On Checking Parallel Real-Time Systems for Linear Duration Invariants

Dang Van Hung∗ and Pham Hong Thai†
The United Nations University
International Institute for Software Technology
P.O.Box 3058, Macau
E-mail: {dvh,pht}@iist.unu.edu

Abstract

In this paper we define timed regular expressions to describe the timed behaviour of parallel real-time systems and consider the problem of checking algorithmically the set of timed behaviours defined by timed regular expressions for a real-time requirement specified by a linear duration invariant. In general, the problem can be solved by using the mixed integer linear programming techniques. We show that in many cases, the problem can be reduced to a finite number of linear programming problems.

1. Introduction

Model checking for real-time system, i.e. to check a real-time system for a real-time requirement algorithmically, has been studied extensively for decades. In general, the problem is very hard, especially for the concurrent and distributed real-time systems. Some model-checking methods for concurrent systems have been proposed (see, i.e. [6]) to avoid state explosion, but they works well for some special cases only.

In order to make the problem easier, many authors have to restrict themselves to a restricted class of systems and a restricted class of real-time requirements. For the class of requirements written as linear duration invariants, i.e. linear constraints on the durations of the system states in the observation time interval, the problem has been considered in [5, 2, 3, 4]. In [5], the authors showed that for finitary timed automata, i.e. timed automata satisfying some restriction conditions, the problem can be solved using the mixed integer linear programming techniques. In [2, 3, 4], the authors show that with some restrictions on the models of real-time systems, the problem can be solved even using only the linear programming techniques. All of these results are interesting, but they are not dealing with the parallelism directly, and hence, could not be applied directly to solve the problem for the parallel and distributed systems.

In this paper, we introduced timed regular expressions to model concurrent and distributed real-time systems. Those systems which are well-structured will have their behaviour described by timed regular expressions. Then, we show that the techniques introduced in [5, 2, 3, 4] can be generalised to solve the problem for this class of real-time systems. Our results in this paper can be summarised as follows. First, we show that for the timed regular expressions in which there is no occurrence of the repetition operator *, the problem can be solved using only the linear programming techniques. Then, we show that in many cases, the problem for the timed regular expressions with occurrence of * can be converted into the problem for the ones without occurrence of *. For the remaining case, we show that the problem can be solved by using the mixed integer linear programming techniques.

The paper is organised as follows. In the next section, we define timed regular expressions and formalise the model-checking problem. The techniques for solving the problem using linear programming are presented in Section 3 and Section 4. Section 5 is devoted to our discussion on the problem for the general case. The last section is the conclusion of the paper.

2. Timed Regular Expressions

In this section, we give a representation of a ‘regular’ class of models of Duration Calculus, which will be taken in this paper as a representation of the behaviour real-time systems. Duration Calculus (DC) was introduced by Zhou Chaochen et al ([1]) as a logics to reason about the states of real-time systems. In DC, states are viewed as boolean functions of the continuous time. The interpretation is that if a state is present (absent) at a time t then its value at t is true, which is denoted by 1 (false, which is denoted by 0, respectively). This enables to define duration of a state
accumulated present time of $T$. Definition 2.1

Let $S$ range over $s, a, v, \ldots$ be a set of states. A DC model represents an observation of the behaviour of states in $S$ in an interval of time. It consists of an interval $[0, T]$ and an interpretation $I$ in the interval $[0, T]$ of the states in $S$, which assigns each state $s$ in $S$ to a boolean function $s_t$ over $[0, T]$ meaning that $s_t(t) = 1$ if and only if state $s$ is present at time $t$ (under interpretation $I$).

A program written in a real-time programming language actually defines a class of DC models over its variables. We consider in this paper a kind of abstract and simple real-time programs represented by so-called Timed Regular Expressions (TRE, for short) defined below. For a TRE $R$, let $\text{state}(R)$ denote the set of states occurring in $R$.

Definition 2.1 TRE are defined recursively by

i. $\epsilon$ is a TRE and $\text{state}(\epsilon) = \emptyset$

ii. For any $s \in S$, for any real numbers $a, b, 0 \leq a \leq b$ (or $b$ may be $\infty$), $(s, [a, b])$ is a TRE and $\text{state}((s, [a, b])) = \{s\}$

iii. If $R_1, R_2$ are TREs, then $R_1^* R_2, R_1 \oplus R_2$ are TREs, and $\text{state}(R_1^* R_2) = \text{state}(R_1) \cup \text{state}(R_2)$, $\text{state}(R_1 \oplus R_2) = \text{state}(R_1) \cup \text{state}(R_2)$

iv. If $R_1, R_2$ are TREs, and $\text{state}(R_1) \cap \text{state}(R_2) = \emptyset$, then $R_1 \otimes R_2$ is a TRE.

A TRE of the form $(s, [a, b])$ is said to be primitive, and for simplicity the primitive of the form $(s, [0, \infty))$ will be written as $s$. The intuitive meaning of a TRE $(s, [a, b])$ is that the state $s$ is present for some time in between $a$ and $b$. As usual, the operator $\sim$ is for sequential composition, the operator $\otimes$ for parallel composition, and $*$ for repetition.

To avoid the heavy use of brackets, we assume the binding order of the operators as follows: the operator $\sim$ binds most tightly, and the operator $\otimes$ binds more tightly than the operator $\oplus$.

As an example, let us take the railroad crossing system [8] and see how TRE can be used to represent the real-time behaviour of the system. We have trains, a railroad crossing monitor, and a gate controller which are subject to the following constraints (see Figure 1).

i. The monitor has four states to express the positions of the train: state $A$ for train approaching beyond 1/2 mile, state $B$ for train approaching within 1/2 mile, state $C$ for train crossing, and state $P$ for train just passed.

ii. The controller has four states to express the positions of the gate: state $U$ for the gate being up, state $MD$ for the gate moving down, state $Dn$ for the gate being down and $MU$ for the gate moving up.

