An On-the-fly Tableau Construction for a Real-Time Temporal Logic

Marc Geilen and Dennis Dams

Faculty of Electrical Engineering, Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: {m.c.w.geilen,d.dams}@tue.nl

Abstract. Temporal logic is a useful tool for specifying correctness properties of reactive programs. In particular, real-time temporal logics have been developed for expressing quantitative timing aspects of systems. A tableau construction is an algorithm that translates a temporal logic formula into a finite-state automaton that accepts precisely all the models of the formula. It is a key ingredient to checking satisfiability of a formula as well as to the automata-theoretic approach to model checking. An improvement to the efficiency of tableau constructions has been the development of on-the-fly versions. In the real-time domain, tableau constructions have been developed for various logics and their complexities have been studied. However, there has been considerably less work aimed at improving and implementing them. In this paper, we present an on-the-fly tableau construction for a linear temporal logic with dense time, a fragment of Metric Interval Temporal Logic that is decidable in PSPACE. We have implemented a prototype of the algorithm and give experimental results. Being on-the-fly, our algorithm is expected to use less memory and to give smaller tableaux in many cases in practice than existing constructions.

1 Introduction

Temporal logic has enjoyed an increased interest ever since it was suggested, in [12], as a formalism for specifying correctness properties of reactive programs. Using this approach, several questions about a program's specification can be phrased in formal terms. When the (abstraction of the) program is finite-state, model checking procedures can be used to verify correctness automatically. Today, a rich variety of temporal logics exist in which properties of various kinds can be expressed. In particular, *real-time* temporal logics have been developed for expressing timing aspects of systems.

A *tableau construction* is an algorithm that translates a temporal logic formula into a finite-state automaton (possibly on infinite words) that accepts precisely all the models of the formula. Tableau constructions play a role in several places in the area of specification and verification. For example, it is the key to deciding satisfiability of formulas. Furthermore, the *automata-theoretic* approach to model checking ([11, 14]) relies on tableau algorithms to turn a temporal formula into an observer of a program's behaviours. Driven by practical needs, tableau constructions are being continuously improved and reimplemented (e.g. [8, 5, 6]). One such improvement has been the development of *on-the-fly* versions of tableau constructions. In general, this means that the tableau automaton is constructed in a lazy way, generating states and transitions as they are needed. At the heart of such on-the-fly tableau constructions is a normal form for temporal formulas in which the constraints on the current state are separated from the constraints on the future states. Another research topic is the search for fragments of temporal logics that reduce the complexity of tableau constructions, which is PSPACE or worse for many common logics. Furthermore, the limits of expressiveness of logics within certain complexity classes are explored ([4, 13]).

In the real-time domain, tableau constructions have been developed for various logics and their complexities have been studied ([1, 9, 13]). The key observation behind the tableau constructions in the case of a real-valued time domain is the observation that, under certain restrictions on the logic, the continuous time axis can be discretised. More precisely, given a real-time temporal logic formula φ, time can be sliced into countably many intervals in such a way that along each interval, the truth value of any subformula of φ is invariant. There has been considerably less work aimed at improving and implementing tableau constructions for real-time temporal logics.

In this paper, we present an on-the-fly tableau construction for a linear temporal logic with dense time. The logic that we consider is based on a fragment of *Metric Interval Temporal Logic* (MITL, see [4]) that is decidable in PSPACE. To the best of our knowledge, our tableau construction is the first on-the-fly construction for this logic. Compared to the tableau construction for MITL presented in [1], our work is distinguished by the following points. Our tableau algorithm is on-the-fly. The technical underpinning of this construction is a generalisation to the real-time case of the normal form procedure for formulas, that separates constraints on the current time interval from those on the rest of the time axis. In order to define this normal form, the logic is extended with timers that can be explicitly set and tested, and with a *Next* operator referring to the beginning of the next time interval. As expected, the correctness of the normal form procedure hinges on the discretisation result mentioned above.

A technical complication of the tableau construction of [1] is caused by the two different forms that interval bounds may have: open or closed. As a result, the transition from an interval of time into the next, adjacent, interval can happen in two ways. In order to distinguish these, an additional clock in the timed tableau automaton is needed and furthermore, the type of transition must be remembered. We impose restrictions on the form of the models of formulas and restrict the timer constraints expressible in the logic, in such a way that it suffices to consider intervals that are left-closed and right open. Apart from saving a timer in the tableau automaton, this simplifies the presentation. We have implemented a prototype of the algorithm and give experimental results.

Section 2 introduces the versions of logic and timed automata we use. Section 3 presents the normal form for formulas. The tableau algorithm, its correctness, and the implementation are the topic of Section 4. Section 5 concludes.

2 Preliminaries

An (ω-)*word* $\bar{w} = \sigma_0\sigma_1\sigma_2\ldots$ over an alphabet Σ is an infinite sequence of symbols from Σ; $\bar{w}(k)$ denotes σ_k and $\bar{w}^k$ refers to the *tail* $\sigma_k\sigma_{k+1}\sigma_{k+2}\ldots$. Indeed, we use the latter notations for other kinds of sequences as well. An *interval* $I = [a, b)$ is a convex, left-closed, and right-open subset of $\mathbb{R}^{\geq 0}$; $l(I)$ ($r(I)$) denotes the lower (upper) bound a (b) and $|I|$ the length of I. We use $I - t$ to denote the interval $\{t' - t \mid t' \in I\}$. An *interval sequence* $\bar{I} = I_0I_1I_2\ldots$ is an infinite sequence of intervals that are *adjacent*, meaning $r(I_i) = l(I_{i+1})$ for every i, for which $l(I_0) = 0$, and which is *diverging*, i.e. any $t \in \mathbb{R}^{\geq 0}$ belongs to some interval I_i. A *timed word* $\bar{u}$ over Σ is a pair $(\bar{w}, \bar{I})$ consisting of an ω-word $\bar{w}$ over Σ and an interval sequence $\bar{I}$. For $t \in \mathbb{R}^{\geq 0}$, $\bar{u}(t)$ denotes the symbol present at time t, this is $\bar{w}(k)$ if $t \in \bar{I}(k)$. For such a t and k, $\bar{u}^t$ is the tail of the timed word, consisting of the word $\bar{w}^k$ and of the interval sequence $[0, r(\bar{I}(k)) - t)$, $\bar{I}(k+1) - t, \ldots$. Timed words $\bar{u}_1$ and $\bar{u}_2$ are *equivalent* (denoted $\bar{u}_1 \equiv \bar{u}_2$) if for all $t \geq 0$, $\bar{u}_1(t) = \bar{u}_2(t)$. A *timed state sequence* over a set *Prop* of propositions is a timed word over the alphabet 2^{Prop}.

