Symbolic Model Checking of Real-Time Systems

G. Logothetis and K. Schneider
University of Karlsruhe
Institute for Computer Design and Fault Tolerance (Prof. Dr.-Ing. D. Schmid)
P.O. Box 6980, 76128 Karlsruhe, Germany
email: {logo, schneide}@informatik.uni-karlsruhe.de
http://goethe.ira.uka.de/fmg

Abstract

We present a new real-time temporal logic for the specification and verification of discrete quantitative temporal properties. This logic is an extension of the well-known logic CTL. Its semantics is defined on discrete time transition systems which are in turn interpreted in an abstract manner instead of the usual stuttering interpretation. Hence, our approach directly supports abstractions of real-time systems by ignoring irrelevant qualitative properties, but without losing any quantitative information. We analyze the complexity of the presented model checking algorithm and furthermore present a fragment of the logic that can be efficiently checked.

1. Introduction

Formal verification methods have been developed to reason about the correctness of a system with respect to a given specification. In particular, model checking [4, 15] of temporal logics has become one of the most successful verification techniques. Using this technique requires to adequately model a system by a finite state transition system so that specifications given in temporal logics [8] can be checked for that model.

In general, model checking procedures suffer from the so-called state explosion problem: The size, i.e., the number of states of the system can exponentially grow with the size of the implementation description. It is therefore often necessary to use abstraction techniques like those given in [6, 12, 14] to neglect irrelevant details so that the verification can concentrate on the necessary facts. As sets of states are thereby collected into abstract states, this means that the number of transitions to reach a certain state from another one is changed. As a consequence, however, information about quantitative time consumption is lost.

Real-time systems must perform certain actions within limited time bounds or should start actions only after some point of time. It is therefore natural to label the transitions of the abstract transition system by numbers that denote the time required to move from one state to another one. In general, there are two possible interpretations of these timed transition systems:

Interpretation $I_1$: A transition from state $s_1$ to state $s_2$ with label $k \in \mathbb{N}$ means that at any time $t_0$, where we are in state $s_1$, we can perform an atomic action that requires $k$ units of time. The action terminates at time $t_0 + k$, where we are in state $s_2$. There is no information about the intermediate points of time $t_0 < t < t_0 + k$.

Interpretation $I_2$: A transition from state $s_1$ to state $s_2$ with label $k > 1$ is seen as abbreviation for a stuttering sequence $s_1 \rightarrow s_1, 1 \rightarrow \ldots \rightarrow s_1, k-1 \rightarrow s_2$ where all the states $s_{1,i}$ have the same variable assignment as state $s_1$.

Clearly, only interpretation $I_1$ can be used in a setting where more powerful abstraction techniques than stuttering simulations are used. It is therefore surprising that none of the previous real-time extensions of CTL is based on interpretation $I_1$, although this is the more general (expressive) one!

The development of discrete real-time extensions of CTL has been initiated in [9], where the temporal operators have been extended by time bounds to limit the number of fix-point iterations required to evaluate the considered temporal expression. The models used in [9] were still traditional finite-state transition systems where each transition requires a single unit of time.

In order to represent real-time systems in a more compact way, [3] introduced timed transition systems, where transitions are labeled by natural numbers that denote the time consumption of the action associated with the transition. The meaning of these timed transitions is in our terms the stuttering interpretation $I_2$.

In [10] a new real-time temporal logic was introduced, where both interpretations $I_1$ and $I_2$ were unfortunately mixed: Different temporal operators of this logic interpret...
transitions differently, i.e., either according to $I_1$ or $I_2$. As the meaning of timed transitions therefore depends on the context, it is impossible to reason about the meaning of timed transitions. In particular, it is not possible to define composition of structures.

Becoming aware of the misleading semantics of [10] (see page 11 of [16]), another temporal logic that is solely based on interpretation $I_2$ has been presented in [16]. In principle, the logic of [16] is defined on unit delay transition systems that are obtained by expanding (cf. definition 6) a given timed transition system. However, there are different expansion algorithms that yield in different transition systems that are not bisimilar to each other. Hence, the verification results obtained for an expanded structure cannot be easily transferred back to the timed transition system.

To summarize, the mentioned real-time extensions of CTL have the following drawbacks:

- None of the previous real-time extensions of CTL is based on interpretation $I_1$ that is necessary to benefit from abstraction techniques.
- None of the previous real-time extensions of CTL has a time bounded next state operator to express facts about actions that correspond with a single transition. As a consequence, facts as the following one can not be expressed: "Is there a non-stop flight from New York to Paris with a duration of at most 9 hours?"
- The problem to compute the set of states of a timed transition system $\mathcal{K}$ where a real-time CTL formula holds, can not be easily translated to an equivalent CTL model checking problem on a unit delay structure: Expansions of timed transition systems may yield in different results [13], and furthermore different expansions are not equivalent to each other [13].

Hence, there has been much confusion and misconception about the definition of a sound and reasonable real-time extension of the famous temporal logic CTL. All mentioned previous approaches [3, 10, 16] are – from a logical perspective – questionable.

In [13], we have presented a real-time extension JCTL of CTL that is based on interpreting timed transition systems with interpretation $I_1$. This directly supports abstractions of real-time systems by ignoring their irrelevant qualitative properties, but without losing their quantitative ones. For example, we can model processes that compute some values within a certain limit of time with a single transition, that does not state anything about the values of the variables during the computation. Moreover, JCTL has a next-state operator equipped with time bounds, so that one can reason about real-time constraints of atomic actions.