When the system starts, the monitor is in state $A$ and the controller is in state $U$. In state $A$, when the monitor detects a train approaching within 1/2 mile, it enters state $B$, and at the same time if the controller is in state $U$ or state $MU$, it must enter state $MD$. Namely, when the gate is up or is moving up, and detects that the monitor enters state $B$, it must start moving down. When the train enters the crossing, the monitor enters state $C$, and when the train has passed, it enters state $P$. When the monitor changes its state from $C$ to $P$ then at the same time the monitor changes its state from $Dn$ to $MU$. This means, when the gate is down, and detects that the monitor enters state $P$, it begins to move up. In addition, due to the speed of trains and the safety distance between trains, it takes at least a time units for a train to go to the crossing, after entering state $B$, and when a train has passed, a new train could come after at least $b$ time units. That means that the monitor stays in $B$ at least $a$ time units each time and in $P$ at least $b$ time units each time. Furthermore, assume that it takes the gate at most $c$ time units to move down, and hence, the controller stays at $MD$ at most $c$ time units each time, where $c \leq a$. Automata modelling the railroad crossing system are depicted in Figure 1. Intuitively, the parallel behaviour of the system is now can be described by the following TRE RCM:

\[
\begin{align*}
(A \otimes U)^- \\
(((B, [a, \infty]))^- C \otimes (M(D, [0, c])^- Dn)^-) \\
((P, [b, \infty]))^- A \otimes (M(U^- U \oplus MU))\star^- \\
(e \oplus (B \otimes (B, [a, \infty]^- C) \otimes (M(D, [0, c]) \oplus (M, [0, c]))^- Dn) \oplus \\
((P \oplus P^- A) \otimes (MU \oplus MU^- U)))&
\end{align*}
\]

Note that the expression RCM is prefix-closed. It expresses all possible observations about the system from beginning up to a time point: given a time point $t$, we can decide in which state each component of the system can be. Figure 2 represents a behaviour of the system in an interval $[0, t]$; state $A$ is true from time $0$ to time $x_1$, and from time $y_2$ to time $x_3$; state $B$ is true from time $x_1$ to time $y_1$, and from time $x_3$ to time $t$; state $P$ is true from time $x_2$ to time $y_2$; state $U$ is true in the intervals $[0, x_1]$ and $[x_2, x_3]$, state $MD$ is true in the intervals $[x_2, z_1]$ and $[x_3, t]$, etc.

So, each TRE defines a class of DC models, which is defined formally as follows.

Definition 2.2 Let $R$ be a TRE. The class $\mathcal{M}(R)$ of models represented by $R$ is defined by

i. A model $\sigma = (I, [0, T])$ is in $\mathcal{M}(\epsilon)$ iff $T = 0$. 
ii. A model \( \sigma = (\mathcal{I}, [0, T]) \) is in \( \mathcal{M}((s, [a, b])) \) iff \( a \leq T \leq b \) and for all \( t \in [0, T] \) \( s_{z}(t) = 1 \), and for all \( s' \neq s \) \( s'_{z}(t) = 0 \).

iii. A model \( \sigma = (\mathcal{I}, [0, T]) \) is in \( \mathcal{M}(R_{1} \overline{\Sigma} R_{2}) \) iff there are \( 0 \leq T' \leq T \), \( \sigma_{1} = (\mathcal{I}_{1}, [0, T']) \in \mathcal{M}(R_{1}) \), \( \sigma_{2} = (\mathcal{I}_{2}, [0, T - T']) \in \mathcal{M}(R_{2}) \) such that for all \( s \in \text{state}(R_{1}) \cup \text{state}(R_{2}) \), \( s_{z}(t) = s'_{z}(t) \) for all \( t \in [0, T'] \) and \( s_{z}(t - T') = s_{z}(t) \) for all \( t \in [T', T] \), and for all \( s' \notin \text{state}(R_{1}) \cup \text{state}(R_{2}) \), \( s'_{z}(t) = 0 \) for all \( t \in [0, T] \). Then, we defined \( \sigma_{1} \overline{\Sigma} \sigma_{2} \) as \( \sigma \).

iv. A model \( \sigma = (\mathcal{I}, [0, T]) \) is in \( \mathcal{M}(R_{1} \bigoplus R_{2}) \) iff there are \( \sigma_{1} = (\mathcal{I}_{1}, [0, T]) \in \mathcal{M}(R_{1}) \), \( \sigma_{2} = (\mathcal{I}_{2}, [0, T]) \in \mathcal{M}(R_{2}) \) such that for all \( t \in [0, T] \), \( s_{z}(t) = s'_{z}(t) \) for all \( s \in \text{state}(R_{1}) \), and \( s_{z}(t) = s_{z}(t) \) for all \( s \in \text{state}(R_{2}) \), and \( s_{z}(t) = 0 \) for all \( s' \notin \text{state}(R_{1}) \cup \text{state}(R_{2}) \), and then we define \( \sigma_{1} \bigoplus \sigma_{2} \) as \( \sigma \).

v. A model \( \sigma = (\mathcal{I}, [0, T]) \) is in \( \mathcal{M}(R_{1} \otimes R_{2}) \) iff \( \sigma \in \mathcal{M}(R_{1}) \) or \( \sigma \in \mathcal{M}(R_{2}) \).

vi. A model \( \sigma = (\mathcal{I}, [0, T]) \in \mathcal{M}(R^{*}) \) iff there is an integer \( k \geq 0 \) such that \( \sigma \in \mathcal{M}(R^{k}) \), where \( R^{0} \equiv e \), and for \( k > 0 \), \( R^{k} \equiv R^{-k} R^{k} \). Or, equivalently, there are models \( \sigma_{1}, \ldots, \sigma_{k} \in \mathcal{M}(R) \) such that \( \sigma = \sigma_{1} \overline{\Sigma} \ldots \overline{\Sigma} \sigma_{k} \). (It should be noted here that \( \overline{\Sigma} \) is associative.)

