](\sigma)$ canonically lifts a Boolean-valued interpretation $\sigma : \mathit{Varname} \to \mathbb{B}$ of the state variables to an interpretation of the state assertion P, e.g. $[\![a \wedge \neg c]\!](\sigma) = \sigma(a) \wedge \neg\sigma(c)$, and χ maps truth values to $\{0,1\}$ according to the convention $\chi(\mathtt{false}) = 0$ and $\chi(\mathtt{true}) = 1$.
A trajectory tr *satisfies a formula* ϕ, denoted $tr \models \phi$, iff any prefix-observation of tr satisfies ϕ — formally, $tr \models \phi$ iff $tr, [0,t] \models \phi$ for each $t \in \mathbb{R}_{\geq 0}$. Note that this is the original definition of satisfaction as used in [ZHR91] and that a different notion, namely $tr \models \phi$ iff $tr, [b,e] \models \phi$ for each $b \leq e \in \mathbb{R}_{\geq 0}$, has been introduced in more recent expositions, e.g. [HZ97]. We stick to the original definition as it yields a strictly more expressive logics, being able to simulate the new definition through encoding ϕ by $\neg(\mathtt{true} \frown \neg\phi)$ and, furthermore, being able to express initialisation properties not covered by the new semantics.

Duration Calculus is interpreted over trajectories

$$tr \in \mathit{Traj} \stackrel{\text{def}}{=} \{tr : \mathit{Time} \to \mathit{Varname} \to \mathbb{B} \mid \mathit{Time} = \mathbb{R}_{\geq 0}, tr \text{ finitely variable}\}$$

that provide a finitely variable, time-dependent, Boolean-valued interpretation of variables. Finite variability means that only finitely many state changes may occur within any finite time interval. Satisfaction of a formula ϕ by a trajectory tr is defined as a limit property over a chain of finite chunks from tr called *observations*, where an observation is a pair $(tr, [a,b]) \in \mathit{Obs} \stackrel{\text{def}}{=} \mathit{Traj} \times \mathit{TimeInterval}$ with $\mathit{TimeInterval}$ being the set of finite closed intervals in $\mathbb{R}_{\geq 0}$. The definition of satisfaction of a formula ϕ by observations and trajectories is given in Table 1. The set of models of ϕ, i.e. trajectories satisfying ϕ, is denoted $\mathcal{M}[\![\phi]\!]$. As usual, we say that ϕ *is valid* iff $\mathcal{M}[\![\phi]\!] = \mathit{Traj}$, and we say that $\mathit{Traj} \setminus \mathcal{M}[\![\phi]\!]$ is the set of *counterexamples of* ϕ.

According to [ZHS93], validity is undecidable for the fragment $\{\lceil P \rceil, l = k\}$. Consequently, the same applies for the fragment $\{\int P = k\}$, as $\lceil P \rceil$ can be encoded as $\int \neg P = 0 \wedge \neg(\int \mathtt{true} = 0)$ and $l = k$ as $\int \mathtt{true} = k$, resp.[2] However, these undecidability results for Duration Calculus crucially depend on the particular notion of trajectory over time used by the Duration Calculus and most other dense-time temporal logics: the notion of *finitely variable trajectories*, i.e. trajectories that can only exhibit a finite, yet *unbounded*, number of state changes within finite time. Unboundedness is crucial to the proof of undecidability, as the proof reduces the halting problem of two-counter machines to a validity problem of certain duration formulae via the following basic proof pattern:

Proof pattern 2.1. Encode the computations of a two-counter machine by the models of an effectively constructed duration formula, representing the counter values by a corresponding number of state changes of two Boolean variables within time intervals of some fixed length k.

Clearly, this proof pattern fails if the number of possible state changes over finite intervals of time is finitely bounded. Yet, given that Duration Calculus is intended to be a design instrument for embedded real-time systems, it is questionable whether the full range of finitely variable trajectories should be regarded crucial to validity of DC formulae. As any technical system is band-limited, the possibility to have arbitrarily short transients, which is offered by finitely variable models, is unimportant for describing technical systems. In most (if not all) application domains, more restrictive constraints on the temporal evolution of trajectories can be justified from physical properties of the systems. Therefore, it makes sense to investigate decidability of the model property for Duration Calculus on suitably sparser model classes. For doing so, we first have to expand on the models that we want to check against DC specifications.

[2] Here and in the sequel, we take the liberty of using the standard abbreviations from propositional logic in the context of state assertions as well as formulae: $\mathtt{true}$ abbreviates an arbitrary tautological state assertion or — depending on context — formulae, e.g. $\neg(s \wedge \neg s)$ for some state assertion s, and $p \vee q$ abbreviates $\neg(\neg p \wedge \neg q)$, etc.

3. A trajectory-based interpretation of timed automata and timed regular expressions

A convenient way to describe real-time behaviour is the timed automata model of R. Alur and D. Dill [AD94]. Timed automata are finite automata equipped with clocks. The clocks can be reset upon transitions and can be compared to integer constants in guard expressions in order to impose time-dependent constraints upon the enabledness of the guarded transitions. In analogy to regular languages, which are the languages recognised by finite automata, the languages recognised by timed (finite) automata are called *timed regular languages*.

We will assume in the remainder that the models to be checked are represented by timed ω-automata. As DC is interpreted over trajectories while timed automata are classically interpreted over timed traces, a common semantic basis for formulating the model property has to be defined. We do so by mapping timed traces to trajectories and finite timed traces to prefix-observations.

3.1. Mapping timed traces to trajectories or observations

A *finite (or infinite) timed trace* over an alphabet α is a string $\langle(a_0,t_0),(a_1,t_1),\ldots,(a_n,t_n)\rangle \in (\alpha \times \mathit{Time})^*$ ($\langle(a_0,t_0),(a_1,t_1),\ldots\rangle \in (\alpha \times \mathit{Time})^\omega$, resp.) with $t_0 = 0$ and $\forall\, i \in \mathbb{N}_{<n} \,.\, t_i < t_{i+1}$ ($\forall\, i \in \mathbb{N} \,.\, t_i < t_{i+1}$, resp.).[3] The interpretation is that letter a_i is recognised at time instant t_i. If the alphabet α is of the form $V \to \mathbb{B}$, with $V \subset \mathit{Varname}$ being finite, an observation-based interpretation of finitely accepting timed automata can be easily derived from their trace-based standard interpretation by assigning to each finite timed trace $\langle(a_0,t_0),(a_1,t_1),\ldots,(a_n,t_n)\rangle$ the set

$$O\left(\langle(a_0,t_0),(a_1,t_1),\ldots,(a_n,t_n)\rangle\right) \stackrel{\text{def}}{=} \left\{ (tr,[t_0,t_n]) \;\middle|\; \begin{array}{l} tr \in \mathit{Traj}, \\ \forall\, i \in \mathbb{N}_{<n} \,.\, \forall\, t \in (t_i,t_{i+1}] \,.\, tr(t)\mid_V = a_i \end{array} \right\}$$

of left-continuous observations. Lifting this to finitely accepting timed automata by assigning to a finitely accepting timed automaton A the set

$$O(A) \stackrel{\text{def}}{=} \bigcup_{t \text{ is accepted by } A} O(t)$$

we obtain an observation-based interpretation of finitely accepting timed automata. Similarly, an infinite timed trace $\langle(a_0,t_0),(a_1,t_1),\ldots\rangle$ is mapped to the set

$$T\left(\langle(a_0,t_0),(a_1,t_1),\ldots\rangle\right) \stackrel{\text{def}}{=} \left\{ tr \;\middle|\; \begin{array}{l} tr \in \mathit{Traj}, \\ \forall\, i \in \mathbb{N} \,.\, \forall\, t \in (t_i,t_{i+1}] \,.\, tr(t)\mid_V = a_i \end{array} \right\}$$

of left-continuous trajectories. Lifting this to timed ω-automata by assigning to an automaton A the set

$$T(A) \stackrel{\text{def}}{=} \bigcup_{t \text{ is accepted by } A} T(t)$$

we obtain a trajectory-based interpretation of timed ω-automata.

We will not expand further upon timed automata here. The reader interested in these devices is referred to the literature, in particular the survey [AD94]. The reason for not going deeper into timed automata is that we will avoid explicit treatment of timed automata in the remainder and instead use a more abstract notation in our constructions: that of timed regular expressions [ACM97].

3.2. Timed regular expressions

For untimed regular languages, representation by regular expressions is often found to be much more convenient than construction of recognising automata. The quest for a similar representation of timed regular

[3] Some authors prefer non-decreasing instead of strictly increasing time sequences. We follow [ACM97] and [Wil94b] in using strictly increasing sequences, which allows us to build upon their results.

languages has been settled by E. Asarin, P. Caspi, and O. Maler in [ACM97], where they define a notion of *timed regular expressions* that characterises the observation sets recognised by timed automata in very much the same way that regular expressions characterise the languages accepted by finite automata. It is worthwhile to note that their construction characterises the observation sets $O(A)$ or trajectory sets $T(A)$ recognised by timed automata and *not* the sets of timed traces that timed automata recognise according to the classical interpretation. This makes a difference, as in timed traces the same symbol may occur multiple times in a row, which remains invisible in the associated observations. E.g., $\langle(0,\mathrm{a}),(0.75,\mathrm{a}),(2.1,\mathrm{b})\rangle \neq \langle(0,\mathrm{a}),(2.1,\mathrm{b})\rangle$, but $O(\langle(0,\mathrm{a}),(0.75,\mathrm{a}),(2.1,\mathrm{b})\rangle) = O(\langle(0,\mathrm{a}),(2.1,\mathrm{b})\rangle)$.[4]

The syntax of timed regular expressions resembles that of untimed regular expressions, except that a means to control time progress is added and that — as timed regular languages are in contrast to untimed ones not closed under complementation [AD94] — intersection is explicitly present. The set of *timed regular expressions* over some finite alphabet α is defined by the context-free grammar

$$
\begin{array}{rcl}
\langle \mathit{timed\,reg\,exp}\rangle & ::= & \langle \mathit{letter}\rangle \mid \langle \mathit{timed\,reg\,exp}\rangle \vee \langle \mathit{timed\,reg\,exp}\rangle \mid \\
 & & \langle \mathit{timed\,reg\,exp}\rangle \wedge \langle \mathit{timed\,reg\,exp}\rangle \mid \langle \mathit{timed\,reg\,exp}\rangle \cdot \langle \mathit{timed\,reg\,exp}\rangle \mid \\
 & & \langle \mathit{timed\,reg\,exp}\rangle^{*} \mid \langle \mathit{timed\,reg\,exp}\rangle_{\langle \mathit{time\,constraint}\rangle} \\
\langle \mathit{timed}\,\omega\,\mathit{reg\,exp}\rangle & ::= & \langle \mathit{timed\,reg\,exp}\rangle \cdot \langle \mathit{timed\,reg\,exp}\rangle^{\omega} \\
\langle \mathit{letter}\rangle & ::\in & \alpha \\
\langle \mathit{time\,constraint}\rangle & ::= & [\langle \mathit{bound}\rangle, \langle \mathit{bound}\rangle] \mid (\langle \mathit{bound}\rangle, \langle \mathit{bound}\rangle] \mid \\
 & & [\langle \mathit{bound}\rangle, \langle \mathit{general\,bound}\rangle) \mid (\langle \mathit{bound}\rangle, \langle \mathit{general\,bound}\rangle) \\
\langle \mathit{bound}\rangle & ::\in & \mathbb{N} \\
\langle \mathit{general\,bound}\rangle & ::\in & \mathbb{N} \cup \{\infty\} \ .
\end{array}
$$

Timed regular expressions are interpreted by sets of observations and timed ω-regular expressions over trajectories. A letter $a \in \alpha$ denotes an observation of arbitrary length that has value a within the observation interval.

$$[\![a]\!] \stackrel{\mathrm{def}}{=} \{(tr,[0,e]) \mid tr \in \mathit{Traj}(\alpha),\ e \in \mathit{Time},\ \forall t \in (0,e]\,.\,tr(t) = a\ \} \quad .$$

Note that actually a set of observations is assigned, as neither the length of the observation nor its value in the left end-point of the observation interval, i.e. at time instant 0, is constrained.