In this paper, we reconsider the logic JCTL and its model checking algorithm, and analyse its complexity. It turns out that the complexity to compute the set of states where a given JCTL formula holds is the same as for previous approaches like [10]. However, we additionally present a non-trivial fragment JCTL$^L$ of JCTL that can be more efficiently checked. In particular, less fixpoints need to be evaluated in this fragment, and the model checking algorithm can ignore the time bounds of the transitions for many of its computations.

This paper is organized as follows: In the next section, we define our version of timed transition systems and our real-time temporal logic JCTL. We will then proceed with the definition of a symbolic model checking procedure for JCTL. After this, we analyze its complexity, and present the fragment JCTL$^L$ of JCTL that can be checked more efficiently.

Though not closely related to this paper, we want to finally mention that beneath the real-time extensions of CTL that are defined on discrete time models, there are also very successful approaches that are based on a continuous model of time [1, 11, 7, 2]. These approaches usually rely on timed automata, i.e., on finite state automata that are endowed by a finite set of real-valued clocks. Most verification procedures based on timed automata require the construction of a so-called region graph to reduce the infinite state space of timed automata to a finite state problem. However, the construction of the region graph is very expensive in practice. Approaches for discrete time models do not suffer from the need of such a construction as they directly use finite state transition systems to model the systems.

2. Syntax and Semantics of the Logic JCTL

2.1. Timed Kripke Structures

We consider systems modeled as timed Kripke structures\footnote{Timed transition systems have been introduced by many authors with different names like timed transitions graphs [3], quantitative temporal structures [10], or timed temporal structures in [16]. Following the CTL notations, we prefer the name timed Kripke structures.} over some set of variables $\mathcal{V}$. These timed Kripke structures are formally defined as follows:

**Definition 1 (Timed Kripke Structures (TKS))** A timed Kripke structure over the variables $\mathcal{V}$ is a tuple $(\mathcal{I}, S, \mathcal{R}, \mathcal{L})$, such that $S$ is a finite set of states, $\mathcal{I} \subseteq S$ is the set of initial states, and $\mathcal{R} \subseteq S \times \mathcal{N} \times S$ is the set of transitions. For any state $s \in S$, the set $\mathcal{L}(s) \subseteq \mathcal{V}$ is the set of variables that hold on $s$. We furthermore demand that for any $(s, t, s') \in \mathcal{R}$, we have $t > 0$ and that for any $s \in S$, there must be a $t \in \mathcal{N}$ and a $s' \in S$ such that $(s, t, s') \in \mathcal{R}$ holds.

Timed Kripke structures may be pictorially drawn as given in Figure 1, where initial states are drawn with double lines.
Some approaches, e.g. [3] label transitions with intervals \([a, b]\) of time. It is easily seen that our TKSs subsume these models since we can add for any \(t \in [a, b]\) a new transition between the considered two states.

It is crucial to understand what is modeled by a TKS. We use interpretation \(I_1\): A transition from state \(s\) to state \(s'\) with label \(k \in \mathbb{N}\) means that at any time \(t_0\) where we are in state \(s\), we can perform an atomic action that requires \(k\) units of time. The action terminates at time \(t_0 + k\), where we are in state \(s'\). In particular, there is no information about the intermediate points of time \(t\) with \(t_0 < t < t_0 + k\).

### 2.2. JCTL as a Real-Time Extension of CTL

To write down specifications in a formal way, we use JCTL, a real-time extension of the temporal logic CTL. For its definition below, we only use a small subset of logical operators that will be extended below by some abbreviations.

**Definition 2 (Syntax of JCTL)** Given a set of variables \(V\), the set of JCTL formulas is the least set satisfying the following rules, where \(\varphi\) and \(\psi\) denote arbitrary CTL formulas, and \(a, b \in \mathbb{N}\) are arbitrary natural numbers:

- \(V \subseteq \text{JCTL}\), i.e., any variable is a JCTL formula
- \(\neg \varphi, \varphi \land \psi \in \text{JCTL}\)
- \(\text{EX}^{[a,b]} \varphi \in \text{JCTL}\)
- \(\text{EX}^{>a} \varphi \in \text{JCTL}\)
- \(\text{E}[\varphi \cup^{[a,b]} \psi] \in \text{JCTL}\)
- \(\text{E}[\varphi \cup^{>a} \psi] \in \text{JCTL}\)
- \(\text{EG}^{[a,b]} \varphi \in \text{JCTL}\)
- \(\text{EG}^{>a} \varphi \in \text{JCTL}\)

The semantics of JCTL is defined with respect to a TKS. For the definition of the semantics, we need the notion of paths. A path \(\pi\) through a timed Kripke structure is a function \(\pi : \mathbb{N} \to S\) such that \(\forall i \in \mathbb{N}. \exists t \in \mathbb{N}. (\pi(i), t, \pi(i+1)) \in \mathcal{R}\) holds (we write the function application with a superscript). Hence, \(\pi(i)\) is the \(i^{th}\) state on path \(\pi\). For a given path \(\pi\), we define an associated time consumption function \(\tau_\pi\), so that \(\pi\) and \(\tau_\pi\) satisfy the condition \(\forall i \in \mathbb{N}. \exists t \in \mathbb{N}. (\pi(i), \tau_\pi(i), \pi(i+1)) \in \mathcal{R}\). Note that \(\tau_\pi\) is not uniquely defined for a fixed path \(\pi\), since we may have more than one transition between two states that are labeled with different numbers. The set of paths starting in a state \(s\) is furthermore denoted as \(\text{Paths}_\mathcal{K}(s)\).