Example

- Let \( R_{1} = (s, [1, 5]) \overline{\Sigma} (u, [1, 7]) \). Then \( (\mathcal{I}_{1}, [0, T]) \) is in \( \mathcal{M}(R_{1}) \), where by \( \mathcal{I}_{1}, s \) is interpreted as 1 in the interval \([0, 4]\), as 0 in the interval \([4, 7]\), and \( u \) is interpreted as 0 in the interval \([0, 4]\), as 1 in the interval \([4, 7]\) under \( \mathcal{I}_{1} \).

- Let \( R = R_{1} \bigoplus (v, [3, 10]) \). Then \( (\mathcal{I}, [0, T]) \) is in \( \mathcal{M}(R) \), where under \( \mathcal{I}, s \) and \( u \) are interpreted, as in the model \( (\mathcal{I}_{1}, [0, T]) \) above, and \( v \) is interpreted as the boolean function 1 in the interval \([0, 7]\).

We are interested in this paper in checking a real-time system for a real-time requirement. We restrict ourselves in the requirements that can be written as a linear duration invariant. Given a set of states \( S \), a linear duration invariant over \( S \) is a Duration Calculus formula of the form

\[
b \leq \ell \leq e \Rightarrow \sum_{s \in S} c_{s} f s \leq M,
\]

where \( b, e, c_{s}, M, (b \leq e) \) are fixed real numbers (\( e \) may be \( \infty \)), \( f s \) denotes the duration of state \( s \) over the time interval of the observation, which is the accumulated time that
the system stays in $s$ in that interval. The above linear duration invariant simply says that if the length of the time interval of the observation is in between $b$ and $c$, then the duration of the system states should satisfy the linear constraints $\sum_{s \in S} c_s \int s \leq M$. For instance, the requirement for the railroad crossing monitor could be written as

$$0 \leq t \leq \infty \Rightarrow \int C - \int Dn \leq 0$$

which means that for any time $t$, for any non negative real number $\epsilon$, if we observe the system from time $t$ to time $t + \epsilon$, then the time that trains are crossing should be no more than the time that the gate is closed. This implies that if a train is crossing then the gate should be down.

Formally, a DC model $\sigma = (T, [0, T])$ satisfies a linear duration invariant $b \leq t \leq c \Rightarrow \sum_{s \in S} c_s \int s \leq M$ if and only if when $b \leq T \leq c$ it holds that $\sum_{s \in S} c_s \int s(t) dt \leq M$.

The model-checking problem in this paper is formulated as follows:

Given a TRE $R$ over $S$, given a linear duration invariant $D$ over $S$. Find a procedure to decide whether $\sigma \models D$ for all $\sigma \in \mathcal{M}(R)$ (abbreviation: $R \models D$).

Note that in this paper, we use a TRE $R$ to express the behaviour of a real-time system in a sense that any linear duration invariant $D$ is satisfied by the system for all time observation intervals if and only if it is satisfied by all models defined by $R$. Thus, the $\text{TRE} RCM$ in the Railroad Crossing Monitor example just represents the observations in the intervals from the time that train is approaching and the gate is up. A TRE to express all the observation intervals of the Railroad Crossing Monitor is a little bit more complicated and is not given here for simplicity.

### 3. Checking Finite TREs for Linear Duration Invariants

Let $R$ be a TRE, $\sigma = (T, [0, T])$ be a model in $\mathcal{M}(R)$, and $D$ be a Linear Duration Invariant. For simplicity, let $d_s(\sigma)$ denote the accumulated time (duration) of state $s \in S$ over the interval $[0, T]$ under the interpretation $T$, i.e. $d_s(\sigma) = \int_0^T s(t) dt$, let $d(\sigma)$ denote the length of the interval $[0, T]$, i.e. $T$, and let $\text{inv}(\sigma)$ denote $\sum_{s \in S} c_s d_s(\sigma)$ (i.e. $\sum_{s \in S} c_s \int s$ evaluated over $\sigma$). Hence, $\sigma \models D$ iff $b \leq d(\sigma) \leq c \Rightarrow \text{inv}(\sigma) \leq M$. For example, for the model $\sigma$ in Figure 2, $d_s(\sigma) = x_1 + x_3 - y_2$, $d_{D_n}(\sigma) = x^2 - z_1$, and $d(\sigma) = t$.

**Lemma 3.1** Let $\sigma = (T, [0, T_1]), \sigma_1 = (T_2, [0, T_2]) \in \mathcal{M}(R_2)$ be DC models. Then,

i. if $\sigma = \sigma_1 \triangleright\sigma_2$ then $d_s(\sigma) = d_s(\sigma_1) + d_s(\sigma_2)$ for all $s \in S$, $d(\sigma) = d(\sigma_1) + d(\sigma_2)$ and $\text{inv}(\sigma) = \text{inv}(\sigma_1) + \text{inv}(\sigma_2)$.

ii. if $\sigma = \sigma_1 \otimes\sigma_2$ then $d_s(\sigma) = d_s(\sigma_1) + d_s(\sigma_2)$ for all $s \in S$, $d(\sigma) = d(\sigma_1) + d(\sigma_2)$, and $\text{inv}(\sigma) = \text{inv}(\sigma_1) + \text{inv}(\sigma_2)$.

**Proof.** Let $\sigma = (T, [0, T]), \sigma_1 = (T_1, [0, T_1]),$ and $\sigma_2 = (T_2, [0, T_2])$

i. If $\sigma = \sigma_1 \triangleright\sigma_2$ then, by Definition 2.2, $T = T_1 + T_2$. Hence $d(\sigma) = d(\sigma_1) + d(\sigma_2)$. Furthermore, for all $s \in S, \text{for all } t \in [0, T]$,

$$s_T(t) = \begin{cases} s_T(t) & \text{if } t \in [0, T_1] \\
 s_{T_2}(t - T_1) & \text{if } t \in (T_1, T] 
\end{cases}$$

Thus,

$$d_s(\sigma) = \int_0^T s_T(t) dt = \int_0^{T_1} s_{T_1}(t) dt + \int_{T_1}^T s_{T_2}(t - T_1) dt = d_s(\sigma_1) + \int_0^{T_2} s_{T_2}(t) dt = d_s(\sigma_1) + d_s(\sigma_2).$$

Consequently, $\text{inv}(\sigma) = \text{inv}(\sigma_1) + \text{inv}(\sigma_2)$.

ii. Let $\sigma_1 \in \mathcal{M}(R_1), \sigma_2 \in \mathcal{M}(R_2)$. If $\sigma = \sigma_1 \otimes\sigma_2$ then, by Definition 2.2, $T = T_1 + T_2$. Hence $d(\sigma) = d(\sigma_1) + d(\sigma_2)$.

