Sādhanā, Vol. 21, Part 2, April 1996, pp. 147–184. © Printed in India.

A real-time interval logic and its decision procedure

Y S RAMAKRISHNA[1+], L K DILLON[2], L E MOSER[1],
P M MELLIAR-SMITH[1] and G KUTTY[1]

[1]Department of Electrical and Computer Engineering, and
[2]Department of Computer Science, University of California, Santa Barbara, CA 93106, USA
[+]Present Address: Computer Science Department, SUNY, Stony Brook, NY 11794-4400, USA

Abstract. Real-Time Future Interval Logic is a temporal logic in which formulæ have a natural graphical representation, resembling timing diagrams. It is a dense real-time logic that is based on two simple temporal primitives: *interval modalities* for the purely qualitative part and *duration predicates* for the quantitative part. This paper describes the logic and gives a decision procedure for satisfiability by reduction to the emptiness problem for Timed Büchi Automata. This decision procedure forms the core of an automated proof-checker for the logic. The logic does not admit instantaneous states, and is invariant under real-time stuttering, properties that facilitate proof methods based on abstraction and refinement. The logic appears to be as strong as one can hope for without sacrificing elementary decidability. Two natural extensions of the logic, along lines suggested in the literature, lead to either non-elementariness or undecidability.

Keywords. Interval logics; concurrent systems; real-time temporal logics; hierarchical refinement.

1. Introduction

Specification and verification of concurrent systems is difficult in part because the many possible alternative interleavings of activities generate a large number of cases that must be considered. The presence of real-time constraints, and their interaction with constraints on the interleavings, makes the problem even more difficult. Propositional temporal logic (PTL) and the propositional μ-calculus are too low-level to capture abstract system requirements easily without including extraneous details that can bias subsequent implementations. Interval logics aid the specification of concurrent systems by providing temporal

A preliminary version of this paper appears in the Proceedings of the 13th FST&TCS, LNCS 761, December 1993, pp 201–220.

modalities designed explicitly to ease the definition of temporal contexts and of properties required to hold in such contexts.

Interval logics also permit natural graphical representations, which are usually more intuitive and easier to understand than their textual counterparts. When expressed graphically, interval logic formulæ resemble the "back-of-the-envelope" timing diagrams that designers typically draw to document and reason about temporal properties of their designs. Interval logics, in their graphical representation, could serve to extend existing design and documentation environments to the more challenging task of verification of concurrent systems.

However, most known interval logics are either non-elementary or even undecidable. In particular, the Interval Temporal Logic of Moszkowski (Halpern *et al* 1983) is provably non-elementary and the Modal Logic of Time-Intervals of Halpern & Shoham (1991) is undecidable. In Ramakrishna *et al* (1992) we presented an interval logic, called Future Interval Logic (FIL), and a decision procedure for it. As far as we are aware, this is the first and indeed the only interval logic known today, with an elementary decision procedure. Examples illustrating the use of FIL in specification and verification appear in Dillon *et al* (1992) and Kutty *et al* (1993). However, FIL is a "timeless" logic, with no quantitative notion of time.

There are numerous applications, however, where a purely qualitative notion of time is insufficient, because correctness depends crucially on real-time constraints between events in a system. This has led to real-time extensions of temporal logics (Jahanian & Mok 1986; Narayana & Aaby 1988; Alur & Henzinger 1989; Emerson *et al* 1990; Lewis 1990). The theory of timed-automata of Alur & Dill (1990) and Alur & Henzinger (1992) has helped clarify fundamental issues regarding the decidability of real-time temporal logics. These results have not, however, been applied to real-time extensions of interval logics (Melliar–Smith 1987; Narayana & Aaby 1988; Razouk & Gorlick 1989) to establish their decidability or to obtain "efficient" decision procedures.

In this paper we extend FIL to real-time. We associate the domain of non-negative reals with a computation, and extend the language of FIL to allow statements about the durations of intervals. This gives a relatively clean extension of FIL. Firstly, the extension is conservative. All tautologies of FIL are tautologies of this logic. Moreover, the tautologies of RTFIL, restricted to the language of FIL, are precisely the tautologies of FIL. Secondly, the extension does not sacrifice decidability. RTFIL is decidable by reduction to the emptiness problem for Timed Büchi Automata; this constitutes the main result of this paper. Finally, the extension is adequate. RTFIL has the expressiveness needed for real-time reasoning. We give an example of its use in Ramakrishna *et al* (1993), where a proof-checker based on the decision procedure presented here is used to verify a simple real-time system.

Our work, like Barringer *et al* (1986) and Alur *et al* (1991) but unlike Narayana & Aaby (1988), Emerson *et al* (1990), Jahanian & Mok (1986) and Razouk & Gorlick (1989), uses a dense model of time. A dense time domain is preferable and to a discrete time domain for specifying concurrent systems because independent events in asynchronous components may occur arbitrarily close in time. It is not possible, therefore, to bound *à priori* the granularity of the underlying time domain, as required for a discrete model. A dense time domain is also preferable for carrying out hierarchical verification, since proofs remain valid under refinement or abstraction. A dense time domain facilitates compositional specification and

verification of real-time systems, where a component's (timing) semantics must be independent of (the timing granularity of) the environment in which it may operate. Dense time is also required when a component interacts with the continuous world; hybrid systems are a good example (Maler *et al* 1991). Numerous real world and process control applications, thus, require a dense model of time.

Unlike most real-time temporal logics, RTFIL is insensitive to instantaneous states.[1] This semantics agrees with our intuition that a property of a system can be "observed" only if it persists for some measurable amount of time. It may be counterproductive, when specifying systems, to impose instantaneous requirements on behaviours. Specifications whose only satisfying models contain instantaneous states obstruct the use of hierarchical refinement in much the same way that the next operator obstructs hierarchical refinement (Barringer *et al* 1986; Lamport 1991) in non-real-time temporal logics. In the case of RTFIL, the absence of instantaneous states, in concert with its restricted syntax, results in the property that, for any model whose valuation function is right-continuous, the valuation function extended to an arbitrary RTFIL formula is also right-continuous. This property of "temporal interpolation" is expected to facilitate proofs based on successive refinement or abstraction, where the refinement mapping defining a predicate at one level may involve an arbitrary RTFIL formula on predicates from an adjacent level.

This paper is organized as follows. Section 2 introduces Real-Time Future Interval Logic (RTFIL) by means of a simple graphical formula. It then defines a textual syntax, intended models, and semantics of RTFIL. Section 3 contains some preliminary definitions and notation. The decision procedure is described in § 4 and its correctness is proved in § 5. We present complexity results in § 6. In § 7 we discuss related work and conclude in § 8 with some open problems and on-going work.

2. The logic

We first provide a very informal introduction to RTFIL and illustrate the graphical representation of formulæ. RTFIL is a linear-time temporal logic. Thus, a formula is interpreted on a linear trace of states, representing a possible execution of a transition system (or a fragment of such an execution). Every trace has an initial state. Traces may, however, be unbounded and may thus represent nonterminating behaviours. We assume that the states of the transition system are continuously observed at all $t \in R$ (the set of non-negative reals); thus, every trace of the system is a dense real-time trace.

The key constructs of RTFIL are the *interval modality* and the *duration constraint*. Syntactically, an interval modality is constructed by means of searches and other (simpler) RTFIL formulæ. Semantically, an interval modality extracts a convex subset from a given dense trace. This convex subset specifies the interval over which a property designated by a nested formula holds. The duration constraint is expressed using the special predicate len, and specifies rational lower and upper bounds on the length of an interval.

An interval is constructed using a pair of *search patterns*; searches are shown dashed with arrowheads, and target formulæ are left-justified below the arrowheads. The semantics

[1] The decidability results presented here do not require this semantics.

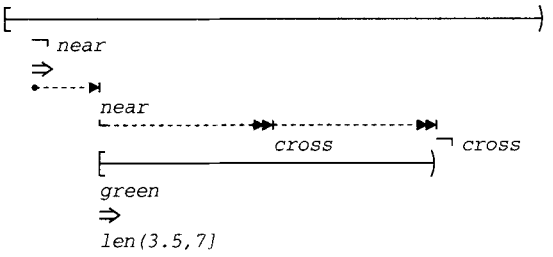


Figure 1. An example specification in Graphical Interval Logic.

of a search that starts at a point in the trace is that the search locates the earliest point in the reflexive future where the target formula holds. When such searches are composed sequentially into a search pattern, every subsequent search begins at the state where the previous search ended. In case the target of a search is not satisfied at any point in the future of the current point within the previous outer interval, the formula is assumed to be true by default if the search is "weak" (shown by a single arrowhead) and false if the search is "strong" (shown by a double arrowhead). Intervals are shown solid with square brackets on the left and parentheses on the right. A formula drawn left-justified below the start of an interval must hold at the first state of that interval, while a formula indented below an interval must hold throughout the interval. In the graphical representation of an RTFIL formula, the horizontal dimension shows progression through the trace (time progresses from left to right) and the vertical dimension describes the composition of formulæ from subformulæ.

The example in figure 1 is a fragment of a road intersection specification. The state predicates *near*, *cross* and *green* are true, respectively, when a car is near an intersection, when it is crossing the intersection and when the signal along that direction is green. It states that, if the signal is *green* whenever a car first approaches the intersection, it takes more than 3.5 seconds but at most 7 seconds to complete the crossing.

Although graphical formulæ such as that above are easier to read and understand than their textual counterparts, the rest of the paper will use a textual syntax for convenience of exposition.

2.1 *Syntax*

The sets of well-formed formulæ (wffs), well-formed search patterns (wfsp), and well-formed interval modalities (wfim) of RTFIL are defined relative to a finite set $\mathcal{P}$ of primitive propositions by the following BNF grammar. We use f for a wff, $p \in \mathcal{P}$ for a primitive proposition, θ for a wfsp, I for a wfim and $d \in Q$ (the set of non-negative rationals) for a duration, each possibly with a subscript.

$$f = \mathsf{true} \mid p \mid \mathsf{len}(0, d] \mid \neg f \mid f_1 \wedge f_2 \mid If$$
$$I = [- \mid \theta) \mid [\theta \mid \rightarrow) \mid [\theta_1 \mid \theta_2)$$
$$\theta = \rightarrow f \mid \rightarrow f, \theta$$

Although in the syntax above, we do not use θ to include the trivial search patterns, "—" and "$\rightarrow$", in the sequel we shall use the meta-variable θ to mean *any* search pattern, trivial or non-trivial, unless explicitly noted otherwise.

$$\Box\left(\neg near \Rightarrow \left(\begin{array}{l}[\rightarrow near \mid \rightarrow)\Diamond(cross \wedge \Diamond\neg cross)\wedge \\ [\rightarrow near \mid \rightarrow near, \rightarrow cross, \rightarrow\neg cross)(green \Rightarrow \mathsf{len}(3.5, 7])\end{array}\right)\right)$$

Figure 2. Textual equivalent of the graphical specification in figure 1.

We consider two special sets of well-formed strings when defining the semantics. The first, the set of wfsp, will be denoted by **srchp**($\mathcal{P}$), and the second, the set of wfim, will be denoted by **imod**($\mathcal{P}$). In addition, we shall call a wff *purely propositional* if it is formed by the following grammar:

$$f = \mathsf{true} \mid p \mid \neg f \mid f_1 \wedge f_2$$

We use false as an abbreviation for $\neg\mathsf{true}$, $f \vee g$ as an abbreviation for $\neg(\neg f \wedge \neg g)$, $\mathsf{len}(d, \infty)$ as an abbreviation for $\neg\,\mathsf{len}(0, d]$, and $\mathsf{len}(d_1, d_2]$ as an abbreviation for $\mathsf{len}(d_1, \infty) \wedge \mathsf{len}(0, d_2]$. The traditional temporal operators are defined by

$$\Diamond f \stackrel{def}{\equiv} \neg[\rightarrow f \mid \rightarrow)\mathsf{false}$$

$$\Box f \stackrel{def}{\equiv} [\rightarrow\neg f \mid \rightarrow)\mathsf{false}$$

$$f \,\mathsf{U}\, g \stackrel{def}{\equiv} [\rightarrow(\neg f \vee g) \mid \rightarrow)g$$

and so on.

In FIL, the formula If, I an interval modality and f a formula, has the semantics "if the interval designated by I exists, then f holds at the initial state within that interval." Syntactically RTFIL is just FIL extended with the timing primitives $\mathsf{len}(0, d]$ for $d \in Q$, the domain of durations. The formula $I\,\mathsf{len}(0, d]$ has the natural interpretation that if the interval I exists then its duration is no more than d. Intuitively, $\mathsf{len}(0, d]$ asserts that the duration of the remaining (suffix) interval is at most d time units. A search to $\mathsf{len}(0, d]$ locates the earliest future point within the current interval, such that the duration of the remaining interval is at most d. Consequently, over an interval of infinite duration $\mathsf{len}(0, d]$ is never satisfied.[2]

A detailed description of the translation of graphical formulæ to the textual syntax is beyond the scope of this paper (details appear in Dillon *et al* (1994)). For purposes of illustration, however, we note that the graphical formula given in figure 1 translates to the textual RTFIL formula given in figure 2.

2.2 *Models*

The models on which we interpret RTFIL formulæ are partial functions from the non-negative reals R (the time domain) to states, which assign valuations to the primitive propositions. We represent a model by a total function $\mathcal{M}$: $R \rightarrow 2^{\mathcal{P}} \cup \{\perp\}$, where $\mathcal{P}$ is the set of primitive propositions and $\perp$ represents undefined.[3] We require a model for RTFIL to satisfy the following requirement of admissibility.

[2]The semantics of formulæ containing wfim that involve timing primitives can be counterintuitive. Thus, while such formulæ are decidable in the logic at no extra cost, their use should probably be avoided.

[3]We assume that all functions and predicates, except equality, are strict, *i.e.* if any of its arguments is $\perp$ then the result of a function or predicate is also $\perp$. For equality, however, we regard $\perp = \perp$ to be true and $\perp = x$ and $x = \perp$ to be false if x is not $\perp$.

DEFINITION 1
[ADMISSIBILITY] A function $F: R \to X \cup \{\perp\}$ is

- *finitely variable* iff, for any two elements $t_1 < t_2$ in R, there are only finitely many changes in F between t_1 and t_2
- *right continuous* iff for any $t \in R$, $\lim_{t' \to t+} F(t') = F(t)$

F is *admissible* iff it is finitely variable, right continuous, dom $F = \{ t \in R \mid F(t) \neq \perp \}$ is a left-closed right-open segment of R, and im $F = \{ x \in X \mid \exists t \in R, F(t) = x \}$ is finite. The above definitions of finite variability and right continuity are stated relative to an arbitrary valuation function on R in order that we can also use them with extended models (see theorem 3 in § 3.2 below), and not just with models. Note that finite variability implies discreteness, but finiteness of the image set is a stronger requirement. These definitions are equivalent to the standard ones in the literature.

Intuitively, the domain of a model represents the interval (or "context") over which a formula is evaluated. Finite variability ensures that a system performs only a finite number of actions in any finite period of time and right continuity guarantees that a property can be observed only if it holds over an interval with a positive duration. Together these conditions imply that corresponding to every proposition p there is a sequence $t_0, t_1, \ldots$ of time values, with $\lim_{i \to \infty} t_i = \infty$, that partition the time domain R into half-open segments $[t_i, t_{i+1})$ over which the valuation of p is constant. We call any model satisfying the above properties an *admissible model*. We write $\perp_{\mathcal{M}}$ for the everywhere undefined model $\perp_{\mathcal{M}}: R \to 2^{\mathcal{P}} \cup \{\perp\}$, which satisfies $\text{dom}(\perp_{\mathcal{M}}) = \emptyset$ and is (trivially) admissible.

An observation regarding the condition of right continuity is in order. The semantics of RTFIL can be generalized to admit models that are not right continuous. However, as long as the semantics are defined so as to be insensitive to instantaneous states, a formula will be satisfiable in the more general class of models precisely if it is satisfiable in the class of right continuous models. Moreover, the semantics of RTFIL are simpler to state and more intuitive if formulæ are interpreted over right continuous models only.

An admissible model $\mathcal{M}$ satisfies an RTFIL formula if the formula is true when evaluated at the initial state of $\mathcal{M}$, where the valuation of formulæ is defined below. If an admissible model represents an entire behaviour of a system, then its domain will be all of R. (To represent a terminating behaviour by such a model, the last state of the behaviour is stuttered.) However, in general, the domain of an admissible model may be any left-closed right-open segment of R.

2.3 *Semantics*

We now give a formal definition of the semantics of RTFIL, which have been explained informally above. The semantics are a natural extension of the FIL semantics (see Ramakrishna *et al* 1992). They are defined here with respect to a dense, rather than a discrete, time domain. Moreover, the syntax of FIL does not contain timing primitives, so that FIL formulæ describe only constraints on the ordering of states.

The semantics make use of the "locator" function λ for locating the result of a search and the "constructor" function C for constructing the subinterval, given the current interval

and the states located by the searches. For brevity, we use $R^{\perp,\infty}$ to denote $R \cup \{\perp, \infty\}$ below.