**Definition 3 (Semantics of JCTL)** Given a TKS \(\mathcal{K} = (I, S, \mathcal{R}, \mathcal{L})\), and \(s \in S\), then the semantics of the logic is recursively defined as follows:

- \(\mathcal{K}, s \models p\) iff \(p \in \mathcal{L}(s)\) for any \(p \in \mathcal{V}\)
- \(\mathcal{K}, s \models \neg \varphi\) iff \(\mathcal{K}, s \models \varphi\) and \(\mathcal{K}, s \models \psi\)
- \(\mathcal{K}, s \models \text{EX}^{[a,b]} \varphi\) iff there is a path \(\pi \in \text{Paths}_\mathcal{K}(s)\) with associated duration function \(\tau_\pi\) with
  \[\left( a \leq \tau_\pi(0) \leq b \right) \land \left( \mathcal{K}, \pi(1) \models \varphi \right)\]
- \(\mathcal{K}, s \models \text{EX}^{>a} \varphi\) iff there is a path \(\pi \in \text{Paths}_\mathcal{K}(s)\) with associated duration function \(\tau_\pi\) and an \(i \in \mathbb{N}\) with
  \[\left( a \leq \sum_{j=0}^{i-1} \tau_\pi(j) \leq b \right) \land \left( \mathcal{K}, \pi(i) \models \psi \right) \land \left( \forall j < i. \mathcal{K}, \pi(j) \models \varphi \right)\]
- \(\mathcal{K}, s \models \text{EX}[\varphi \cup^{[a,b]} \psi]\) iff there is a path \(\pi \in \text{Paths}_\mathcal{K}(s)\) with associated duration function \(\tau_\pi\) and an \(i \in \mathbb{N}\) with
  \[\left( a \leq \sum_{j=0}^{i-1} \tau_\pi(j) \leq b \right) \land \left( \mathcal{K}, \pi(i) \models \psi \right) \land \left( \forall j < i. \mathcal{K}, \pi(j) \models \varphi \right)\]
- \(\mathcal{K}, s \models \text{EG}^{[a,b]} \varphi\) iff there is a path \(\pi \in \text{Paths}_\mathcal{K}(s)\) with associated duration function \(\tau_\pi\), such that for all \(i \in \mathbb{N}\), we have
  \[\left( a \leq \sum_{j=0}^{i-1} \tau_\pi(j) \leq b \right) \to \left( \mathcal{K}, \pi(i) \models \varphi \right)\]
- \(\mathcal{K}, s \models \text{EG}^{>a} \varphi\) iff there is a path \(\pi \in \text{Paths}_\mathcal{K}(s)\) with associated duration function \(\tau_\pi\), such that for all \(i \in \mathbb{N}\), we have
  \[\left( a \leq \sum_{j=0}^{i-1} \tau_\pi(j) \leq b \right) \to \left( \mathcal{K}, \pi(i) \models \varphi \right)\]

![Figure 1. A Timed Kripke Structure](image-url)
Given a TKS $\mathcal{K}$ and a JCTL formula $\varphi$, we denote the set of states of $\mathcal{K}$ where $\varphi$ holds as $[\varphi]_{\mathcal{K}}$.

Intuitively, $\mathcal{K}, s \models \text{EX}^{[\alpha, \beta]} \varphi$ means that the state $s$ has a direct successor state $s'$ that satisfies $\varphi$ and can be reached in time $t \in [a, b]$. $\mathcal{K}, s \models \text{EX}^{a} \varphi$ means that the state $s$ has a direct successor state $s'$ that satisfies $\varphi$ and can be reached in time $t \geq a$.

$\mathcal{K}, s \models \text{E}[\varphi U_{\alpha, \beta} \psi]$ means that there is a path $\pi$ starting in $\pi(0) = s$ and a number $i \in \mathbb{N}$ so that for the first $i$ states $\pi(0), \pi(1), \ldots, \pi(i-1)$ the property $\varphi$ holds, and $\psi$ holds on $\pi(i)$, and the time $t := \sum_{j=0}^{i-1} \tau_{\pi(j)}$ required to reach state $\pi(i)$ satisfies the numerical relation $a \leq t$ and $t \leq b$.

$\mathcal{K}, s \models \text{EG}^{[\alpha, \beta]} \varphi$ means that there is a path $\pi$ starting in $\pi(0) = s$ and a number $i \in \mathbb{N}$ so that for the first $i$ states $\pi(0), \pi(1), \ldots, \pi(i)$ the property $\varphi$ holds, and $\psi$ holds on $\pi(i)$, and the time $t := \sum_{j=0}^{i-1} \tau_{\pi(j)}$ required to reach state $\pi(i)$ satisfies the numerical relation $a \leq t$.

$\mathcal{K}, s \models \text{EG}^{\alpha} \varphi$ means that there is a path $\pi$ starting in $\pi(0) = s$, such that for any state $\pi(i)$ that is reached within a time $t := \sum_{j=0}^{i-1} \tau_{\pi(j)}$ with $t \in [a, b]$, we have $\pi(i)$. Hence, $\varphi$ holds in the interval $[a, b]$.