Also by Definition 2.2, for all $t \in [0, T]$ for all $s \in S$, $s_T(t) = s_{T_1}(t)$ if $s \in \text{state}(R_1)$, $s_T(t) = s_{T_2}(t)$ if $s \in \text{state}(R_2)$, and $s_T(t) = 0$ otherwise. Since $\text{state}(R_1) \cap \text{state}(R_2) = \emptyset$, $d_s(\sigma) = d_s(\sigma_1) + d_s(\sigma_2)$. Consequently, $\text{inv}(\sigma) = \text{inv}(\sigma_1) + \text{inv}(\sigma_2)$ as well.

We will not distinguish the TREs that define the same set of DC models.

**Definition 3.1** For arbitrary TREs $R_1, R_2$, we say $R_1 \equiv R_2$ if $\mathcal{M}(R_1) = \mathcal{M}(R_2)$.

The following theorem follows immediately from Definition 2.2.

**Theorem 3.1** For arbitrary TREs $R, R_1, R_2$

i. $(R_1 \oplus R_2) \sim R \equiv (R_1 \sim R) \oplus (R_2 \sim R)$ and $R \sim (R_1 \oplus R_2) \equiv (R_1 \sim R) \oplus (R_2 \sim R)$

ii. $(R_1 \oplus R_2) \otimes R \equiv (R_1 \otimes R) \oplus (R_2 \otimes R)$ and $R \otimes (R_1 \oplus R_2) \equiv (R_1 \otimes R) \oplus (R_2 \otimes R)$

iii. $(R_1 \oplus R_2)^* \equiv ((R_1^*) \sim (R_2^*))^*$
Theorem 3.1 implies that any TRE $R$ can be written as $R_1 \oplus R_2 \oplus \ldots \oplus R_k$, where each $R_i$ is a TRE in which there is no occurrence of $\oplus$.

In this paper, we are interested in checking a TRE for the linear duration invariant $D$. So, we will not distinguish the TRES that define the same set of models for $D$ and define:

**Definition 3.2** We say that $R_1$ and $R_2$ are $D$-equivalent, denoted by $R_1 \equiv_D R_2$, iff $R_1 \models D$ if and only if $R_2 \models D$.

Of course, if $R_1 \equiv R_2$ then $R_1 \equiv_D R_2$.

**Theorem 3.2** For arbitrary TRES $R_1$, $R_2$

i. $R_1 \sim R_2 \equiv_D R_2 \sim R_1$

ii. $(R_1 \oplus R_2)^* \equiv_D (R_1)^* \oplus (R_2)^*$

iii. $(R_1^* \sim R_2^*)^* \equiv_D (R_1^* \sim R_2^*)$

iv. $(R_1^* \otimes R_2^*)^* \equiv_D (R_1^* \otimes R_2^*)$

**Proof.** The theorem follows immediately from Lemma 3.1.

**Definition 3.3**

- A TRE $R$ in which there is no occurrence of the combinator $\oplus$ is said to be finite. Otherwise, $R$ is said to be infinite.
- A simple TRE is a finite TRE in which there is no occurrence of the combinator $\oplus$.

For example, $((s, [1, 5]) \sim (u, [1, 7])) \otimes ((v_1, [3, 10]) \oplus (v_2, [2, 9]))$ is a finite TRE, and $((s, [1, 5]) \sim (u, [1, 7])) \otimes (v, [3, 10])$ is a simple TRE.

We are going to show that for a finite TRE $R$, $R \models D$ is decidable. By Theorem 3.1 and Definition 3.3, we will only consider the simple TRES because any finite TRE $R$ can be written as $R_1 \oplus \ldots \oplus R_k$, where each $R_i$ is a simple TREs, and $R \models D$ iff for all $i$ $(i \leq k)$, $R_i \models D$.

Let $R$ be a simple TRE. We associate a set of linear constraints $C(R)$, the set of durations $\{d_s(R) | s \in S\}$ and execution time $d(R)$ to $R$ as follows. Let $\text{Var}(C(R))$ denote the set of variables occurring in $C(R)$.

**Definition 3.4**

- Let $R = (s, [a, b])$. Then $C(R) = \{a \leq t \leq b\}$, $d_s(R) = t$, $d(R) = t$ where $t$ is a real variable, and $d_s'(R) = 0$ for all $s' \neq s$.
- Let $R = R_1 \sim R_2$. By renaming the variables if necessary, we can assume that $\text{Var}(C(R_1)) \cap \text{Var}(C(R_2)) = \emptyset$. Then, $C(R) = C(R_1) \cup C(R_2)$, $d_s(R) = d_s'(R_1) + d_s'(R_2)$ and $d(R) = d(R_1) + d(R_2)$.
- Let $R = R_1 \otimes R_2$. Assume that $\text{Var}(C(R_1)) \cap \text{Var}(C(R_2)) = \emptyset$. Then, $C(R) = C(R_1) \cup C(R_2) \cup \{d(R_1) \sim d(R_2)\}$, $d_s(R) = d_s'(R_1) + d_s'(R_2)$ for all $s \in S$, and $d(R) = d(R_1)$.

Let $\text{inv}(R)$ denote $\sum_{s \in S} c_s d_s(R)$.