$\vee$, $\wedge$, and $\cdot$ denote union, intersection, and concatenation of observation sets.

$$
\begin{aligned}
&[\![\phi \vee \psi]\!] \stackrel{\mathrm{def}}{=} [\![\phi]\!] \cup [\![\psi]\!] \ , \\
&[\![\phi \wedge \psi]\!] \stackrel{\mathrm{def}}{=} [\![\phi]\!] \cap [\![\psi]\!] \ , \\
&[\![\phi \cdot \psi]\!] \stackrel{\mathrm{def}}{=} \left\{ (tr,[0,e]) \,\middle|\, \exists m \in [0,e]\,.\, \begin{pmatrix} (tr,[0,m]) \in [\![\phi]\!] \wedge \\ (t \mapsto tr(t+m),[0,e-m]) \in [\![\psi]\!] \end{pmatrix} \right\} \quad .
\end{aligned}
$$

The star-operator denotes the timed version of the Kleene star, i.e. finitely iterated catenation of the enclosed timed regular expression:

$$[\![\phi^{*}]\!] \stackrel{\mathrm{def}}{=} \bigcup_{i \in \mathbb{N}} [\![\phi^{i}]\!] \ ,$$

where ϕ^i denotes $\phi \cdot \phi \cdot \ldots \cdot \phi$ with i occurrences of ϕ for $i > 0$ and *point* for $i = 0$. Here, *point* is an abbreviation for an arbitrary timed regular expression that corresponds to all point observations, e.g. for the timed regular expression $a_{[0,0]}$ with arbitrary $a \in \alpha$.

---

[4] However, it should be noted that in the original definition of Asarin, Caspi, and Maler, finitely accepting timed automata and timed regular expressions are interpreted by a slightly different notion of observation which is called 'signal' by the authors. Signals are *partial* functions from *Time* to α that have a left-open, right-closed interval as domain. The set of signals assigned to a timed trace $\langle(a_0,t_0),(a_1,t_1),\ldots,(a_n,t_n)\rangle$ in [ACM97] is

$$O_{\mathrm{ACM}}(A) \stackrel{\mathrm{def}}{=} \left\{ tr\big|_{(t_0,t_n]} \,\middle|\, \begin{array}{l} tr \in \mathit{Traj}(\alpha), \\ \forall i \in \mathbb{N}_{<n}\,.\,\forall t \in (t_i,t_{i+1}]\,.\,tr(t) = a_i \end{array} \right\} \ .$$

This difference in the shape of observations is, however, marginal and does not affect the theory developed. We therefore prefer to use the same kind of observations for interpreting DC and timed regular expressions.

Finally, ϕ_I denotes those interpretations of ϕ that have a length that obeys the constraint I:

$$[\![\phi_I]\!] \stackrel{\text{def}}{=} \left\{(tr,[0,e]) \,\middle|\, (tr,[0,e]) \in [\![\phi]\!],\ e \in \tilde{I}\right\} \quad ,$$

where $\tilde{I}$ is that interval in *Time* that is the natural interpretation of the constraint I. E.g., for $I = (n,m]$ it is $\{x \in \mathit{Time} \mid n < x \leq m\}$.

For a timed ω-regular expression, the semantics is

$$[\![\phi \cdot \psi^\omega]\!] \stackrel{\text{def}}{=} \left\{ tr \,\middle|\, \exists\, e_0, e_1, \ldots \in \mathbb{R}_{\geq 0} \,.\, \begin{pmatrix} (tr,[0,e_0]) \in [\![\phi]\!] \wedge \\ \forall\, i \in \mathbb{N} \,.\, (tr,[e_i,e_{i+1}]) \in [\![\psi]\!] \end{pmatrix} \right\} \quad .$$

The design goal of timed regular expressions has been to obtain a timed counterpart of Kleene's theorem relating finite automata and regular expressions [Kle56]. This goal has been achieved for Asarin's, Caspi's, and Maler's interpretation of timed automata and timed regular expressions, as the main theorems of [ACM97] show. As our interpretation of timed automata and timed regular expressions is only marginally different from theirs (and in particular consistently different for timed automata and timed regular expressions, s.t. it can be derived from theirs by applying the same mapping to both their interpretation of timed automata and their interpretation of timed regular languages), these results carry over to our interpretation. This yields the following theorem.

Theorem 3.1 [ACM97]. Every set of observations that is definable by a timed regular expression can be accepted by a finitely accepting timed automaton. Every set of trajectories that is definable by a timed ω-regular expression can be accepted by a timed ω-automaton. $\Box$

As the construction given in [ACM97] in support of this theorem is effective this suggests that timed regular expressions are an adequate means for description of observation or trajectory sets that can be recognised by timed automata. By decidability of the emptiness problem for languages accepted by timed automata [AD94, Theorem 4.16] this implies furthermore

Corollary 3.2. It is decidable whether a timed regular expression denotes the empty set of observations and it is decidable whether a timed ω-regular expression denotes the empty set of trajectories. $\Box$

4. The model-checking problem

We will now turn attention to automatic verification of timed automata against DC specifications. We start by introducing the model-checking problem and investigating it on finitely variable models. We will show that the undecidability results for dense-time DC then carry over to the model-checking problem and hence model-checking is impossible for all the above-mentioned subsets of DC. Nevertheless, we will be able to outline a procedure for some subset without general negation. Besides being useful itself, this result is a stepping stone towards the more general model-checking procedures obtained from behavioural restrictions.

The purpose of *model-checking* is to provide an *automatic* check of whether a given device satisfies a specification, i.e. is a model of the specification formulae. Here, we assume that devices are represented by timed ω-automata (e.g. timed Büchi or timed Muller automata [AD94]) representing their dynamic behaviour and that specifications are formalised through DC formulae. W.l.o.g. we require that automata alphabets have the form $V \to \mathbb{B}$, with $V \subset \mathit{Varname}$ being finite, s.t. automata map to a corresponding set $T(A) \subseteq \mathit{Traj}$ of possible trajectories as shown in Section 3.

We then call A *a model of a specification* ϕ iff each possible trajectory of A satisfies ϕ, i.e. iff $T(A) \subseteq \mathcal{M}[\![\phi]\!]$. We say that A *subject to a behavioural constraint* $X \subset \mathit{Traj}$ *is a model of a specification* ϕ iff each trajectory of A that conforms to the behavioural constraint satisfies ϕ, i.e. iff $T(A) \cap X \subseteq \mathcal{M}[\![\phi]\!]$. Accordingly, *model-checking timed automata against formulae from some given subset of DC is possible (possible with respect to behavioural constraint X)* iff there is a decision procedure that, given a timed automaton A and a formula ϕ from that subset, decides whether A is a model of ϕ (or whether A subject to X is a model of ϕ, resp.).

Now, it is easy to see that model-checking timed automata against any of the above-mentioned fragments of DC featuring metric time is impossible:

Theorem 4.1. It is in general undecidable whether a timed automaton A is a model of some $\{\lceil\!\lceil P \rceil\!\rceil, l = k\}$-formula ϕ. The same holds true for $\{\int P = \int Q\}$-formulae and for $\{\int P = k\}$-formulae.

Proof. It is straightforward to construct a timed automaton A with $T(A) = \mathit{Traj}$. Such an automaton is a model of a formula ϕ iff ϕ is valid. Hence, deciding the model property subsumes deciding validity. However, the $\{\llbracket P \rrbracket, l = k\}$-fragment and the $\{\int P = \int Q\}$-fragment of DC are both undecidable according to [ZHS93, Theorems 11 and 12]. Consequently, model-checking timed automata against $\{\llbracket P \rrbracket, l = k\}$-formulae or $\{\int P = \int Q\}$-formulae is impossible.

Finally, the $\{\llbracket P \rrbracket, l = k\}$-fragment can be effectively embedded into the $\{\int P = k\}$-fragment through encoding $\llbracket P \rrbracket$ by $(\int \neg P = 0) \wedge \neg(\int \mathtt{true} = 0)$ and $l = k$ by $\int \mathtt{true} = k$. Hence, undecidability of the $\{\llbracket P \rrbracket, l = k\}$-fragment of Duration Calculus implies undecidability of the $\{\int P = k\}$-fragment and thus implies impossibility of model-checking for this subset also. □

Thus, model-checking timed automata without behavioural restrictions against DC specifications is impossible, unless one is willing to restrict the specification language by breaking the trinity of metrics, negation, and chop. E.g., abandoning full negation we obtain the following model-checking procedure:

Theorem 4.2. Let A be a timed automaton and ϕ a $\{\llbracket P \rrbracket, l < k, l = k, l > k\}$-formula that contains exactly one, outermost negation.[5] Then it is decidable whether A is a model of ϕ.

Proof. Due to the similarity of timed regular expressions with negation-free formulae, an effective mapping of formulae of the $\{\llbracket P \rrbracket, l < k, l = k, l > k\}$-fragment of Duration Calculus that do contain *exactly one, outermost, negation* to timed ω-regular expressions and thus to timed ω-automata that recognise their left-continuous counterexamples of finite variability can be defined: first, we observe that the left-continuous counterexamples of $\neg\phi$ are exactly the trajectories starting with a prefix-observation satisfying ϕ, and thus define

$$\mathit{Counterexamples}(\neg\phi) \stackrel{\text{def}}{=} \mathit{PrefixModels}(\phi) \cdot \alpha^\omega \quad ,$$

where $\alpha \stackrel{\text{def}}{=} \mathit{free}(\phi) \to \mathbb{B}$ is the set of Boolean valuations of the free variables of ϕ. Then we capture the (left-continuous) prefix-observations satisfying a negation-free $\{\llbracket P \rrbracket, l < k, l = k, l > k\}$-formula by defining

$$\begin{aligned}
\mathit{PrefixModels}(\llbracket P \rrbracket) &\stackrel{\text{def}}{=} \left(\left(\bigvee\nolimits_{a \in \alpha, a \models P} a \right)^* \right)_{(0,\infty)} \quad , \\
\mathit{PrefixModels}(l < k) &\stackrel{\text{def}}{=} (\alpha^*)_{[0,k)} \quad , \\
\mathit{PrefixModels}(l = k) &\stackrel{\text{def}}{=} (\alpha^*)_{[k,k]} \quad , \\
\mathit{PrefixModels}(l > k) &\stackrel{\text{def}}{=} (\alpha^*)_{(k,\infty)} \quad , \\
\mathit{PrefixModels}(\phi \wedge \psi) &\stackrel{\text{def}}{=} \mathit{PrefixModels}(\phi) \wedge \mathit{PrefixModels}(\psi) \quad , \\
\mathit{PrefixModels}(\phi \vee \psi) &\stackrel{\text{def}}{=} \mathit{PrefixModels}(\phi) \vee \mathit{PrefixModels}(\psi) \quad , \\
\mathit{PrefixModels}(\phi \frown \psi) &\stackrel{\text{def}}{=} \mathit{PrefixModels}(\phi) \cdot \mathit{PrefixModels}(\psi) \quad .
\end{aligned}$$

Thus, ϕ can be effectively mapped to a timed automaton A_ϕ recognising its left-continuous counterexamples. As timed automata are effectively closed under intersection, a timed automaton $A \times A_\phi$ that recognises the empty language iff no left-continuous trajectory of A is a counterexample to ϕ can be constructed effectively. All trajectories of A are by definition left-continuous, thus $A \times A_\phi$ recognises the empty language iff A is a model of ϕ. As language emptiness is decidable for timed automata [AD94], this yields a decision procedure for A being a model of ϕ. □

Unfortunately, it follows from Theorem 4.1 that this procedure cannot be extended to deal with inner negation also. While abandoning full negation surely is a dramatic restriction to expressiveness — e.g. straightforward specifications of what the system should do can no longer be mixed with specifications of

[5] I.e., there are no inner negations in formulae, yet one at the outermost level. The atomic formulae are of the forms $\llbracket P \rrbracket$, $l < k$, $l = k$, $l > k$, and the connectives available are conjunction, disjunction, and chop. Note that this subset covers the so-called DC-implementables [Rav95].