2.1 Real-time temporal logic

We consider a restricted version of the real-time temporal logic MITL of [4], MITL$_{\leq}$, with formulas of the following form ($d \in \mathbb{N}$).

$$\varphi ::= \text{true} \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathsf{U}_{\leq d}\varphi_2$$

Formulas of this form are called *basic*, in order to distinguish them from formulas using an extended syntax that is to be defined in Section 3.1. A formula is interpreted over a timed state sequence $\bar{\rho}$ as follows.

- $\bar{\rho} \models \text{true}$ for every timed state sequence $\bar{\rho}$;
- $\bar{\rho} \models p$ iff $p \in \bar{\rho}(0)$;
- $\bar{\rho} \models \neg\varphi$ iff not $\bar{\rho} \models \varphi$;
- $\bar{\rho} \models \varphi_1 \vee \varphi_2$ iff $\bar{\rho} \models \varphi_1$ or $\bar{\rho} \models \varphi_2$;
- $\bar{\rho} \models \varphi_1 \mathsf{U}_{\leq d}\varphi_2$ iff there is some $0 \leq t \leq d$, such that $\bar{\rho}^t \models \varphi_2$ and for all $0 \leq t' < t$, $\bar{\rho}^{t'} \models \varphi_1$.

Note that no basic MITL$_{\leq}$ formula can discriminate between equivalent timed state sequences. The restriction of the bound d to finite naturals simplifies the presentation as it yields a logic in which only safety properties can be expressed, thus avoiding the need to handle acceptance conditions. It is straightforward to extend our results to a logic which also includes an unbounded Until operator.

2.2 φ-fine interval sequences

In the on-the-fly tableau constructions of [5, 8] for untimed logics, a formula is rewritten so that the constraints on the current state are separated from those on the remainder of the state sequence. This is possible because of the discrete nature of the sequence. In the dense time case, there is no such thing as a next state. The discretisation suggested by the interval sequence $\bar{I}$ of a timed state sequence $\bar{\rho}$ is, in general, not fine enough for our purposes: when interpreted over tails of $\bar{\rho}$, the truth value of a formula may vary along a single interval of $\bar{I}$.

Definition 1. *Let $\varphi \in$ MITL$_\leq$. An interval sequence $\bar{I}$ is called* φ-fine *for timed state sequence $\bar{\rho}$ if for every syntactic subformula ψ of φ, every $k \geq 0$, and every $t_1, t_2 \in \bar{I}(k)$, we have $\bar{\rho}^{t_1} \models \psi$ iff $\bar{\rho}^{t_2} \models \psi$. In case that $\bar{I}$ is φ-fine for a timed state sequence $(\bar{\sigma}, \bar{I})$, also $(\bar{\sigma}, \bar{I})$ will be called φ-fine.*

In [1] (Lemma 4.11) it was shown that the intervals of a timed state sequence can always be refined so that the value of a given MITL formula does not change within any interval. Although the timed state sequences have a slightly more restrictive definition in our case, a similar lemma can be proved for the restricted logic MITL$_\leq$.

Lemma 1. *Let φ be a basic* MITL$_\leq$ *formula and $\bar{\rho}$ a timed state sequence. Then there exists a φ-fine timed state sequence that is equivalent with $\bar{\rho}$.*

Note that this lemma also implies that, when confining to φ-fine timed state sequences, it suffices to consider intervals, which are left-closed and right-open by our definition. In particular, there is no need to introduce a more general notion of interval, as in [1].

2.3 Timed automata

The target of our tableau construction are timed automata in the style of Alur and Dill ([3]). We use a variation that is adapted to our needs. The automata use *timers* that decrease as time advances, possibly taking negative values in $\mathbb{R}$. They may be set to nonnegative integer values and may be compared to zero in a restricted way. Given a set T of timers, a *timer valuation* $\nu \in TVal(T)$ is a mapping $T \to \mathbb{R}$. For $t \in \mathbb{R}^{\geq 0}$, $\nu - t$ denotes the timer valuation that assigns $\nu(x) - t$ to any timer x in the domain of ν. A *timer setting* $TS \in TSet(T)$ is a partial mapping $T \to \mathbb{N}$. We use $TS[x := d]$ (where $x \in T$ and $d \in \mathbb{N}$) to denote the timer setting that maps x to d and other timers to the same value as TS. $[x := d]$ is short for $\varnothing[x := d]$. For a timer valuation ν and a timer setting TS, $TS(\nu)$ is the timer valuation that maps any timer x in the domain of ν to $TS(x)$ if defined, and to $\nu(x)$ otherwise. The set $TCond(T)$ of *timer conditions over* T is $\{x > 0, x \leq 0 \mid x \in T\}$. As suggested by Lemma 1 above, our timed automata may be restricted in such a way that the period of time during which control resides in a location is always a (left-closed, right-open) interval.

Definition 2. *Let Σ be an alphabet. A* timed automaton *$A = \langle L, T, L_0, Q, TC, E\rangle$ over Σ consists of*

- *a finite set L of* locations*;*
- *a set T of timers;*
- *a finite set L_0 of* initial extended locations *$(\ell_0, \nu_0) \in L \times TVal(T)$, where ν_0 assigns integer values to the timers;*

– *a mapping* $Q : L \to 2^{\Sigma}$ *labelling every location with a set of symbols from the alphabet;*
– *a mapping* $TC : L \to 2^{TCond(T)}$ *labelling every location with a set of timer conditions over* T*;*
– *a set* $E \subseteq L \times TSet(T) \times L$ *of* edges *labelled by timer settings.*

An *extended location* λ is a pair (ℓ, ν) consisting of a location ℓ and a timer valuation ν. In the context of an automaton with the set T of timers, we use $\bar{0}$ to denote the timer valuation that maps every timer in T to 0.

A *timed run* describes the path taken by the timed automaton when accepting a timed word. It gives the location of the automaton and the values of its timers at any moment, by recording the sequence of locations, the intervals during which the automaton resides in those locations, and the timer values at the beginning of each such interval.