For instance, let $R$ be

$$(\{(s, [1, 5]) \sim (u, [1, 7])\} \otimes (v, [3, 10])).$$

Then, we can associate to each primitive in $R$ a variable, say to $(s, [1, 5])$ variable $x$, to $(u, [1, 7])$ variable $y$ and to $(v, [3, 10])$ variable $z$. Then, $C(R) = \{1 \leq x \leq 5, 1 \leq y \leq 7, 3 \leq z \leq 10, x + y = z\}$, $d(R) = 2$, $d_s(R) = x$, $d_u(R) = y$, and $d_v(R) = z$.

For a solution $w$ of the system of linear constraints $C(R)$, denote by $d_s(R)(w)$, $d(R)(w)$, and $\text{inv}(R)(w)$ respectively, the value of $d_s(R)$, $d(R)$ and $\text{inv}(R)$ evaluated over $w$. For vectors $w_1 = (t_1, t_2, \ldots, t_p)$ and $w_2 = (u_1, u_2, \ldots, u_q)$, we denote by $(w_1, w_2)$ the vector $(t_1, t_2, \ldots, t_p, u_1, u_2, \ldots, u_q)$, and if $p = q$ we denote by $w_1 + w_2$ the vector $(t_1 + u_1, t_2 + u_2, \ldots, t_p + u_p)$.

Let $D(R)$ denote $\max(\text{inv}(R))$ subject to $C(R) \cup \{b \leq d(R) \leq e\}$.

**Theorem 3.3** For a simple TRE $R$, $R \models D$ iff $D(R) \leq M$.

**Proof.**

First, we prove that for any model $\sigma \in M(R)$, there exists a solution $w$ of $C(R)$ and vice-versa, such that for all $s \in S$, $d_s(\sigma) = d_s(R)(w)$, and $d(\sigma) = d(R)(w)$ (hence, $\text{inv}(\sigma) = \text{inv}(R)(w)$). This means that there is one-to-one corresponding between $M(R)$ and the set of solutions of $C(R)$. The proof goes by induction on the structure of $R$.

- Let $R = (s, [a, b])$. Given $\sigma = (T, [0, T]) \in M(R)$. By Definition 2.2, $a \leq T \leq b$ and for all $t \in [0, T]$, $s_T(t) = 1$. This implies that $d_s(\sigma) = T$ and $d(\sigma) = T$. On the other hand, by Definition 3.4, $C(R)$ contains only one constraint $a \leq t \leq b$. Since $a \leq T \leq b$, so $w = (T)$ is a solution of $C(R)$. Hence, $d_s(\sigma) = d_s(R)(w)$, and $d(\sigma) = d(R)(w)$.

Reversely, given $w = (t_0)$ be a solution of $C(R)$, i.e. $t = t_0$ and $a \leq t_0 \leq b$. Let $\sigma = (T, [0, t_0])$ be a model such that for all $t \in [0, t_0]$, $s_T(t) = 1$, and for all $s' \neq s, s_{s'}(t) = 0$. Then obviously, $\sigma \in M(R)$, and $d_s(\sigma) = d_s(R)(w)$, $d(\sigma) = d(R)(w)$. 

Let $R = R_1 \setminus R_2$, where $\text{state}(R_1) \cap \text{state}(R_2) = \emptyset$. Given $\sigma \in \mathcal{M}(R)$. By Definition 2.2 and by Lemma 3.1, $d_\sigma(w) = d_\sigma(1) + d_\sigma(2)$, $\forall s \in S$ and $d(\sigma) = d(\sigma_1) + d(\sigma_2).

From the inductive hypothesis, we can find solutions $w_1$ of $C(R_1)$ such that for any $s \in S$, $d_\sigma(w_1) = d_\sigma(R_1)(w_1)$ and $d(\sigma) = d(R_1)(w_1)$, $(i = 1, 2)$. Consequently, $d(\sigma_1) = d(\sigma_2)$.

Now, the theorem follows immediately: since any model $\sigma \in \mathcal{M}(R)$ corresponds to a solution $w$ of $C(R)$ and vice-versa, such that $d_\sigma(w) = d_\sigma(1), \forall s \in S$ and $d(\sigma) = d(R)(w)$, which implies $b \leq d(\sigma) \leq e$ if $b \leq d(R)(w) \leq e$ and $\text{inv}(\sigma) \leq M$ iff $\text{inv}(R)(w) \leq M$, we have that $R \models D$ iff $D(R) \leq M$.

From Theorem 3.3, for a simple TRE $R$, checking $R \models D$ can be done by solving the linear programming problem to find $D(R)$ and comparing it to $M$.

Example Let $R = (\{s, [1, 5]\} \setminus [u, [1, 7]]) \otimes [v, [3, 10]]$.

Let $x, y, z$ be variables associated to the primitives $\{s, [1, 5]\}, [u, [1, 7]], [v, [3, 10]]$ respectively. $R \models D$ can be checked by solving the linear programming problem

$$\max \{2x - z\} \text{ subject to } \begin{cases} 1 \leq x \leq 5 \\ 1 \leq y \leq 7 \\ 3 \leq z \leq 10 \\ z = x + y \\ 4 \leq z \leq 8 \end{cases}$$

and checking whether it is less than 5. It is easy to see that the solution of the linear programming problem is $x = 5, y = 1, z = 6$ and the maximal value of the objective function is 4, which is less than 5.

Let $R$ be an infinite TRE. By replacing each occurrence of the operator $*$ (repetition) with an integer variable $k_i$, we obtain a finite TRE and can associate a finite number of linear programming problems to it. However, because the set of values of $k_i$'s is infinite, the number of linear programming problems is also infinite. It is therefore impossible to solve all of these problems.

In the following sections, we will introduce a technique to reduce an infinite TRE to a finite TRE which is $D$-equivalent to it, and therefore an infinite TRE could be checked for $D$.

4. Reducing TREs to finite TREs

From now on we assume that any primitive occurring in TRE $R$ is not of the form $(s, [0, 0])$ because removing the primitives of the form $(s, [0, 0])$ from $R$ does not change the set $\mathcal{M}(R)$. 