The new atomic formulae $l < k$ and $l > k$, as well as the new connective $\vee$, take the obvious semantics. They were added because they are no longer encodable due to the lack of full negation.

what it should never do — it should be noted that the remaining logics is nevertheless useful: it covers the so-called DC implementables [Rav95] that A.P. Ravn considers to be a stepping stone from DC specifications to implementations.

5. Bounded variability

Considering the fact that any physically realizable reactive system is subject to band-limitation, an easily justifiable assumption on *realistic* device models is that reactions can only come arbitrarily close in time if they originate from different subsystems. As the number of subsystems in a given system is finite, this implies that the number of events observable within a time unit is bounded by a system-dependent natural number. An appropriate behavioural model is that of *n-bounded trajectories*, which are those trajectories that exhibit at most n state changes over any unit-length interval of time, where $n \in \mathbb{N}_{\geq 1}$ is a system-dependent parameter. Such trajectories provide a natural notion of band-limitation and an appropriate n is often straightforwardly determined from the structure of the system. E.g. a system-level description of a parallel composition of k independently clocked devices with clock frequency f requires variability $n = \lfloor kf \rfloor + 1$. Furthermore, due to the universal nature of band-limitation, the n-bounded model is far more powerful than it may at first seem: in [Frä97, chapter 13] it was shown that n-bounded trajectories even suffice for a physically adequate behavioural description of *asynchronous* circuits at the *gate level*.

With respect to the model-checking problem it is interesting to note that model-checking devices against $\{\llbracket P \rrbracket, l = k\}$-specifications (with full negation) is possible if the device dynamics can be described by the n-bounded trajectories of timed automata:

Theorem 5.1. Let A be a timed automaton and ϕ a formula of the $\{\llbracket P \rrbracket, l = k\}$-fragment of Duration Calculus. Then it is decidable whether A subject to Traj_n, where Traj_n is the set of n-bounded trajectories, is a model of ϕ.

Proof. In the proof of Theorem 4.2, an effective mapping $\mathit{PrefixModels}(\phi)$ of negation-free $\{\llbracket P \rrbracket, l < k, l = k, l > k\}$-formulae to timed regular expressions and thus to timed automata recognising their satisfying left-continuous prefix-observations has been outlined. According to [Wil94a], the timed regular languages, i.e. the languages recognised by timed automata, are furthermore effectively closed under complementation relative to the set of n-bounded trajectories. I.e., there is an effective procedure that assigns to any timed automaton A a timed automaton recognising the language $\mathrm{pref}(\mathit{Traj}_n) \setminus O(A)$, where $\mathrm{pref}(\mathit{Traj}_n)$ is the set of left-continuous prefix observations of n-bounded variability. Hence, the model-checking procedure of Theorem 4.2 can be extended to deal with inner negation also if only the n-bounded counterexamples are of interest. Therefore, any formula ϕ of the $\{\llbracket P \rrbracket, l = k\}$-fragment of DC can be effectively assigned a timed automaton A_ϕ recognising its left-continuous n-bounded counterexamples. This effectively reduces the model-checking problem $T(A) \cap \mathit{Traj}_n \subseteq \mathcal{M}\llbracket \phi \rrbracket$ to the timed regular language emptiness problem $T(A) \cap T(A_\phi) = \emptyset$. This language emptiness problem is decidable due to the effective closure of timed regular languages under intersection and decidability of emptiness of a timed regular language [AD94]. $\square$

Unfortunately, above result does not generalise to the $\{\int P = k\}$-subset of DC, which features accumulated durations and thus is considerably more expressive than the $\{\llbracket P \rrbracket, l = k\}$-fragment:

Theorem 5.2. For specifications ϕ from the $\{\int P = k\}$-fragment of Duration Calculus it is in general undecidable whether a timed automaton A subject to Traj_n is a model of ϕ.

Proof. As there are timed automata A with $T(A) \supseteq \mathit{Traj}_n$, an appropriate model-checking procedure would be able to decide whether $\mathit{Traj}_n \subseteq \mathcal{M}\llbracket \phi \rrbracket$, i.e. whether all n-bounded trajectories satisfy ϕ. However, in [Frä97, Theorem 6.7] it was shown (using a real-time pumping lemma) that this property subsumes the validity problem of the $\{\int P = k\}$-fragment in the following sense: for any $\{\int P = k\}$-formula ψ there is an effectively constructible $\{\int P = k\}$-formula ψ_{slow} s.t. ψ_{slow} is satisfied by all n-bounded trajectories iff ψ is valid.[6]

[6] The construction of ϕ_{slow} from a formula ϕ is amazingly simple. Let tf be a state variable that does not occur in ϕ. ϕ_{slow} is then derived from ϕ by replacing any duration $\int P$ occurring in ϕ by the duration $\int (P \wedge \mathit{tf})$. The intuitive effect of this procedure is that tf plays the role of a flag indicating which parts of the trajectory are relevant to durations (and thus to satisfaction of the formula). Whenever tf is true, the duration $\int P$ actually accumulates. But if tf is false over some interval of time then

As the validity problem of the $\{\int P = k\}$-fragment is undecidable, it is thus in general undecidable whether all n-bounded trajectories satisfy a formula ϕ of the $\{\int P = k\}$-fragment, and consequently it is also in general undecidable whether a timed automaton subject to $Traj_n$ is a model of ϕ. $\square$

6. Time-wise discrete trajectories

In order to obtain model-checking procedures for the $\{\int P = k\}$-fragment, we may try to reduce the model class that is regarded crucial to the correctness problem still further. If the devices to be designed are fully synchronous and evenly clocked, it makes sense to consider trajectories that are changing state only at evenly spaced time instants. This is captured by studying DC formulae over *time-wise discrete trajectories*, where a trajectory $tr \in Traj$ is called time-wise discrete iff it has discontinuities (i.e., state changes) only in time instants which are multiples of the time unit. The set of time-wise discrete trajectories is denoted $DTraj$.

The restriction to time-wise discrete models allows model-checking even with respect to specifications in the $\{\int P = k\}$-fragment of DC:

Theorem 6.1. Let A be a timed automaton and ϕ a $\{\int P = k\}$-formula. Then it is decidable whether A subject to $DTraj$ is a model of ϕ.

Proof. For any formula ϕ of the $\{\int P = k\}$-fragment, $DTraj \setminus \mathcal{M}[\![\phi]\!]$ is an unrolling of an ω-regular language to real-time based on the convention that one letter per time unit is traversed; a corresponding ω-automaton Aut_ϕ can be effectively constructed (the construction is presented in Appendix A). Similarly, $T(A) \cap DTraj$ corresponds to an analogous unrolling of an ω-regular language to real-time and a corresponding ω-automaton Aut_A can again be effectively constructed. Due to these constructions, the problem of deciding $T(A) \cap DTraj \subseteq \mathcal{M}[\![\phi]\!]$ can be effectively reduced to the ω-regular language emptiness problem $\mathcal{L}_{Aut_A} \cap \mathcal{L}_{Aut_\phi} = \emptyset$, which is decidable [Tho90]. $\square$

As the time-wise discrete trajectories of timed automata provide an adequate model of the behaviour of evenly clocked devices, above theorem implies that synchronous circuits can well be model-checked against $\{\int P = k\}$ specifications. As e.g. the ProCoS gas-burner requirements specification [RRH93] can be expressed in this subset of DC, this allows the automatic verification of synchronously clocked embedded controllers for the gas-burner directly against the *requirements* specification. In fact, we have gone even a bit further in [Frä97, Chapter 9], where we have used the construction of Aut_ϕ in combination with effective strategy construction procedures from the theory of ω-regular games [Tho95] for automatically synthesising a synchronous controller directly from the requirements specification of the gas-burner.

7. Computational complexity

It is the complexity of the new model-checking procedures where the bad news comes: even for the simplest subsets of DC that feature chop and negation — not even metric time is necessary — the worst case complexity of the model-checking problem is non-elementary in the size of the specification, irrespective of the particular trajectory class used. The reason is that chop and negation are akin to concatenation and complement of languages s.t. the non-elementary emptiness problem of extended regular languages can be linearly encoded in DC (see Theorem A.14 for the details of such an encoding).

However, the non-elementary complexity of the decision procedure need not be an obstacle to practical applications, as it is only through deep nesting of chop and negation that the worst-case complexity is encountered. Such nesting hardly ever occurs in practice, except within iterated application of some derived standard operators. In most cases, more efficient decision procedures than obtained through just unfolding their definitions can be easily devised for these derived operators. While such a special treatment of derived operators does not influence the worst-case complexity, it does reduce complexity of checking formulae where deep nesting of negation and chop *only* occurs through iterated application of derived operators. In a

this interval does not contribute to any duration in ϕ_{slow} such that registration of the passage of time effectively pauses during this period. Hence, introduction of *tf* allows us to introduce 'pauses' into the trajectories which do not affect satisfaction of the formula ϕ_{slow}, but can serve to reduce variability of the trajectory to a finitely bounded value. Details of the construction can be found in [Frä97, Chapter 6.2].

prototypic implementation of a decision procedure for validity of $\{\int P = k\}$-formulae over time-wise discrete trajectories performed by the author, enhanced treatment of timed leads-to and some other frequently encountered derived operators resulted in a dramatic increase of performance when applied to the gas-burner example.