DEFINITION 2
The search-locator function

$$\lambda: \mathbf{srchp}(\mathcal{P}) \times (2^{\mathcal{P}} \cup \{\perp\})^R \times R^{\perp,\infty} \to R^{\perp,\infty}$$

is defined by

– If $\mathcal{M} = \perp_{\mathcal{M}}$ or $t = \perp$ then

$$\lambda(\theta, \langle \mathcal{M}, t \rangle) = \perp$$

– If $\mathcal{M} \neq \perp_{\mathcal{M}}$ and $t \neq \perp$ then

$$\begin{aligned}
\lambda(-, \langle \mathcal{M}, t \rangle) &= t \\
\lambda(\to, \langle \mathcal{M}, t \rangle) &= \sup \operatorname{dom} \mathcal{M} \\
\lambda(\to a, \langle \mathcal{M}, t \rangle) &= \begin{cases} \perp, \text{ if } \langle \mathcal{M}, t' \rangle \not\models a \text{ for all } t' \geq t, t' \in \operatorname{dom} \mathcal{M} \\ \inf\{t' \mid t' \geq t, \langle \mathcal{M}, t' \rangle \models a\}, \text{ otherwise} \end{cases} \\
\lambda(\to a, \theta, \langle \mathcal{M}, t \rangle) &= \lambda(\theta, \langle \mathcal{M}, \lambda(\to a, \langle \mathcal{M}, t \rangle) \rangle)
\end{aligned}$$

The model-constructor function

$$\mathcal{C}: \mathbf{imod}(\mathcal{P}) \times (2^{\mathcal{P}} \cup \{\perp\})^R \times R \to (2^{\mathcal{P}} \cup \{\perp\})^R$$

is defined by

$$\mathcal{C}([\theta_1 \mid \theta_2), \langle \mathcal{M}, t \rangle) = \mathcal{M}_{\lambda(\theta_1, \langle \mathcal{M}, t \rangle), \lambda(\theta_2, \langle \mathcal{M}, t \rangle)}$$

where $\mathcal{M}_{t_1,t_2}$ with $t_1, t_2 \in R^{\perp,\infty}$, represents the subinterval model defined by

$$\mathcal{M}_{t_1,\perp} = \mathcal{M}_{\perp,t_2} = \perp_{\mathcal{M}}$$

and $\mathcal{M}_{t_1,t_2}$ is the restriction of $\mathcal{M}$ to $[t_1, t_2)$ if $t_1 \neq \perp$ and $t_2 \neq \perp$.

DEFINITION 3
[SEMANTICS] The valuation of an RTFIL formula is defined at a point $t \in \operatorname{dom} \mathcal{M}$ in an admissible model $\mathcal{M} \in (2^{\mathcal{P}} \cup \{\perp\})^R$ using the satisfaction relation defined below.
If $\mathcal{M} = \perp_{\mathcal{M}}$ then

– $\langle \mathcal{M}, t \rangle \models f$

If $\mathcal{M} \neq \perp_{\mathcal{M}}$ then

– $\langle \mathcal{M}, t \rangle \models \mathsf{true}$ and $\langle \mathcal{M}, t \rangle \not\models \mathsf{false}$
– $\langle \mathcal{M}, t \rangle \models p$, for $p \in \mathcal{P}$ iff $p \in \mathcal{M}(t)$
– $\langle \mathcal{M}, t \rangle \models \neg f$ iff $\langle \mathcal{M}, t \rangle \not\models f$
– $\langle \mathcal{M}, t \rangle \models f \wedge g$ iff $\langle \mathcal{M}, t \rangle \models f$ and $\langle \mathcal{M}, t \rangle \models g$

- $\langle \mathcal{M}, t \rangle \models \mathsf{len}(0, d]$ iff $t < \sup \mathrm{dom}\, \mathcal{M} \leq t + d$
- $\langle \mathcal{M}, t \rangle \models If$ iff $\langle \mathcal{M}', \inf \mathrm{dom}\, \mathcal{M}' \rangle \models f$ where $\mathcal{M}' = \mathcal{C}(I, \langle \mathcal{M}, t \rangle)$

We say that f is true at t in $\mathcal{M}$ iff $\langle \mathcal{M}, t \rangle \models f$ and that it is false otherwise.

A formula f is *satisfiable* iff there exists an admissible model $\mathcal{M} \in (2^{\mathcal{P}} \cup \{\perp\})^R$ such that $\mathrm{dom}\, \mathcal{M} = R$ and $\langle \mathcal{M}, 0 \rangle \models f$. We then say that $\mathcal{M}$ is a *satisfying model* for f. A formula f is *valid* iff every admissible model $\mathcal{M}$ for which $\mathrm{dom}\, \mathcal{M} = R$ is a satisfying model for f.

The theorem below follows from the definition of admissibility and from the semantics, by induction on the structure of an RTFIL formula. The proof of this theorem is subsumed by that of theorem 3, which appears in the next section.

Theorem 1. *Let f be any RTFIL formula and let $\mathcal{M}$ be an admissible model. Then for any $t \in R$, $\langle \mathcal{M}, t \rangle \models f$ iff there exists $\epsilon > 0$ such that for all $t \leq t' < t + \epsilon$, $\langle \mathcal{M}, t' \rangle \models f$.*

This theorem motivates the choice of $\mathsf{len}(0, d]$ and, by negation, $\mathsf{len}(d, \infty)$ as timing primitives. If, for instance, we had chosen $\mathsf{len}[d, \infty)$ (with the intuitive semantics) as the basic timing primitive then, if $\mathcal{M}$ is an admissible model with $\mathrm{dom}\, \mathcal{M} = [0, 1)$, we would have $\langle \mathcal{M}, 0 \rangle \models \mathsf{len}[1, \infty)$ although $\langle \mathcal{M}, t \rangle \not\models \mathsf{len}[1, \infty)$ for any $t > 0$, and theorem 1 would no longer be valid.

The significance of theorem 1 springs from the fact that it ensures that any refinement mapping definable in RTFIL preserves admissibility. The absence of such a property would make refinement proofs difficult, since a refinement mapping on a given level might possibly produce an inadmissible model at the next lower level. This means that, at every stage, in order to apply further refinements, one would first have to prove that the previous mapping preserved admissibility. Moreover, it would overly restrict the applicable mappings.

3. Preliminaries

This section introduces important concepts required by the decision procedure. In particular, it describes the timed automata used in deciding satisfiability and the various concepts of reductions and clocks used in the construction of the automata for the decision procedure.

3.1 *Timed Büchi automata and timed ω-strings*

The approach we use for our decision procedure is closely related to the procedure for the untimed logic in Ramakrishna *et al* (1992). The first step in that procedure is the construction of a Büchi Automaton (BA) for a formula, such that the formula is satisfied iff the automaton has a non-empty language. This is the basic automata-theoretic approach (Wolper 1987). However, since RTFIL deals with real-time rather than only order relations, the notion of automata on infinite strings is now extended to that of timed automata on timed strings.

DEFINITION 4
[TIMED ω-STRING] A timed ω-string over the alphabet Σ is an infinite sequence $\langle(\sigma_i, t_i)\rangle_{i\in\omega}$ in $(\Sigma \times R)^\omega$ such that $\langle t_i\rangle_{i\in\omega}$ is an unbounded strictly monotonically increasing sequence, with $t_0 > 0$.

Observe that an admissible model for our logic identifies a timed string over the alphabet $2^{\mathcal{P}}$. In fact, in some of our subsequent proofs we shall use this representation for RTFIL models rather than the one we gave in the previous section.

The following definition of Timed Büchi Automaton (TBA) is a special case of the TBA described in Alur & Dill (1990).

DEFINITION 5
A Timed Büchi Automaton $\mathcal{A}$ is a tuple $\langle \Sigma, S, C, \rho, S_I, S_F\rangle$ where

- Σ is a finite input alphabet
- S is a finite set of states
- C is a finite set of clocks
- $\rho: S \times \Sigma \to 2^{S\times 2^C \times 2^{\Phi(C)}}$ is the transition function where $\Phi(C)$, the set of clock conditions, is the set of inequalities of the form $c \leq t$ and $c = t$, for $c \in C$ and $t \in Q$
- $S_I \subseteq S$ is the set of possible initial states
- $S_F \subseteq S$ is the set of accepting states.

The transition function ρ defines for each state s and input σ a set of triples, where each triple $\langle s', C', \phi\rangle \in \rho(s, \sigma)$ specifies a next state s', a set C' of clocks reset with that transition and a set ϕ of clock conditions that must be satisfied at the moment of the transition. We say that a clock assignment $\gamma \in R^C$ satisfies a set of clock conditions $\phi \subseteq \Phi(C)$ iff the set of inequalities $\phi[c \leftarrow \gamma(c)]$ obtained by replacing each clock variable c in ϕ by the corresponding value $\gamma(c)$ is satisfied.[4] If $\langle s', C', \phi\rangle \in \rho(s, \sigma)$, we say that ρ allows the transition $s \xrightarrow{\sigma, C', \phi} s'$.

A run of $\mathcal{A}$ on an ω-string $\sigma = \langle(\sigma_i, t_i)\rangle_i \in (\Sigma \times R)^\omega$ is an ω-string $\mathcal{R}(\mathcal{A}, \sigma) = \langle(s_i, \gamma_i)\rangle_i \in (S \times R^C)^\omega$ satisfying

- *Initiality*: $s_0 \in S_I$, and for all $c \in C$, $\gamma_0(c) = 0$
- *Transitions*: for each i, there is a set of clocks $C_i \subseteq C$ and a finite set $\phi_i \subset \Phi(C)$ of clock conditions such that
  - ρ allows the transition $s_i \xrightarrow{\sigma_i, C_i, \phi_i} s_{i+1}$
  - the inequalities in $\phi_i[c \leftarrow \gamma_i(c) + t_i - t_{i-1}]$ are satisfied, where $t_{-1} = 0$
  - $\gamma_{i+1}(c) = 0$ for all $c \in C_i$
  - $\gamma_{i+1}(c) = \gamma_i(c) + t_i - t_{i-1}$ for all $c \in C \setminus C_i$

[4] As usual the empty set of conditions imposes no conditions and so is always satisfied.

We write $(s_0, \gamma_0) \xrightarrow{\sigma_0, t_0} (s_1, \gamma_1) \xrightarrow{\sigma_1, t_1} \cdots$ when these conditions hold. Such a run is *accepting* iff the set $\{ i \mid s_i \in \mathbf{S}_F \}$ is infinite. The language of a TBA is non-empty iff there is a timed ω-string over its alphabet on which it has an accepting run.

Intuitively, a TBA reads a *timed* Σ*-string* and makes transitions satisfying its transition function. It has a finite set of clocks, which proceed at the same rate, and which it can reset with a transition or compare with rational constants. Transitions must satisfy the associated clock conditions for the input string to be consumed. The operational semantics of the run shown above are that the automaton stays in state s_i at time t, $t_{i-1} \leq t < t_i$. At time t_i it moves into state s_{i+1} resetting the clocks in C_i. The remaining clocks have meanwhile advanced by the time spent in s_i. The semantics of the input string σ are that it is a model $\mathcal{M}_\sigma$ such that for $t_{i-1} \leq t < t_i$ and $i \in \omega$, $\mathcal{M}_\sigma(t) = \sigma_i$. We say that the TBA $\mathcal{A}$ *consumes* a timed Σ-string when there exists a run of $\mathcal{A}$ on the string and that it *accepts* the string when some such run is accepting.

We note for the sequel that a BA can be regarded as a TBA whose set of clocks is empty. We take this as our definition of BA below. Because the set of clocks of a BA is empty, its transition function is regarded as a function $\rho: S \times \Sigma \to 2^S$, and it ignores the timing information on a timed ω-string.

DEFINITION 6

[UNTIMING] We define a polymorphic untiming function as follows

- When given a timed ω-string $\langle \sigma_i, t_i \rangle_{i \in \omega}$, it returns the untimed ω-string

$$\mathbf{untime}(\langle \sigma_i, t_i \rangle_{i \in \omega}) = \langle \sigma_i \rangle_{i \in \omega}$$

- When given a TBA $\mathcal{A} = \langle \Sigma, \mathbf{S}, \mathbf{C}, \rho, \mathbf{S}_I, \mathbf{S}_F \rangle$, it returns the BA

$$\mathbf{untime}(\mathcal{A}) = \langle \Sigma, \mathbf{S}, \rho', \mathbf{S}_I, \mathbf{S}_F \rangle$$

where the transition function $\rho': \mathbf{S} \times \Sigma \to 2^{\mathbf{S}}$ is defined by

$$\rho'(s, \sigma) = \{ s' \mid \langle s', C, \phi \rangle \in \rho(s, \sigma) \}$$

Lemma 1. For a timed ω-string σ and TBA $\mathcal{A}$, if $\mathcal{A}$ accepts σ then $\mathbf{untime}(\mathcal{A})$ *accepts* $\mathbf{untime}(\sigma)$.

Proof. The statement follows immediately from the definition of untiming. □

Observe that the admissibility requirement makes the acceptance criterion for our TBAs slightly more restrictive than that in Alur & Dill (1990). However, it is not difficult to see that, because of our more restricted edge conditions Φ, if there is any accepting run of the TBA by the less restrictive definition of Alur & Dill (1990), there is a also an admissible model on which the TBA has an accepting run by our definition.[5] Thus, the emptiness algorithm of Alur & Dill (1990) suffices for our purpose.

[5]The latter model is obtained by simply closing each interval of the former model on the left (and opening its successor interval on the right) – that this does not violate any of the transition conditions in the course of an identical run of our automaton on the latter model is easy to establish, using the fact that edge conditions are of the form $c = t$ or $c \leq t$ only.

Theorem 2. Alur & Dill (1990) *It is decidable whether the language of a TBA is empty.*

3.2 *Subformulæ, reductions and extensions*

The concept of subformula closure set, reductor set and reductions on interval formulæ for FIL were introduced in Ramakrishna *et al* (1992). The first is well-known in automata-theoretic approaches in conventional temporal logics. The latter were introduced to simplify the statement of the FIL decision procedure and correspond, roughly, to the so-called rewrite rules used in the method of semantic tableaux.

The definitions that follow are straightforward extensions of those appearing in Ramakrishna *et al* (1992) to take into account the presence of duration formulæ, *i.e.* those involving predicates of the form $\mathsf{len}(0, d]$.

The subformula closure $\mathbf{scl}(f)$ captures the idea that in deciding the satisfiability of the formula f, one need only consider formulæ in the set $\mathbf{scl}(f)$. The formulæ in the set intuitively represent all the "verification conditions" arising in an on-line strategy to verify if f is satisfied by an arbitrary model. As in Fischer & Ladner (1977) our closure is an *extended subformula closure*, sometimes also called *Fischer–Ladner Closure*, in the sense that $\mathbf{scl}(f)$ may contain formulæ that are not syntactic subformulæ of f.

Notation 1. Let I be an interval modality and let F be a set of formulæ. Then $I.F$ denotes the set of formulæ $\{If \mid f \in F\}$. If F is empty then so is $I.F$.

DEFINITION 7
[SUBFORMULA CLOSURE] The *subformula closure* $\mathbf{scl}(f)$ of a formula f is the smallest set such that[6]

a) $f \in \mathbf{scl}(f)$.

b) $\mathsf{true} \in \mathbf{scl}(f)$ and $\mathsf{false} \in \mathbf{scl}(f)$.

c) $f_1 \in \mathbf{scl}(f)$ iff $\neg f_1 \in \mathbf{scl}(f)$.

d) if $f_1 \wedge f_2 \in \mathbf{scl}(f)$ then $f_1 \in \mathbf{scl}(f)$ and $f_2 \in \mathbf{scl}(f)$.

e) if $[\rightarrow a, \theta_1 \mid \theta_2) f_1 \in \mathbf{scl}(f)$ or $[\theta_1 \mid \rightarrow a, \theta_2) f_1 \in \mathbf{scl}(f)$ then $[\theta_1 \mid \theta_2) f_1 \in \mathbf{scl}(f)$.

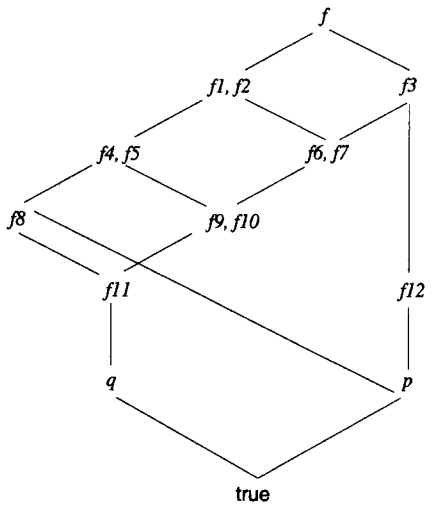
f) if $[\rightarrow a \mid \theta_2) f_1 \in \mathbf{scl}(f)$ then if θ_2 is not $\rightarrow$ then $[- \mid \theta_2) f_1 \in \mathbf{scl}(f)$ and if θ_2 is $\rightarrow$ then $f_1 \in \mathbf{scl}(f)$.

g) if any of $[\rightarrow a, \theta_1 \mid \theta_2) f_1$, $[\theta_1 \mid \rightarrow a, \theta_2) f_1$, $[\rightarrow a \mid \theta_2) f_1$, or $[\theta_1 \mid \rightarrow a) f_1$ is in $\mathbf{scl}(f)$ then $a \in \mathbf{scl}(f)$.

h) if $[\theta_1 \mid \theta_2) f_1 \in \mathbf{scl}(f)$ then

- if θ_1 is not $-$ then $[\theta_1 \mid \rightarrow)\mathsf{false} \in \mathbf{scl}(f)$, and
- if θ_2 is not $\rightarrow$ then $[\theta_2 \mid \rightarrow)\mathsf{false} \in \mathbf{scl}(f)$.