Finally, $\mathcal{K}, s \models \text{EG}^{[\alpha, \beta]} \varphi$ means that there is a path $\pi$ starting in $\pi(0) = s$, such that for any state $\pi(i)$ that is reached within a time $t := \sum_{j=0}^{i-1} \tau_{\pi(j)}$ with $t \geq a$, we have $\pi(i)$. Hence, $\varphi$ holds for all states on $\pi$ that are reached at time $a$ or after time $a$.

In the above definition, we have only used basic operators of the logic. Of course, we must introduce some further operators to express some properties directly. For this reason, we give the following abbreviations:

**Definition 4 (Further Temporal Operators)** We define further temporal operators in JCTL as follows, where $p$ is an arbitrary variable:

- **Boolean Operators:**
  - $1 := p \lor \neg p$
  - $0 := p \land \neg p$
  - $\varphi \lor \psi := \neg \neg (\neg \varphi \land \neg \psi)$
  - $\neg \varphi := \neg \varphi \lor \psi$

- **EX Operators:**
  - $\text{EX}^{0} \varphi := \text{EX}^{[0, \beta]} \varphi$
  - $\text{EX}^{a} \varphi := \text{EX}^{[0, a]} \varphi$
  - $\text{EX}^{< a} \varphi := \text{EX}^{[0, a-1]} \varphi$
  - $\text{EX}^{a} \varphi := \text{EX}^{[a, \beta]} \varphi$
  - $\text{EX}^{< a} \varphi := \text{EX}^{[a-1, \beta]} \varphi$

- **E[· U ·] Operators:**
  - $\text{E}[\varphi U_{\alpha} \psi] := \text{E}[\varphi U^{\alpha} \psi]$
  - $\text{E}[\varphi U^{< a} \psi] := \text{E}[\varphi U^{[0, a]} \psi]$
  - $\text{E}[\varphi U^{< a} \psi] := \text{E}[\varphi U^{[0, a-1]} \psi]$

- **EG Operators:**
  - $\text{EG}^{\alpha} \varphi := \text{EG}^{[\alpha, \beta]} \varphi$
  - $\text{EG}^{< a} \varphi := \text{EG}^{[0, \beta]} \varphi$
  - $\text{EG}^{< a} \varphi := \text{EG}^{[0, a-1]} \varphi$
  - $\text{EG}^{a} \varphi := \text{EG}^{[a, \beta]} \varphi$
  - $\text{EG}^{< a} \varphi := \text{EG}^{[a-1, \beta]} \varphi$

- Let $\kappa$ be any time constraint, i.e., $[a, b]$, then we define the following operators:
  - $\text{AX}^{\alpha} \varphi := \neg \text{EX}^{\alpha} \neg \varphi$
  - $\text{A}[\varphi U^{a} \psi] := \neg \text{E}[\neg (\neg \varphi U^{a} \psi)] \lor \neg \text{EG}^{a} \neg \psi$
  - $\text{AF}^{\alpha} \varphi := \text{E}[\varphi U^{a} \psi]$
  - $\text{AG}^{< a} \varphi := \neg \text{E}[\neg \varphi U^{a} \psi]$
  - $\text{E}[\varphi U^{a} \psi] := \text{E}[\neg \varphi U^{a} \psi]$
  - $\text{E}[\varphi U^{< a} \psi] := \text{E}[\neg \varphi U^{[0, a-1]} \psi]$

The definitions of the further EX, E[· U ·], and EX operators should be clear. $\text{EG}^{\alpha} \varphi$ holds in state $s$ iff a state can be reached where $\varphi$ holds and that state can be reached within a time that satisfies the time constraint $\kappa$.

$\text{E}[\varphi U^{a} \psi]$ means that there must be a path $\pi$ and a number $i \in \mathbb{N}$ such that $\varphi$ holds on $\pi(i)$, and all states $\pi(j)$ with $j \leq i$ do not satisfy $\psi$, and the time required to reach $\pi(i)$ satisfies the time constraint $\kappa$. $\text{E}[\cdot U \cdot]$ is defined in a similar way. Finally, the versions with the $\chi$ path quantifier are defined such that the corresponding path property must hold for all paths leaving the state.

As can be seen, the $\text{EG}^{[\alpha, \beta]} \varphi$ operator states that some property holds for all states that can be reached within $[a, b]$, while $\text{E}[\cdot U \cdot]$ states a property for some point of time in that interval. Nevertheless, the following lemma holds, which shows that the $\text{EG}^{< a} \varphi$ operator makes both a universal and an existential statement (compare $G_{2}$ and $G_{3}$ in the following lemma). This is due to the fact, that the equation $\text{EG}^{< a} \varphi = \text{E}[\varphi U^{< a} 1]$ is valid:

**Lemma 1 (Semantics of $\text{EG}^{< a} \varphi$)** Given a JCTL formula $\varphi$ and a number $k \in \mathbb{N}$. Then, the following properties are equivalent for any TKS $\mathcal{K} = (I, S, R, L)$:
Definition 5

(G1) \( \mathcal{K}, s \models \text{EG}^k \varphi \)

(G2) there is a path \( \pi \in \text{Path}_{\mathcal{K}}(s) \) starting in state \( s \), such that for all numbers \( i \in \mathbb{N} \), we have

\[
\left( \sum_{j=0}^{i-1} \pi_{s(j)} \leq k \right) \rightarrow \left( \mathcal{K}, \pi_{s(i)} \models \varphi \right)
\]