Another complexity issue arises due to the architecture-dependent parameter n in model-checking n-bounded models. The procedure used for complementing timed automata with respect to n-bounded models proposed in [Wil94a] introduces a number of clocks that depends linearly on n, with the size of the individual clock constraints being independent of n. Therefore, the number of clocks as well as the accumulated size of the clock constraints in the automaton A_ϕ recognizing the n-bounded counterexamples of a $\{\llbracket P \rrbracket, l = k\}$-formula ϕ may grow linearly in n. As checking emptiness of timed automata is PSPACE-complete [AD94], we conjecture that the space-complexity of n-bounded model-checking of $\{\llbracket P \rrbracket, l = k\}$-formulae is polynomial in n. In practice, this might be a much more serious obstacle than the non-elementary complexity in the size of the specification formula, as specifications tend to remain relatively small (at least with respect to the nesting depth of negation and chop) while the architecture-dependent parameter n, modelling the band-limit of the switching dynamics in relation to the time unit, will be a very large constant for modern architectures if natural time units (e.g., seconds) are used. Note that using a smaller time unit would not change anything, as then the timing constants in the specification formula and thus the clock constraints in the generated timed automata would scale up accordingly, also yielding a polynomial-space blowup in the emptiness check.

8. Restricted variability and refinement

A “popular” argument in favour of finitely variable trajectories is that they support refinement of operations by always being able to accommodate an arbitrary finite number of additional state changes between two existent state changes. Vice versa, it was sometimes suspected that modelling or verification techniques that are explicitly confined to a given finite amount of temporal variation will not fully support operation refinement due to a possible saturation problem: a trajectory of bounded variation may be saturated in the sense that it cannot accommodate further state changes, which would prohibit splitting a state change into multiple ones during operation refinement. We will now investigate whether this problem arises with our modelling techniques.

8.1. n-bounded variability

For n-bounded trajectories, the saturation problem could take the following form: an n-bounded trajectory may be saturated in the sense that there are already n state changes within some unit interval of time, not allowing for introduction of further state changes.

However, it is easy to show that this saturation problem may only occur with pathological refinements, namely in cases where the refinement chain has anyhow entered a blind alley, no longer allowing for a viable implementation. The key issue is to adjust the variation parameter n in n-bounded verification such that the variation of even the fastest available (or affordable) implementation technology can be catered for. In order to accomplish this, we just need a safe lower bound for the minimum delay between two successive state changes on the same port within this technology. Let $\delta \in \mathbb{R}_{>0}$ time units be such a lower bound. As this is a safe lower bound for the minimum delay in the fastest technology, it is a safe lower bound for all technologies in the design space. Consequently, any port in any implementation envisaged can exhibit at most $\lfloor \delta^{-1} \rfloor + 1$ state changes per time unit. The behaviours of any design with k ports are thus clearly $k(\lfloor \delta^{-1} \rfloor + 1)$-bounded. When setting up a chain of refinement steps, it suffices to construct the models of the specification formula with this resolution, as higher resolution is anyhow non-implementable:

Lemma 8.1. Let A be a timed automaton with alphabet $\alpha = V \to \mathbb{B}$, and I an implementation with $\mathrm{Ports}(I) \supseteq V$ and in an implementation technology with a minimum delay of at least δ time units between two successive state changes on the same port. Then I refines A, i.e. $\mathcal{B}(I) \subseteq T(A)$, where $\mathcal{B}(I)$ is the set of possible behaviours of the implementation I, iff $\mathcal{B}(I) \cap \mathit{Traj}_n \subseteq T(A) \cap \mathit{Traj}_n$ for $n \geq |\mathrm{Ports}(I)|\ (\lfloor \delta^{-1} \rfloor + 1)$.

Proof. $\mathcal{B}(I) \subseteq T(A)$ trivially implies $\mathcal{B}(I) \cap \mathit{Traj}_n \subseteq T(A) \cap \mathit{Traj}_n$. For the remaining implication assume $\mathcal{B}(I) \cap \mathit{Traj}_n \subseteq T(A) \cap \mathit{Traj}_n$. Then

$$
\begin{array}{ll}
& T(A) \\
= & \{ tr \in \mathit{Traj} \mid \exists\, tr' \in T(A) \,.\, tr'|_V = tr|_V \} \qquad [A \text{ does only constrain dynamics of } V] \\
\supseteq & \{ tr \in \mathit{Traj} \mid \exists\, tr' \in T(A) \cap \mathit{Traj}_n \,.\, tr'|_V = tr|_V \} \qquad [T(A) \supseteq T(A) \cap \mathit{Traj}_n] \\
\supseteq & \{ tr \in \mathit{Traj} \mid \exists\, tr' \in \mathcal{B}(I) \cap \mathit{Traj}_n \,.\, tr'|_V = tr|_V \} \qquad [\mathcal{B}(I) \cap \mathit{Traj}_n \subseteq T(A) \cap \mathit{Traj}_n] \\
\supseteq & \{ tr \in \mathit{Traj} \mid \exists\, tr' \in \mathcal{B}(I) \cap \mathit{Traj}_n \,.\, tr'|_{\mathrm{Ports}(I)} = tr|_{\mathrm{Ports}(I)} \} \qquad [\mathrm{Ports}(I) \supseteq V] \\
\supseteq & \{ tr \in \mathit{Traj} \mid \exists\, tr' \in \mathcal{B}(I) \,.\, tr'|_{\mathrm{Ports}(I)} = tr|_{\mathrm{Ports}(I)} \} \qquad [I\text{'s dynamics is } n\text{-bounded}] \\
\supseteq & B(I) \qquad \Box
\end{array}
$$

This means that with respect to refinement, n-bounded modelling comes at no cost: whenever an intermediate design A is no blind alley, i.e. has an implementation, then its restriction to n-bounded behaviours can also be refined to an implementation — even to the same implementation. If an intermediate design is a blind alley then it obviously cannot be refined to an implementation in both settings. However, this problem may become apparent earlier in n-bounded modelling, since $T(A) \cap \mathit{Traj}_n$ may be empty — which is even automatically detectable — while $T(A) \neq \emptyset$ still holds, deluding the designer to believe in implementability. In this respect, as well as with respect to automation, n-bounded modelling is an advantage. However, the price to be paid is that some information on the available implementation technologies has to be incorporated into earlier design phases, namely the lower bound δ on possible delays.

8.2. Time-wise discrete behaviour

Time-wise discrete modelling is obviously more restrictive with respect to the refinement capabilities offered. As is well-known from discrete-time modelling techniques (which time-wise discrete modelling is in a sense akin to, while it still is a dense-time technique with respect to the interpretation of e.g. the chop modality of Duration Calculus), adoption of time-wise discrete modelling incorporates an early commitment towards at least a synchronously clocked interface design and even to a particular sampling and update rate of that interface. Therefore, time-wise discrete modelling and verification techniques, as proposed in Section 6, do indeed seriously constrain the design space and thus limit the refinement capabilities.

9. Discussion

We have been able to show that by suitable restriction of the model class used in behavioural descriptions of system dynamics, model-checking for large subsets of Duration Calculus becomes feasible. The particular restrictions adopted are motivated by physical properties of practical verification problems, namely band-limitation of reactive systems and synchronicity of clocked systems. Table 2 provides an overview over the decidability results for model-checking timed automata against various fragments of dense-time Duration Calculus.

Note that our results are distinguished from other approaches towards model-checking dense-time DC in that they exploit physically justifiable restrictions on model dynamics for obtaining decidability rather than severely restricting expressiveness of the specification language. To the best of our knowledge, all other approaches (except [SHP98], which builds upon our results concerning the time-wise discrete setting) are based on either restricting the use of negation, as in [BLR95] and [Lak96], or on restricting the use of chop, as in [ZZYL94].

Finally, some of our decidability results have been further elaborated towards certain target technologies, e.g. towards synchronously clocked embedded controllers in [Frä96b] and [Frä97, Chapter 8], where for the first time ever an automatic controller synthesis technique based on dense-time Duration Calculus has been devised, and in [Frä97, Chapters 10–13], where an automatic verification procedure for asynchronous circuit correctness against $\{[\![P]\!], l = k\}$-Specifications is elaborated. The automatic synthesis procedure has been applied to the ProCoS gas-burner problem in [Frä97, Chapter 9] and [FMO99].

Table 2. Decidability of the model property for timed automata with behavioural restrictions with respect to specifications from different fragments of Duration Calculus

		Behaviour class of timed automaton		
		finitely variable	n-bounded	time-wise discrete
Subset of DC	$\{\lceil P \rceil\}$	Decidable *[ZHS93, Theorem 10]*†	Decidable*	Decidable*
	$\{\lceil P \rceil, l < k, l = k, l > k\}$ with exactly one, outermost, negation	Decidable *Theorem 4.2*‡	Decidable*	Decidable*
	$\{\lceil P \rceil, l = k\}$	Undecidable *[ZHS93, Theorem 11]*†	Decidable *Theorem 5.1*	Decidable§
	$\{\int P = k\}$	Undecidable§	Undecidable *Theorem 5.2*	Decidable *Theorem 6.1*
	$\{\int P = \int Q\}$	Undecidable *[ZHS93, Theorem 12]*†	Undecidable¶	Undecidable¶
			asynchronous circuits	synchronous circuits
		Corresponding device classes		

* As $\{\lceil P \rceil\}$ and $\{\lceil P \rceil, l < k, l = k, l > k\}$ with outermost negation are subsets of $\{\lceil P \rceil, l = k\}$, decidability results transfer from the latter to both former fragments.

† Strictly speaking, these results deal with decidability of the validity problem of DC formulae. However, using standard techniques, they can be extended to deal with decidability of the model property also.

‡ Note that this fragment of DC properly covers the so-called DC-implementables [Rav95]. Concerning decidability of the model property for another superset of the DC-implementables, a related result has been obtained by Y. Lakhnech in [Lak96].

§ As $\{\lceil P \rceil, l = k\}$ is a subset of $\{\int P = k\}$, decidability results transfer from the latter to the former fragment, and undecidability results vice versa.

¶ The proof of Theorem 12 of [ZHS93] does not depend on finite variability and thus generalises to the n-bounded and the time-wise discrete case.

Acknowledgements. Without the tight interaction in the ESPRIT Working Group "ProCoS", the research reported herein would not have come about. Special thanks go to my academic teacher Hans Langmaack for supporting this line of work over many years, to Markus Müller-Olm for reviewing earlier versions of the decidability results with scrutiny, and to Michael R. Hansen for organising the Duration Calculus Workshop at ESSLLI'98 and thus forcing me to condense the results into a single paper.

References

[ACM97] Eugene Asarin, Paul Caspi, and Oded 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.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[BLR95] Ahmed Bouajjani, Yassine Lakhnech, and Riadh Robbana. From duration calculus to linear hybrid automata. In Pierre Wolper, editor, *Computer Aided Verification (CAV '95)*, volume 939 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[FMO99] Martin Fränzle and Markus Müller-Olm. Compilation and synthesis for real-time embedded controllers. In Ernst-Rüdiger Olderog and Bernhard Steffen, editors, *Correct System Design — Recent Insights and Advances*, volume 1710 of *Lecture Notes in Computer Science*, pages 256–287. Springer-Verlag, 1999.

[Frä96a] Martin Fränzle. Decidability of duration calculi on restricted model classes. ProCoS Technical Report Kiel MF 21/1, Christian-Albrechts-Universität Kiel, Germany, July 1996.

[Frä96b] Martin Fränzle. Synthesizing controllers from duration calculus. In Bengt Jonsson and Joachim Parrow, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT '96)*, volume 1135 of *Lecture Notes in Computer Science*, pages 168–187. Springer-Verlag, 1996.