[6] As usual we identify $\neg\neg f_1$ with f_1, $\neg\mathsf{true}$ with false and $\neg\mathsf{false}$ with true.



$$
\begin{aligned}
f &\stackrel{def}{\equiv} [\rightarrow p \mid \rightarrow p, \rightarrow q)\neg\,\mathsf{len}(0,3]\\
f_1 &\stackrel{def}{\equiv} [- \mid \rightarrow p, \rightarrow q)\neg\,\mathsf{len}(0,3]\\
f_2 &\stackrel{def}{\equiv} [- \mid \rightarrow p, \rightarrow q)\,\mathsf{len}(0,3]\\
f_3 &\stackrel{def}{\equiv} [\rightarrow p \mid \rightarrow q)\neg\,\mathsf{len}(0,3]\\
f_4 &\stackrel{def}{\equiv} [- \mid \rightarrow p, \rightarrow q)\mathsf{false}\\
f_5 &\stackrel{def}{\equiv} [- \mid \rightarrow p, \rightarrow q)\mathsf{true}\\
f_6 &\stackrel{def}{\equiv} [- \mid \rightarrow q)\neg\,\mathsf{len}(0,3]\\
f_7 &\stackrel{def}{\equiv} [- \mid \rightarrow q)\,\mathsf{len}(0,3]\\
f_8 &\stackrel{def}{\equiv} [\rightarrow p, \rightarrow q \mid \rightarrow)\mathsf{false}\\
f_9 &\stackrel{def}{\equiv} [- \mid \rightarrow q)\mathsf{false}\\
f_{10} &\stackrel{def}{\equiv} [- \mid \rightarrow q)\mathsf{true}\\
f_{11} &\stackrel{def}{\equiv} [\rightarrow q \mid \rightarrow)\mathsf{false}\\
f_{12} &\stackrel{def}{\equiv} [\rightarrow p \mid \rightarrow)\mathsf{false}
\end{aligned}
$$

Figure 3. Example illustrating the subformula closure definition.

i) if $[- \mid \theta)f_1 \in \mathbf{scl}(f)$ and f_1 is purely propositional then $f_1 \in \mathbf{scl}(f)$ (recall that duration predicates are *not* purely propositional)

j) if $[- \mid \theta)f_1 \in \mathbf{scl}(f)$ then $[- \mid \theta).\mathbf{scl}(f_1) \subseteq \mathbf{scl}(f)$.

Example 1. Let f be the formula $[\rightarrow p \mid \rightarrow p, \rightarrow q)\neg\,\mathsf{len}(0,3]$, where $p, q \in \mathcal{P}$ and let $f_1, \cdots, f_{12}$ represent the subformulæ as shown in figure 3. Then $\mathbf{scl}(f)$ consists of precisely the formulæ f, $f_1, \cdots f_{12}$, p, q, **true** and all their negations. This is shown in the figure in the form of a Hassé Diagram, where a formula f' (and its negation) is in the subformula closure of another formula f'' (or its negation) if either f' and f'' are at the same "node" (such as, for instance, f_1 and f_2) or if f' is below f'' and reachable from it (such as for instance f_{11} and f_1). We assume that at every node, a formula and its negation are both present although, for clarity, we do not explicitly show the negation.

Consider now a model $\mathcal{M}$ and $t \in R$, such that the formulæ p and $[\rightarrow p \mid \rightarrow p, \rightarrow q)\neg\,\mathsf{len}(0, d]$ are both satisfied at $\langle \mathcal{M}, t \rangle$. Clearly, the "search" to p starting at t will locate the current point, so that, as a result, the formula $[- \mid \rightarrow q)\neg\,\mathsf{len}(0, d]$ must also hold at t. Moreover, the formula $[- \mid \rightarrow q)\,\mathsf{len}(0, d]$ must *not* hold at t, unless either q holds at t (the interval "collapses") or q never holds for any $t' \geq t$ (the search "fails"), *i.e.* unless q or $[\rightarrow q \mid \rightarrow)\mathsf{false}$ also holds at t. This notion of a set of formulæ *forcing* the truth of other formulæ is closely related to the concept of (finitary) "forcing" in descriptive set theory, and motivates the following series of definitions, culminating with lemma 2.

DEFINITION 8

[REDUCTOR SET] The *reductor set* $\mathbf{red}(f)$ of a formula f is the smallest set of wff, not containing f, such that

– if f is of the form $[\rightarrow a, \theta_1 \mid \theta_2)f_1$, $[\rightarrow a \mid \theta_2)f_1$, $[\theta_1 \mid \rightarrow a, \theta_2)f_1$

or $[\theta_1 \mid \rightarrow a) f_1$ then $a \in \mathbf{red}(f)$

- if f is of the form $\neg f_1$ then $\mathbf{red}(f_1) \subseteq \mathbf{red}(f)$
- if f is of the form $[\theta_1 \mid \theta_2) f_1$ then
  - if θ_1 is not $-$ then $[\theta_1 \mid \rightarrow)\mathsf{false} \in \mathbf{red}(f)$ and
  - if θ_2 is not $\rightarrow$ then $[\theta_2 \mid \rightarrow)\mathsf{false} \in \mathbf{red}(f)$
- if f is of the form $[- \mid \theta_2) f_1$ then $[- \mid \theta_2).\mathbf{red}(f_1) \subseteq \mathbf{red}(f)$
- if f is of any other form then $\mathbf{red}(f) = \emptyset$.

DEFINITION 9
[REDUCIBILITY] Let a and f be formulæ. Then f is *a-reducible* iff $a \in \mathbf{red}(f)$. Otherwise it is *a-irreducible*. If S is a set of formulæ, then f is *S-reducible* iff it is a-reducible for some $a \in S$.

DEFINITION 10
[REDUCTION] Let a, f be such that $a \in \mathbf{red}(f)$. Then the wff f' is an *a-reduct* of f, written $f' \prec_a f$, iff one of the following holds

- f is of the form $[\rightarrow a, \theta_1 \mid \theta_2) f_1$ or $[\theta_1 \mid \rightarrow a, \theta_2) f_1$ and f' is $[\theta_1 \mid \theta_2) f_1$
- f is of the form $[\rightarrow a \mid \theta_2) f_1$ and f' is $[- \mid \theta_2) f_1$
- f is of the form $[\theta_1 \mid \rightarrow a) f_1$ and f' is true
- f is of the form $[\theta_1 \mid \theta_2) f_1$, a is either $[\theta_1 \mid \rightarrow)\mathsf{false}$ or $[\theta_2 \mid \rightarrow)\mathsf{false}$, and f' is true
- f is of the form $[\rightarrow a \mid \rightarrow) f_1$ and f' is f_1
- f is of the form $\neg f_1$, f' is $\neg f_1'$ and $f_1' \prec_a f_1$
- f is of the form $[- \mid \theta_2) f_1$, a is $[- \mid \theta_2) b$, f' is $[- \mid \theta_2) f_1'$ and $f_1' \prec_b f_1$.

When f is reducible to f' through a chain of reductions with respect to formulæ in a set S, we say $f' \prec_S^* f$.

Example 2. Continuing with example 1, figure 4 illustrates the definitions just given. In the figure, if a formula f' is reachable from a formula f'' by a direct edge labelled with a formula a, then $f' \prec_a f''$. Thus, the fanout labels of a node f' are precisely the formulæ in $\mathbf{red}(f')$. For instance, f is p-reducible but q-irreducible. Moreover, p transitively reduces f to f_6. This reduced formula is now q-reducible, so that $\mathsf{true} \prec_{\{p,q\}}^* f$. Note also that f_8 directly reduces f to true.

Observe that for a wff a, the parameterized reduction operator $\prec_a$ on wff, has been defined so that $f' \prec_a f$ guarantees that $a \Rightarrow (f' \equiv f)$ and $\mathbf{scl}(f') \subset \mathbf{scl}(f)$. This is formalized in the next lemma, which helps motivate the construction of the untimed automaton described in § 4.2.

Lemma 2. Let f, f', a be formulæ and $\mathcal{M}$ be a model such that $\langle \mathcal{M}, t \rangle \models a$ and $f' \prec_a f$. Then $\langle \mathcal{M}, t \rangle \models f$ iff $\langle \mathcal{M}, t \rangle \models f'$.

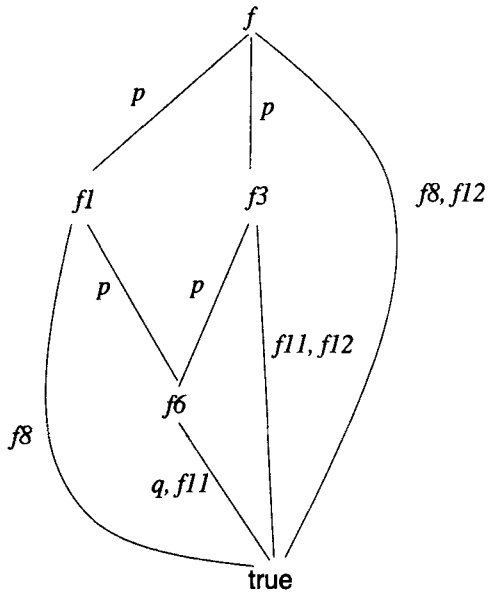


Figure 4. Example illustrating reductions.

Proof. The proof is by a case analysis of the reduction rule used in the reduction (definition 10). The presence of the last rule requires an induction. We use induction on the number of applications of the last rule in the reduction. The base case involves an application of one of the earlier rules, which can be proved easily using the semantics of FIL.

For the induction step, let f be $[- \mid \theta_2) f_1$, a be $[- \mid \theta_2) b$ and f' be $[- \mid \theta_2) f_1'$. To establish the forwards implication, assume that $\langle \mathcal{M}, t \rangle \models f$, $\langle \mathcal{M}, t \rangle \models a$ and $f' \prec_a f$. Using the semantics of the logic, it is clear that $\langle \mathcal{M}', t \rangle \models f_1$, and $\langle \mathcal{M}', t \rangle \models b$, where $\mathcal{M}' = \mathcal{M}_{[t, \lambda(\theta_2, \langle \mathcal{M}, t \rangle))}$. But, by the definition of reduction, $f_1' \prec_b f_1$, so that by the induction hypothesis $\langle \mathcal{M}', t \rangle \models f_1'$. Again, from the semantics, we have $\langle \mathcal{M}, t \rangle \models [- \mid \theta_2) f_1'$ as required. The backwards implication follows similarly. □

COROLLARY 1
Let $\langle \mathcal{M}, t \rangle \models a$ *for all* $a \in S$ *and let* $f' \prec_S^* f$. *Then* $\langle \mathcal{M}, t \rangle \models f$ *iff* $\langle \mathcal{M}, t \rangle \models f'$.

Example 3. Observe, in our running example, that $p \Rightarrow (f \equiv f_1)$, $(p \wedge q) \Rightarrow f$ and $f_8 \Rightarrow f$. Note also that, for a formula f, the formulæ which are the (transitive) reducts of f give rise to a complete lattice under the relation "is a reduct of."

We have so far represented models as mappings from R to the powerset of primitive propositions. It is a useful abstraction for the description of the decision procedure and for the subsequent correctness proofs to *extend* this mapping so that it provides valuations to every formula in $\mathbf{scl}(f)$.

DEFINITION 11
[MODEL EXTENSION] Given an admissible model $\mathcal{M} \in (2^{\mathcal{P}})^R$, its *extension* with respect to an RTFIL formula f is the function $\mathcal{M}^f : R \to 2^{\mathbf{scl}(f)}$ satisfying $\mathcal{M}^f(t) = \{f_1 \mid f_1 \in \mathbf{scl}(f), \langle \mathcal{M}, t \rangle \models f_1\}$. We call $\mathcal{M}^f$ an *extended model*, and each set $\mathcal{M}^f(t)$ an *extended state.*

It is easy to see that for an arbitrary model, extension is well-defined and, thus, that corresponding to every model there is a unique extension with respect to a given formula. Moreover, every state $\mathcal{M}^f(t)$ in an extended model $\mathcal{M}^f$ is

- *consistent* in the sense that for any formula $f' \in \mathbf{scl}(f)$, $f' \in \mathcal{M}^f(t)$ only if $\neg f' \notin \mathcal{M}^f(t)$
- *complete* (up to elements in $\mathbf{scl}(f)$) in the sense that for any formula $f' \in \mathbf{scl}(f)$ either $f' \in \mathcal{M}^f(t)$ or $\neg f \in \mathcal{M}^f(t)$.

Theorem 3. *Admissibility of models is preserved under extension.*

Recall that the real-line is partitioned by any primitive proposition P into a sequence of segments over which the valuation of P is constant. We may extend this concept to arbitrary subsets of formulæ in $\mathbf{scl}(f)$, such that two points $t_1 \leq t_2 \in R$ are in the same equivalence class iff all points t such that $t_1 \leq t \leq t_2$ yield the same valuation for all formulæ in the set. Intuitively, our proof of theorem 3 uses the fact that the partition of the real-line induced by any RTFIL formula f, not involving duration predicates, is at most as fine as the coarsest partition that refines the partitions induced by the formulæ in $\mathbf{scl}(f) \setminus \{f, \neg f\}$. Moreover, if every equivalence class belonging to one of a finite set of partitions is left-closed and right-open, then so is every equivalence class in the coarsest partition that refines these partitions. For formulæ containing duration predicates, we note that there is at most one (right-continuous) change in the valuation of a duration predicate in any finite segment of R, and no change in any infinite segment of R.

The proof of theorem 3 makes heavy use of the following lemma, the proof of which is straightforward.

Lemma 3. Let X_1 and X_2 be finitely variable and right continuous functions from R to finite subsets of a set S. Let $P(b_1, \cdots, b_n)$ be a boolean function of n variables $b_1, \cdots, b_n$, and let $x_1, \ldots, x_n$ be elements of S. Then the functions[7]

1. $X: R \to 2^S$ *defined by* $X(t) = X_1(t) \cup X_2(t)$
2. $B: R \to \{true, false\}$ *defined by* $B(t) = P[b_i \leftarrow (x_i \in X_1(t))]_i$

are also finitely variable (FV) and right-continuous (RC).

Proof of theorem 3. Let $\mathcal{M}$ be an admissible model. Then dom $\mathcal{M}^f$ = dom $\mathcal{M}$. Moreover, since $\mathbf{scl}(f)$ is finite for any formula f, clearly $\mathcal{M}^f$ is image finite. It remains to prove that $\mathcal{M}^f$ is right-continuous and finitely variable. The proof is by induction on the inclusion order induced by the subformula closure.

For the first of two base cases, we note that

$$\mathcal{M}^p(t) = \begin{cases} \{\mathsf{true}, p\} & \text{if } p \in \mathcal{M}(t) \\ \{\mathsf{true}, \neg p\} & \text{otherwise} \end{cases}$$

Finite variability and right continuity of $\mathcal{M}^p$ then follows easily from that of $\mathcal{M}$ for any $p \in \mathcal{P}$.

[7] The abbreviation $P[x_i \leftarrow y_i]_i$ denotes simultaneous substitution of y_i for x_i, for every i.

For the remaining base case, we note that $\sup \operatorname{dom} \mathcal{M} \neq t$ for $t \in \operatorname{dom} \mathcal{M}$, so that, for any $t \in \operatorname{dom}(\mathcal{M})$, $d \in Q$,

$$\mathcal{M}^{\mathsf{len}(0,d]}(t) = \begin{cases} \{\mathsf{true}, \mathsf{len}(0,d]\} & \text{if } \sup \operatorname{dom} \mathcal{M} - t \leq d \\ \{\mathsf{true}, \neg\, \mathsf{len}(0,d]\} & \text{otherwise} \end{cases}$$

Thus there is at most one right-continuous change in the valuation of $\mathcal{M}^{\mathsf{len}(0,d]}$ over $\operatorname{dom} \mathcal{M}^{\mathsf{len}(0,d]}$.

For the induction step, we consider two sample cases. The remaining cases are similar.

CASE 1. Consider $\mathcal{M}^{f_1 \wedge f_2}$. We have

$$\mathcal{M}^{f_1 \wedge f_2}(t) = \mathcal{M}^{f_1}(t) \cup \mathcal{M}^{f_2}(t) \cup X(t)$$

where

$$X(t) = \begin{cases} \{f_1 \wedge f_2\} & \text{if } f_1 \in \mathcal{M}^{f_1}(t) \text{ and } f_2 \in \mathcal{M}^{f_2}(t) \\ \{\neg(f_1 \wedge f_2)\} & \text{otherwise} \end{cases}$$

Clearly X is FV and RC by the second clause of lemma 3, since $\mathcal{M}^{f_1}$ and $\mathcal{M}^{f_2}$ are. By the first clause of lemma 3, so is $\mathcal{M}^{f_1 \wedge f_2}$.