Definition 3. *A* timed run $\bar{r}$ *of a timed automaton* $A = \langle L, T, L_0, Q, TC, E \rangle$ *is a triple* $\left(\bar{\ell}, \bar{I}, \bar{\nu}\right)$ *consisting of a sequence of locations, an interval sequence, and a sequence of timer valuations, such that:*

– *[Consecution] for all* $k \geq 0$*, there is an edge* $\left(\bar{\ell}(k), TS_k, \bar{\ell}(k+1)\right) \in E$ *such that* $\bar{\nu}(k+1) = TS_k(\bar{\nu}(k) - |\bar{I}(k)|)$*;*
– *[Timing] for all* $k \geq 0$ *and* $t \in \bar{I}(k)$*, the timer valuation at time* t*,* $\bar{\nu}(k) - (t - l(\bar{I}(k)))$*, satisfies all timer conditions in* $TC(\bar{\ell}(k))$*.*

In this case we also say that $\bar{r}$ *is a run* from *extended location* $(\bar{\ell}(0), \bar{\nu}(0))$*, or an* $(\bar{\ell}(0), \bar{\nu}(0))$-run *(or simply an* $\bar{\ell}(0)$-run*). We write* $\bar{r}(t)$ *to denote the location of* $\bar{r}$ *at time* t*, i.e.* $\bar{\ell}(k)$ *if* $t \in \bar{I}(k)$*. Given a timed word* $\bar{u}$*,* $\bar{r}$ *is a run* for $\bar{u}$ *if*[1]

– *[Symbol match] for all* $t \geq 0$*,* $\bar{u}(t) \in Q(\bar{r}(t))$*.*

A accepts $\bar{u}$ *if it has a run for* $\bar{u}$ *from an initial extended location. The* timed language $\mathcal{L}(A)$ *of* A *is the set of all timed words that it accepts.*

It follows from Definition 3 that the language of a timed automaton is closed under equivalence. Together with Lemma 1, this will allow us to restrict our attention to φ-fine sequences when arguing the correctness of the tableau construction for a basic $\mathrm{MITL}_{\leq}$ formula φ.

3 Disjunctive Temporal Normal Form

Central to tableau constructions for temporal logics is the observation that every formula can be rewritten into a normal form in which the constraints on the current state are (syntactically) separated from the constraints on the future states. For example, in the case of Propositional Linear-time Temporal Logic (LTL), any formula can be written as a disjunction of terms of the form $\pi \wedge \bigcirc \varphi$,

[1] As locations are labelled with *sets* of symbols, a single run corresponds in general to a set of timed words.

where π is a conjunction of propositions dictating the constraints on the current state and the LTL formula φ is "guarded" by the *Next* operator $\bigcirc$, meaning that it must hold in the future starting from the next state. In order to define a similar normal form for $\mathrm{MITL}_{\leq}$, the logic needs to be extended by adding a Next operator and by introducing timers and timer settings into the syntax of the logic.

3.1 Extending the logic

In order to define the normal form, the logic is extended with timers that can be bound by timer setting operators or which can be free. A formula is interpreted in a timer environment which provides a value for every free timer. Moreover we assume that formulas are in positive normal form (negations only occur in front of propositions) using the dual operators (false, $\wedge$, V). The timer set operator can set timers to integer values, and timers can be compared to 0 only by checking the condition $x > 0$ or $x \leq 0$. Compared to other timed logics that use freeze/reset quantifiers (see e.g. [2, 10]), the syntax of our logic is restricted: The arguments of an Until or Release operator must be basic $\mathrm{MITL}_{\leq}$ formulas.

Definition 4. $\mathrm{MITL}_{\leq}$ *is redefined. Besides the basic formulas defined in Section 2.1 we add formulas of the following form, where φ, φ_1, and φ_2 are basic formulas, $d \in \mathbb{N}$, TS is a timer setting, and x is a timer.*

$$\psi ::= \varphi \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid TS.\psi \mid x > 0 \mid x \leq 0 \mid \varphi_1 \mathsf{V}_{<d} \varphi_2 \mid \varphi_1 \mathsf{U}_{\leq x} \varphi_2 \mid \varphi_1 \mathsf{V}_{<x} \varphi_2 \mid \bigcirc \psi.$$

The semantics is extended as follows. We write $\bar{\rho} \models_\nu \psi$ to denote that the timed state sequence $\bar{\rho}$ satisfies ψ in the context of the timer valuation ν.

$\bar{\rho} \models_\nu \varphi$ *iff* $\bar{\rho} \models \varphi$;
$\bar{\rho} \models_\nu \psi_1 \vee \psi_2$ *iff* $\bar{\rho} \models_\nu \psi_1$ *or* $\bar{\rho} \models_\nu \psi_2$;
$\bar{\rho} \models_\nu \psi_1 \wedge \psi_2$ *iff* $\bar{\rho} \models_\nu \psi_1$ *and* $\bar{\rho} \models_\nu \psi_2$;
$\bar{\rho} \models_\nu \varphi_1 \mathsf{V}_{<d} \varphi_2$ *iff for all* $0 \leq t < d$, $\bar{\rho}^t \models_{\nu - t} \varphi_2$ *or there is some* $0 \leq t' < t$, *such that* $\bar{\rho}^{t'} \models_{\nu - t'} \varphi_1$;
$\bar{\rho} \models_\nu TS.\psi$ *iff* $\bar{\rho} \models_{TS(\nu)} \psi$;
$\bar{\rho} \models_\nu \varphi_1 \mathsf{U}_{\leq x} \varphi_2$ *iff there is some* $0 \leq t \leq \nu(x)$, *such that* $\bar{\rho}^t \models_{\nu - t} \varphi_2$ *and for all* $0 \leq t' < t$, $\bar{\rho}^{t'} \models_{\nu - t'} \varphi_1$;
$\bar{\rho} \models_\nu \varphi_1 \mathsf{V}_{<x} \varphi_2$ *iff for all* $0 \leq t < \nu(x)$, $\bar{\rho}^t \models_{\nu - t} \varphi_2$ *or there is some* $0 \leq t' < t$, *such that* $\bar{\rho}^{t'} \models_{\nu - t'} \varphi_1$;
$\bar{\rho} \models_\nu x > 0$ *iff* $\nu(x) > 0$;
$\bar{\rho} \models_\nu x \leq 0$ *iff* $\nu(x) \leq 0$;
$\bar{\rho} \models_\nu \bigcirc\psi$ *iff* $(\bar{\sigma}^1, \bar{I}^1) \models_{\nu - |\bar{I}(0)|} \psi$ *where* $\bar{\rho} = (\bar{\sigma}, \bar{I})$.