[Frä97] Martin Fränzle. *Controller Design from Temporal Logic: Undecidability need not matter.* Dissertation, Technische Fakultät der Christian-Albrechts-Universität Kiel, Germany, 1997. Available as Bericht Nr. 9710, Institut für Informatik und Prakt. Mathematik der Christian-Albrechts-Universität Kiel, Germany, and via WWW under URL `http://ca.informatik.uni-oldenburg.de/~fraenzle/diss.html`.

[HZ97] Michael R. Hansen and Zhou Chaochen. Duration calculus: Logical foundations. *Formal Aspects of Computing*, 9(3):283–330, 1997.

[Kle56] S. C. Kleene. Representations of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–42. Princeton University Press, 1956.

[Lak96] Yassine Laknech. *Specification and Verification of Hybrid and Real-Time Systems.* Dissertation, Technische Fakultät der Christian-Albrechts-Universität Kiel, Germany, 1996.

[LdRV94] 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*. Springer-Verlag, 1994.

[MS73] A. R. Meyer and L. Stockmeyer. Nonelementary word problems in automata and logic. In *Proc. AMS Symposium on Complexity of Computation*, April 1973.

[Rav95] Anders P. Ravn. *Design of Embedded Real-Time Computing Systems.* Doctoral dissertation, Department of Computer Science, Danish Technical University, Lyngby, DK, October 1995. Available as technical report ID-TR: 1995-170.

[RRH93] Anders P. Ravn, Hans Rischel, and Kirsten M. Hansen. Specifying and verifying requirements of real-time systems. *IEEE Transactions on Software Engineering*, 19(1):41–55, January 1993.

[SHP98] Manoranjan Satpathy, Dang Van Hung, and Paritosh K. Pandya. Some decidability results for duration calculus under synchronous interpretation. In A. P. Ravn and H. Rischel, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, volume 1486 of *Lecture Notes in Computer Science*, pages 186–197. Springer-Verlag, 1998.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 4, pages 133–191. North-Holland, 1990.

[Tho95] Wolfgang Thomas. On the synthesis of strategies in infinite games. In E. W. Meyer and C. Puech, editors, *Symposium on Theoretical Aspects of Computer Science (STACS 95)*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, March 1995.

[Wil94a] Thomas Wilke. *Automaten und Logiken zur Beschreibung zeitabhängiger Systeme.* Dissertation, Technische Fakultät der Christian-Albrechts-Universität Kiel, Germany, 1994.

[Wil94b] Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In Langmaack et al. [LdRV94], pages 694–715.

[ZHR91] Zhou Chaochen, C. A. R. Hoare, and Anders P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.

[ZHS93] Zhou Chaochen, Michael R. Hansen, and Peter 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.

[ZZYL94] Zhou Chaochen, Zhang Jingzhong, Yang Lu, and Li Xiaoshan. Linear duration invariants. In Langmaack et al. [LdRV94], pages 86–109.

A. Effectively representing time-wise discrete counterexamples of $\{\int P = k\}$-formulae

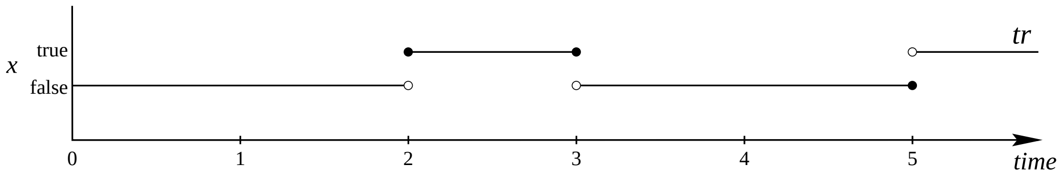
The time-wise discrete trajectories, i.e. those trajectories *tr* satisfying

$$\forall\, k \in \mathbb{N} \,.\, \forall\, \delta, \delta' \in (0,1) \,.\, tr(k+\delta) = tr(k+\delta') \quad ,$$

are the largest currently known class of trajectories that permits an effective representation of counterexamples to $\{\int P = k\}$-formulae. Here, by effective representation of counterexamples we mean that at least existence of counterexamples in the given class of trajectories should be decidable. In the case of time-wise discrete trajectories we can obtain this through a canonical representation of time-wise discrete counterexamples by ω-automata. This construction was first pursued in [Frä96a] and later, based on this, a slightly different construction was given in [SHP98]. We follow here the exposition of [Frä97, Chapter 6.3] which is directly based on [Frä96a].

We start by arbitrarily choosing a finite set $V \subset \mathit{Varname}$. As V is finite, the alphabet $\alpha \overset{\text{def}}{=} V \to \mathbb{B}$ is finite. A digitisation $\mathit{wrd}_V : \mathit{DTraj} \to \alpha^\omega$ that maps time-wise discrete observations to infinite strings over α is defined by

$$\mathit{wrd}_V(tr)(m) \overset{\text{def}}{=} \lim_{t \geq m} tr(t)\big|_V \qquad \text{for each } m \in \mathbb{N}.$$



wrd_V maps time-wise discrete trajectories to infinite strings. Given a time-wise discrete trajectory tr it yields the sequence of valuations of variables from V that tr assigns to the integer-aligned unit intervals of time. Hence, given that $V = \{x\}$ and tr is the trajectory depicted above,

$$wrd_V(tr) = \langle x \mapsto \texttt{false}, x \mapsto \texttt{false}, x \mapsto \texttt{true}, x \mapsto \texttt{false}, x \mapsto \texttt{false}, x \mapsto \texttt{true}, \ldots \rangle \ .$$

Note that wrd_V is independent of the border values of the unit intervals and takes the values inside the intervals instead.

Fig. 1. Mapping a time-wise discrete trajectory to an infinite string

wrd_V maps a time-wise discrete trajectory tr to the sequence of values which are assigned to the integer-aligned unit-intervals of time by tr. This is illustrated in Fig. 1. wrd_V is used for assigning to each $\{\int P = k\}$-formula ϕ with $free(\phi) \subseteq V$ a family of languages over α that will turn out to have an effective construction:

Definition A.1. Let ϕ be a formula with $free(\phi) \subseteq V$. For each $l, r \in [0, 1)$,

$$\mathcal{L}_{l,r}(\phi) \stackrel{\text{def}}{=} \left\{ wrd_V(tr)[0, \lfloor e \rfloor] \,\middle|\, \begin{array}{l} tr \in DTraj,\ e \in Time, \\ e \geq l \ \wedge \ e = \lfloor e \rfloor + r, \\ tr, [l, e] \models \phi \end{array} \right\}$$

is the set of digitisations of those time-wise discrete finite observations that satisfy ϕ, start in time instant l and end with offset r to an integer time instant.

Through its definition, emptiness of $\mathcal{L}_{l,r}(\phi)$ is related to satisfiability of ϕ by time-wise discrete observations that obey certain restrictions on their start and end times, namely that they start in l and end with offset r to an integer time instant. The rationale behind this construction is that emptiness of each of the languages $\mathcal{L}_{0,r}(\neg\phi)$ characterises existence of time-wise discrete counterexamples of ϕ:

Lemma A.2. Let ϕ be a $\{\int P = k\}$-formula with $free(\phi) \subseteq V$. ϕ has a time-wise discrete counterexample iff $\bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \neq \emptyset$. More precisely, the images of time-wise discrete counterexamples under wrd_V obey

$$wrd_V(DTraj \setminus \mathcal{M}[\![\phi]\!]) = \bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \cdot \alpha^\omega \ .$$

Proof. First, we will show the inclusion $wrd_V(DTraj \setminus \mathcal{M}[\![\phi]\!]) \subseteq \bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \cdot \alpha^\omega$. Therefore assume that $tr \in DTraj \setminus \mathcal{M}[\![\phi]\!]$. I.e., tr is time-wise discrete and there is $e \in Time$ such that $tr, [0, e] \models \neg\phi$. Then, by Definition A.1, $wrd_V(tr)[0, \lfloor e \rfloor] \in \mathcal{L}_{0,e-\lfloor e \rfloor}(\neg\phi)$ and thus $wrd_V(tr) \in \bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \cdot \alpha^\omega$, which proves the desired first inclusion.

For showing the converse inclusion let $w \in \bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \cdot \alpha^\omega$. Then, $w = u \cdot v$ with $u \in \mathcal{L}_{0,r}(\neg\phi)$ for some $r \in [0, 1)$ and $v \in \alpha^\omega$. By Definition A.1 this implies that there is $tr \in DTraj$ and $e = \lfloor e \rfloor + r$ such that $tr, [0, e] \models \neg\phi$ and $wrd_V(tr)[0, \lfloor e \rfloor] = u$. Now, taking $tr' \in DTraj$ such that

$$tr'(t) \stackrel{\text{def}}{=} \begin{cases} tr(t) & \text{, if } t \leq \lceil e \rceil \ , \\ w(\lfloor t \rfloor), & \text{if } t > \lceil e \rceil \ , \end{cases}$$

we obtain a time-wise discrete trajectory tr' with $wrd_V(tr') = w$ and, as tr' coincides with tr on $[0, e)$, with $tr', [0, e] \models \neg\phi$. I.e., $w \in wrd_V(DTraj \setminus \mathcal{M}[\![\phi]\!])$, which yields the second inclusion. ☐

In the sequel we will reveal a procedure for constructing $\bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \cdot \alpha^\omega$. The revelation proceeds in two steps: We start with a sequence of lemmata that characterise the dependency of $\mathcal{L}_{l,r}(\phi)$ on l, r, and the syntactic structure of ϕ. Then we deduce from that characterisation an explicit representation of $\bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \cdot \alpha^\omega$ by a regular language.

The dependency of $\mathcal{L}_{l,r}(\phi)$ on l, r, and the syntactic structure of ϕ is subject of Lemmata A.3 to A.8. We start by reducing duration equations $\int P = k$ to just two simple forms of duration inequations:[7]

Lemma A.3. Suppose P is a state assertion with $\mathit{free}(P) \subseteq V$ and let $k \in \mathbb{N}_{>0}$. Then

$$\begin{aligned}
\mathcal{L}_{l,r}(\textstyle\int P = 0) &= \mathcal{L}_{l,r}(\neg \textstyle\int P > 0) \ , \\
\mathcal{L}_{l,r}(\textstyle\int P = 1) &= \mathcal{L}_{l,r}((\neg \textstyle\int P < 1) \wedge \neg(\textstyle\int P > 0 \frown \neg \textstyle\int P < 1) \ , \\
\mathcal{L}_{l,r}(\textstyle\int P = k+1) &= \mathcal{L}_{l,r}(\textstyle\int P = k \frown \textstyle\int P = 1) \ .
\end{aligned}$$

Note that this characterisation allows to determine $\mathcal{L}_{l,r}(\int P = k)$ for arbitrary $k \in \mathbb{N}$ from the characterisations of $\mathcal{L}_{l,r}(\int P < 1)$ and $\mathcal{L}_{l,r}(\int P > 0)$ together with chop and Boolean connectives.

Proof. From definition A.1 it is obvious that $\mathcal{L}_{l,r}(\phi) = \mathcal{L}_{l,r}(\psi)$ if ϕ and ψ are logically equivalent. Hence, above equations follow from the logical equivalences of DC. $\square$

Next, we determine $\mathcal{L}_{l,r}(\int P \sim k)$ for those simple inequations.