(G3) there is a path \( \pi \in \text{Path}_{\mathcal{K}}(s) \) starting in state \( s \) and a number \( i \in \mathbb{N} \) such that

\[
\left( \sum_{j=0}^{i-1} \pi_{s(j)} \leq k < \sum_{j=0}^{i} \pi_{s(j)} \right) \land \left( \forall j \leq i, \mathcal{K}, \pi_{s(j)} \models \varphi \right)
\]

(G4) there is a path \( \pi \in \text{Path}_{\mathcal{K}}(s) \) starting in state \( s \) and a number \( i \in \mathbb{N} \) such that

\[
\left( \sum_{j=0}^{i-1} \pi_{s(j)} > k \right) \land \left( \forall j < i, \mathcal{K}, \pi_{s(j)} \models \varphi \right)
\]

(G5) \( \mathcal{K}, s \models \text{E}[\varphi \cup \text{F}^k 1] \)

The proof of the above lemma is not very difficult. We just make use of the well-ordering of natural numbers, i.e., if there is a number with some property, then there is also a least number with the same property.

2.3. Expansion of TKSs

Normal Kripke structures are special cases of TKS that are obtained by restricting TKSs so that \( (s, t, s') \in \mathcal{R} \) implies \( t = 1 \). To avoid confusion, we call the ‘normal Kripke structures’ unit delay structures in the following:

Definition 5 (Unit Delay Structure (UDS)) A TKS \( \mathcal{K} = (I, S, \mathcal{R}, \mathcal{L}) \) is called to be unit delay structure if \( I_\mathcal{K} := \{ t | (s, t, s') \in \mathcal{R} \} = \{1\} \).

The previously mentioned approaches to define real-time logics that rely on interpretation \( I_2 \) are forced to expand TKSs in order to define the semantics of their logics. For example, the following expansion algorithm could be used for that purpose:

Definition 6 (Expansion of TKS) Given a TKS \( \mathcal{K} = (I, S, \mathcal{R}, \mathcal{L}) \), we compute \( (I_\mathcal{K}, S_\mathcal{K}, \mathcal{R}_\mathcal{K}) = \text{expand}(I, S, \mathcal{R}) \) with the function expand as defined in Figure 2. Moreover, we define for any \( (s, u) \in S_\mathcal{K} \), the label function \( \mathcal{L}_\mathcal{K}((s, u)) := \mathcal{L}(s) \). The expansion of \( \mathcal{K} \) is then the unit delay structure \( \mathcal{K}_\mathcal{K} := (I_\mathcal{K}, S_\mathcal{K}, \mathcal{R}_\mathcal{K}, \mathcal{L}_\mathcal{K}) \).

As can be seen, the expansion relies on interpretation \( I_2 \), since we defined \( \mathcal{L}_\mathcal{K}((s, u)) := \mathcal{L}(s) \), i.e., the states of \( \mathcal{K}_\mathcal{K} \) have the same variable assignments as the corresponding states of \( \mathcal{K} \). For conciseness, we use the following definition:

<table>
<thead>
<tr>
<th>Function expand(I, S, R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_\mathcal{K} := { (s, 1)</td>
</tr>
<tr>
<td>( \mathcal{R}_\mathcal{K} := { } );</td>
</tr>
<tr>
<td>for ( (s, t, s') \in R ) do</td>
</tr>
<tr>
<td>for ( i := 2 ) to ( t ) do</td>
</tr>
<tr>
<td>( S_\mathcal{K} := S_\mathcal{K} \cup { (s, i) } );</td>
</tr>
<tr>
<td>( \mathcal{R}<em>\mathcal{K} := \mathcal{R}</em>\mathcal{K} \cup { ((s, i-1), (s, i)) } );</td>
</tr>
<tr>
<td>end for;</td>
</tr>
<tr>
<td>( \mathcal{R}<em>\mathcal{K} := \mathcal{R}</em>\mathcal{K} \cup { ((s, t), (s', 1)) } );</td>
</tr>
<tr>
<td>end for;</td>
</tr>
<tr>
<td>( I_\mathcal{K} := { (s, 1)</td>
</tr>
<tr>
<td>return ( (I_\mathcal{K}, S_\mathcal{K}, \mathcal{R}_\mathcal{K}) );</td>
</tr>
</tbody>
</table>

Figure 2. Expansion of TKS to UDS

Definition 7 (Tracks of a State) Given a TKS \( \mathcal{K} = (I, S, \mathcal{R}, \mathcal{L}) \), its expanded structure \( \mathcal{K}_\mathcal{K} := (I_\mathcal{K}, S_\mathcal{K}, \mathcal{R}_\mathcal{K}, \mathcal{L}_\mathcal{K}) \), and a state \( s \in S \). Then, we define \( \text{Track}_{\mathcal{K}}(s) = \{ (s', u) \in S_\mathcal{K} | (s, s') \in \mathcal{R} \} \).

We emphasize that expansions of TKS can be performed in many different ways that are not equivalent to each other [13]. Furthermore, real-time model checking problems cannot be simply reduced to ordinary CTL model checking problems, although this is widely believed [13]. For this reason, we must extend the usual CTL model checking procedure to capture JCTL. This is shown in the next section.