CASE 2. Consider now the case of $\mathcal{M}^f$ with $f = [\rightarrow a, \theta_1 \mid \rightarrow b, \theta_2) f'$.

From the definitions of extension and subformula closure we have

$$\mathcal{M}^f(t) = \bigcup_{i=1}^{4} \mathcal{M}^{f_i}(t) \cup \mathcal{M}^a(t) \cup \mathcal{M}^b(t) \cup X(t)$$

with

$$X(t) = \begin{cases} \{f\} & \text{if } \begin{cases} f_1 \in \mathcal{M}^{f_1}(t), \text{ or} \\ f_2 \in \mathcal{M}^{f_1}(t), \text{ or} \\ a \in \mathcal{M}^a(t) \text{ and } f_3 \in \mathcal{M}^{f_3}(t), \text{ or} \\ b \in \mathcal{M}^b(t) \text{ and } f_4 \in \mathcal{M}^{f_4}(t), \text{ or} \\ B(t) \end{cases} \\ \{\neg f\} & \text{otherwise} \end{cases}$$

where

$$f_1 = [\rightarrow a, \theta_1 \mid \rightarrow)\mathsf{false}$$
$$f_2 = [\rightarrow b, \theta_2 \mid \rightarrow)\mathsf{false}$$
$$f_3 = [\theta_1 \mid \rightarrow b, \theta_2) f'$$
$$f_4 = [\rightarrow a, \theta_1 \mid \theta_2) f'$$

and $B(t)$ is a boolean condition defined by

$$B(t) = \begin{cases} \exists t' > t \left(\begin{array}{l} a \in \mathcal{M}^a(t') \wedge f_3 \in \mathcal{M}^{f_3}(t') \\ \forall t'' \left(t \leq t'' < t' \Rightarrow \neg b \in \mathcal{M}^b(t'') \neg a \in \mathcal{M}^a(t'') \right) \end{array} \right) \\ \text{or} \\ \exists t' > t \left(\begin{array}{l} b \in \mathcal{M}^b(t') f_4 \in \mathcal{M}^{f_4}(t') \\ \forall t'' \left(t \leq t'' < t' \Rightarrow \neg b \in \mathcal{M}^b(t'') \neg a \in \mathcal{M}^a(t'') \right) \end{array} \right) \end{cases}$$

We now show that $B(t)$ is itself RC and FV. By the induction hypothesis each of the functions $\mathcal{M}^a$, $\mathcal{M}^b$, $\mathcal{M}^{f_3}$ and $\mathcal{M}^{f_4}$ is RC and FV. Consider now an arbitrary point $t \in \text{dom}\,\mathcal{M}$. We have the following possibilities. Either $a \in \mathcal{M}^a(t)$ or $b \in \mathcal{M}^b(t)$ or neither. In the first two cases $B(t)$ is false, and continues to be false at least up to (but possibly not including) the least t' where neither $a \in \mathcal{M}^a(t')$ nor $b \in \mathcal{M}^b(t')$. Consider therefore the third case, for which $\neg a \in \mathcal{M}^a(t)$ and $\neg b \in \mathcal{M}^b(t)$. Now we have two cases depending on whether there is any point $t' > t$ where either $a \in \mathcal{M}^a(t')$ or $b \in \mathcal{M}^b(t')$.

- Assume not. Then clearly B continues to be false for all $t' \geq t$.
- In the alternative case, let $t' > t$ be the least point such that either $a \in \mathcal{M}^a(t')$ or $b \in \mathcal{M}^b(t')$. Then B is false on $[t, t')$ if

$$\neg((a \in \mathcal{M}^a(t')f_3 \in \mathcal{M}^{f_3}(t')) \vee (b \in \mathcal{M}^b(t')f_4 \in \mathcal{M}f_4(t'))),$$

  and otherwise B is true on $[t, t')$.

This establishes the RC of B.

Let $D_{\mathcal{M}^a}$ represent the set of points at which $\mathcal{M}^a$ has a (left) discontinuity, and similarly $D_{\mathcal{M}^b}$ for $\mathcal{M}^b$. For a subset S of R and $t \in R$, let $S \downarrow t = \{ s \in S \mid s \leq t \}$. The FV condition for $\mathcal{M}^a$ is then equivalent to saying that $D_{\mathcal{M}^a} \downarrow t$ is finite for any $t \in R$. By the induction hypothesis $\mathcal{M}^a$ and $\mathcal{M}^b$ are FV, so each of $D_{\mathcal{M}^a}$ and $D_{\mathcal{M}^b}$ has this property and, therefore, so also does $D_{\mathcal{M}^a} \cup D_{\mathcal{M}^b}$, and *a fortiori* any subset of $D_{\mathcal{M}^a} \cup D_{\mathcal{M}^b}$. As our argument above for RC of B clearly shows, B is constant between any two consecutive points (in the usual ordering) in $D_{\mathcal{M}^a} \cup D_{\mathcal{M}^b}$. Therefore, $D_B \subseteq D_{\mathcal{M}^a} \cup D_{\mathcal{M}^b}$, giving FV for B.

Now, using lemma 3 we obtain RC and FV, first for X, and then for $\mathcal{M}^f$. □

Note that right continuity of $\mathcal{M}^f$ for an arbitrary f gives us theorem 1 as a corollary to theorem 3.

The above theorem plays a crucial role in our completeness proof. The automata that we build in the sequel operate on extended models. Satisfying models for f are obtained by restricting the extended models accepted by the automaton for f to the set $\mathcal{P}$ of primitive propositions.

Our definition of reductions yields the following property of extensions, which helps motivate the construction of the untimed automaton in § 4.2.

Notation 2. In what follows, $\mathcal{I}$ represents a string of *zero or more* interval modalities of the form $[- \mid \theta)$, which we refer to as *current modalities*.

Lemma 4. Let $\mathcal{M}$ be an admissible model and $f_1 \in \mathbf{scl}(f)$ be $\mathcal{M}^f(t)$-irreducible. Let $t' \in R$ be the least $t' > t$ such that $\mathcal{M}^f(t) \neq \mathcal{M}^f(t')$. Then

a) if f_1 is $\mathcal{I}[\theta_1|\theta_2)f_2$ where θ_1 is not $-$ then $\langle \mathcal{M}, t\rangle \models f_1$ iff $\langle \mathcal{M}, t'\rangle \models f_1$

b) if f_1 is $\mathcal{I}\neg[\theta_1|\theta_2)f_2$ where θ_1 is not $-$ then $\langle \mathcal{M}, t\rangle \models f_1$ iff both $\langle \mathcal{M}, t'\rangle \models f_1$ and $\langle \mathcal{M}, t'\rangle \not\models \mathcal{I}\mathsf{false}$

c) if f_1 is $\mathcal{I}\,\mathsf{len}(0, d]$ and $\langle \mathcal{M}, t\rangle \models f_1$ then $\langle \mathcal{M}, t'\rangle \models \mathcal{I}\neg\,\mathsf{len}(0, d]$ iff $\langle \mathcal{M}, t'\rangle \models \mathcal{I}\mathsf{false}$

d) if f_1 *is* $\mathcal{I}\neg\,\mathsf{len}(0, d]$ *and* $\langle \mathcal{M}, t\rangle \models f_1$ *then* $\langle \mathcal{M}, t'\rangle \not\models \mathcal{I}\mathsf{false}$.

Intuitively, in the first case, if $\mathcal{I}[\theta_1 \mid \theta_2)$ can be constructed, it lies in the strict future of t and, therefore, in the reflexive future of t'. In the second case, $[\theta_1 \mid \theta_2)$ *can* be constructed within $\mathcal{I}$ (its surrounding context), so $\mathcal{I}$ cannot collapse at t'. For the third case $\mathcal{I}$ must collapse at t' since its duration cannot increase in going from t to t'. Finally, for the last case, $\mathcal{I}$ cannot collapse before its duration becomes less than d (at the earliest such point $\mathcal{I}\,\mathsf{len}(0, d]$ must hold).

In the following proof, we say that "a search $\rightarrow a$ at t *resolves* at a point $t' \geq t$ in a model $\mathcal{M}$" when either

- $t' = t$ and $\langle \mathcal{M}, t\rangle \models a$, or
- $t' > t$, $\langle \mathcal{M}, t'\rangle \models a$ and for all t'' such that $t \leq t'' < t'$, $\langle \mathcal{M}, t''\rangle \not\models a$.

Proof sketch of lemma 4. Proofs of each of the four clauses are sketched below.

[CLAUSE 1.] We sketch only the proof of the forwards direction, the reverse direction follows by similar arguments. From the definition of reductions, we know that since f_1 is $\mathcal{M}^f(t)$-irreducible, all searches in $\mathcal{I}$, θ_1 and θ_2 must resolve in the strict future of t and not before t'. The semantics of searches immediately gives us $\langle \mathcal{M}, t'\rangle \models f_1$. Note that none of the searches in $\mathcal{I}$, θ_1 or θ_2 can "fail" since our definition of reductions ensures the reducibility of f_1 to true in $\mathcal{M}^f(t)$ in such a case.

[CLAUSE 2.] Once again, we shall sketch only a proof for the forwards direction. For the forwards direction, the proof that the first consequent follows is essentially along the lines of the last case. We show why the second consequent, $\langle \mathcal{M}, t'\rangle \not\models \mathcal{I}\mathsf{false}$, also holds. Assume for a contradiction that $\langle \mathcal{M}, t\rangle \models \mathcal{I}\neg[\theta_1 \mid \theta_2)f_2$, $\langle \mathcal{M}, t'\rangle \models \mathcal{I}\mathsf{false}$ and f_1 is $\mathcal{M}^f(t)$-irreducible. As in the last clause, then, all of the searches in $\mathcal{I}$, θ_1 and θ_2 will resolve in the strict future and not before t'. Since all modalities in $\mathcal{I}$ are current, the left endpoints of all these intervals are at t. With the above, $\langle \mathcal{M}, t'\rangle \models \mathcal{I}\mathsf{false}$ implies that the right endpoint of one of the intervals in $\mathcal{I}$ was located at t'. From the semantics, therefore, in fact $\langle \mathcal{M}, t'\rangle \models \mathcal{I}f'$ for an arbitrary formula f', and in particular, $\langle \mathcal{M}, t'\rangle \models \mathcal{I}[\theta_1 \mid \rightarrow)\mathsf{false}$. Now, since $\mathcal{I}[\theta_1 \mid \theta_2)f$ is $\mathcal{M}^f(t)$-irreducible, so also is $\mathcal{I}[\theta_1 \mid \rightarrow)\mathsf{false}$. By the reverse direction of clause 1 above, therefore, $\langle \mathcal{M}, t\rangle \models \mathcal{I}[\theta_1 \mid \rightarrow)\mathsf{false}$. But $\mathcal{I}[\theta_1 \mid \rightarrow)\mathsf{false} \in \mathbf{red}(f_1)$ thus contradicting the assumption of irreducibility of f_1 in $\mathcal{M}^f(t)$.

[CLAUSE 3.] From the semantics, we know that all searches in $\mathcal{I}$ resolve in the strict future, not before t'. Thus the right endpoint of the instance of interval $\mathcal{I}$, cannot be before t'. If it is at t', then some search in $\mathcal{I}$ caused an interval to "collapse" at t', so that from the semantics $\langle \mathcal{M}, t'\rangle \models \mathcal{I}\mathsf{false}$ and, therefore, also $\langle \mathcal{M}, t'\rangle \models \mathcal{I}\neg\,\mathsf{len}(0, d]$. If not (*i.e.* if the right endpoint is in the future of t'), assume that the right endpoint is located at some $t_1 > t'$, then from the semantics, $t_1 \leq t + d$. Moreover, since the instance of interval $\mathcal{I}$ beginning at t' also ends at t_1, surely the duration of that interval is also less than d, since from the above $t_1 < t' + d$. In this case, both $\langle \mathcal{M}, t'\rangle \not\models \mathcal{I}\neg\,\mathsf{len}(0, d]$ and $\langle \mathcal{M}, t'\rangle \not\models \mathcal{I}\mathsf{false}$.

[CLAUSE 4.] The argument here is quite similar to the previous. The right end-point of the instance of interval $\mathcal{I}$ starting at t ends either at t' or later. In the first case, there must exist a point between t and t' at which an instance of the interval $\mathcal{I}$ (also ending at t') has duration at most d. At this time (say, t_1), we have $\langle \mathcal{M}, t_1\rangle \models \mathcal{I}\,\mathsf{len}(0, d]$, contradicting

Table 1. Example illustrating model extension.

	$[0, 1)$	$[1, 4)$	$[4, 7)$	$[7, \infty)$
true	1	1	1	1
p	0	1	1	1
q	0	0	0	1
f	1	1	0	1
f_1	1	1	0	1
f_2	0	0	1	1
f_3	1	1	0	1
f_4	0	0	0	1
f_5	1	1	1	1
f_6	1	1	0	1
f_7	0	0	1	1
f_8	0	0	0	0
f_9	0	0	0	1
f_{10}	1	1	1	1
f_{11}	0	0	0	0
f_{12}	0	0	0	0

the assumption that t' is the first time greater than t at which the $\mathcal{M}^f$ changes. Thus, we need only consider the second case. In such a case, the instance of $\mathcal{I}$ starting at t' cannot collapse at t', giving us the result. □

Example 4. Let $\mathcal{M}$ be defined by $\mathcal{M}(t) = \emptyset$ for $t \in [0, 1)$, $\mathcal{M}(t) = \{p\}$ for $t \in [1, 7)$, and $\mathcal{M}(t) = \{p, q\}$ for $t \in [7, \infty)$. The reader can verify that $\mathcal{M}^f(t)$ is defined by the matrix shown in table 1, where the f_i are as defined in figure 3. In the table, a row denotes an interval I of R. A formula appearing in a column is in $\mathcal{M}^f(t)$, $t \in I$, iff the entry in that column is a 1 and its negation is in $\mathcal{M}^f(t)$ iff the entry in that column is a 0. The example also illustrates the ideas in lemmas 2 and 4.

Finally, we introduce the following notation that we use to describe the construction of the eventuality automaton.

DEFINITION 12
[BASIS] Let f, f' be formulæ and let S be a set of formulæ such that $f' \prec_S^* f$ and f' is S-irreducible. Then f' is a *basis formula* for f with respect to S, and is denoted by $f' = \langle f \rangle_S$.

Note that the basis formula for any formula with respect to a given set is unique. The proof relies on the local confluence property of $\prec_S$ and the absence of infinite descending chains. This ensures global confluence by Newman's Diamond Lemma (Newman 1942). It is useful to bear this in mind (and we shall implicitly assume this in our subsequent exposition) although we do not require this property for any of our subsequent proofs.

Example 5. For the case of example 4, for instance, $f_6 = \langle f \rangle_{\mathcal{M}^f(t)}$ for $t \in [1, 7)$ and $\mathsf{true} = \langle f \rangle_{\mathcal{M}^f(t)}$ for $t \in [7, \infty)$. Note also that f is irreducible at $t \in [0, 1)$ and is (trivially) its own basis with respect to $\mathcal{M}^f(t)$, $t \in [0, 1)$.

3.3 *Interval reductions, clocks and conditions*

In Example 4 there are no formulæ involving nested interval modalities. However, in general, a formula may involve nested modalities, so that for ease in describing our constructions, we require the more general machinery below.

Roughly speaking, the essential "real-time" unit of manipulation by the TBA is a timed current interval formula of the form $\mathcal{I}\,\mathsf{len}[0, d]$ or $\mathcal{I}\neg\,\mathsf{len}[0, d]$, where $\mathcal{I} = [-|\theta_1)[-|\theta_2) \cdots [-|\theta_n)$ is a string of zero or more current interval modalities. For the case of such formulæ, we also need the concept of an interval-reduct. Interval reduction is a relation on strings of current interval modalities and is parameterized by a set of formulæ.

DEFINITION 13
[INTERVAL REDUCTION] Let $\mathcal{I}$ and $\mathcal{I}'$ denote strings of current interval modalities and let S be a set of RTFIL formulæ. Then $\mathcal{I}'$ is an *interval reduct* of $\mathcal{I}$ with respect to S iff $\mathcal{I}'\mathsf{true} \prec_S^* \mathcal{I}\mathsf{true}$. We represent this by $\mathcal{I}' \sqsubset_S^* \mathcal{I}$ and we say that $\mathcal{I}$ is S-reducible.

Note that $\mathcal{I}'$ above may be the "empty" sequence of modalities (which we suppress), which is irreducible with respect to any S. We shall simply say "$\mathcal{I}'$ is a reduct of $\mathcal{I}$" instead of "$\mathcal{I}'$ is an interval reduct of $\mathcal{I}$," when there is no confusion.

Among the possible reductions on an interval modality is a special kind of reduction called a *collapsing reduction*. A collapsing reduction may trigger the checking of clock conditions on a transition that was just taken, and so our procedure must treat it differently from a non-collapsing reduction. This will become clear later when we describe the TBA construction.

DEFINITION 14
[COLLAPSING REDUCTIONS] Let $\mathcal{I} = I_1 I_2 \cdots I_n$ and $\mathcal{I}' = I_1' I_2' \cdots I_m'$ be such that $\mathcal{I}' \sqsubset_S^* \mathcal{I}$ and $m < n$. Then $\mathcal{I}'$ is a *collapsed reduct* of $\mathcal{I}$ and the corresponding operation is a *collapsing reduction*, written $\sqsubset_S^{\downarrow}$.