Lemma A.4. Let P be a state assertion with $\mathit{free}(P) \subseteq V$. Then

$$\mathcal{L}_{l,r}(\textstyle\int P < 1) = \begin{cases} \widetilde{\neg P}^* \cdot \alpha & \text{if } \ l = 0 \ , \\ (\widetilde{\neg P}^* \cdot \alpha) \cup (\alpha \cdot \widetilde{\neg P}^*) & \text{if } \ 0 < l \leq r \ , \\ \alpha \cdot \widetilde{\neg P}^* \cdot \alpha & \text{if } \ l > r \ , \end{cases}$$

where $\widetilde{P} \stackrel{\text{def}}{=} \left\{ \sigma|_V \;\middle|\; \begin{array}{l} \sigma \in V \to \mathbb{B}, \\ [\![P]\!](\sigma) = \mathtt{true} \end{array} \right\}$ is the set of valuations in the alphabet α that satisfy the state assertion P. For convenience, we define $\emptyset^* \stackrel{\text{def}}{=} \{\varepsilon\}$, i.e. $\widetilde{\mathtt{false}}^* = \{\varepsilon\}$.

Proof. Figure 2 illustrates which observations $(tr, [l, e])$ with $l \leq e = \lfloor e \rfloor + r$ satisfy $\int P < 1$ depending on the choice of $l, r \in [0, 1)$. It is straightforward to construct the image of these model classes under the mapping $(tr, [l, e]) \mapsto \mathit{wrd}_V(tr)[0, \lfloor e \rfloor]$. For $l = 0$ this image is $\widetilde{\neg P}^* \cdot \alpha$ such that $\mathcal{L}_{0,r}(\int P < 1) = \widetilde{\neg P}^* \cdot \alpha$. Similarly, for $0 < l \leq r$ it is $(\widetilde{\neg P}^* \cdot \alpha) \cup (\alpha \cdot \widetilde{\neg P}^*)$, yielding the proposition for this case. Finally, for $l > r$ the side condition $l \leq e = \lfloor e \rfloor + r$ implies $e \geq 1$ such that the image of $(tr, [l, e]) \mapsto \mathit{wrd}_V(tr)[0, \lfloor e \rfloor]$ contains only words of length greater or equal to 2 in that case. Together with the structure of the models explained in Figure 2 this shows that $\mathcal{L}_{l,r}(\int P < 1) = \alpha \cdot \widetilde{\neg P}^* \cdot \alpha$ whenever $l > r$. $\square$

Lemma A.5. Let P be a state assertion with $\mathit{free}(P) \subseteq V$. Then

$$\mathcal{L}_{l,r}(\textstyle\int P > 0) = \begin{cases} \alpha^* \cdot \widetilde{P} \cdot \alpha^+ & \text{if } \ r = 0 \ , \\ \alpha^* \cdot \widetilde{P} \cdot \alpha^* & \text{if } \ r > l \ , \\ \alpha^{\geq 2} \cap \left(\alpha^* \cdot \widetilde{P} \cdot \alpha^*\right) & \text{if } \ 0 < r \leq l \ , \end{cases}$$

where $\alpha^{\geq 2} \stackrel{\text{def}}{=} \{w \in \alpha^* \mid \mathrm{len}(w) \geq 2\}$.

Proof. By a similar inspection of the model class as in the proof of Lemma A.4. $\square$

Lemma A.6. Suppose ϕ is a formula with $\mathit{free}(\phi) \subseteq V$. Then

$$\mathcal{L}_{l,r}(\neg\phi) = \begin{cases} \alpha^+ \setminus \mathcal{L}_{l,r}(\phi) & \text{if } \ l \leq r, \\ \alpha^{\geq 2} \setminus \mathcal{L}_{l,r}(\phi) & \text{if } \ l > r. \end{cases}$$

Proof. For the case $l \leq r$ we show $\mathcal{L}_{l,r}(\neg\phi) = \alpha^+ \setminus \mathcal{L}_{l,r}(\phi)$ by proving that

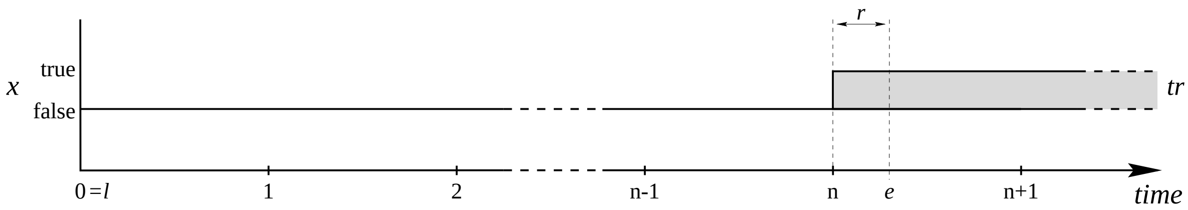
1. $\mathcal{L}_{l,r}(\phi) \cup \mathcal{L}_{l,r}(\neg\phi) = \alpha^+$ and

---

[7] According to the syntax of Sect. 2, duration inequations are not genuine formulae of DC, but have too be understood as abbreviations for formulae built from duration equations. E.g., $\int P > 0$ denotes $\neg \int P = 0$ and $\int P < 1$ denotes $\neg(\int P = 1 \frown \mathtt{true})$. However, as the decision procedures for inequalities are considerably simpler, we prefer to use the converse encoding here. Note that there is no circularity involved.

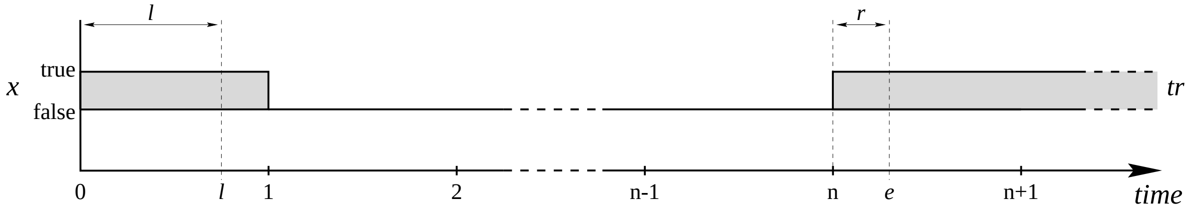
Time-wise discrete observations $(tr, [l, e])$ that satisfy $\int x < 1$, start in $l \in [0, 1)$, and end in e with offset $r \in [0, 1)$ to an integer time instant can have one of the following shapes, depending on l and r. Shaded parts indicate that both a trajectory assigning `true` to x in that phase and a trajectory assigning `false` to x in that phase satisfy $\int x < 1$.

Case 1. $0 = l \leq r < 1$.



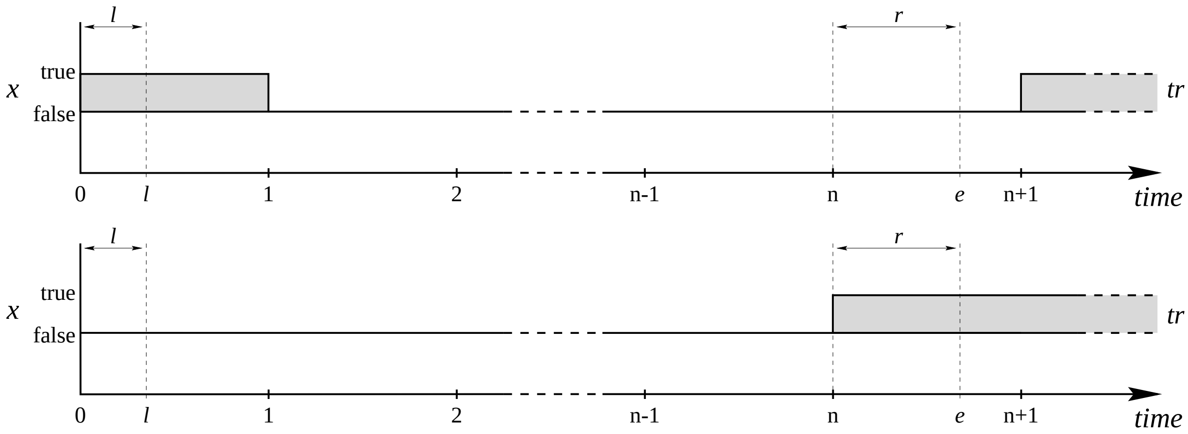
If $l = 0$ then all non-empty intersections of integer-aligned unit intervals with $[l, e]$ except the rightmost one have length 1. Hence, in order to satisfy $\int x < 1$, x has to be interpreted by `false` throughout all integer-aligned unit intervals except perhaps the rightmost, i.e. incomplete, one.

Case 2. $0 \leq r < l < 1$.



If $l > 0$ then all non-empty intersections of integer-aligned unit intervals with $[l, e]$ except the leftmost and the rightmost one have length 1. Hence, in order to satisfy $\int x < 1$, x has to be interpreted by `false` throughout all integer-aligned unit intervals except perhaps the leftmost or the rightmost one. As $r < l$, $\int x < 1$ is even satisfied if x is interpreted by `true` in both the leftmost and the rightmost integer-aligned unit interval, as then $[\![\int x]\!](tr, [l, \mathrm{n} + r]) = 1 - l + r < 1$.

Case 3. $0 < l \leq r < 1$.



As in case 2, all integer-aligned unit intervals intersecting with $[l, e]$ except the leftmost and the rightmost one have length 1. Hence, x has to be interpreted by `false` throughout all integer-aligned unit intervals except perhaps the leftmost or the rightmost one. However, as $l \leq r$, $\int x < 1$ would be violated if x were interpreted by `true` in both the leftmost and the rightmost integer-aligned unit interval, as then $[\![\int x]\!](tr, [l, \mathrm{n} + r]) = 1 - l + r \geq 1$. Therefore, x may only evaluate to `true` in *either* the leftmost *or* the rightmost aligned unit interval intersecting with $[l, e]$, which gives rise to the two possible shapes of *tr* that are depicted above.

Fig. 2. Time-wise discrete observations satisfying $\int x < 1$

2. $\mathcal{L}_{l,r}(\phi) \cap \mathcal{L}_{l,r}(\neg\phi) = \emptyset$.

We start by showing property (1). Therefore let $k \in \mathbb{N}_{\geq 1}$ and $w \in \alpha^k$. Furthermore let tr be a time-wise discrete trajectory such that

$$\forall\, k \in \mathbb{N}_{<k}\,.\, \forall\, \delta \in (0,1)\,.\, tr(i+\delta)(v) = w(i)(v)$$

holds for each $v \in V$. Such a $tr \in DTraj$ exists obviously and it satisfies $w = wrd_V(tr)[0, \lfloor k-1+r \rfloor]$. Now either $tr, [l, k-1+r] \models \phi$ or $tr, [l, k-1+r] \models \neg\phi$ holds due to the semantics of DC. Hence, by definition of $\mathcal{L}_{l,r}$, $w \in \mathcal{L}_{l,r}(\phi)$ or $w \in \mathcal{L}_{l,r}(\neg\phi)$. As $k \in \mathbb{N}_{\geq 1}$ and $w \in \alpha^k$ were arbitrarily chosen, this shows $\mathcal{L}_{l,r}(\phi) \cup \mathcal{L}_{l,r}(\neg\phi) \supseteq \alpha^+$.