3. Real Time Model Checking on TKSs

In this section, we present a model checking algorithm for our real-time logic JCTL. We have implemented this algorithm in our verification tool JERRY by using the CUDD BDD library [17]. The underlying algorithms for the basic operators are given in Figure 4.

The essential idea to reason about the real-time constraints is to move fronts of tracks on a virtual expanded structure. However, we emphasize that we do never expand the structure. Furthermore, we do not run into semantic problems since the result of any evaluation of a logical operator is a set of states instead of a set of tracks. Hence, all calculations are independent of the virtual expansion. This is achieved by abstraction of the set of tracks, so that the semantics of the evaluated temporal operator is respected. In general, there are two possibilities: On the one hand, a state \( s \) belongs to the result if its main track \((s, 1)\) belongs to the track set, on the other hand it may be sufficient if anyone of its tracks \((s, t)\) belongs to the track set. The choice between the two possibilities depends on the semantics of the considered temporal operator (see Figure 4).

The key function for the evaluation of all real-time constraints is the function MoveFront. Given a set of tracks
$T_\psi$ and a set of states $S_\psi$, this function computes the set of tracks that have a path of a certain length through the tracks $S_\psi \times N$. The precise specification is as follows:

**Lemma 2 (Correctness of MoveFront)** Given a TKS $K = (I, S, R, L)$, a set of states $S_\psi$, and a set of tracks $T_\psi$, the function MoveFront as given in Figure 4 satisfies the following equations for $\sim \in \{=, \geq\}$ (cf. Figure 3):

$$(s_0, t) \in \text{MoveFront}(\sim, k, S_\psi, T_\psi) \iff \exists s_1, \ldots, s_{n-1} \in S_\psi, (s_n, d) \in T_\psi,$$

where $n = \#\text{preTracks}(\text{MoveFront}(=, k, S_\psi, T_\psi))$.

Hence, MoveFront($\sim, k, S_\psi, T_\psi$) computes the set of tracks that have a path through any expanded structure of length $\ell$ with $\ell \sim k$ to a track in $T_\psi$ which runs only through tracks of $S_\psi \times N$.

**Proof:** The correctness easily follows by induction on $k$, when we observe that our algorithm and the right hand sides of the above equivalence both satisfy the following recursion equations (note that primitive recursive definitions are uniquely determined):

- MoveFront($\sim, 0, S_\psi, T_\psi$) = $T_\psi$
- MoveFront($\sim, k + 1, S_\psi, T_\psi$) = $(S_\psi \times N) \cap \text{preTracks}(\text{MoveFront($\sim, k, S_\psi, T_\psi$)})$
- MoveFront($\sim, k, S_\psi, T_\psi$) = $\left\{ (t_0, \ldots, t_{n-1}) \in T_\psi \cup ((S_\psi \times N) \cap \text{preTracks}(T_\psi)) \right\}$

Using these equations, we can easily prove that $T_0 = \text{MoveFront($\sim, i, S_\psi, T_\psi$)}$ is an invariant of the loop in the algorithm given in Figure 4 for MoveFront. This directly implies the correctness of the above lemma.

Please note that in the algorithm given in Figure 4 to implement the function MoveFront, we use $(i \neq k) \land (T_0 \neq T_1)$ as loop condition instead of $(i \neq k)$ (which would also be correct). The reason for this is that whenever $T_0 = T_1$ holds for an iteration $i < k$, then it follows that all fronts MoveFront($\sim, i, S_\psi, T_\psi$), ..., MoveFront($\sim, k, S_\psi, T_\psi$) would be identical, so that we already have the result in this case.

Now consider the function $\text{StatesEU}^{[a,b]}$. We first compute the set of tracks $T_0$ that can reach a track of $S_\psi \times \{1\}$ in exactly $a$ steps (where only tracks of the states $S_\psi$ are traversed). After this, we move the tracks $T_0$ by further $b - a$ unit delay steps through the tracks of the states $S_\psi$. By the above lemma, we then obtain

$$(s'_0, t') \in T_1 \iff$$

$$(\exists s'_1, \ldots, s'_m \in S_\psi, \exists \delta'_0, \ldots, \delta'_{m-1} \in \mathbb{N},)$$

$$\exists s_0, \ldots, s_{n-1} \in S_\psi, \exists \delta_0, \ldots, \delta_{n-1} \in \mathbb{N},$$

$$\exists s_n \in S_\psi, \exists \delta \in \mathbb{N},$$

$$(\forall \delta'_{m-1} (s'_1, t'_1, s'_{m-1}) \in R \land)$$

$$(s'_m = s_0) \in R \land)$$

$$(\forall \delta'_{m-1} (s'_1, t'_1, s'_{m-1}) \in R \land)$$

$$a = (\sum_{i=0}^{m-1} \delta'_i) + 1 - \delta \land)$$

$$b \geq (\sum_{i=0}^{m-1} \delta'_i) + (\sum_{i=0}^{n-1} \delta_i) + 1 - \delta \land)$$

$$t \leq \delta \land t' \leq \delta \land)$$

Note that $(\sum_{i=0}^{m-1} \delta'_i) + (\sum_{i=0}^{n-1} \delta_i) + 1 - \delta \land t \leq \delta \land t' \leq \delta \land)$$

is the number of unit delay steps that are required to reach track $(s_n, 1)$ in $S_\psi \times \{1\}$ from track $(s'_0, t')$. By the above result, it follows that this time is in the interval $[a, b]$ (consider the case $m = 0$, where the second MoveFront went along a single transition, and the case where $m > 0$ holds). The final step is to translate this result (given for tracks) to sets of states. If $(s'_0, 1) \in T_1$ holds, then we clearly see that state $s_0$ belongs to $E(\varphi \cup \{[a,b] \varphi\})$. If, on the other hand, $(s'_0, t') \in T_1$ holds for some $t' > 1$, and $(s'_0, 1)$ is not included in $T_1$, then it follows by the above formula that the time to reach $(s_n, 1)$ from $(s'_0, 1)$ is larger than $b$, so that $s'_0 \notin E(\varphi \cup \{[a,b] \varphi\})$.