The important property of interval reductions that we require for the sequel is as follows. Suppose $\mathcal{M}$ is admissible, $t \in R$ and $\mathcal{I}$ is $\mathcal{M}^f(t)$-irreducible. Suppose further that there is a next (least) time $t' > t$ such that $\mathcal{M}^f(t') \neq \mathcal{M}^f(t)$. Then $\mathcal{I}$ is $\mathcal{M}^f(t')$-reducible to $\mathcal{I}'$ if $\mathcal{I}$ is of the form $\mathcal{I}_1[- \mid \rightarrow a)\mathcal{I}_2$ or $\mathcal{I}_1[- \mid \rightarrow a, \theta)\mathcal{I}_2$ where $\mathcal{I}_1 a \in \mathcal{M}^f(t')$. Intuitively, then, $\mathcal{I}$ is equivalent to the syntactically simpler formula $\mathcal{I}'$ when evaluated at t'. Moreover, the reduction of $\mathcal{I}$ in $\mathcal{M}^f(t')$ is collapsing in the case that $\mathcal{I}$ has the first form. Essentially, this means that, if the interval $\mathcal{I}$ is evaluated at time t, it will "end" at time t' and, if it is evaluated at t', it will be empty.

Example 6. Continuing with Example 4, the modality $[- \mid \rightarrow q)$ collapses at all $t \in [7, \infty)$. The modality $[- \mid \rightarrow p, \rightarrow q)$ reduces to $[- \mid \rightarrow q)$ at $t \in [1, \infty)$ and collapses at

$t \in [7, \infty)$. In each case the "set" with respect to which the collapse or reduction occurs is $\mathcal{M}^f(t)$, for the appropriate t.

We also use reductions on intervals to keep track of the "remaining searches" of an interval as it is timed by an active clock of the automaton.

The clock closure and clock condition sets defined below represent the clocks and associated conditions required by a TBA during the satisfiability procedure. Thus, while deciding a formula f, the automaton $\mathcal{A}(f)$ never needs any timers other than those in **clocks**(f) and the conditions appearing on its transitions are contained in the set **clkconds**(f).

DEFINITION 15
[CLOCK CLOSURES] Given a formula f its *clock closure set*, denoted **clocks**(f), is the smallest set satisfying the following conditions:

1. if $\mathcal{I}\,\mathsf{len}(0, d] \in \mathbf{scl}(f)$ then $c_{\mathcal{I}}^{\alpha,\mathcal{I},d} \in \mathbf{clocks}(f)$
2. if $c_{\mathcal{I}_1}^{\alpha,\mathcal{I},d} \in \mathbf{clocks}(f)$ and $\mathcal{I}_2 \sqsubset_S^* \mathcal{I}_1$, for $S \subseteq \mathbf{scl}(f)$, then $c_{\mathcal{I}_2}^{\alpha,\mathcal{I},d} \in \mathbf{clocks}(f)$
3. if $c_{\mathcal{I}_1}^{\alpha,\mathcal{I},d} \in \mathbf{clocks}(f)$ then $c_{\mathcal{I}_1}^{\beta,\mathcal{I},d} \in \mathbf{clocks}(f)$

DEFINITION 16
[CLOCK CONDITION SET] Given a formula f, its *clock condition set*, **clkconds**(f) is the set of conditions of the form

- $c \leq d$ for all $c = c_{\mathcal{I}_1}^{\alpha,\mathcal{I},d} \in \mathbf{clocks}(f)$
- $c = d$ for all $c = c_{\mathcal{I}_1}^{\beta,\mathcal{I},d} \in \mathbf{clocks}(f)$

Intuitively, α-clocks enforce upper-bound constraints and β-clocks enforce lower-bound constraints. States in the TBA for a formula will contain "clock-activity sets," which indicate the clocks that are active. The clock $c_{\mathcal{I}_1}^{\gamma,\mathcal{I},d}$ (where γ is either α or β) will be made active at a state within an instance of an interval $\mathcal{I}$ when it is necessary to time $\mathcal{I}$, and $\mathcal{I}_1$ is the interval that remains to complete the instance of $\mathcal{I}$.

Example 7. Let f be $[\rightarrow p \mid \rightarrow p, \rightarrow q)\neg\,\mathsf{len}(0, 3]$. Then **clocks**($f$) contains the clocks, $c_{[\rightarrow p,\rightarrow q)}^{\alpha,[\rightarrow p,\rightarrow q),3}$, $c_{[\rightarrow q)}^{\alpha,[\rightarrow p,\rightarrow q),3}$, $c_{[\rightarrow q)}^{\alpha,[\rightarrow q),3}$, and their β counterparts. [8] The clock condition associated with $c = c_{[\rightarrow q)}^{\alpha,[\rightarrow q),3}$ is $c \leq 3$ and with its β-counterpart $c' = c_{[\rightarrow q)}^{\beta,[\rightarrow q),3}$ is $c' = 3$.

As in Ramakrishna *et al* (1992), let the number of logical connectives and primitive propositions in an RTFIL formula be its *size*, and the depth of nesting of interval modalities, plus one, be its *depth*. The following lemma is straightforward.

Lemma 5. For an RTFIL formula f of size n and depth k, $|\mathbf{scl}(f)| = O(n^k)$ and $|\mathbf{clocks}(f)| = O(n^{2k})$.

[8] We are using the abbreviation $[\theta)$ for $[- \mid \theta)$.

4. Decision procedure

We now have most of the formal machinery required to describe the construction of the TBA $\mathcal{A}_m(f)$ corresponding to a formula f, whose satisfiability is being checked. The construction of $\mathcal{A}_m$ is described in four steps.

In the first step, we construct a BA $\mathcal{A}_u(f)$ containing timing assertions in its states. This construction is similar to the construction of the local automaton for the untimed case Ramakrishna *et al* (1992). Intuitively, the automaton produced in this first step ensures that all timing-independent safety conditions are satisfied and also checks some simple consistency conditions relating to real-time. The BA $\mathcal{A}_u(f)$ accepts the untiming of any timed string corresponding to a model of f, but may also accept other strings, since it does not fully take into account the real-time constraints imposed by f. The states of $\mathcal{A}_u(f)$ are annotated by timing assertions that encode these constraints.

The second step is the heart of the construction. This step constructs a TBA, $\mathcal{A}_t(f)$, from $\mathcal{A}_u(f)$ in such a manner that all timing assertions, of the form $\mathcal{I}\,\mathsf{len}(0, d]$ and $\mathcal{I}\neg\,\mathsf{len}(0, d]$, annotating the states of $\mathcal{A}_u(f)$ are encoded as timer related actions of the TBA. Each state of the TBA $\mathcal{A}_t(f)$ has a set of "active clocks," a subset of $\mathbf{clocks}(f)$, that is uses to enforce the timing assertions. The edges of $\mathcal{A}_t(f)$ have timer resetting and comparison actions. $\mathcal{A}_t(f)$, thus, ensures that all timing based properties are handled properly, in addition to the timeless safety conditions. In this connection, it is useful to note that a time-bounded liveness property is really a safety property; the time bound must not pass before the liveness property is satisfied. That the requisite time must eventually pass — the condition of non-Zenoness — is an *implicit* liveness condition.

In order to take care of the timeless liveness conditions, we construct the eventuality automaton $\mathcal{A}_e(f)$ in the third step of the construction of $\mathcal{A}_m$. The eventuality automaton is a pure BA, without any timers. It is constructed in essentially the same manner as for FIL (Ramakrishna *et al* 1992).

The final automaton $\mathcal{A}_m(f)$ is a product of $\mathcal{A}_t(f)$ and $\mathcal{A}_e(f)$. The formula f is satisfiable iff the TBA $\mathcal{A}_m(f)$ accepts some timed string. We use the procedure by Alur & Dill (1990) to solve the emptiness problem.

An interesting aspect of RTFIL is reflected in this construction. The local automaton $\mathcal{A}_t(f)$ might consume non-Zeno runs, but $\mathcal{A}_m(f)$ does not. This is because, in RTFIL, unlike for instance MITL (Alur *et al* 1991), there is an implicit liveness condition associated with every timing constraint, namely, the right endpoint of an interval satisfying the timing constraint is eventually found. This allows us to, in effect, dispense with the "progressiveness check" that Alur & Dill (1990) require while checking the emptiness of the final TBA.

4.1 *Hintikka sets*

Most constructive decision procedures use sets of formulæ to construct the "components" of a canonical model for a given formula. The formulæ in the sets, like the states in the model extensions above, give a complete characterization of that component of the model in terms of not only the atomic formulæ (primitive propositions), but also more

complicated formulæ. Following tradition (Smullyan 1968; Emerson 1990) we call such a set of formulæ a *Hintikka set* for an RTFIL formula.

DEFINITION 17
[HINTIKKA SET] A *Hintikka set for a formula* f is a subset $\mathbf{s}$ of $\mathbf{scl}(f)$ satisfying the following conditions:

1. for all $f_1 \in \mathbf{scl}(f)$, $f_1 \in \mathbf{s}$ iff $\neg f_1 \notin \mathbf{s}$
2. for all $\mathcal{I}\mathsf{true} \in \mathbf{scl}(f)$ such that $\mathcal{I}\mathsf{true}$ is s-irreducible, $\mathcal{I}\mathsf{true} \in \mathbf{s}$ and $\mathcal{I}\mathsf{false} \notin \mathbf{s}$
3. for all $\mathsf{len}(0, d] \in \mathbf{scl}(f)$, $\neg\,\mathsf{len}(0, d] \in \mathbf{s}$
4. for all $\mathcal{I}f_1 \in \mathbf{scl}(f)$ such that $\mathcal{I}f_1$ is s-irreducible, $\mathcal{I}f_1 \in \mathbf{s}$ iff $\mathcal{I}\neg f_1 \notin \mathbf{s}$
5. for all $\mathcal{I}(f_1 \wedge f_2) \in \mathbf{scl}(f)$ such that $\mathcal{I}(f_1 \wedge f_2)$ is s-irreducible, $\mathcal{I}(f_1 \wedge f_2) \in \mathbf{s}$ iff $\mathcal{I}f_1 \in \mathbf{s}$ and $\mathcal{I}f_2 \in \mathbf{s}$
6. for all $\mathcal{I}f_1 \in \mathbf{scl}(f)$ such that f_1 is purely propositional and $\mathcal{I}f_1$ is s-irreducible, $\mathcal{I}f_1 \in \mathbf{s}$ iff $f_1 \in \mathbf{s}$ (note that $\mathsf{len}(0, d]$ is *not* propositional)
7. for all $\mathcal{I}\neg[-|\theta_2)f_1 \in \mathbf{scl}(f)$ such that $\mathcal{I}\neg[-|\theta_2)f_1$ is s-irreducible, $\mathcal{I}\neg[-|\theta_2)f_1 \in \mathbf{s}$ iff $\mathcal{I}[-|\theta_2)\neg f_1 \in \mathbf{s}$
8. for all $f_1, f_2 \in \mathbf{scl}(f)$ such that $f_1 \prec^*_{\mathbf{s}} f_2$, $f_1 \in \mathbf{s}$ iff $f_2 \in \mathbf{s}$

The set of all Hintikka sets for f is denoted $\mathbf{H}(f)$.

As a result of the first rule, Hintikka sets are complete and consistent in the sense of p 160. However, they may contain temporal inconsistencies that may make them unsatisfiable. The completeness proof for our decision procedure uses the fact that if a set is not Hintikka then it is unsatisfiable. Thus, it suffices to consider Hintikka sets in the automaton construction, as we shall see shortly.

Lemma 6. Any complete subset of $\mathbf{scl}(f)$ *that is not Hintikka is not satisfiable.*

Proof. Assume that $\mathbf{s}$ is a complete subset of $\mathbf{scl}(f)$ that is not Hintikka. We use a case analysis on the condition in definition 17 that $\mathbf{s}$ violates. Consider for instance the last condition. Assume that $f_1 \prec^*_{\mathbf{s}} f_2$, $f_1 \in \mathbf{s}$ but $f_2 \notin \mathbf{s}$. Since $\mathbf{s}$ is complete, $\neg f_2 \in \mathbf{s}$. Let $\mathcal{M}$ be a satisfying model for $\mathbf{s}$. Then $\langle \mathcal{M}, 0\rangle \models f_1$ and for all $a \in \mathbf{s}$, $\langle \mathcal{M}, 0\rangle \models a$, so that by Corollary 1, $\langle \mathcal{M}, 0\rangle \models f_2$. But $\neg f_2 \in \mathbf{s}$ and, thus, $\langle \mathcal{M}, 0\rangle \models \neg f_2$, a contradiction. The case of $f_1 \notin \mathbf{s}$ and $f_2 \in \mathbf{s}$ is similar.

Arguments for the remaining cases can be done in a similar manner using the semantics of the logic to exhibit a contradiction. □

It follows that each state $\mathcal{M}^f(t)$ of the extended model $\mathcal{M}^f$ is Hintikka. However, not every ω-sequence of Hintikka sets is the extension of a model, because the consecution of states in the sequence might be unsatisfiable.

Example 8. When $\mathcal{M}^f$ is constant throughout the interval $[t_1, t_2)$, let $\mathcal{M}^f[t_1, t_2)$ denote its value in that interval. In Example 4, it is clear that the sets $S_1 = \mathcal{M}^f[0, 1)$, $S_2 = \mathcal{M}^f[1, 4)$, $S_3 = \mathcal{M}^f[4, 7)$, $S_4 = \mathcal{M}^f[7, \infty)$ are Hintikka. In this case the conjunction of formulæ in

a Hintikka set is satisfiable. However, consider the set $S_5 = (S_1 \setminus \{\neg f_{11}\}) \cup \{f_{11}\}$. This set is Hintikka by our definition above, but is not satisfiable, because the conjunction of $\neg f_8$ and f_{11} cannot be satisfied in any model. Such "temporal conflicts" are detected by the consecution and acceptance conditions of $\mathcal{A}_e(f)$ and $\mathcal{A}_t(f)$, as will become clear in the sequel.

4.2 *Untimed construction*

Having obtained the candidate states for $\mathcal{A}_u(f)$ as Hintikka sets above, we must now connect them together appropriately. Compared to FIL Ramakrishna *et al* (1992), the only new feature now is the presence of formulæ of the form $\mathcal{I}\,\mathsf{len}(0, d]$ and $\mathcal{I}\neg\,\mathsf{len}(0, d]$. Reductions on such formulæ in a given state are essentially as before. However, consecution of two different states imposes further conditions on the timing assertions that these two states may contain, in addition to the reducibility of non-current interval formulæ from one state to the next.

DEFINITION 18
[UNTIMED CONSTRUCTION] $\mathcal{A}_u(f)$ is the BA with

- Input alphabet $2^{\mathbf{scl}(f)}$
- State set $\mathbf{H}(f)$
- Non-deterministic transition function ρ_u defined on $\mathbf{H}(f) \times 2^{\mathbf{scl}(f)}$ such that ρ_u allows $\mathbf{s} \xrightarrow{\mathbf{i}} \mathbf{t}$ iff
  1. $\mathbf{i} = \mathbf{s}$
  2. if $\mathcal{I}[\theta_1|\theta_2)f_1 \in \mathbf{s}$ is $\mathbf{s}$-irreducible and θ_1 is not $-$, then $\mathcal{I}[\theta_1|\theta_2)f_1 \in \mathbf{t}$
  3. if $\mathcal{I}\neg[\theta_1|\theta_2)f_1 \in \mathbf{s}$ is $\mathbf{s}$-irreducible and θ_1 is not $-$, then $\mathcal{I}\neg[\theta_1 \mid \theta_2)f_1 \in \mathbf{t}$ and $\mathcal{I}\mathsf{false} \notin \mathbf{t}$
  4. if $\mathcal{I}\,\mathsf{len}(0, d] \in \mathbf{s}$ is $\mathbf{s}$-irreducible, then if $\mathcal{I}\neg\,\mathsf{len}(0, d] \in \mathbf{t}$ then $\mathcal{I}$ has a collapsing reduction in $\mathbf{t}$
  5. if $\mathcal{I}\neg\,\mathsf{len}(0, d] \in \mathbf{s}$ is $\mathbf{s}$-irreducible, then $\mathcal{I}\mathsf{false} \notin \mathbf{t}$
- Accepting state set $\mathbf{H}(f)$
- Initial state set $\{\mathbf{s} \in \mathbf{H}(f) \mid f \in \mathbf{s}\}$

The first transition rule ensures that the automaton consumes only Hintikka sets. The remaining transition rules reflect the conditions stated in lemma 4. Observe that ρ_u is reflexive, allowing the automaton to (non-deterministically) stay in state $\mathbf{s}$ whenever input with $\mathbf{i} = \mathbf{s}$.