On the other hand, $0 \leq \lfloor e \rfloor$ for any $e \in Time$ such that $wrd_V(tr)[0, \lfloor e \rfloor] \neq \varepsilon$ for each $tr \in DTraj$ and $e \in Time$. Hence, $\varepsilon \notin \mathcal{L}_{l,r}(\phi)$ and $\varepsilon \notin \mathcal{L}_{l,r}(\neg\phi)$ by definition of $\mathcal{L}_{l,r}$. This implies that $\mathcal{L}_{l,r}(\phi) \cup \mathcal{L}_{l,r}(\neg\phi) \subseteq \alpha^+$. Hence, $\mathcal{L}_{l,r}(\phi) \cup \mathcal{L}_{l,r}(\neg\phi) = \alpha^+$.

We show property (2) by deriving a contradiction from its negation. Therefore assume that $\mathcal{L}_{l,r}(\phi) \cap \mathcal{L}_{l,r}(\neg\phi) \neq \emptyset$, i.e. $w \in \mathcal{L}_{l,r}(\phi) \cap \mathcal{L}_{l,r}(\neg\phi)$ exists. By definition of $\mathcal{L}_{l,r}$ this implies that $tr_\phi, tr_{\neg\phi} \in DTraj$ and $k_\phi, k_{\neg\phi} \in \mathbb{N}$ exist with

a) $tr_\phi, [l, k_\phi + r] \models \phi$,
b) $tr_{\neg\phi}, [l, k_{\neg\phi} + r] \models \neg\phi$, and
c) $wrd_V(tr_\phi)[0, \lfloor k_\phi + r \rfloor] = w = wrd_V(tr_{\neg\phi})[0, \lfloor k_{\neg\phi} + r \rfloor]$.

Property (c) obviously implies $k_\phi = k_{\neg\phi}$ as $k_\phi + 1 = \mathrm{len}(wrd_V(tr_\phi)[0, \lfloor k_\phi + r \rfloor]) = \mathrm{len}(w)$ and similarly $k_{\neg\phi} + 1 = \mathrm{len}(w)$. Furthermore, property (c) implies by definition of wrd_V and time-wise discreteness of tr_ϕ and $tr_{\neg\phi}$ that $\{t \in [0, k_\phi + 1] \mid tr_\phi(t) \neq tr_{\neg\phi}(t)\}$ is a set of measure 0 s.t. no formula of DC can distinguish between observation $(tr_\phi, [0, k_\phi + 1])$ and $(tr_{\neg\phi}, [0, k_\phi + 1])$. But then

$$\begin{aligned} & tr_\phi, [l, k_\phi + r] \models \phi \\ \iff\;& tr_{\neg\phi}, [l, k_{\neg\phi} + r] \models \phi && [\text{DC cannot distinguish}] \\ \iff\;& tr_{\neg\phi}, [l, k_{\neg\phi} + r] \not\models \neg\phi && [\text{Semantics of DC}] \end{aligned}$$

which is a contradiction to properties (a) and (b). As these inconsistent properties were deduced from the single assumption $\mathcal{L}_{l,r}(\phi) \cap \mathcal{L}_{l,r}(\neg\phi) \neq \emptyset$, this assumption cannot hold, which yields property (2). Finally, properties (1) and (2) together give $\mathcal{L}_{l,r}(\neg\phi) = \alpha^+ \setminus \mathcal{L}_{l,r}(\phi)$. This completes the case $l \leq r$.

The proof for the case $l > r$ is similar, the only major difference being that $e = \lfloor e \rfloor + r \wedge e \geq l > r$ implies $e \geq 1$ such that then $\mathrm{len}(wrd_V(tr)[0, \lfloor e \rfloor]) \geq 2$ for any $tr \in DTraj$. Hence, $\mathrm{len}(w) < 2$ implies $w \notin \mathcal{L}_{l,r}(\phi) \cup \mathcal{L}_{l,r}(\neg\phi)$, i.e. $\mathcal{L}_{l,r}(\phi) \cup \mathcal{L}_{l,r}(\neg\phi) \subseteq \alpha^{\geq 2}$. A similar argument as in the case $l \leq r$ shows that $\mathcal{L}_{l,r}(\phi) \cup \mathcal{L}_{l,r}(\neg\phi) = \{w \in \alpha^* \mid \mathrm{len}(w) \geq 2\} = \alpha^{\geq 2}$. Furthermore, $\mathcal{L}_{l,r}(\phi) \cap \mathcal{L}_{l,r}(\neg\phi) = \emptyset$ can be shown exactly as in the case $l \leq r$. Hence, $\mathcal{L}_{l,r}(\neg\phi) = \alpha^{\geq 2} \setminus \mathcal{L}_{l,r}(\phi)$ if $l > r$. $\square$

Lemma A.7. Let ϕ_1 and ϕ_2 be formulae with $\mathit{free}(\phi_i) \subseteq V$. Then

$$\mathcal{L}_{l,r}(\phi_1 \vee \phi_2) = \mathcal{L}_{l,r}(\phi_1) \cup \mathcal{L}_{l,r}(\phi_2)\ .$$

Proof. Straightforward from the semantics of DC. $\square$

Lemma A.8. Let ϕ_1 and ϕ_2 be formulae with $\mathit{free}(\phi_i) \subseteq V$. Then

$$\mathcal{L}_{l,r}(\phi_1 \frown \phi_2) = \bigcup_{x \in [0,1)} \mathcal{L}_{l,x}(\phi_1) \hat{}\, \mathcal{L}_{x,r}(\phi_2)\ ,$$

where $X \hat{}\, Y \stackrel{\text{def}}{=} \{u \hat{}\, v \mid u \in X, v \in Y, (u,v) \in \mathrm{dom}\,(\hat{}\,)\}$ is the lifted version of concatenation with one letter overlap $\hat{}\, : \alpha^* \times \alpha^* \xrightarrow{\text{partial}} \alpha^*$ which is defined by

$$u \hat{}\, v \stackrel{\text{def}}{=} \begin{cases} u \cdot (v[1, \mathrm{len}(v) - 1]) & \text{iff } u(\mathrm{len}(u) - 1) = v(0), \\ \text{undefined} & \text{iff } u(\mathrm{len}(u) - 1) \neq v(0). \end{cases}$$

Note that this lemma characterises $\mathcal{L}_{l,r}(\phi_1 \frown \phi_2)$ through an uncountably infinite union of languages.

Proof. First, we observe that $v[0,n] = v[0,m]\hat{}v[m,n]$ for each $m \le n \le \mathrm{len}(v) - 1$. Hence,

$$wrd_V(tr)[0, \lfloor e \rfloor] = wrd_V(tr)[0, \lfloor m \rfloor]\hat{}\, wrd_V(tr)[\lfloor m \rfloor, \lfloor e \rfloor] \tag{1}$$

for each $tr \in DTraj$ and $m \le e \in Time$. Furthermore, for each $tr \in DTraj$ and $k \in \mathbb{N}$, $t \mapsto tr(t + k)$ is a time-wise discrete trajectory that satisfies

$$wrd_V(tr)[k, \lfloor e \rfloor] = wrd_V(t \mapsto wrd_V(t + k))[0, \lfloor e - k \rfloor] \ . \tag{2}$$

In order to show that $\mathcal{L}_{l,r}(\phi_1 \frown \phi_2) \subseteq \bigcup_{x \in [0,1)} \mathcal{L}_{l,x}(\phi_1)\hat{}\mathcal{L}_{x,r}(\phi_2)$ let $w \in \mathcal{L}_{l,r}(\phi_1 \frown \phi_2)$. By definition of $\mathcal{L}_{l,r}$ this implies that there is $tr \in DTraj$ and $e \in Time$ with $e = \lfloor e \rfloor + r$ such that $w = wrd_V(tr)[0, \lfloor e \rfloor]$ and $tr, [l, e] \models \phi_1 \frown \phi_2$. According to the semantics of chop this implies existence of $m \in Time$ with $l \le m \le e$ such that $tr, [l, m] \models \phi_1$ and $tr, [m, e] \models \phi_2$ hold.

Let $x \stackrel{\mathrm{def}}{=} m - \lfloor m \rfloor$ and $k \stackrel{\mathrm{def}}{=} \lfloor m \rfloor$. Due to the semantics of DC, observations can be freely shifted over time without changing satisfaction of formulae, such that $(t \mapsto tr(t + k)), [x, e - k] \models \phi_2$ follows from $tr, [m, e] \models \phi_2$. As k is integer, $e = \lfloor e \rfloor + r$ implies $e - k = \lfloor e - k \rfloor + r$, such that $(t \mapsto tr(t+k)), [x, e-k] \models \phi_2$ yields $wrd_V(t \mapsto tr(t + k))[0, \lfloor e - k \rfloor] \in \mathcal{L}_{x,r}(\phi_2)$ according to Definition A.1. By equation (2), $wrd_V(t \mapsto tr(t + k))[0, \lfloor e - k \rfloor] = wrd_V(tr)[k, \lfloor e \rfloor]$ such that

$$wrd_V(tr)[k, \lfloor e \rfloor] \in \mathcal{L}_{x,r}(\phi_2) \ . \tag{3}$$

Furthermore, $m = \lfloor m \rfloor + x$ and $tr, [l, m] \models \phi_1$ yield

$$wrd_V(tr)[0, \lfloor m \rfloor] \in \mathcal{L}_{l,x}(\phi_1) \tag{4}$$

according to Definition A.1. It follows that

$$\begin{array}{ll} & w \\ = & wrd_V(tr)[0, \lfloor e \rfloor] \qquad [\text{Choice of } wrd_V \text{ and } e] \\ = & wrd_V(tr)[0, \lfloor m \rfloor]\hat{}\, wrd_V(tr)[\lfloor m \rfloor, \lfloor e \rfloor] \qquad [\text{Equation (1)}] \\ = & wrd_V(tr)[0, k]\hat{}\, wrd_V(tr)[k, \lfloor e \rfloor] \qquad [k = \lfloor m \rfloor] \\ \in & \mathcal{L}_{l,x}(\phi_1)\hat{}\mathcal{L}_{x,r}(\phi_2) \qquad [\text{(3) and (4)}] \\ \subseteq & \displaystyle\bigcup_{y \in [0,1)} \mathcal{L}_{l,y}(\phi_1)\hat{}\mathcal{L}_{y,r}(\phi_2) \ . \qquad [x = m - \lfloor m \rfloor \in [0, 1)] \end{array}$$

This proves the inclusion $\mathcal{L}_{l,r}(\phi_1 \frown \phi_2) \subseteq \bigcup_{x \in [0,1)} \mathcal{L}_{l,x}(\phi_1)\hat{}\mathcal{L}_{x,r}(\phi_2)$ as $w \in \mathcal{L}_{l,r}(\phi_1 \frown \phi_2)$ was arbitrarily chosen. The converse inclusion follows analogously. $\square$

Lemmata A.3 to A.8 provide a recursive — except for Lemma A.3 even structurally inductive — characterisation of $\mathcal{L}_{l,r}(\phi)$ for arbitrary $l, r \in [0, 1)$ and $\{\int P = k\}$-formulae ϕ with $free(\phi) \subseteq V$. For the base cases, Lemmata A.3 to A.5 even provide regular expressions such that $\mathcal{L}_{l,r}(\phi)$ can be seen to be effectively constructible in the base cases. Negation and disjunction preserve effectiveness, as they map to language difference and language union (Lemmata A.6 and A.7), which are both effective in the realm of regular languages. Unfortunately, Lemma A.8, which deals with chop, involves an uncountably infinite union of languages such that it is not immediately clear whether chop preserves regularity.