Note that with the $\bigcirc$ operator it is now possible to discriminate between equivalent timed state sequences. In our tableau construction for the timed case, formulas will also be rewritten to separate "now" from "next". More precisely, the "now" part will refer to the current *interval* of time, and so it will be thought of as being true during a non-zero *period*. In contrast, the "next" part of a formula in normal form will refer to the *first point* of the next interval. In addition, in rewriting a formula, we will also separate the setting of the timers to be effected when entering the current interval. The rewrite rules achieving this will be presented in the next subsection. Of the equivalences on which they are based, the most important ones are displayed below. Here, $\psi \equiv \psi'$ abbreviates $\bar{\rho} \models_\nu \psi$ iff $\bar{\rho} \models_\nu \psi'$, where $\bar{\rho}$ is a timed state sequence and ν a timer valuation, that are both implicitly universally quantified. For some equivalences, conditions on $\bar{\rho}$ and ν are required which are then explicitly listed. φ_1 and φ_2 are basic $\text{MITL}_{\leq}$ formulas, $d \in \mathbb{N}$, and x and y are timers.

$$\varphi_1 \mathsf{U}_{\leq d} \varphi_2 \equiv [y := d].(\varphi_1 \mathsf{U}_{\leq y} \varphi_2) \tag{1}$$

$$\varphi_1 \mathsf{V}_{< d} \varphi_2 \equiv [y := d].(\varphi_1 \mathsf{V}_{< y} \varphi_2) \tag{2}$$

$$\varphi_1 \mathsf{U}_{\leq x} \varphi_2 \equiv \varphi_2 \vee (x > 0 \wedge \varphi_1 \wedge \bigcirc(\varphi_1 \mathsf{U}_{\leq x} \varphi_2)) \tag{3}$$

$$\text{if } \bar{\rho} \text{ is both } \varphi_1\text{-fine and } \varphi_2\text{-fine, and } \nu(x) \geq 0$$

$$\varphi_1 \mathsf{V}_{\leq d} \varphi_2 \equiv \varphi_2 \wedge (\varphi_1 \vee \bigcirc(\varphi_1 \mathsf{V}_{< d} \varphi_2)) \tag{4}$$

$$\text{if } \bar{\rho} \text{ is } \varphi_1 \mathsf{V}_{\leq d} \varphi_2\text{-fine}$$

$$\varphi_1 \mathsf{V}_{< x} \varphi_2 \equiv x \leq 0 \vee (\varphi_2 \wedge (\varphi_1 \vee \bigcirc(\varphi_1 \mathsf{V}_{< x} \varphi_2))) \tag{5}$$

$$\text{if } \bar{\rho} \text{ is both } \varphi_1\text{-fine and } \varphi_2\text{-fine}$$

Lemma 2. *The equivalences between* $\text{MITL}_{\leq}$ *formulas presented above hold.*

Proof. We will only prove the equivalence 4. Let $\bar{\rho}$ be a timed state sequence that is fine for $\varphi_1 \mathsf{V}_{\leq d} \varphi_2$ and ν a timer valuation then $\bar{\rho} \models_\nu \varphi_1 \mathsf{V}_{\leq d} \varphi_2$ iff $\bar{\rho} \models_\nu \varphi_2 \wedge (\varphi_1 \vee \bigcirc(\varphi_1 \mathsf{V}_{< d} \varphi_2))$ ($\Rightarrow$) Let $\bar{\rho} \models_\nu \varphi_1 \mathsf{V}_{\leq d} \varphi_2$. Obviously, $\bar{\rho}$ satisfies φ_2. Assume $\bar{\rho} \not\models_\nu \varphi_1$, then we need to show that $\bar{\rho} \models_\nu \bigcirc(\varphi_1 \mathsf{V}_{< d} \varphi_2)$. Since $\bar{\rho}$ is $\varphi_1 \mathsf{V}_{\leq d} \varphi_2$-fine, if $\varphi_1 \mathsf{V}_{\leq d} \varphi_2$ does not hold at the first moment of the second interval, then this is the first moment where it does not hold. Thus at the first moment of the second interval $\varphi_1 \mathsf{V}_{< d} \varphi_2$ holds. ($\Leftarrow$) obvious.

One may wonder why the timer condition $\nu(x) \geq 0$ in rule 3 is listed external to the formula, instead of making it part of the right-hand side: $(x \geq 0 \wedge \varphi_2) \vee (x > 0 \wedge \varphi_1 \wedge \bigcirc(\varphi_1 \mathsf{U}_{\leq x} \varphi_2))$. The reason is that the truth value of a condition of the form $x \geq 0$ may change in a "left-open fashion": it may be true in the first (singular) point of an interval, but false in the remainder. By syntactically restricting the timer conditions to the forms $x > 0$ and $x \leq 0$ only, it is clearer that the "now" parts of a formula in normal form can indeed be made true during a full interval. Indeed, the condition $\nu(x) \geq 0$ will turn out to be always fulfilled when equivalence 3 is used as a rewrite rule in the tableau construction. This is also the reason for the introduction of the formula $\varphi_1 \mathsf{V}_{< d} \varphi_2$: Unfolding the formula $\varphi_1 \mathsf{V}_{\leq x} \varphi_2$ would lead to a timer condition $x < 0$.

3.2 Rewrite rules

Definition 5. *An* $\text{MITL}_{\leq}$ *formula is said to be in disjunctive temporal form if it is of the form* $\bigvee_{i=1}^{k} TS_i.(\Pi_i \wedge \bigcirc \Phi_i)$ *where* $k \geq 0$ *(for* $k = 0$ *the formula equals* false *represented as the empty disjunction), the* Π_i *are conjunctions of* *atomic propositions, negated atomic propositions and timer conditions, and the* Φ_i *conjunctions of* $\textsc{Mitl}_{\leq}$ *formulas.*

For presentational purposes we introduce some notation. We will identify a set Ψ of formulas with the conjunction $\bigwedge \Psi$, the empty conjunction being equivalent to true. A *term* is a triple $\langle TS, Now, Next \rangle$, where TS is a timer setting and Now and $Next$ are sets of $\textsc{Mitl}_{\leq}$ formulas. Such a term is identified with the $\textsc{Mitl}_{\leq}$ formula $TS.(Now \wedge \bigcirc Next)$. A set of terms will be identified with the *disjunction* of the formulas represented by the individual terms. Note that an arbitrary $\textsc{Mitl}_{\leq}$ formula ψ is represented by the set $\{\langle \varnothing, \{\psi\}, \varnothing \rangle\}$ of terms. We now introduce a number of rewrite rules on sets of terms, that transform any such set into disjunctive temporal form. These rules are presented in Figure 1, which is interpreted as follows. Consider a set $\Phi \cup \{\langle TS, Now \cup \{\psi\}, Next \rangle\}$ of terms. The row in the table in which the Case field coincides with the shape of the $\textsc{Mitl}_{\leq}$ formula ψ determines how the set is rewritten. For a timer setting TS, the function $rename_{TS}$ renames the timers occurring in any syntactical object apart from the timers in TS. The following lemma states that the disjunctive