The correctness of $EG^{[a,b]} \varphi$ is seen as follows: $T_0$ is the set of tracks that have a path of length $b - a$ through $S_\psi \times N$ in the expanded structure. $T_1$ is the set of tracks of $S_\psi \times N$ that can reach $T_0$ while only traversing tracks of $S_\psi \times N$. Hence, $T_1$ is the set of tracks that have a path of length $\geq b - a$ through $S_\psi \times N$ in the expanded structure. Finally, $T_2$ is the set of tracks of $S \times N$ that can reach $T_1$ in exactly $a$ steps. Clearly, we only should consider those tracks with time stamp 1 of $T_2$.

The correctness of the other two functions to evaluate $E(\varphi \cup \{[a,b] \varphi\})$ and $EG^{\geq a} \varphi$ are proved in a similar way. Therefore, we obtain the following correctness result:

**Theorem 1 (Correctness of Function States)** For any TKS $K = (I, S, R, L)$ and any JCTL formula $\varphi$, the function $\text{States}$ given in Figure 4 satisfies the following equations for $\varphi$ and any state $s$ of $K$.

Figure 3. Correctness of MoveFront
Figure 4. Model Checking of JCTL Formulas on a Timed Kripke Structure $\mathcal{K} = (I, S, R, L)$
4. Complexity of JCTL

In this section, we analyze the complexity of the presented model checking algorithm for JCTL. For this reason, we first note that there is a model checking procedure for CTL, e.g., the one given in [5], that runs in time $O(|\phi|(|R|+|S|))$. This procedure is able to evaluate any CTL operator in time $O(|\phi|(|R|+|S|))$.

It is easily seen that if we expand a TKS to a UDS, the number of states and transition is multiplied with the maximum delay time $\tau_K$ that appears in $K$. However, the runtime of the JCTL model checking procedure is not in $O(\tau_K |\phi|(|R|+|S|))$, since the number of iterations does also depend on the time constraints of the temporal operators that may enforce more than $\tau_K |S|$ iterations. The crucial part of our complexity analysis is the complexity of the MoveFront function. To this end, we note that the following holds:

**Lemma 3 (Complexity of MoveFront (I))** Given a TKS $K = (\mathcal{I}, \mathcal{S}, \mathcal{R}, \mathcal{L})$, a set of states $\mathcal{S}_p$, and a set of tracks $\mathcal{T}_0$. Let moreover be $K_e = (\mathcal{I}_e, \mathcal{S}_e, \mathcal{R}_e, \mathcal{L}_e)$ the expanded structure of $K$ according to definition 6, and $\varphi$ and $\psi$ formulas so that the equations $\mathcal{S}_e \times \mathcal{N} \cap \mathcal{S}_e = [\varphi]_{K_e}$ and $\mathcal{T}_0 = [\psi]_{K_e}$ hold. Then, we have

$$\text{MoveFront}(\varphi, \psi, \mathcal{T}_0) = [\Phi(\varphi, k, \varphi, \psi)]_{K_e},$$

where the formula $\Phi(\varphi, k, \varphi, \psi)$ is recursively defined as follows:

- $\Phi(\varphi, 0, \varphi, \psi) = \psi$
- $\Phi(\varphi, k+1, \varphi, \psi) = \varphi \land \exists \varphi \Phi(\varphi, k, \varphi, \psi)$
- $\Phi(\varphi, k+1, \varphi, \psi) = \left(\begin{array}{l}
  \text{let } y = \Phi(\varphi, k, \varphi, \psi) \\
  \text{end if } y \land \varphi \land \exists \varphi
\end{array}\right)$

Moreover, if common subformulas are shared, it is easily seen that $\Phi(\varphi, k, \varphi, \psi)$ is $O(k)$ holds. Hence, $\text{MoveFront}(\varphi, \psi, \mathcal{T}_0)$ can be computed in time $O(k(|\mathcal{R}_e|+|\mathcal{S}_e|))$, and $\text{MoveFront}(\varphi, \psi, \mathcal{T}_0)$ even in time $O(\min\{k, |\mathcal{S}_e|\}(|\mathcal{R}_e|+|\mathcal{S}_e|))$.