- Let $R = R_1 \setminus R_2$. Given $\sigma \in \mathcal{M}(R)$. By Definition 2.2, $\sigma = \sigma_1 \otimes \sigma_2$ with $\sigma_i \in \mathcal{M}(R_i)$ $(i = 1, 2)$ and by Lemma 3.1, $d_\sigma(w) = d_\sigma(1) + d_\sigma(2), \forall s \in S$ and $d(\sigma) = d(\sigma_1) + d(\sigma_2)$.

- Reversely, let $w$ be a solution of $C(R)$. Since $C(R) = C(R_1) \cup C(R_2)$ and $\text{Var}(C(R_1)) \cap \text{Var}(C(R_2)) = \emptyset$, we can partition $w$ to $w_1$ and $w_2$, i.e., $w = (w_1, w_2)$, where $w_i$ is a solution of $C(R_i)$ $(i = 1, 2)$ for which $d_\sigma(w_1) = d_\sigma(1)(w_1)$ and $d(\sigma) = d(R_1)(w_1)$. Hence, for any $s \in S$, $d_\sigma(R)(w) = d_\sigma(R_1)(w_1) + d_\sigma(R_2)(w_2)$ and $d(R)(w) = d(R_1)(w_1) + d(R_2)(w_2)$. By the inductive hypothesis, there are models $\sigma_i \in \mathcal{M}(R_i)$, such that for any $s \in S$, $d_\sigma(\sigma_i) = d_\sigma(R_i)(w_i)$ and $d(\sigma) = d(R)(w)$, $(i = 1, 2)$.
Let $R$, $R'$ be TRE’s. If there is an occurrence of $R'$ in $R$, then $R'$ is called sub-expression of $R$. For example, let $R = (s, [1, 5]) \neg (u, [1, 7]) \overset{\ast}{\otimes} (v, [3, 10])^*$. Then $(s, [1, 5]), (u, [1, 7]), (v, [3, 10]), (s, [1, 5]) \neg (u, [1, 7]), (v, [3, 10])^*$ and $R$ are sub-expressions of $R$. A sub-expression $R'$ of $R$ can occur at many different positions in $R$. In the sequel, when we talk about a subexpression of $R$, we mean an occurrence of its in $R$. Thus, for the simplicity, we will identify a subexpression with one of its occurrences.

An TRE $R$ for which $M(R) = \emptyset$ is called an empty TRE and denoted by $\Lambda$. For example, $(s, [3, 5]) \overset{\ast}{\otimes} (v, [6, 9])$ is an empty TRE. We will show how to recognise an empty expression in the next section.

For any TRE $R$ in which there is no occurrence of an empty sub-expression, we associate with the numbers $m(R), M(R)$ as in the following definition. Roughly speaking, $m(R)$ is a lower bound and $M(R)$ is an upper bound of the set $\{d(\sigma) \mid \sigma \in M(R)\}.$

**Definition 4.1**

- If $R = \epsilon$, then $m(R) = 0$ and $M(R) = 0$.
- If $R = (s, [a, b])$, then $m(R) = a$ and $M(R) = b$ ($b$ may be $\infty$).
- If $R = R_1^\ast R_2$, then $m(R) = 0$ and $M(R) = \infty$.
- If $R = R_1 \neg R_2$, then $m(R) = m(R_1) + m(R_2)$ and $M(R) = M(R_1) + M(R_2)$.
- If $R = R_1 \otimes R_2$, then $m(R) = \min(m(R_1), m(R_2))$ and $M(R) = \max(M(R_1), M(R_2))$.
- If $R = R_1 \otimes R_2$, then $m(R) = \max(m(R_1), m(R_2))$ and $M(R) = \min(M(R_1), M(R_2))$.

From Definition 4.1, it is easy to see that if $R$ is a simple TRE and $M(R) < \infty$ then $m(R), M(R)$ are minimum and maximum of the set $\{d(\sigma) \mid \sigma \in M(R)\}$. Furthermore, for any TRE $R$, for any $\sigma \in M(R)$, $m(R) \leq d(\sigma) \leq M(R)$. For example, let

$$R = ((s, [1, 5]) \neg (u, [1, 7]) \overset{\ast}{\otimes} (v, [3, 10]).$$

Then, $m(R) = 3$ and $M(R) = 10$. This means that for any $\sigma \in M(R)$, $3 \leq d(\sigma) \leq 10$.

An important remark should be made here is that for any simple TRE $R$, for any real number $r$ such that $m(R) \leq r \leq M(R)$, there is a model $\sigma \in M(R)$ for which $d(\sigma) = r$. Therefore, checking the emptiness of a simple TRE $R$ is trivial. Hence, for a simple TRE $R$, $m(R) = 0$ means that for any primitive $(s, [a, b])$ occurring in $R$ the lower bound $a$ should be 0.

Note that for any non empty TREs $R_1, R_2, R_1 \otimes R_2$ may be empty although $R_1 \neg R_2$, $R_1 \otimes R_2, R_1^\ast$ cannot be empty.

If $R$ is not an empty TRE, we can find out $R_1$ such that $R_1$ has no empty sub-expression and that $M(R) = M(R_1)$. Thus, from now on, unless otherwise stated, we assume that all TREs under our consideration are not empty TREs and do not have any empty sub-expression.

Let $R_1, R_2$ be TRE’s. As discussed earlier, if $R = R_1 \otimes R_2$ then any $\sigma \in M(R)$ is constructed from models $\sigma_1 \in M(R_1)$ and $\sigma_2 \in M(R_2)$ such that $d(\sigma_1) = d(\sigma_2)$. Hence, the execution time of $R_1$ is limited by the execution time of $R_2$ and vice-versa. In general, the execution time of $R'$, where $R'$ is an arbitrary sub-expression of $R_1$, is not only bounded by $m(R')$ and $M(R')$ but also by $m(R_2)$ and $M(R_2)$. This means that the execution time of a sub-expression $R'$ in a TRE $R$ is constrained by the operator $\otimes$ and by its occurrence position in $R$. To capture these constraints we define the quantities $m(R', R)$ and $M(R', R)$ as lower and upper bounds of the time execution of $R'$ when it occurs at fixed position in $R$. $m(R', R)$ and $M(R', R)$ are defined recursively as follows.