Case	$\Phi \cup \{\langle TS, Now \cup \{\psi\}, Next \rangle\}$ reduces to:
$\psi = \text{true}$	$\Phi \cup \{\langle TS, Now, Next \rangle\}$
$\psi = \text{false}$	Φ
$\psi = \psi_1 \vee \psi_2$	$\Phi \cup \{\langle TS, Now \cup \{\psi_1\}, Next \rangle, \langle TS, Now \cup \{\psi_2\}, Next \rangle\}$
$\psi = \psi_1 \wedge \psi_2$	$\Phi \cup \{\langle TS, Now \cup \{\psi_1, \psi_2\}, Next \rangle\}$
$\psi = \varphi_1 \mathsf{U}_{\leq d} \varphi_2$	$\Phi \cup \{\langle TS, Now \cup \{[x := d].(\varphi_1 \mathsf{U}_{\leq x} \varphi_2)\}, Next \rangle\}$
$\psi = \varphi_1 \mathsf{U}_{\leq x} \varphi_2$	$\Phi \cup \{\langle TS, Now \cup \{\varphi_2 \vee (x > 0 \wedge \varphi_1 \wedge \bigcirc(\varphi_1 \mathsf{U}_{\leq x} \varphi_2))\}, Next \rangle\}$
$\psi = \varphi_1 \mathsf{V}_{\leq d} \varphi_2$	$\Phi \cup \{\langle TS, Now \cup \{\varphi_2 \wedge (\varphi_1 \vee \bigcirc(\varphi_1 \mathsf{V}_{< d} \varphi_2))\}, Next \rangle\}$
$\psi = \varphi_1 \mathsf{V}_{< d} \varphi_2$	$\Phi \cup \{\langle TS, Now \cup \{[x := d].(\varphi_1 \mathsf{V}_{< x} \varphi_2)\}, Next \rangle\}$
$\psi = \varphi_1 \mathsf{V}_{< x} \varphi_2$	$\Phi \cup \{\langle TS, Now \cup \{x \leq 0 \vee (\varphi_2 \wedge (\varphi_1 \vee \bigcirc(\varphi_1 \mathsf{V}_{< x} \varphi_2)))\}, Next \rangle\}$
$\psi = TS'.\psi'$	$\Phi \cup \{\langle TS \cup TS'', Now \cup \{\psi''\}, Next \rangle\}$ where $TS''.\psi'' = rename_{TS}(TS'.\psi')$
$\psi = \bigcirc \psi'$	$\Phi \cup \{\langle TS, Now, Next \cup \{\psi'\} \rangle\}$

Fig. 1. Rewrite rules for $\textsc{Mitl}_{\leq}$.

temporal form is a normal form under these rules.

Lemma 3. *Let* ψ *be an* $\textsc{Mitl}_{\leq}$ *formula and* ψ' *be obtained from* ψ *by repeated application of rules from Figure 1 until no more rule applies. This process terminates, and* ψ' *is in disjunctive temporal form. Furthermore, for every timed state sequence* $\bar{\rho}$ *and every timer valuation* ν *such that*

- $\bar{\rho}$ *is fine for every basic syntactic subformula of* ψ*;*
- $\nu(x) \geq 0$ *for every timer* x *that is free in* ψ *and that occurs in a subformula of the form* $\psi_1 \mathsf{U}_{\leq x} \psi_2$*,*

we have $\bar{\rho} \models_\nu \psi$ *iff* $\bar{\rho} \models_\nu \psi'$*.*

Proof. Termination and the fact that ψ' is in disjunctive temporal form are easily seen. The equivalence of ψ and ψ' under the stated conditions follows from Lemma 2 and a few distributivity rules for the $\bigcirc$ and timer set operators.

Depending on the order in which terms from Φ and formulas from *Now* are selected, different normal forms may be obtained. In the sequel, we assume the existence of a deterministic procedure *NF* that computes a particular normal form for any given formula. The following lemma states that the timer setting *TS* and *Now* and *Next* parts of a term in a normal form have the properties we set out for. It will be used in the correctness proof in Section 4.3.

Lemma 4. *Let $\psi \in \text{MITL}_{\leq}$, ν be a timer valuation, and $\bar{\rho}$ a timed state sequence with interval sequence $\bar{I}$ that is fine for all basic subformulas of ψ. If $\bar{\rho} \models_\nu \psi$, then there is some term $\langle TS, Now, Next\rangle \in NF(\psi)$ such that for all $0 \leq t < l(\bar{I}(1))$, $\bar{\rho}^t \models_{TS(\nu)-t} Now$, and furthermore $\bar{\rho}^{l(\bar{I}(1))} \models_{TS(\nu)-l(\bar{I}(1))} Next$.*

Proof. Follows by induction on the number of rewrite steps in the normal form procedure.

Example The (basic) $\text{MITL}_{\leq}$ formula $\Diamond_{\leq 2} p = \text{true}\mathsf{U}_{\leq 2} p$ has an equivalent formula in disjunctive temporal form:

$$\text{true}\mathsf{U}_{\leq 2} p \equiv ([x := 2].p) \vee ([x := 2].\,(x > 0 \wedge \bigcirc(\text{true}\mathsf{U}_{\leq x} p)))$$

Here, the equivalence holds for any timer valuation ν and any timed state sequence $\bar{\rho}$ that is fine for p. In terms of the normal form procedure, the rewriting process of $\text{true}\mathsf{U}_{\leq 2} p$ proceeds as follows (we write $\Phi_1 \Rightarrow \Phi_2$ to express that Φ_2 is obtained from Φ_1 by one or more steps in the procedure).

$$\begin{gathered}
\{\langle \varnothing, \{\text{true}\mathsf{U}_{\leq 2} p\}, \varnothing\rangle\} \Rightarrow \{\langle [x := 2], \{\text{true}\mathsf{U}_{\leq x} p\}, \varnothing\rangle\} \Rightarrow \\
\{\langle [x := 2], \{p \vee (x > 0 \wedge \text{true} \wedge \bigcirc(\text{true}\mathsf{U}_{\leq x} p))\}, \varnothing\rangle\} \Rightarrow \\
\{\langle [x := 2], \{p\}, \varnothing\rangle, \langle [x := 2], \{x > 0, \text{true}, \bigcirc(\text{true}\mathsf{U}_{\leq x} p)\}, \varnothing\rangle\} \Rightarrow \\
\{\langle [x := 2], \{p\}, \varnothing\rangle, \langle [x := 2], \{x > 0\}, \{\text{true}\mathsf{U}_{\leq x} p\}\rangle\}
\end{gathered}$$