The complexity can also be directly derived from the implementation of MoveFront: Note that the value of $T_0$ monotonically grows in function calls of MoveFront, but not for calls of MoveFront. Therefore, $\text{MoveFront}(\varphi, \psi, \mathcal{T}_0)$ runs in time $O(\min\{k, |\mathcal{S}_e|\}(|\mathcal{R}_e|+|\mathcal{S}_e|))$, but $\text{MoveFront}(\varphi, \psi, \mathcal{T}_0)$ requires time $O(k(|\mathcal{R}_e|+|\mathcal{S}_e|))$. For this reason, we have the following result:

**Theorem 2 (Complexity of JCTL)** For any TKS $K = (\mathcal{I}, \mathcal{S}, \mathcal{R}, \mathcal{L})$ and any JCTL formula $\varphi$, the function $\text{States}$ given in Figure 4 runs in time $O(\tilde{k}_\varphi |\mathcal{R}_e| |\mathcal{S}_e|)$, where $\tilde{k}_\varphi := \max\{t \mid \exists s, s' \in \mathcal{R} \land (s, s', t, t') \in \mathcal{R}\}$ is the maximum delay time of $K$, and $\tilde{k}_\varphi$ is the maximum number used in time constraints in $\varphi$.

The proof can be obtained by induction along the JCTL formulas. The induction steps are thereby obtained by the following facts, where $\text{Time}(f)$ denotes the runtime of function $f$:

1. $\text{Time}(\text{preStates}) \in O(|\mathcal{R}_e|)$
2. $\text{Time}(\text{preTracks}) \in O(\tilde{k}_\varphi |\mathcal{R}_e| |\mathcal{S}_e|)$
3. $\text{Time}(\text{StatesEU}) \in O(k(\tilde{k}_\varphi |\mathcal{R}_e| |\mathcal{S}_e|))$
4. $\text{Time}(\text{StatesEG}) \in O(k(\tilde{k}_\varphi |\mathcal{R}_e| |\mathcal{S}_e|))$

Hence, all operators can be evaluated in time $O(\tilde{k}_\varphi \tilde{k}_\varphi |\mathcal{R}_e| |\mathcal{S}_e|)$. As a formula $\varphi$ may contain $\{\varphi\}$ operators, the above theorem follows.

Hence, one key to define a more efficient fragment of JCTL is to avoid calls of the form MoveFront, since this is not as efficient as MoveFront. Using a specialized algorithm similar to the one given in [5], we can even improve the complexity for computing MoveFront:

**Lemma 4 (Complexity of MoveFront (II))** Given a TKS $K = (\mathcal{I}, \mathcal{S}, \mathcal{R}, \mathcal{L})$ and a set of tracks $\mathcal{T}_0$. Let moreover be $K_e = (\mathcal{I}_e, \mathcal{S}_e, \mathcal{R}_e, \mathcal{L}_e)$ the expanded structure of $K$ according to definition 6, and $\varphi$ and $\psi$ formulas so that the equations $\mathcal{S}_e \times \mathcal{N} \cap \mathcal{S}_e = [\varphi]_{K_e}$ and $\mathcal{T}_0 = [\psi]_{K_e}$ hold. Then, there is an algorithm to compute $\text{MoveFront}(\varphi, \psi, \mathcal{T}_0)$ in time $O(|\mathcal{R}_e| + |\mathcal{S}_e|)$.

The specialized algorithm is similar to the one given in [5], but it needs to additionally take care of chains of the paths that have reached a certain track. For this reason, we maintain for any track $s_i \in \mathcal{S}_e$, a list $\mathcal{T}_i = [s_1, \ldots, s_m]$ such that $(s_i, s_j)$ is a transition in $\mathcal{R}_e$. Any track $s_j$ will be marked with a number $m_j$, later on that is the minimal length of a path to reach the set $\mathcal{T}_i$ from $s_i$. Furthermore, we create lists $L_\ell$ during the computations that contain the tracks that are marked with the number $\ell$. The algorithm performs then the following steps:

**Step 1:** We eliminate all $s_{ij}$ in each list $\mathcal{T}_i$ that does not belong to $\mathcal{T}_\psi$. This can be done in time $O(|\mathcal{R}_e|)$, since we look at each transition once.

**Step 2:** We mark the tracks $\mathcal{T}_0$ with the number 0, and list them in our first list $L_0$. Let $\ell := 0$. This step is performed in time $O(|\mathcal{S}_e|)$, since we look at each track at most once.
Step 3: For each track $s_i$ in $L_e$, and each track $s_{i,j}$ of $T_i$, we mark $s_{i,j}$ with $\ell+1$ if it is not already marked, and put $s_{i,j}$ in list $L_{e+1}$ in this case. We eliminate $s_{i,j}$ from $T_i$. We then increment $\ell$, and repeat this step until $\ell = k$ holds or no transitions are left in the lists $T_i$.

Clearly, the repeated execution of step 3 does look at each model checking based on BDDs. However, in practice, the latter approaches are in general superior. The next step is to define a subset of JCTL that can be computed without calls MoveFront($\cdot$, $K_i, \cdot$). Clearly, we must therefore forbid the interval constraints, and there-
model checking to qualitative fixpoints. This is due to the fact, that we can separate between a quantitative and a qualitative aspect in these formulas. Note again, that the qualitative fixpoint is computed on the abstract structure with an ordinary CTL model checking procedure.

5. Conclusions

We have presented a new real-time temporal logic JCTL that is directly defined on timed Kripke structures using interpretation $I_1$. This logic overcomes some semantic problems of previous real-time extensions of CTL and allows us to use abstractions without loosing quantitative information. We analysed the complexity of the JCTL model checking problem and presented a non-trivial fragment JCTL$^S$ that can be checked more efficiently than JCTL. Our first experimental results are promising and allow to check practical JCTL$^S$ model checking problems in runtimes that are comparable to CTL model checking.

References