**Definition 4.2**

- Let $R = R', m(R', R) = 0$ and $M(R', R) = \infty$ (no additional constraint).
- Let $R = R_1 \ast R_2 (R_2 \ast R_1, R_1^\ast), \sigma \in \{\ast, \otimes\}$ and $R'$ occurs in $R_1$. Then $m(R', R) = m(R', R_1)$ and $M(R', R) = M(R', R_1)$ (no additional constraint).
- Let $R = R_1 \otimes R_2 (R_2 \otimes R_1)$, and let $R'$ occur in $R_1$. Then $m(R', R) = \max(m(R', R_1), m(R_2))$ and $M(R', R) = \min(M(R', R_1), M(R_2))$ (additional constraint enforced by the operator $\otimes$).

For example, let

$$R = ((s, [1, 5]) \neg (u, [1, 7]) \overset{\ast}{\otimes} (v, [3, 10])^*.$$

Let $R' = (s, [1, 5]) \neg (u, [1, 7])$ then $m(R', R) = 0$ and $M(R', R) = \infty$.

Let $R' = (v, [3, 10])^*$. Then $m(R', R) = 2$ and $M(R', R) = 12$.

Denote $M(R', R) = \{\sigma \in M(R') \mid m(R', R) \leq d(\sigma) \leq M(R', R)\}$.

From the above discussion, it can be seen that only the models in $M(R', R)$ can participate in constructing the models of $R$. Therefore, from now on, if $R'$ is considered as a sub-expression (occurring at a fixed position) of $R$ then we can identify $M(R', R)$ to its subset $M(R', R)$. By induction on the structure of TREs, we can prove the following lemmas.
Lemma 4.1 Let $R_1$, $R_2$, $R'$ be arbitrary TREs. If for any model $\sigma_1 \in M(R_1)$, there exists a model $\sigma_2 \in M(R_2)$ such that $d(\sigma_1) = d(\sigma_2)$ and $\text{inv}(\sigma_1) \leq \text{inv}(\sigma_2)$ then for any model $\sigma_1' \in M(R_1) \setminus R'$ $(M(R_1 \cup R'))$, $(M(R_1 \setminus R))$ there exists a model $\sigma_2' \in M(R_2 \setminus R')$ $(M(R_2 \cup R'))$, $(M(R_2 \setminus R))$ such that $d(\sigma_1') = d(\sigma_2')$ and $\text{inv}(\sigma_1') \leq \text{inv}(\sigma_2')$.

Lemma 4.2 Let $R$, $R_1$, $R_2$ be arbitrary TREs. If

i. For any model $\sigma_1 \in M(R_1)$, there exists a model $\sigma_2 \in M(R_2)$ such that $d(\sigma_1) = d(\sigma_2)$ and $\text{inv}(\sigma_1) \leq \text{inv}(\sigma_2)$, and

ii. For any model $\sigma_2 \in M(R_2)$, there exists a model $\sigma_1 \in M(R_1)$ such that $d(\sigma_2) = d(\sigma_1)$ and $\text{inv}(\sigma_2) \leq \text{inv}(\sigma_1)$,

then by replacing an occurrence of $R_1$ in $R$ with $R_2$, we obtain a new expression $R'$ which is $D$-equivalent to $R$, i.e. $R' \equiv_D R$.

Let $[x]$ be the floor of a real variable $x$, which is the maximal integer which are not greater than $x$.

Theorem 4.1 Let $A$ be a simple TRE with $m(A) = 0$. Let $A'$ be the TRE obtained from $A$ by replacing each primitive $(s, [0, b])$ of $A$ with $(s, [0, \infty))$ (remember that $b > 0$ as assumed earlier). Then, by replacing an occurrence of $A^*$ in a TRE $R$ with $A'$, we obtain a new expression $R'$ which is $D$-equivalent to $R$.

Proof.

For the simplicity of our presentation, assume that primitives of $A$ and $A'$ are listed as sequences $\{(s_i, [0, b_i]), i = 1, 2, \ldots, m\}$ and $\{(s_i, [0, \infty)), i = 1, 2, \ldots, m\}$. Let $C(A)$ and $C(A')$ respectively be the systems of linear constraints associated to $A$ and $A'$ (Definition 3.4). Recall that each linear constraint in $C(A)$ is of the form $0 \leq t_i \leq b_i$ which corresponds to the constraint $0 \leq t_i \in C(A')$, or of the form $\sum_{k=1}^{n} t_{ik} = \sum_{j=1}^{m} t_{jk}$, which corresponds to the same constraint in $C(A')$.

Let $\sigma \in M(A^*)$. It follows that $\sigma = \sigma_1 \overline{\sigma_2 \cdots \sigma_k}$, where $\sigma_i \in M(A) (1 \leq i \leq k)$. Let $w_1, w_2, \ldots, w_k$ be the solutions of the system of linear constraints $C(A)$ corresponding to $\sigma_1, \sigma_2\ldots, \sigma_k$ respectively as in the proof of Theorem 3.3. From the above notice, it is obviously that $w' = w_1 + w_2 + \ldots + w_k$ is a solution of $C(A')$. Let $\sigma'$ be a model in $M(A')$ corresponding to solution $w'$. Hence, for any $\sigma \in M(A^*)$ we can find a model $\sigma' \in C(A')$ satisfying $d(\sigma') = d(w') = d(w_1) + d(w_2) + \ldots + d(w_k) = d(\sigma_1) + d(\sigma_2) + \ldots + d(\sigma_k) = d(\sigma)$ and $d_s(\sigma') = d_s(w') = d_s(w_1) + d_s(w_2) + \ldots + d_s(w_k) = d_s(\sigma_1) + d_s(\sigma_2) + \ldots + d_s(\sigma_k) = d_s(\sigma), \forall s \in S$, and hence, $\text{inv}(\sigma') = \text{inv}(\sigma)$.