4 Tableau Construction

4.1 The tableau algorithm

The construction of a tableau automaton for a basic $\text{MITL}_{\leq}$ formula φ, is based upon the normal form introduced in the previous section. The number of formulas that may occur in the *Now* and *Next* sets of the normal form terms is limited to syntactic subformulas of φ and a number of formulas derived from them such as timer conditions and Until or Release formulas indexed by a timer. However, the procedure introduces new timers when applying the reduction for $\psi = TS'.\psi'$. If not applied carefully, this could lead to an unbounded number of timers and locations of the tableau automaton. To prevent this, we use a unique timer x_ψ for every Until or release formula ψ occurring in φ. It follows from the normal form procedure that the only use of the timer of an Until (Release) will be in the $\varphi_1 \mathsf{U}_{\leq x} \varphi_2$ ($\varphi_1 \mathsf{V}_{<x} \varphi_2$) variant of its corresponding formula. Then we can apply the following equivalences to limit the number of timers.

$$(\varphi_1 \mathsf{U}_{\leq x} \varphi_2) \wedge ([y := d].(\varphi_1 \mathsf{U}_{\leq y} \varphi_2)) \equiv \varphi_1 \mathsf{U}_{\leq x} \varphi_2 \qquad \text{if } \nu(x) \leq d \tag{6}$$

$$(\varphi_1 \mathsf{V}_{<x} \varphi_2) \wedge ([y := d].(\varphi_1 \mathsf{V}_{<y} \varphi_2)) \equiv [y := d].(\varphi_1 \mathsf{V}_{<y} \varphi_2) \qquad \text{if } \nu(x) \leq d \tag{7}$$

The validity of these rules follows straightforwardly from the semantics. The use of the case $\psi = TS'.\psi'$ can be circumvented by replacing the corresponding rules in the normal form procedure with the following new rules for the cases $\varphi_1 \mathsf{U}_{\leq d} \varphi_2$ and $\varphi_1 \mathsf{V}_{<d} \varphi_2$, based on the equivalences presented above. In the remainder, $NF(\psi)$ refers to the updated version of the normal form procedure.

CASE	$\Phi \cup \{\langle TS, Now \cup \{\psi\}, Next\rangle\}$ REDUCES TO:
$\psi = \varphi_1 \mathsf{U}_{\leq d} \varphi_2$	$\Phi \cup \{\langle TS[x_\psi := d], Now \cup \{\varphi_1 \mathsf{U}_{\leq x_\psi} \varphi_2\}, Next\rangle\}$ if $\{x_\psi > 0, \varphi_1 \mathsf{U}_{\leq x_\psi} \varphi_2\} \cap Now = \varnothing$
$\psi = \varphi_1 \mathsf{U}_{\leq d} \varphi_2$	$\Phi \cup \{\langle TS, Now \cup \{\varphi_1 \mathsf{U}_{\leq x_\psi} \varphi_2\}, Next\rangle\}$ if $\{x_\psi > 0, \varphi_1 \mathsf{U}_{\leq x_\psi} \varphi_2\} \cap Now \neq \varnothing$
$\psi = \varphi_1 \mathsf{V}_{<d} \varphi_2$	$\Phi \cup \{\langle TS[x_\psi := d], Now \cup \{\varphi_1 \mathsf{V}_{<x_\psi} \varphi_2\}, Next\rangle\}$

The tableau automaton of an MITL$_{\leq}$ formula φ is computed in the following way.

Definition 6. *Let φ be a basic* MITL$_{\leq}$ *formula and Prop be the set of atomic propositions that occur in φ. Then the tableau automaton A_φ of φ is the automaton $\langle L, T, L_0, Q, TC, E\rangle$ over the alphabet 2^{Prop}, where*

- *T is the set of all timers x_ψ for every Until or Release formula ψ that occurs as a syntactic subformula of φ.*
- *The locations (L), initial extended locations (L_0) and transitions (E) are computed by the procedure depicted in Figure 2. The locations $\ell \in L$ are pairs $(Now, Next)$ of sets. The first item of the location ℓ is denoted by $Now(\ell)$ and the last by $Next(\ell)$.*
- *$Q(\ell) = \{\sigma \in 2^{Prop} \mid \forall_{p \in Prop} p \in Now(\ell) \Rightarrow p \in \sigma, \neg p \in Now(\ell) \Rightarrow p \notin \sigma\}$. That is, a location ℓ is labelled with all sets of propositions that are consistent with the atomic propositions and the negated atomic propositions in $Now(\ell)$.*
- *$TC(\ell) = \{\xi \in TCond(T) \mid \xi \in Now(\ell)\}$, the location is labelled with all timer conditions in $Now(\ell)$.*

4.2 Example

If we take the formula $\Box_{\leq 100} \Diamond_{\leq 5} p = \text{false} \mathsf{V}_{\leq 100} (\text{true} \mathsf{U}_{\leq 5} p)$ and apply the tableau algorithm, we arrive at the automaton represented in Figure 3. Only the formulas in the *Now* set of the locations have been depicted. Initial extended locations are represented by a small arrow not originating from any location leading to the initial location and labelled with a timer setting that yields the initial timer valuation when applied to the timer valuation that assigns 0 to every timer. There is a timer associated with the formula $\Diamond_{\leq 5} p$ named x and a timer associated with the formula $\Box_{\leq 100} \Diamond_{\leq 5} p$ named y.

```
L_0 := {((Now, Next), TS(0̄)) | ⟨TS, Now, Next⟩ ∈ NF(φ)}
LNew := {(Now, Next) | ((Now, Next), TS) ∈ L_0 }
L := ∅, E := ∅
while LNew ≠ ∅ do
  Let (Now, Next) ∈ LNew
  LNew := LNew\{(Now, Next)}
  L := L ∪ {(Now, Next)}
  for every (TS', Now', Next') ∈ NF(Next) do
    E := E ∪ {((Now, Next), TS', (Now', Next'))}
    if (Now', Next') ∉ L then LNew := LNew ∪ {(Now', Next')}
    od
od
```

Fig. 2. Algorithm for constructing the locations and edges of the on-the-fly tableau automaton.

4.3 Correctness

In this section we will give a sketch of the proof that the tableau construction is correct, i.e. that for any basic $\text{MITL}_{\leq}$ formula φ, the tableau automaton of φ accepts precisely those timed state sequences that satisfy φ, as expressed by the following theorem.