Example 9. Consider the Hintikka sets $S_1, \cdots, S_4$ of the last example, and $\mathcal{M}^f$ of Example 4. If we feed $\mathbf{untime}(\mathcal{M}^f)$ to $\mathcal{A}_u(f)$ as an untimed ω-string, then the resulting run is shown in figure 5. The vertices represent states of the automaton and the edge labels represent letters of the input string. Note that the automaton $\mathcal{A}_u(f)$ has many other states and

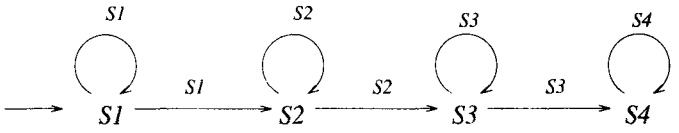


Figure 5. A run of $\mathcal{A}_u$ for example 9.

transitions, but for brevity only those in the locus of this run are shown. The reader can verify that the transition conditions given above are satisfied for each transition shown.

4.3 *Timing augmentation*

The timing augmentation systematically examines each state of the automaton built above, starting from an initial state, adding activity indicators to its states and clock conditions to its transitions, and splitting states when necessary. State-splitting occurs when different paths from an initial state to some state of $\mathcal{A}_u(f)$ require different sets of timers to be active. The resulting automaton is the required local TBA.

The augmentation is described in two steps. First, we replicate the states of $\mathcal{A}_u(f)$, pairing the replicas with subsets of **clocks**(f), to obtain the states of $\mathcal{A}_t(f)$. Intuitively, for $\langle \mathbf{s}, \mathbf{a}_s \rangle \in \mathbf{H}(f) \times 2^{\mathbf{clocks}(f)}$, the clock-activity set $\mathbf{a}_s$ represents the clocks that are active in this replica of the state $\mathbf{s}$ of $\mathcal{A}_u(f)$. We then define the transition function of $\mathcal{A}_t(f)$ to permit only "legal" transitions between the states produced by this replication process. While this style of exposition clarifies the underlying mechanics, it is generally more expedient to perform a breadth-first traversal of $\mathcal{A}_u(f)$, adding clock-activity sets to its states and splitting states as required. Although the worst-case behaviour of this latter augmentation procedure may be as bad as the naïve method of the description, in general, the latter procedure never creates many unreachable replicas.

For expositional reasons, we allow the transitions of $\mathcal{A}_t(f)$ to copy the value of a clock c_1 into a clock c_2 provided that c_1 has the form $c_{\mathcal{I}_1}^{\gamma,\mathcal{I},d}$, c_2 has the form $c_{\mathcal{I}_2}^{\gamma,\mathcal{I},d}$, and $\mathcal{I}_2 \sqsubset^* \mathcal{I}_1$. Thus, in addition to clock resetting actions, we allow restricted copying actions. This method of description clarifies the underlying reasoning better than a direct encoding into a conventional TBA. A slightly unnatural clock-naming scheme would allow us to rename the clocks in $\mathcal{A}_t(f)$ while eliminating the copying actions on its transitions. For instance, it is easy to see that instead of the copy action $c_2 \leftarrow c_1$, a "shadow clock" $c_{1,2}$ could be started simultaneously with c_1 and used in place of c_2 following the copy action. This simple-minded scheme, however, increases the number of clocks quadratically and increases the number of states by a factor exponential in the number of clocks. A slightly more sophisticated scheme, taking account of properties of interval reductions, allows us to encode copy actions without increasing the number of clocks, while keeping the number of states essentially the same. Note for this that the clocks form a natural partial order under the copying relation. We give details of this construction in the next subsection.

Note also that clock-activity sets are not mentioned in the definition of TBAs given earlier or in the original definition in Alur & Dill (1990). It is easy, however, to modify the definition of TBAs and the emptiness algorithm in Alur & Dill (1990). to handle clock-activity sets in a straightforward manner; see Dill (1989), for instance, where a similar concept is used.

Below we formalize the operations of clock activation and deactivation, which we use in our construction of the TBA $\mathcal{A}_t$.

DEFINITION 19
[CLOCK OPERATIONS] The transition $\langle \mathbf{s}, \mathbf{a}_s \rangle \xrightarrow{i,C,\phi} \langle \mathbf{t}, \mathbf{a}_t \rangle$ of $\mathcal{A}_t(f)$ *activates* the clock c iff one of the following holds

1. $c = c_{\mathcal{I}}^{\alpha,\mathcal{I},d}$, $\mathcal{I}\,\mathsf{len}(0,d] \in \mathbf{t}$, $\mathcal{I}$ is irreducible in $\mathbf{t}$, and $\mathcal{I}\,\mathsf{len}(0,d] \in \mathbf{s}$ only if $\mathcal{I}$ is reducible in $\mathbf{s}$
2. $c = c_{\mathcal{I}}^{\beta,\mathcal{I},d}$, $\mathcal{I}\neg\,\mathsf{len}(0,d] \in \mathbf{s}$, $\mathcal{I}\,\mathsf{len}(0,d] \in \mathbf{t}$, and $\mathcal{I}$ is irreducible in both $\mathbf{s}$ and $\mathbf{t}$
3. $c = c_{\mathcal{I}_2}^{\gamma,\mathcal{I},d} \notin \mathbf{a}_s$, $\mathcal{I}_2$ is irreducible in $\mathbf{t}$, $c_{\mathcal{I}_1}^{\gamma,\mathcal{I},d} \in \mathbf{a}_s$, $\mathcal{I}_1$ is irreducible in $\mathbf{s}$, $\mathcal{I}_2 \sqsubset_{\mathbf{t}}^* \mathcal{I}_1$, and this reduction is not collapsing

The transition *deactivates* clock c iff $c = c_{\mathcal{I}_1}^{\gamma,\mathcal{I},d}$ is in $\mathbf{a}_s$ and $\mathcal{I}_1$ is reducible in $\mathbf{t}$.

We now define the automaton $\mathcal{A}_t(f)$.

DEFINITION 20
[TIMING AUGMENTATION] Let $\mathcal{A}_u(f)$ be an untimed automaton such as obtained above. Then its *timing augmentation*, denoted $\mathcal{A}_t(f)$, is the TBA with:

- State set $\mathbf{H}(f) \times 2^{\mathbf{clocks}(f)}$
- Input Alphabet $2^{\mathbf{scl}(f)}$
- Clock Set $\mathbf{clocks}(f)$
- Non-deterministic transition function ρ_t defined on $(\mathbf{H}(f) \times 2^{\mathbf{clocks}(f)}) \times 2^{\mathbf{scl}(f)}$ such that ρ_t allows the transition $\langle \mathbf{s}, \mathbf{a}_s \rangle \xrightarrow{i,C,\phi} \langle \mathbf{t}, \mathbf{a}_t \rangle$ iff

  1. $\mathbf{s} \xrightarrow{i} \mathbf{t}$ is allowed by ρ_u
  2. $\mathbf{a}_t$ consists of all clocks that are activated by the transition and all clocks of $\mathbf{a}_s$ that are not deactivated by the transition
  3. if the transition activates c_2 by the third rule of definition 19, $c_1 = c_{\mathcal{I}_1}^{\gamma,\mathcal{I},d}$ and $c_2 = c_{\mathcal{I}_2}^{\gamma,\mathcal{I},d}$ are the clocks in this rule, and $\gamma = \beta$, then for all $c_1' = c_{\mathcal{I}_1'}^{\beta,\mathcal{I},d} \in \mathbf{a}_s$ such that $\mathcal{I}_1' \neq \mathcal{I}_1$, it is not the case that $\mathcal{I}_2 \sqsubset_{\mathbf{t}}^* \mathcal{I}_1'$
  4. C contains the reset action "$c \leftarrow 0$" iff the transition activates c by either the first or the second rule of definition 19
  5. C contains the copy action "$c_2 \leftarrow c_1$" iff the transition activates c_2 by the third rule of definition 19, $c_1 = c_{\mathcal{I}_1}^{\gamma,\mathcal{I},d}$ and $c_2 = c_{\mathcal{I}_2}^{\gamma,\mathcal{I},d}$ are the clocks in this rule, and if $\gamma = \alpha$, then for all $c_1' = c_{\mathcal{I}_1'}^{\alpha,\mathcal{I},d} \in \mathbf{a}_s$ such that $\mathcal{I}_2 \sqsubset_{\mathbf{t}}^* \mathcal{I}_1'$ we have $\mathcal{I}_1 \sqsubset_{\mathbf{t}}^* \mathcal{I}_1'$
  6. ϕ contains the clock condition $c \leq d$ iff $c = c_{\mathcal{I}_1}^{\alpha,\mathcal{I},d} \in \mathbf{a}_s$
  7. ϕ contains the clock condition $c = d$ iff $c = c_{\mathcal{I}_1}^{\beta,\mathcal{I},d} \in \mathbf{a}_s$ and $\mathcal{I}_1$ has a collapsing reduction in $\mathbf{t}$

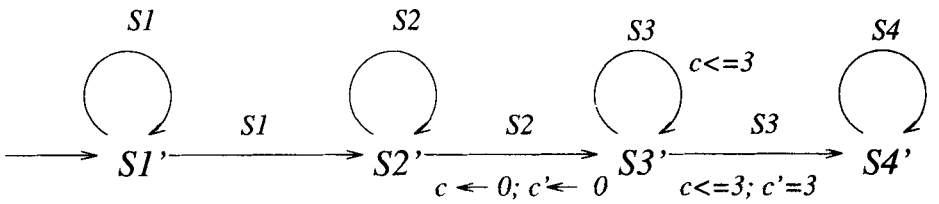


Figure 6. A run of $\mathcal{A}_t$ as in example 10.

- Initial state set $\{\langle \mathbf{s}, \mathbf{a}_s \rangle\}_{\mathbf{s},\mathbf{a}_s}$ such that $f \in \mathbf{s}$ and $\mathbf{a}_s = \{c_{\mathcal{I}}^{\alpha,\mathcal{I},d}\}_{\alpha,\mathcal{I},d}$ such that $\mathcal{I}\,\mathsf{len}(0,d] \in \mathbf{s}$ and $\mathcal{I}$ is $\mathbf{s}$-irreducible
- Accepting state set $\mathbf{H}(f) \times 2^{\mathbf{clocks}(f)}$

The intuition behind the augmentation procedure is as follows. Rule 4.3 ensures that any model of $\mathcal{A}_t(f)$, when untimed, is accepted by $\mathcal{A}_u(f)$. Rules 4.3 and 4.3 ensure that the appropriate clocks get started whenever there is a new upper- or lower-bound condition to verify, and that conditions are remembered until discharged. Rules 4.3 and 4.3 ensure that the upper- and lower-bound timers are compared with their prescribed limits when the ends of intervals are reached. Rule 4.3 frees up timers for reuse. The condition for α-clocks in the last part of that rule states that if there are two running instances of an interval that reduce to the same one, the older instance continues to be timed for the upper-bound. Rule 4.3 guarantees that such a condition will not arise for β-clocks.

Example 10. Recall example 9, where we illustrated an accepting run of $\mathcal{A}_u(f)$. Figure 6 shows the corresponding accepting run of $\mathcal{A}_t(f)$ on our now familiar $\mathcal{M}^f$. The states of $\mathcal{A}_t(f)$ shown in the figure are $S'_1 = \langle S_1, \emptyset\rangle$, $S'_2 = \langle S_2, \emptyset\rangle$, $S'_3 = \langle S_3, \{c, c'\}\rangle$, $S'_4 = \langle S_4, \emptyset\rangle$, where $c = c_{[\to q)}^{\alpha,[\to q),3}$ and $c' = c_{[\to q)}^{\beta,[\to q),3}$ are the clocks of Example 7. The edge labels also indicate associated clock conditions and/or clock actions.

Although the role of clock c is superfluous in the run shown above, in general it may be required.

4.3a *Eliminating copying of clocks* Notice that the only clause in the transition conditions of $\mathcal{A}_t$ that requires copying of one clock's value into another is clause 5. In the following we describe how such copying actions can be eliminated in order to obtain a conventional TBA. First of all, we note that we may rename the timers in $\mathbf{clocks}(f)$ so that each clock $c_{\mathcal{I}'}^{\gamma,\mathcal{I},d}$ is replaced by a unique clock $c_i^{\gamma,\mathcal{I},d}$, $i \in \{1, \cdots, m\}$ where $m = |\{\mathcal{I}' \mid \mathcal{I}' \sqsubset^* \mathcal{I}\}|$. Further, we associate with each state a *tagging function*, which associates with each clock active in that state an element of $\{\mathcal{I}' \mid \mathcal{I}' \sqsubset^* \mathcal{I}\}$. The essential idea is that, instead of copying one clock c value into another c' on a transition, we simply update the tag function on c in the next state. The tag function, thus, keeps track of the remaining suffix of the interval being timed by a clock. This will not work in case a transition also resets the active clock, following a copying action, since the old value would get "clobbered." In such a case, (*i.e.* if the transition also resets the source clock of a copy action), we simply pick an inactive clock, with the same superscript (perhaps with the lowest subscript among those available) and activate it. It is not difficult to see that in

every such case an inactive clock will always be available. This takes care of all cases of copying. When a clock's tag collapses, indicating the end of an interval, it is compared with its upper or lower bound as appropriate, and returned to the pool of inactive clocks.

We need only show that the number of clocks suffice, *i.e.* there is always a clock of the required kind available, when we want to pick an inactive one. But this is clear from the fact that for a clock with superscript $\mathcal{I}$ there cannot be more than $|\{\mathcal{I}' \mid \mathcal{I}' \sqsubset^* \mathcal{I}\}|$ copies ever required, since there will never be more than that many instances of the interval $\mathcal{I}$ active simultaneously. We have thus eliminated all copying actions while keeping the number of clocks the same as before. However, in comparison with our original construction which involves copying between clocks, there is an increase in the number of states, because of the association of active clock sets and tag functions with states. In fact, the total number of states in the resulting TBA is now bounded above by $|\mathbf{H}(f)| \cdot 2^{O(n^{2k} \cdot k \cdot \log n)}$.

4.4 *Eventuality automaton*

This is essentially the same as the construction in Ramakrishna *et al* (1992) to which we refer the reader for more details and intuition.

DEFINITION 21
[EVENTUALITY AUTOMATON] $\mathcal{A}_e(f)$ is the BA with

- Input Alphabet $2^{\mathbf{scl}(f)}$
- State Set $2^{\mathbf{E}(f)}$, where $\mathbf{E}(f)$ is the subset of $\mathbf{scl}(f)$ that contains all formulæ of the form $\neg[\theta \mid \rightarrow)\mathsf{false}$
- Deterministic transition function ρ_e defined on $2^{\mathbf{E}(f)} \times 2^{\mathbf{scl}(f)}$ such that $\mathbf{s} \xrightarrow{\mathbf{i}} \mathbf{t}$ satisfies
  1. $\mathbf{t} = \{f_1 \in \mathbf{E}(f) \cap \mathbf{i} \mid f_1 \text{ is } \mathbf{i}\text{-irreducible}\}$ when $\mathbf{s} = \emptyset$
  2. $\mathbf{t} = \{(f_1)_\mathbf{i} \in \mathbf{E}(f) \mid f_1 \in \mathbf{s}\}$ when $\mathbf{s} \neq \emptyset$
- Accepting state set $\{\emptyset\}$
- Initial state set $\{\emptyset\}$

Note, in particular, that $\mathcal{A}_e(f)$ handles only unbounded liveness conditions. Time-bounded liveness conditions are handled by the combination of $\mathcal{A}_e(f)$ and $\mathcal{A}_t(f)$; $\mathcal{A}_e(f)$ ensures that the required state is eventually reached (without regard to real-time) and $\mathcal{A}_t(f)$ ensures that the related timing constraints are met when the state is reached. A similar "communication" (via the "input" string) also occurs in the purely untimed case of FIL while dealing with eventualities that are bounded within intervals (Ramakrishna *et al* 1992): for checking an eventuality within a bounded context, the local automaton checks that the context does not end before the eventuality is found, a pure safety property; the eventuality automaton checks that the right end-point of the enclosing context *does* eventually occur, a pure liveness property.

Example 11. In our running example, we have $\mathbf{E}(f) = \{\neg f_8, \neg f_{11}, \neg f_{12}\}$. As in the previous two examples, we illustrate the accepting run of $\mathcal{A}_e(f)$ on $\mathcal{M}^f$ in figure 7. The states shown are $\emptyset$, $E_1 = \{\neg f_8, \neg f_{11}, \neg f_{12}\}$, and $E_2 = \{\neg f_{11}\}$.

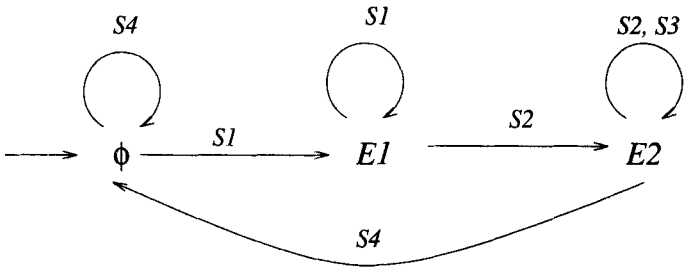


Figure 7. A run of $\mathcal{A}_e$ for example 11.

Note how $\mathcal{A}_e(f)$ is always one step "behind" $\mathcal{A}_u(f)$: $\mathcal{A}_u(f)$ is non-deterministic, while $\mathcal{A}_e(f)$ is fully deterministic, allowing precisely one transition on any input. Note also that both automata do not cycle, in the terminology of fundamental mode asynchronous automata; *i.e.* on any input stream consisting of precisely one input letter, there is at most one state change.