Fortunately, this particular infinite union of languages consists of only finitely many different constituents according to the following lemma, such that regularity of the language $\mathcal{L}_{l,r}(\phi)$ is also preserved under chop.

Lemma A.9 (Finiteness of the family of languages $(\mathcal{L}_{l,r}(\phi))_{l,r \in [0,1)}$).
Let $l_1, l_2, r_1, r_2 \in [0, 1)$ and let ϕ be a $\{\int P\}$-formula such that $free(\phi) \subseteq V$. If

(i)	$0 = l_1 = r_1$ and $0 = l_2 = r_2$	or	(ii)	$0 = l_1 < r_1$ and $0 = l_2 < r_2$	or
(iii)	$0 < l_1 < r_1$ and $0 < l_2 < r_2$	or	(iv)	$0 < l_1 = r_1$ and $0 < l_2 = r_2$	or
(v)	$0 = r_1 < l_1$ and $0 = r_2 < l_2$	or	(vi)	$0 < r_1 < l_1$ and $0 < r_2 < l_2$	

then $\mathcal{L}_{l_1,r_1}(\phi) = \mathcal{L}_{l_2,r_2}(\phi)$ holds. I.e. $\mathcal{L}_{l,r}(\phi)$ depends only on sgn(l), sgn(r), and sgn$(l-r)$ rather than on the exact values of l and r. Therefore, the family of languages $(\mathcal{L}_{l,r}(\phi))_{l,r\in[0,1)}$ contains at most six different languages for each ϕ.

Proof. This property follows through a straightforward induction from the characterisations of $\mathcal{L}_{l,r}(\phi)$ provided by Lemmata A.3 to A.8. Note that the case distinction (i) to (vi) is indeed taken from the intersection of the case distinctions made in those lemmata such that the proposition can immediately be seen to hold for atomic formulae and to be preserved by the connectives $\neg$, $\vee$, and $\frown$. Hence, structural induction reveals that the proposition is true for all formulae built from atomic formulae using the Boolean connectives and chop. $\square$

As this lemma does not only show that the apparently uncountable union of languages in Lemma A.8 consists of only finitely many different constituents, but furthermore provides an effective reduction to such a finite union, it shows that beyond regularity of the language even effective constructibility of the corresponding regular expression is preserved under chop. Thus, all operators of DC preserve effectiveness of the construction:

Corollary A.10. $\mathcal{L}_{l,r}(\phi)$ is regular and can be effectively constructed for each $l,r \in [0,1)$ and each $\{\int P = k\}$-formula ϕ with $\mathit{free}(\phi) \subseteq V$. $\square$

Furthermore, Lemma A.9 shows that $\bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\phi)$ is a union of at most two different languages, as $\mathcal{L}_{0,r_1}(\phi) = \mathcal{L}_{0,r_2}(\phi)$ for any $r_1, r_2 \in (0,1)$. As these two languages are effectively constructible regular languages due to Corollary A.10, this implies

Corollary A.11. $\bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\phi)$ is an effectively constructible regular set for each $\{\int P = k\}$-formula ϕ with $\mathit{free}(\phi) \subseteq V$.

Proof. An appropriate regular expression can be built from the characterisations provided by Lemmata A.3 to A.9. $\square$

As the images under wrd_V of time-wise discrete counterexamples to ϕ are, according to Lemma A.2, of the form $\bigcup_{r\in[0,1)} \mathcal{L}_{0,r}(\neg\phi) \cdot \alpha^\omega$, this implies

Corollary A.12. Let ϕ be a formula from the $\{\int P = k\}$-fragment of DC with $\mathit{free}(\phi) \subseteq V$. Then the image of the time-wise discrete counterexamples to ϕ under wrd_V, i.e. $\mathit{wrd}_V(\mathit{DTraj} \setminus \mathcal{M}[\![\phi]\!])$, is an ω-regular language, and there is an effective construction for an ω-automaton recognising it. $\square$

As the set V was arbitrarily chosen, this shows

Theorem A.13 (Decidability of $\{\int P = k\}$-formulae over time-wise discrete models). Let ϕ be an arbitrary $\{\int P = k\}$-formula. Then

1. $\mathit{DTraj} \setminus \mathcal{M}[\![\phi]\!]$ is an unrolling of an ω-regular language to real-time based on the convention that one letter per time unit is traversed. An ω-automaton Aut_ϕ recognising that ω-regular language can be effectively constructed, and
2. it is decidable whether ϕ has a time-wise discrete counterexample.

Proof. Given an arbitrary formula ϕ, the construction of Corollary A.12 can be performed with $V = \mathit{free}(\phi)$. This yields an ω-automaton Aut_ϕ with $\mathit{DTraj} \setminus \mathcal{M}[\![\phi]\!] = \mathit{wrd}^{-1}_{\mathit{free}(\phi)}(\mathcal{L}_{\mathit{Aut}_\phi})$, which shows the first part of the theorem. As emptiness of $\mathcal{L}_{\mathit{Aut}_\phi}$ is decidable, the second part follows from totality and surjectivity of $\mathit{wrd}_{\mathit{free}(\phi)}$. $\square$

An analysis of the extended regular expressions that the construction of Lemmata A.3 to A.9 assigns to formulae reveals that this particular decision procedure for existence of time-wise discrete counterexamples — and hence also any model-checking procedure built on it — inherits non-elementary complexity from the language emptiness problem for extended regular expressions [MS73]. Furthermore, it is easy to see that in general one cannot do better:

Theorem A.14 (Non-elementariness of propositional DC). The three problems of

1. deciding existence of a finitely variable counterexample to a $\{\lceil P \rceil\}$-formula,

2. deciding existence of an n-bounded counterexample to a $\{\lceil\!\lceil P \rceil\!\rceil\}$-formula, and
3. deciding existence of a time-wise discrete counterexample to a $\{\lceil\!\lceil P \rceil\!\rceil\}$-formula

are all non-elementary.

Note that thus the same applies for any decision procedure dealing with the richer subsets $\{\lceil\!\lceil P \rceil\!\rceil, l = k\}$ and $\{\int P = k\}$.

Proof. The emptiness problem of extended regular expressions, which is non-elementary according to [MS73], has a linear encoding in DC: let re be an extended regular expression over an alphabet of the form $\alpha = V \to \mathbb{B}$, with $V \subset \mathit{Varname}$ being finite. For any $a \in \alpha$ let P_a be the state assertion

$$P_a \stackrel{\text{def}}{=} \bigwedge_{\substack{u \in V,\\ a(u)=\mathtt{true}}} u \quad \wedge \quad \bigwedge_{\substack{v \in V,\\ a(v)=\mathtt{false}}} \neg v \quad .$$

We will encode words $w = \langle w_1, \ldots, w_n \rangle \in \alpha^*$ by observations consisting of time intervals first satisfying P_{w_1}, then P_{w_2}, and so on, with the individual characters being delimited by phases where a further variable $\mathit{mark} \in \mathit{Varname} \setminus V$ holds true. I.e., w is encoded by those observations that satisfy

$$\mathit{enc}_w \stackrel{\text{def}}{=} \lceil\!\lceil P_{w_1} \wedge \neg \mathit{mark} \rceil\!\rceil \frown \lceil\!\lceil \mathit{mark} \rceil\!\rceil \frown \lceil\!\lceil P_{w_2} \wedge \neg \mathit{mark} \rceil\!\rceil \frown \lceil\!\lceil \mathit{mark} \rceil\!\rceil \frown \ldots \frown \lceil\!\lceil P_{w_n} \wedge \neg \mathit{mark} \rceil\!\rceil \frown \lceil\!\lceil \mathit{mark} \rceil\!\rceil \quad .$$

A common property of all observations satisfying enc_w for some $w \in \alpha^*$ is (a) that they end in a phase where mark holds continously true and (b) that within any time interval where mark is continuously false the encoded character remains constant, i.e. some of the predicates P_a for $a \in \alpha$ holds throughout that phase. (a) and (b) can be formalised by the $\{\lceil\!\lceil P \rceil\!\rceil\}$-formula

$$\mathit{well\_encoded} \stackrel{\text{def}}{=} \begin{pmatrix} (\mathtt{true} \frown \lceil\!\lceil \mathit{mark} \rceil\!\rceil) & \wedge \\ \neg(\mathtt{true} \frown (\lceil\!\lceil \neg \mathit{mark} \rceil\!\rceil \wedge \neg(\bigvee_{a \in \alpha} \lceil\!\lceil P_a \rceil\!\rceil)) \frown \mathtt{true}) & \end{pmatrix} \quad .$$

Now, it is easy to show by structural induction that for any extended regular expression re the $\{\lceil\!\lceil P \rceil\!\rceil\}$-formula $F(re)$ defined by

$$\begin{aligned} F(a) &\stackrel{\text{def}}{=} \lceil\!\lceil P_a \wedge \neg \mathit{mark} \rceil\!\rceil \frown \lceil\!\lceil \mathit{mark} \rceil\!\rceil \quad , \\ F(re_1 \vee re_2) &\stackrel{\text{def}}{=} F(re_1) \vee F(re_2) \quad , \\ F(re_1 \cdot re_2) &\stackrel{\text{def}}{=} F(re_1) \frown F(re_2) \quad , \\ F(\neg re_1) &\stackrel{\text{def}}{=} \mathit{well\_encoded} \wedge \neg F(re_1) \end{aligned}$$

is satisfied by exactly those observations which satisfy enc_w for some $w \in \mathcal{L}_{re}$. I.e.,

$$\left\{ tr, [0,e] \,\middle|\, \begin{array}{l} tr \in \mathit{Traj}, e \in \mathit{Time}, \\ tr, [0,e] \models F(re) \end{array} \right\} = \bigcup_{w \in \mathcal{L}_{re}} \left\{ tr, [0,e] \,\middle|\, \begin{array}{l} tr \in \mathit{Traj}, e \in \mathit{Time}, \\ tr, [0,e] \models \mathit{enc}_w \end{array} \right\} \quad .$$

As $\{tr, [0,e] \mid tr \in \mathit{Traj}, e \in \mathit{Time}, tr, [0,e] \models \mathit{enc}_w\} \neq \emptyset$ for each $w \in \alpha^*$, this implies that $F(\neg re)$ has a finitely variable counterexample iff $\mathcal{L}_{re} \neq \emptyset$. Thus F provides a linear encoding of the emptiness problem of extended regular expressions into the $\{\lceil\!\lceil P \rceil\!\rceil\}$-fragment of DC, which proves the first part of the theorem due to the non-elementary complexity of language emptiness for extended regular expressions [MS73].

The other two conjectures follow similarly, as $\{tr, [0,e] \mid tr \in \mathit{Traj}_n, e \in \mathit{Time}, tr, [0,e] \models \mathit{enc}_w\} \neq \emptyset$ for each $w \in \alpha^*$ and each $n \in \mathbb{N}_{\geq 1}$, and as $\{tr, [0,e] \mid tr \in \mathit{DTraj}, e \in \mathit{Time}, tr, [0,e] \models \mathit{enc}_w\} \neq \emptyset$ for each $w \in \alpha^*$. $\square$