Theorem 1. *Let φ be a basic $\text{MITL}_{\leq}$ formula, A_φ the corresponding tableau automaton. Then for every timed state sequence $\bar{\rho}$, A_φ accepts $\bar{\rho}$ iff $\bar{\rho} \models \varphi$.*

This theorem follows from soundness (every state sequence accepted by A_φ satisfies φ) and completeness (every state sequence satisfying φ is accepted by A_φ) of the construction as expressed by Lemmas 7 and 10 below. The structure of the proof is similar to the correctness proof of the untimed on-the-fly tableau of [8]. In this section, we assume that φ is a basic $\text{MITL}_{\leq}$ formula and $A_\varphi = \langle L, T, L_0, Q, TC, E\rangle$ its tableau automaton.

Soundness In this paragraph it is demonstrated that the automaton accepts only timed state sequences that satisfy φ. It is shown that whenever a formula ψ leads to the normal form term that corresponds to a particular location ℓ, then any state sequence for which there is an ℓ-run satisfies the formula ψ. To do this, we associate with a location $\ell = (Now, Next)$, reached after performing the timer setting TS, the set $Old(TS, \ell)$ as the set of all formulas for which $\langle TS, Now, Next\rangle$ is a term of the normal form. Note that in [8], a similar Old set is computed during the construction of the tableau and is part of a location.

Definition 7. *Let TS be a timer setting and $\ell = (Now, Next)$.*

$$Old(TS, \ell) = \bigcup \{Now' \mid \langle TS, Now, Next\rangle \in NF(\langle TS', Now', Next'\rangle) \text{ for some } TS', Next'\}$$

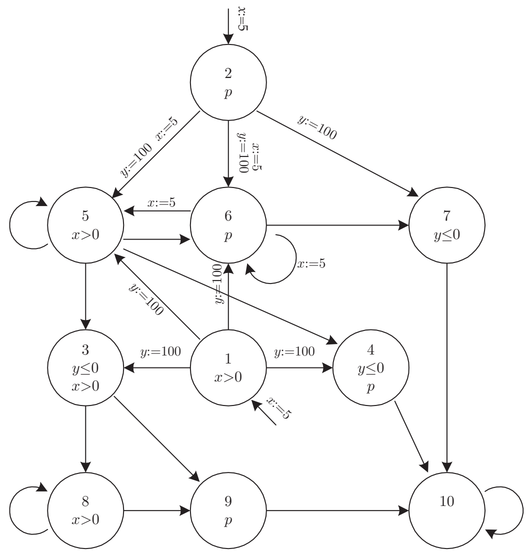

Fig. 3. Example tableau automaton of the formula $\Box_{\leq 100}\Diamond_{\leq 5}p$

The main lemma is the following, claiming that any formula in the *Old* set of a particular location is dealt with correctly.

Lemma 5. *Let $\bar{\rho}$ be a timed state sequence and $\bar{r} = \left(\bar{\ell}, \bar{I}, \bar{\nu}\right)$ be a run of A_φ for $\bar{\rho}$, taking the edges $\left(\bar{\ell}(k), TS_{k+1}, \bar{\ell}(k+1)\right)$. Furthermore, let TS be a timer setting such that $\bar{\nu}(0) = TS(\nu)$ for some timer valuation ν and $\psi \in Old(TS, \bar{\ell}(0))$. Then $\bar{\rho} \models_{\bar{\nu}(0)} \psi$.*

Proof. By induction on the structure of ψ. We only show the cases related to the Until formula.

- If $\varphi_1 \mathsf{U}_{\leq d} \varphi_2 \in Old(TS, \bar{\ell}(0))$, then it can be shown by the reduction of $\varphi_1 \mathsf{U}_{\leq d} \varphi_2$ in the disjunctive temporal form procedure that $\varphi_1 \mathsf{U}_{\leq x} \varphi_2 \in Old(TS, \bar{\ell}(0))$. By induction it follows that $\bar{\rho} \models_{\bar{\nu}(0)} \varphi_1 \mathsf{U}_{\leq x} \varphi_2$. Since x cannot be larger than d in $\bar{\nu}(0)$, it follows that $\bar{\rho} \models_{\bar{\nu}(0)} \varphi_1 \mathsf{U}_{\leq d} \varphi_2$.
- If $\varphi_1 \mathsf{U}_{\leq x} \varphi_2 \in Old(TS, \bar{\ell}(0))$, then by the construction of the automaton and the reduction of $\varphi_1 \mathsf{U}_{\leq x} \varphi_2$ in the normal form procedure, there is some k, such that $\varphi_2 \in Old(TS_k, \bar{\ell}(k))$ and for every $0 \leq t < l(\bar{I}(k))$, $\varphi_1 \in Old(\varnothing, \bar{\ell}(t))$ and $x > 0 \in Old(\varnothing, \bar{\ell}(t))$. Moreover it can be shown that the timer x is never set to d in any TS_i, $0 < i \leq k$, and thus $l(\bar{I}(k)) \leq \bar{\nu}(0)(x)$. By induction it follows that $\bar{\rho} \models_{\bar{\nu}(0)} \varphi_1 \mathsf{U}_{\leq x} \varphi_2$. $\square$

One can furthermore show that the *Old* set of every initial location contains the original formula φ.

Lemma 6. *For every initial extended location (ℓ, ν) of A_φ, there exists some timer setting TS such that $\varphi \in Old(TS, \ell)$ and $\nu = TS(\bar{0})$.*

Proof. Follows straightforwardly from the definitions of the automaton and the disjunctive temporal form procedure.

From the Lemmas 5 and 6, it follows immediately that every state sequence accepted by the tableau automaton A_φ satisfies φ.

Lemma 7. *Let $\bar{\rho}$ be a timed state sequence. If A_φ accepts $\bar{\rho}$, then $\bar{\rho} \models \varphi$.*

Completeness In this paragraph it is demonstrated that every timed state sequence that satisfies φ is accepted by the tableau automaton. The normal form procedure guarantees that if a timed state sequence $\bar{\rho}$ satisfies a formula ψ, then there is a term in the normal form of ψ, that is satisfied by $\bar{\rho}$. Moreover, it has been shown in Lemma 4 that the formulas in the *Now* set of the term hold during the entire first interval of $\bar{\rho}$. Since the remainder of the state sequence satisfies the formulas in the *Next* set, there is a transition that can be taken by the automaton. This argument can be repeated to construct a run of the automaton for $\bar{\rho}$.

The following lemma states that if a timed state sequence is φ-fine and satisfies all formulas in the *Next* set of a location, then there is an edge in the automaton to a new location where the first interval satisfies all *Now* formulas and the tail of the state sequence satisfies all *Next* formulas again.