4.5 *Combining the automata*

The decision procedure is now straightforward. We construct $\mathcal{A}_u(f)$ and augment it using the timing construction to obtain $\mathcal{A}_t(f)$. We then take the product of $\mathcal{A}_t(f)$ with the eventuality automaton $\mathcal{A}_e(f)$, where $\mathcal{A}_e$ is run on the untiming of the input string. Finally, we check the emptiness of the resulting timed automaton $\mathcal{A}_m(f)$, using the emptiness algorithm of Alur & Dill (1990). We thus have our main theorem.

Theorem 4. [DECISION PROCEDURE] *Given an RTFIL formula f, it is decidable whether or not f is satisfiable.*

The main lemma required in the proof of theorem 4 is

Lemma 7. The language of $\mathcal{A}_m(f)$ is empty iff f is not satisfiable.

Proof. The proof follows from the Completeness and Soundness lemmas below. The proofs of the two lemmas follow the usual format of playing off the semantics of formulæ against the allowed runs of the automaton, and are sketched in the next section.

Lemma 8. [COMPLETENESS] *Let f be an RTFIL formula and $\mathcal{M}$ a satisfying model for it. Then $\mathcal{M}^f$ is accepted by $\mathcal{A}_m(f)$.*

Lemma 9. [SOUNDNESS] *Let f be an RTFIL formula, $\mathcal{M}'$ a timed string accepted by $\mathcal{A}_m(f)$, and $\mathcal{M}$ the restriction of $\mathcal{M}'$ to the primitive propositions. Then $\mathcal{M} \models f$.*

The construction for our decision procedure shows, once again, that RTFIL is invariant under finite infinitesimal timed stuttering. This was stated and proved directly in theorem 1, but is further clarified by noting that the local TBA $\mathcal{A}_t(f)$ has a reflexive transition relation with the self-loops containing edge conditions of the form $c \leq d$ only and no clock actions.

5. Proof of correctness

We devote the next two sections to proving the Soundness and Completeness lemmas.

5.1 *Completeness*

Throughout this section we assume that $\mathcal{M}^f$ is the extension of a satisfying model $\mathcal{M}$ for f, as stated in the Completeness Lemma. Moreover, we use the timed ω-string representation for $\mathcal{M}^f$. It is easy to see that admissibility of $\mathcal{M}^f$ implies that there is a timed ω-string representation for it. Note that any of the uncountably many representations suffices for our purposes. However, for convenience, we use a "canonical" representation, with $\mathcal{M}^f$ represented by the timed ω-string $\langle \sigma_i, t_i \rangle_{i \in \omega}$ defined inductively as follows (let $t_{-1} = 0$):

$$\sigma_i = \mathcal{M}^f(t_{i-1})$$
$$t_i = \inf(\{\{t > t_{i-1} \mid \mathcal{M}\}^f(t) \neq \mathcal{M}^f(t_{i-1})\} \cup \{\lfloor t_{i-1} \rfloor + 1\})$$

The proof of the Completeness Lemma follows from lemmas 12 and 14. Proofs or these lemmas make use of several intermediate lemmas.

Lemma 10. $\mathcal{A}_u(f)$ *accepts* **untime**$(\mathcal{M}^f)$.

Proof. Observe first that since all states of $\mathcal{A}_u(f)$ are accepting, we need only show that there is an infinite run of $\mathcal{A}_u$ that consumes $\langle \sigma_i \rangle_{i \in \omega} =$ **untime**$(\mathcal{M}^f)$. Since each state σ_i is Hintikka, it is a state of $\mathcal{A}_u$. By clause 1 in the definition of ρ_u (see definition 18), α_U CAN CONSUME the input symbol σ_i iff it is in the state σ_i. Thus, if $\mathcal{A}_u$ has an infinite run consuming $\mathcal{M}^f$, that run is unique. That it has an infinite run is shown by induction on the length of the run.

BASE CASE. Since $\langle \mathcal{M}, 0 \rangle \models f$, we have $f \in \mathcal{M}^f(0)$ and, therefore, $f \in \sigma_0$. Thus σ_0 is an initial state of $\mathcal{A}_u$.

INDUCTIVE STEP. We need to show that $\sigma_{i+1} \in \rho_u(\sigma_i, \sigma_i)$. Assume not. Then $\sigma_i \neq \sigma_{i+1}$, since ρ_u allows self-loops. This means that t_{i+1} is the least t' satisfying $t' > t_i$ and $\mathcal{M}^f(t_i) \neq \mathcal{M}^f(t')$. But then the assumption that the transition $\sigma_i \longrightarrow \sigma_{i+1}$ violates one of the last four transition requirements of ρ_u. But using the definition of extension, this contradicts lemma 4. □

Lemma 11. If a timed ω-string $\langle \sigma_i, t_i \rangle_{i \in \omega}$ is accepted by $\mathcal{A}_t$, then the acceptance run is unique.

Proof. From the definition of the transition function of $\mathcal{A}_t$, the BA $\mathcal{A}_u$ must accept the untimed string $\langle \sigma_i \rangle_i$. From the proof of lemma 10, the run of $\mathcal{A}_u$ on $\langle \sigma_i \rangle_i$ must be unique. Recall that the state of $\mathcal{A}_t$ consists of two components: a Hintikka set and a set of active clocks. From the above, it is clear that the "Hintikka component" of the run of $\mathcal{A}_t$ on $\langle \sigma_i, t_i \rangle_i$ is unique. What remains to be shown is that the "clock component" is also unique.

To see that this is indeed the case, we note from definition 19, that the clocks that are active in the state following a transition are uniquely determined by the Hintikka component of the states adjoining the transition, and the clocks that are active in the state prior to the

transition. Moreover, the clocks that are active in the initial state are uniquely determined by the Hintikka component of the state. Since the Hintikka component is unique and determined, so also is the clock component, and the result follows. □

Lemma 12. $\mathcal{A}_t(f)$ *accepts* $\mathcal{M}^f$.

Proof. Assume for a contradiction, that it does not. From lemma 10 we know that $\mathcal{A}_u$ accepts **untime**$(\mathcal{M}^f)$. If $\mathcal{A}_t$ rejects, it must be because some clock condition, introduced as a result of the timing augmentation is not satisfied along the run. Before we proceed with the proof, we introduce some terminology.

DEFINITION 22
For a run of $\mathcal{A}_t$ over an extended model $\mathcal{M}^f$, a *timer thread* $c^{\gamma,\mathcal{I},d}[t_a, t_d)$ is a finite chain of clocks $\langle c^{\gamma,\mathcal{I},d}_{\mathcal{I}_1}, \cdots, c^{\gamma,\mathcal{I},d}_{\mathcal{I}_n}\rangle$ such that

1. $\mathcal{I} = \mathcal{I}_1$
2. for all $i \in \{1, \cdots, n-1\}$, $\mathcal{I}_{i+1} \sqsubset^* \mathcal{I}_i$
3. a transition at time t_b activates $c^{\gamma,\mathcal{I},d}_{\mathcal{I}}$
4. there is a strictly monotonically increasing sequence of time values $\langle t_1, \cdots, t_{n-1}\rangle$ with $t_b < t_1$ and $t_{n-1} < t_e$, such that a transition at time t_i copies $c^{\gamma,\mathcal{I},d}_{\mathcal{I}_i}$ into $c^{\gamma,\mathcal{I},d}_{\mathcal{I}_{i+1}}$, and no transition at any time strictly between t_i and t_{i+1} deactivates $c^{\gamma,\mathcal{I},d}_{\mathcal{I}_i}$
5. a transition at time t_e deactivates $c^{\mathcal{I},\gamma,d}_{\mathcal{I}_n}$

A timer thread is *incomplete* if the deactivating transition at t_e also copies the last clock to another clock.

A timer thread is *useless* if the transition at t_e deactivates the last clock without copying it and the remaining interval $\mathcal{I}_n$ does not collapse in the state following the transition.

A timer thread is a *complete useful* thread if $\mathcal{I}_n$ has a collapsing reduction in the state following the transition at t_e.

The intuition behind this terminology is as follows. A complete useful timer thread represents a successful verification of a timing constraint. A useless thread represents a verification that was started but was later abandoned, because the corresponding timing constraint was subsumed by another timing constraint whose verification was in progress. An incomplete thread represents a verification that is in progress and that may be either completed into a successful verification or abandoned in the future. The reader should observe that, with the transition function for $\mathcal{A}_t$ defined in definition 20, if $\gamma = \beta$, then every incomplete thread eventually completes usefully. On the other hand, incomplete α-threads may become useless. It is easy to see that each active clock in any run belongs to precisely one incomplete thread and, moreover, each incomplete thread correponds to precisely one active clock in any state. Also, threads may complete usefully or become useless as a run progresses, but they never fork or merge. Therefore, starting from an active clock and tracing back along the thread to which it belongs, one can locate its "ultimate ancestor", or the initial clock created for the verification of a timing constraint. The value

of the active clock indicates the time that the thread has been active since its ultimate ancestor was activated.

Proof of lemma 12 Cont'd. We need to show that $\mathcal{A}_t(f)$ consumes the timed ω-string $\langle \sigma_i, t_i \rangle_{i \in \omega}$ representing $\mathcal{M}^f$. We show that for all $j \in \omega$, the TBA $\mathcal{A}_t(f)$ consumes the prefix $\langle \sigma_i, t_i \rangle_{i=0,\ldots,j-1}$ in a run τ^j that ends in the state $\langle \sigma_j, \mathbf{a}_j \rangle$, where $\mathbf{a}_j$ consists of the clocks that terminate the incomplete timer threads induced by τ^j.

The base case of $j = 0$ follows immediately from the definition of initial states of $\mathcal{A}_t$.

For the induction assume that the above holds for j.

We first show that ρ_t allows the transition $\langle \sigma_j, \mathbf{a}_j \rangle \xrightarrow{\sigma_j, C, \phi} \langle \sigma_{j+1}, \mathbf{a}' \rangle$ where $\mathbf{a}'$ consists of all clocks activated by the transition and all clocks of $\mathbf{a}_j$ not deactivated by the transition, C satisfies clauses 4.3 and 4.3 of definition 20 for the timing augmentation, and ϕ satisfies clauses 4.3 and 4.3. Because of lemmas 10 and 11 all we need show is that $\mathbf{a}_j$ does not contain a pair of active β-clocks representing two distinct incomplete timer threads $c^{\beta,\mathcal{I},d}[t_{i(1)}, t_j)$ and $c^{\beta,\mathcal{I},d}[t_{i(2)}, t_j)$, with $t_{i(1)} \neq t_{i(2)}$, which merge at t_j. The starting of a β-clock at $t_{i(1)}$ implies from definition 19, clause 2, that $\mathcal{I} \neg \mathsf{len}(0, d] \in \sigma_{i(1)}$ and $\mathcal{I}\,\mathsf{len}(0, d] \in \sigma_{i(1)+1}$. The semantics then imply that $t_j + x = t_{i(1)} + d$, where x represents the common suffix that will be timed by the thread following the merge at t_j. Arguing similarly for the case of the second clock, we have $t_j + x = t_{i(2)} + d$, thus together contradicting the assumption that $t_{i(1)} \neq t_{i(2)}$.

Next we show that the run τ^{j+1} of $\mathcal{A}_t(f)$ obtained by extending τ^j by the above transition consumes $\langle \sigma_i, t_i \rangle_{i=0,\ldots,j}$. For this we need only show that the timing conditions required by clauses 6 and 7 of definition 20 are not violated. For the case of clause 6, consider an active α-clock in $\mathbf{a}_j$ representing the timer-thread $c^{\alpha,\mathcal{I},d}[t_i, t_j)$. But the starting of an α-clock at t_i implies from definition 19, clause 1, that $\mathcal{I}\,\mathsf{len}(0, d] \in \sigma_{i+1}$. The semantics then tell us that $t_j \leq t_i + d$. The value of the clock, $t_j - t_i$ cannot then exceed d. The case of clause 7 is similar.

Finally, we note that $\mathbf{a}'$ consists of the clocks terminating the incomplete timer threads induced by the run τ^{j+1}. But this follows immediately from definition 22 for timer threads and the definition 19 for the clock operations. □

For the proof of lemma 14, the following simpler lemma is useful.

Lemma 13. Let τ^u and τ^e represent, respectively, the runs of $\mathcal{A}_u(f)$ and $\mathcal{A}_e(f)$ on some ω-string $\sigma \in (2^{\mathbf{scl}(f)})^\omega$. Then for all $i \in \omega$, $\tau^e_i \subset \tau^u_i$.

Proof. We first make the following observation about the statement of the lemma. As we have seen, on any arbitrary ω-string on which $\mathcal{A}_u$ has a run, it has a unique run. Moreover, as we show below in the proof of the next lemma, $\mathcal{A}_e$ has a unique run on an arbitrary input. Thus the runs τ^u and τ^e are unique.

The proof of the lemma follows by induction on the index i of the run, as follows:
BASE CASE. Every Hintikka set has *some* element, thus $\tau^u_0 \neq \emptyset$; but $\tau^e_0 = \emptyset$.
INDUCTIVE STEP. Assume that $\tau^e_n \subset \tau^u_n$. We consider two cases.
CASE 1 $[\tau^e_n = \emptyset]$. First note that a Hintikka set contains its basis, since it is closed under reductions, and every Hintikka set contains $\mathsf{true} \notin \mathbf{E}(f)$. From the transition conditions

of $\mathcal{A}_e$, therefore, τ^e_{n+1} is a subset of the nth input which is τ^u_n. Further, by the transition conditions of $\mathcal{A}_u$, τ^u_{n+1} contains all irreducible formulæ in τ^u_n, so that $\tau^e_{n+1} \subset \tau^u_{n+1}$.
CASE 2 $[\tau^e_n \neq \emptyset]$. Since τ^e_{n+1} contains the irreducible subset of τ^e_n and τ^u_{n+1} contains all irreducible formulæ of τ^u_n, using the induction hypothesis, we have $\tau^e_{n+1} \subset \tau^u_{n+1}$. □

Lemma 14. $\mathcal{A}_e(f)$ *accepts* **untime**$(\mathcal{M}^f)$.

Proof. Observe, first, that $\mathcal{A}_e(f)$ is deterministic, and in every state has a (unique) transition for every input letter from $2^{\mathbf{scl}(f)}$. Thus, there is a unique infinite run of $\mathcal{A}_e$ on $\sigma = \langle\sigma_i\rangle_{i\in\omega} = \mathbf{untime}(\mathcal{M}^f)$ that consumes σ; call this run τ^e. Since $\tau^e_0 = \emptyset$, if τ^e is not accepting, then there is a largest i such that $\tau^e_i = \emptyset$.

From the second transition rule for $\mathcal{A}_e$ in definition 21 and the definition of reduction, we can conclude for any two consecutive states $\mathbf{s} \neq \emptyset$ and $\mathbf{t} \neq \mathbf{s}$ of $\mathcal{A}_e$ such that $\rho_e(\mathbf{s}, \mathbf{i}) = \mathbf{t}$, that size($\mathbf{t}$) < size($\mathbf{s}$).[9] By the well-foundedness of size, there is some $j > i$, such that for all $k \geq j$, $\sigma_k = \sigma_j \neq \emptyset$. Thus there is some formula $\neg[\theta \mid \rightarrow)\mathsf{false} \in \sigma_k$ for all $k \geq j$. Without loss of generality assume that θ is $\rightarrow a, \theta'$.

By the definitions of ρ_e and reducibility, then, $a \notin \sigma_k$ for all $k \geq j$. Completeness of σ implies that $\neg a \in \sigma_k$ for all $k \geq j$, whence the definition of an extension and semantics yield $\langle\mathcal{M}, t_j\rangle \models [\rightarrow a, \theta' \mid \rightarrow)\mathsf{false}$.

But from the proof of lemma 10 we have $\tau^u_k = \mathcal{M}^f(t_k)$, where $\tau^u = \langle\sigma^u_i\rangle_{i\in\omega}$ denotes the infinite run of $\mathcal{A}_u$ on σ. By lemma 13, $\tau^e_k \subset \tau^u_k$, so $\neg[\rightarrow a, \theta \mid \rightarrow)\mathsf{false} \in \mathcal{M}^f(t_k)$. By the definition of extension and semantics, then $\langle\mathcal{M}, t_k\rangle \not\models [\rightarrow a, \theta' \mid \rightarrow)\mathsf{false}$, a contradiction. □

5.2 *Soundness*

The proof consists of showing that given a (timed) string in the language of $\mathcal{A}_m$, one can construct a satisfying model for f. Let $\langle\sigma_i, t_i\rangle_{i\in\omega}$ be a string in the language of $\mathcal{A}_m$, and let $\mathcal{M}'$ be defined by

$$\mathcal{M}'(t) = \{\, f_1 \in \mathbf{scl}(f) \mid\ f_1 \in \sigma_i, t \in [t_{i-1}, t_i) \,\}$$

where we have assumed $t_{-1} = 0$. Moreover, let $\mathcal{M}$ be defined by

$$\mathcal{M}(t) = \{\, p \in \mathcal{P} \mid\ p \in \mathcal{M}'(t) \,\}$$

To prove the lemma, we want to show that $\langle\mathcal{M}, 0\rangle \models f$.