Lemma 8. *Let $\ell \in L$, $\bar{\rho} = \left(\bar{\sigma}, \bar{I}\right)$ a φ-fine timed state sequence, and ν a timer valuation, such that $\bar{\rho} \models_\nu Next(\ell)$. Then there exists an edge $(\ell, TS, \ell') \in E$ such that for all $0 \leq t < l(\bar{I}(1))$, we have that $\bar{\rho}^t \models_{TS(\nu)-t} Now(\ell')$ and $\bar{\rho}^{l(\bar{I}(1))} \models_{TS(\nu)-l(\bar{I}(1))} Next(\ell')$.*

Proof. Follows from Lemma 4 and the construction of the tableau automaton.

Similarly we can use Lemma 4 to show that if a timed state sequence $\bar{\rho}$ satisfies φ, then there is an appropriate initial extended location to start a run for $\bar{\rho}$.

Lemma 9. *Let $\bar{\rho} = \left(\bar{\sigma}, \bar{I}\right)$ be a φ-fine timed state sequence such that $\bar{\rho} \models \varphi$. Then there is some $(\ell, \nu) \in L_0$ such that for all $0 \leq t < l(\bar{I}(1))$, $\bar{\rho}^t \models_{\nu - t} Now(\ell)$ and $\bar{\rho}^{l(\bar{I}(1))} \models_{\nu - l(\bar{I}(1))} Next(\ell)$.*

From Lemma 9 and repeatedly applying Lemma 8 to construct a run, it follows that A_φ accepts all timed state sequences that satisfy φ.

Lemma 10. *Let $\bar{\rho}$ be a timed state sequence. If $\bar{\rho} \models \varphi$, then A_φ accepts $\bar{\rho}$.*

4.4 Implementation

In order to validate the tableau construction described in this section, a prototype implementation of the algorithm has been made. Table 1 shows a few of the formulas that have been tested and the numbers of locations, edges and timers of the corresponding tableau automata. The size of the automata grows relatively mildly with the size of the formula. As we know of no other implementations, we have been unable to collect comparative results.

Table 1. Numbers of states, transitions and timers of the tableaux of different formulas

Formula	Numb. of states	Numb. of transitions	Numb. of timers
$\neg\Diamond_{\leq 5}p$	4	6	1
$\Box_{\leq 100}\Diamond_{\leq 5}p$	10	22	2
$\Diamond_{\leq 5}(\Box_{\leq 1}p \vee \Box_{\leq 1}q)$	11	21	3
$p\mathsf{U}_{\leq 1}(q\mathsf{U}_{\leq 1}(r\mathsf{U}_{\leq 1}s))$	14	30	3
$p \Rightarrow (\Box_{\leq 5}(q \Rightarrow \Box_{\leq 1}r))$	15	48	2
$(p \Rightarrow \Diamond_{\leq 5}q)\,\mathsf{U}_{\leq 100}\Box_{\leq 5}\neg p$	21	64	3
$(((p\mathsf{U}_{\leq 4}q)\,\mathsf{U}_{\leq 3}r)\,\mathsf{U}_{\leq 2}s)\,\mathsf{U}_{\leq 1}t$	60	271	4

5 Conclusions and Future Work

We have presented an on-the-fly tableau construction for the fragment $\text{MITL}_{\leq}$ of MITL. Technically, this required the introduction of explicit timers and timer set operators into the logic, as well as a Next operator. Within this extended syntax, we could then define equivalence-preserving rewrite rules that syntactically separate any given formula into three parts: the timer settings, the constraints on the current time interval, and the constraints on the future intervals. The resulting normal form procedure is the first ingredient to the tableau construction. The second ingredient is given by two more equivalences that are used to restrict the number of timers introduced in the tableau automaton to one timer per Until and per Release formula.

In [4] it has been shown that the construction of tableaux for $\text{MITL}_{0,\infty}$, a slightly different fragment of MITL, is PSPACE-complete, and this result can be adapted to the case of $\text{MITL}_{\leq}$. Thus, the theoretical worst-case complexity of our construction is the same as that of the construction in [4]. Being on-the-fly, we expect our algorithm to use less memory and to give smaller tableaux in many cases in practice. However, as we know of no other implementations, we have been unable to collect comparative results.

By restricting the interval sequences of models to be left-closed and right-open, and subscripts of (basic) Until formulas to be of the form $\leq d$, we could confine ourselves to timed automata of a restricted form, so that in the tableaux no extra clocks are needed to distinguish between transitions from a right-open into a left-closed interval and transitions from a right-closed into a left-open interval. To demonstrate the construction, we have implemented the algorithm and shown experimental results for several formulas. We are currently extending the results of this paper to also include Until formulas without time bounds, leading to timed automata with acceptance conditions. We also consider an implementation of the presented algorithm in the context of the SHESim platform for specification and simulation ([7]). Another direction for future work is the generalisation of optimisations that have been developed for (on-the-fly) tableau constructions in the untimed case (see e.g. [5, 6, 8]).

References

1. R. Alur. *Techniques for automatic verification of real-time systems.* PhD thesis, Stanford University, 1991.
2. R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and Computation*, 104:2–34, 1993.
3. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
4. R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, January 1996.
5. M. Daniele, F. Giunchiglia, and M. Y. Vardi. Improved automata generation for linear temporal logic. In N. Halbwachs and D. Peled, editors, *Computer Aided Verification: 11th International Conference Proceedings, CAV'99, Trento, Italy, July 6-10, 1999 (LNCS 1633)*, pages 249–260. Springer, 1999.
6. K. Etessami and G. Holzmann. Optimizing Büchi automata. To appear in Proceedings of CONCUR'2000, 2000.
7. M.C.W. Geilen and J.P.M. Voeten. Object-oriented modelling and specification using SHE. In R.C. Backhouse and J.C.M. Baeten, editors, *Proceedings of the First International Symposium on Visual Formal Methods VFM'99*, pages 16–24. Computing Science Reports 99/08 Department of Mathematics and Computer Science, Eindhoven University of Technology, 1999.
8. R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. IFIP/WG6.1 Symp. Protocol Specification Testing and Verification (PSTV95), Warsaw Poland*, pages 3–18. Chapman & Hall, June 1995.
9. T. Henzinger. It's about time: real-time logics reviewed. In D. Sangiorgi and R. de Simone, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR 1998)*, pages 439–454, Berlin, 1998. Springer.
10. T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(1):193–244, June 1994.
11. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 97–107. ACM SIGACT/SIGPLAN, 1985.
12. A. Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.
13. J.-F. Raskin. *Logics, automata and classical theories for deciding real time.* PhD thesis, Facultés Universitaires Notre-Dame de la Paix, Namur (Belgium), June 1999.
14. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Logic in Computer Science*, pages 332–344. IEEE TC-MFC, IEEE Computer Society Press, 1986.