Lemma 15. For any $t \in R$ *and* $f_1 \in \mathbf{scl}(f)$, $f_1 \in \mathcal{M}'(t)$ *iff* $\langle\mathcal{M}, t\rangle \models f_1$.

Proof. For a given t, we induct on the inclusion order induced by **scl** on the formulae in $\mathbf{scl}(f)$. Let $t \in [t_{i-1}, t_i)$ as defined above.

BASE CASE. Consider a primitive proposition $p \in \mathcal{P}$. For the forwards direction, let $p \in \mathcal{M}'(t)$, so $p \in \mathcal{M}(t)$, whence the semantics give us $\langle\mathcal{M}, t\rangle \models p$. For the backwards direction, let $\langle\mathcal{M}, t\rangle \models p$, so $p \in \mathcal{M}(t)$, so $p \in \mathcal{M}'(t)$, by construction.

[9] For a set of formulæ F, let size$(F) = \Sigma_{f\in F}$size(f).

INDUCTIVE STEP. Assume that the lemma holds for all $f' \in \mathbf{scl}(f_1)$ where $\mathbf{scl}(f') \subset \mathbf{scl}(f_1)$. We can show then by a case analysis of the structure of the f_1, that the lemma then holds for f_1 also. The details are routine and extremely tedious and are therefore skipped. However, we illustrate below a sample case to illustrate the argument.

Consider the case of $f_1 = [\theta_1 \mid \theta_2) f_2$. Assume for the forwards direction that $f_1 \in \mathcal{M}'(t)$. We have two subcases depending on whether or not f_1 is $\mathcal{M}'(t)$-reducible.

SUBCASE 1 [REDUCIBLE]. By our construction, f_1 is σ_i-reducible. Let $f_1' \in \sigma_i$ and $F \subset \sigma_i$ be such that $f_1' \prec_F^* f_1$. Then since σ_i is Hintikka, $f_1' \in \sigma_i$, so by construction $f_1' \in \mathcal{M}'(t)$ as well as $F \subset \mathcal{M}'(t)$. Now as the subformula closure of all the reductors and reducts of a formula f are strictly contained in $\mathbf{scl}(f)$, the induction hypothesis and Corollary 1 give us the result.

SUBCASE 2 [IRREDUCIBLE]. By construction, $f_1 \in \sigma_i$ is σ_i-irreducible. By our earlier observations, since $\sigma = \langle \sigma_i \rangle_{i \in \omega}$ is accepted by $\mathcal{A}_u(f)$, we may consider σ to be the run of $\mathcal{A}_u$ on σ. By the transition conditions of $\mathcal{A}_u$, $f_1 \in \sigma_j$ for all $j, i \leq j \leq k$ where k is the least index greater than i such that f_1 is σ_k-reducible. To see that such a finite k must exist, use the acceptance criteria for $\mathcal{A}_e$ along with the fact that both $\neg[\theta_1 \mid \rightarrow) f_2 \in \sigma_i$ and $\neg[\theta_2 \mid \rightarrow) f_2 \in \sigma_i$, since f_1 is σ_i-irreducible. At index k, we use an argument identical to Subcase 1 above to establish that $\langle \mathcal{M}, t_{k-1} \rangle \models f_1$. Using the fact that f_1 is irreducible in the intervening period, allows us to use the induction hypothesis on $\mathbf{red}(f_1)$ and $(k-1-i)$ applications of lemma 4 to conclude that $\langle \mathcal{M}, t \rangle \models f_1$.

The backwards direction is similar. For some more details, we refer the reader to Ramakrishna (1993). □

The soundness lemma follows since f is in $\mathcal{M}^f(0)$.

6. Complexity

Let f be an RTFIL formula of size n and depth k, and let T be the size of the encoding of largest finite timing constant appearing in f. By lemma 5, $|\mathbf{scl}(f)| = O(n^k)$. Clearly, $\mathcal{A}_u(f)$ and $\mathcal{A}_e(f)$ can have at most $2^{O(n^k)}$ states each. The timing augmentation can introduce up to $O(n^{2k})$ clocks. Following the elimination of copying actions, thus, $\mathcal{A}_t(f)$ (and consequently also $\mathcal{A}_m(f)$) can have at most $2^{O(n^{2k} \cdot k \cdot \log n)}$ states and $O(n^{2k})$ clocks. The final emptiness check has a complexity of $O(C! \cdot (S+E) 2^{T \cdot \log T})$, where C is the size of the clock-set, S and E are the number of states and edges in the TBA, and T is the size of the binary encoding of the constants appearing on the edge conditions of the TBA (Alur & Dill 1990). The overall complexity of the decision procedure is thus $2^{O(n^{2k} \cdot 2k \cdot \log n + T \cdot \log T)}$.

The main source of the blow-up is due to the large number of clocks. Note, however, that usually the number of clocks will be much less than that indicated by the large upper-bound because timing conditions in specifications will generally involve relations between a few simple predicates rather than long sequences of events. As a result the overall complexity will be closer to $2^{O(n^k + C \cdot k \log n + T \cdot \log T)}$, where C is the number of clocks introduced in the timing augmentation. Comparing this with the $2^{O(n^k)}$ upper-bound for FIL, the price for real-time is seen to be an additional factor exponential in the number of timers and the

constants appearing in the specification. However, the decision procedure is still doubly exponential (deterministic time), essentially the same as for the timeless logic FIL. In fact, by combining the *PSPACE*-containment of the emptiness problem for TBAs (Alur & Dill 1990) with the *EXPSPACE*-encoding of the automaton constructed in the last section, it can be shown that RTFIL is in *EXPSPACE*. [10]

The procedure given can be adapted in a straightforward manner to obtain a model-checking algorithm for RTFIL having the same complexity with respect to input formula and linear in the size of the input model (for instance, in the form of a fair-transition system).

Analogous to the result in Ramakrishna *et al* (1992) we can show that if we bound the largest constant appearing in a formula and the largest depth of nesting of interval modalities, then this bounded version of satisfiability for RTFIL is *PSPACE*-complete in the size of the formula. This result is more indicative of the type of scaling behaviour one would expect for the logic.

7. Related work

The idea of bounding the duration of intervals was first articulated by Melliar-Smith in an early paper on real-time interval logic (Melliar–Smith 1987). Subsequent proposals for real-time interval logics appear in Narayana & Aaby (1988) and Razouk & Gorlick (1989). However, none of these proposals provided decision procedures for the logics presented. In fact the logic of Razouk & Gorlick (1989) is so powerful that it is highly undecidable. The logics of Narayana & Aaby (1988) and of Melliar–Smith (1987) allow the expression of the forbidden "punctuality" construct of Alur *et al* (1991), so that they can be shown to be undecidable if interpreted over a dense time domain.

Consider an extension of RTFIL by allowing searches of the form $\rightarrow + d$ for $d \in Q$. The semantics of such a search is that it locates a point t' in the future of the point t where the search began such that $t' = t + d$. It thus allows relatively natural expression of many real-time constructs. However, it is not difficult to show that this simple extension (with no other restrictions) makes the resulting logic undecidable (Ramakrishna 1993). The proofs of undecidability of all these logics follow essentially along the lines of Alur *et al* (1991), by reduction from the halting problem for two-counter Minsky machines.

Another possible extension is to consider backwards searches, for instance $\leftarrow f$. We have shown that even in the absence of real time, this construct leads to non-elementariness (decidability of the logic with backwards searches, but without real-time, follows by translation to S1S). The proof of non-elementariness (Ramakrishna 1993) is by reduction from the non-emptiness of complement problem for extended star-free regular expressions.

Decidable dense real-time logics are relatively rare because a dense real-time logic must tread a fine line between expressiveness and undecidability. The logics RTFIL and MITL (Alur *et al* 1991) adopt different compromises, and neither, we believe, is as expressive as the other. MITL appears to have no direct way of expressing RTFIL formulæ that

[10] However, the best lower-bound we have is the *PSPACE* lower-bound for FIL. We refer the reader to Ramakrishna (1993) for related comments.

constrain the length of an interval defined between the endpoints of a sequence of (more than two) searches. Correspondingly, RTFIL cannot express the MITL construct $p\mathcal{U}_I q$, which requires q to occur within the time bounds denoted by I (while not constraining its occurrence outside that interval), and p to hold until that occurrence.[11]

In effect, RTFIL defines events in relation to other events, and then imposes real-time constraints on their relative occurrence. In contrast, MITL first defines real-time intervals and then requires events within those intervals, possibly in relation to other events. Thus, it appears that MITL may be better suited for synchronized real-time systems, where the synchronization is by real-time, whereas RTFIL may be more appropriate for asynchronous real-time systems. A natural question, then, is whether there is a reasonable combination of the two logics that retains decidability. We conjecture that the answer is in the affirmative, and a decision procedure for the combination would follow from a suitable "composition" of the procedures for the two logics. This is the case, for instance, for FIL and PTL($\mathcal{S}, \mathcal{U}$), where such a "combined" decision procedure follows from purely automata-theoretic methods (Ramakrishna 1993).

The Duration Calculus (Chaochen *et al* 1991) differs from RTFIL in that it treats intervals as primitive. It is well-suited to describing and reasoning about cumulative behaviour, a feature especially useful for hybrid systems. The operator $\int$ in that logic, for instance, allows one to bound the duration of a (fragment of a) computation for which a predicate holds. This ability to integrate over non-convex intervals, combined with the "non-local" character of the logic makes it very expressive. However, as shown in Chaochen *et al* (1993), over dense time the simplest real-time fragment of the calculus is undecidable, and even without real-time the simplest fragment is non-elementary. We are currently investigating an extension of RTFIL with ageing operators, inspired by the $\int$ operator of the Duration Calculus.

8. Conclusion

We have presented a real-time interval logic RTFIL which conservatively extends the timeless logic FIL. The logic extends FIL in a natural way to allow real-time specification, without sacrificing decidability. We have presented a formal semantics for the logic and have given a decision procedure for it. That RTFIL involves an additional exponential factor proportional to the number of clocks and the constants appearing in the specification should come as no surprise for those familiar with other dense-time logics.

A prototype RTFIL theorem-prover based on a tableau-theoretic analogue of the decision procedure given in this paper has been implemented and used to verify some simple real-time systems. However, further work is required before the system can become the basis of a practical verification system for real-life examples. Apart from the use of efficient data-structures, such as binary decision diagrams for state-encoding, efficient heuristics,

[11] In each case; the introduction of auxiliary predicates mitigates the problem. Note also that the logic TPTL (Alur & Henzinger 1989), with "freeze" quantification, can express the RTFIL property given earlier. Unfortunately, TPTL is undecidable when interpreted over a dense time domain. We must add, however, that MITL extended with past operators *can* express this property, although apparently less succinctly (see Ramakrishna 1993). This logic has a decidable validity problem.

such as those used in Alur *et al* (1992) will need to be used in order to reduce the space requirements for the verification. Since our procedure is automata-theoretic, it can directly benefit from any advances in verification technology based on ω-automata.

We are also devising a proof calculus for the logic in the style of the natural deduction calculi that are now gaining popularity in many applications. The success or failure of an "expensive" logic such as RTFIL would depend crucially upon whether one is able to obtain a clean proof system. We consider our decision procedure an important first step in this direction. For instance, our reduction and transition rules can be seen as a form of "rewrite rules" for a tableau proof system. The incorporation of timers in a formal manner into such tableaux, however, presents non-trivial difficulties. One approach might be to use time variables with such operations as resetting, assignment, comparison and difference, to simulate the role of timers. However, such an approach is probably far too low-level to be useful. On the other hand, some appropriate mixture of automated inference within such a proof system, along with user assistance at crucial points, may be feasible.

Finally, from a more theoretical standpoint, there are interesting expressiveness questions regarding RTFIL and some other decidable real-time logics. The apparent duality between our approach and that of other real-time temporal logics, as outlined in the previous section, clearly merits further study. Another interesting direction involves identifying a natural decidable fragment of *parametric* RTFIL, in the sense of Alur *et al* (1993).

We thank Rajeev Alur for useful discussions, and for helpful comments on the conference version of the paper.

The research was partially supported by NSF/DARPA grant CCR-9014382.

References

Alur R, Dill D 1990 *Automata for Modelling Real-Time Systems*, Proc 17th ICALP, LNCS 443, pp 322–335

Alur R, Henzinger T 1989 A really temporal logic. *Proc. 30th FOCS*, pp 164–169

Alur R, Henzinger T 1992 Back to the future: Towards a theory of timed regular languages. *Proc. 33rd FOCS*, pp 177–186

Alur R, Feder T, Henzinger T 1991 The benefits of relaxing punctuality. *Proc. 10th PODC*, pp 139–152

Alur R, Itai A, Kurshan R P, Yannakakis M 1992 Timing verification by successive approximation. *Proc. 4th CAV*, LNCS 663, pp 137–150

Alur R, Henzinger T, Vardi M Y 1993 Parametric real-time reasoning. *Proc 25th STOC*, pp 592–601

Barringer H, Kuiper R, Pnueli A 1986 A really abstract concurrent model and its temporal logic. *Proc. 18th POPL*, pp 173–183

Chaochen Z, Hoare C A R, Ravn A P 1991 A calculus of durations. *Inf. Process. Lett.* 40: 269–276

Chaochen Z, Hansen M R, Sestoft 1993 Decidability and undecidability results for the duration calculus. *Proc 10th STACS*, LNCS 665, pp 58–68

Dill D 1989 Timing assumptions and verification of finite-state concurrent systems. *Proc. Int. Workshop Automatic Verification Methods for Finite-State Systems*, LNCS 407, pp 196–212

Dillon L K, Kutty G, Moser L E, Melliar-Smith P M, Ramakrishna Y S 1992 Graphical specifications for concurrent software systems. *Proc. 14th ICSE*, pp 214–224

Dillon L K, Kutty G, Moser L E, Melliar-Smith P M, Ramakrishna Y S 1994 A graphical interval logic for specifying concurrent systems. *ACM Trans. Software Eng. Methodol.* 3: 131–165

Emerson E A, Mok A, Sistla A P, Srinivasan J 1990 Quantitative temporal reasoning.*Proc. 1st CAV*, LNCS 531, pp 136–145

Emerson E A 0000 Temporal and modal logic. In *Handbook of theoretical computer science. Volume B. Formal models and semantics* (ed.) J van Leeuwen (Cambridge: MIT Press) pp 789–840

Fischer M J, Ladner R E 1977 Propositional modal logic of programs. *Proc. 9th ACM STOC* pp 286–294

Halpern J, Shoham Y 1991 A propositional modal logic of time intervals. *J. Assoc. Comput. Mach.* 38: 935–962

Halpern J, Manna Z, Moszkowski B 1983 A hardware semantics based on temporal intervals. *Proc 10th ICALP* LNCS, pp 278–291

Jahanian F, Mok A 1986 Safety analysis of timing properties in real-time systems. *IEEE Trans. Software Eng.* 12: 890–904

Kutty G, Ramakrishna Y S, Moser L E, Dillon L K, Melliar-Smith P M 1993 A graphical interval logic toolset for verifying concurrent systems. *Proc. 4th CAV* LNCS 697, pp 138–153

Lamport L 1991 The temporal logic of actions. DEC SRC Tech. Report 79

Lewis H 1990 A logic of concrete time intervals. *Proc. 5th LICS*, pp 380–389

Maler O, Manna Z, Pnueli A 1991 From timed to hybrid systems. *Proc. REX Workshop "Real-time: Theory in practice."* LNCS 600, pp 447–484

Melliar–Smith P M 1987 Extending interval logic to real-time systems.*Proc. Conf. Temporal Logic in Specification*, Altrincham, England, LNCS 398, pp 224–242

Melliar-Smith P M 1988 A graphical representation of interval logic.*Proc. Int. Conf. on Concurrency*, Hamburg, FRG, LNCS 335 (Berlin: Springer-Verlag) pp 106–120

Narayana K T, Aaby A A 1988 Specification of real-time systems in real-time temporal interval logic.*Proc 9th IEEE RTSS*, pp 86–95

Newman M H A 1942 On theories with a combinatorial definition of "equivalence." *Ann. Math.* 43: 223–243

Ramakrishna Y S 1993 *Interval logics for temporal specification and verification.* Ph D dissertation, Dept.of Electrical and Computer Engineering, University of California, Santa Barbara

Ramakrishna Y S, Dillon L K, Moser L E, Melliar-Smith P M, Kutty G 1992 An automata-theoretic decision procedure for future interval logic. *Proc. 12th FST&TCS*, LNCS 652, pp 51–67

Ramakrishna Y S, Melliar-Smith P M, Moser L E, Dillon L K, Kutty G 1993 Really visual temporal Reasoning.*Proc. 14th Real-Time Systems Symposium*, Rayleigh-Durham, pp 262–273

Razouk R R, Gorlick M M 1989 A real-time interval logic for reasoning about executions of real-time programs. *Proc. ACM SIGSOFT'89, 3rd TAV, SIGSOFT SE Notes* 114: 10–19

Schwartz R L, Melliar-Smith P M, Vogt F 1983 An interval logic for higher-level temporal reasoning. *Proc. 2nd PODC*, pp 173–186

Smullyan R M 1968 *First-order logic* (Berlin: Springer-Verlag)

Wolper P 1987 On the relation of programs and computations to models of temporal logic. *Proc. Conf. Temporal Logic in Specification*, LNCS 398, pp 75–123