Reversely, let $\sigma' \in M(A')$ and let $w' = (u_1, u_2, \ldots, u_m)$ be the solution of $C(A')$ corresponding to $\sigma'$. Let $k = \max \{\lfloor u_i/b_i \rfloor + 1 \mid i = 1, 2, \ldots, m\}$ and $w = (u_1/k, u_2/k, \ldots, u_m/k)$. It is easy to prove the $w$ is a solution of $C(A)$. Let $\sigma_0$ be the model in $M(A)$ corresponding to $w$ and $\sigma = \sigma_0 \sigma_0 \cdots \sigma_0$ (k times). Then $\sigma \in M(A^*)$ and $d(\sigma) = k \times d(\sigma_0) = k \times d(w) = d(w') = d(\sigma')$, $d_s(\sigma) = k \times d_s(\sigma_0) = k \times d_s(w) = d_s(w') = d_s(\sigma')$ for all $s \in S$. Hence, $\text{inv}(\sigma') = \text{inv}(\sigma)$.

The theorem now follows immediately from Lemma 4.2.

\[ \square \]

Theorem 4.2 Let $A$ be a simple TRE with $m(A) > 0$. Let $A^*$ be an occurrence of the TRE $A'$ in a TRE $R$ for which $M(A^*, R) < \infty$ or $e < \infty$ (recall that $e$ is the upper bound of the observation time period in the premise $b \leq \ell \leq e$ of the linear duration invariant $D$). Let $A' = \bigoplus_{i=0}^{k} A_i$, where $k = \min \{M(A^*, R), e/m(A)\} + 1$. Then by replacing the occurrence $A^*$ in $R$ with $A'$, we obtain a new expression $R'$ which is $D$-equivalent to $R$.

Proof.

It is obviously that $M(A') \subset M(A^*)$. Hence, by Lemma 4.1, for any model $\sigma' \in M(R')$, there exists model $\sigma \in M(R)$ such that $d(\sigma) = d(\sigma')$ and $\text{inv}(\sigma) \leq \text{inv}(\sigma')$.

Conversely, since $A^*$ occurs in $R$, as mentioned earlier, only the models in the set $M(A^*, R)$ can participate in constructing the models of $R$. Let $\sigma \in M(A^*, R)$. By the definition $M(A^*, R)$, $\sigma = \sigma_1 \overline{\sigma_2 \cdots \sigma_j}$, where $\sigma_i \in M(A) (1 \leq i \leq j)$. Hence $d(\sigma) = d(\sigma_1) + d(\sigma_2) + \ldots + d(\sigma_j) \geq j \times m(A)$. Because $d(\sigma) \leq M(A^*, R)$ and $d(\sigma) \leq e, j \times m(A) \leq d(\sigma) \leq \min \{M(A^*, R), e\}$. Therefore, $j \leq \min \{M(A^*, R), e\}/m(A) \leq k$, which implies that $\sigma \in M(A')$. Hence, applying the lemma 4.1, we have that for any model $\sigma \in M(R)$, there exists a model $\sigma' \in M(R')$ such that $d(\sigma') = d(\sigma)$ and $\text{inv}(\sigma') \leq \text{inv}(\sigma)$. Finally, by Lemma 4.2, $R' \equiv_D R$.

\[ \square \]

Let $R'$ be a sub-expression of $R$. $R'$ is said to be under $\otimes$ if there is a sub-expression of form $R_1 \otimes R_2$ of $R$ such that $R'$ is an occurrence in $R_1$ or in $R_2$. If $A^*$ is not under $\otimes$, then by definition 4.2, $M(A^*, R) = \infty$.

Given a simple TRE $A$. Let $\text{maxinv}(A)$ denote the maximal value of $\{\text{inv}(\sigma) \mid \sigma \in M(A)\}$. $\text{maxinv}(A)$ can be calculated by solving the linear programming problem: finding the maximum of the objective function $\sum_{s \in S} c_s d_s(A)$ subject to the set of constraints $C(A)$.

Lemma 4.3 Let $B$ be a real number, $A'$ be a sub-expression of $R$ which is not under $\otimes$, where $A$ is a simple TRE with $m(A) > 0$, $\text{maxinv}(A) \leq 0$. Assume that $e = \infty$. Furthermore, let $R'$ be obtained from $R$ by replacing the occurrence $A^*$ in $R$ by $A' = \bigoplus_{i=0}^{k} A_i$ with $k = \lfloor B/m(A) \rfloor + 1$. 

Then, for any model σ ∈ ℳ(R) such that d(σ) ≥ B, there exists a model σ′ ∈ ℳ(R′) such that d(σ′) ≥ B, and inv(σ) ≤ inv(σ′).

**Theorem 4.3** Let c = ∞, and A∗ be a sub-expression of R such that A∗ is not under ⊗, where A is a simple TRE with m(A) > 0. Then

i. If maxinv(A) ≤ 0, then by replacing A∗ in R with \( A = \bigodot_{i=0}^{k} A^i \), where \( k = \lfloor b/m(A) \rfloor + 1 \), we obtain a new expression R′ such that R′ = D.

ii. If maxinv(A) > 0, then R ≠ D.

**Proof.**

i. It is obviously that \( ℳ(A^i) ⊂ ℳ(A∗) \). Hence, by Lemma 4.1, for any model σ′ ∈ ℳ(R′), there exists model σ ∈ ℳ(R) such that d(σ) = d(σ′) and inv(σ) ≤ inv(σ′). By Lemmas 4.1 and 4.2, it follows R |= D => R′ |= D.

The other direction is proved as follows. By Lemma 4.3, for any σ ∈ ℳ(R) such that d(σ) ≥ b there is σ′ ∈ ℳ(R′) such that d(σ′) ≥ b and inv(σ′) ≥ inv(σ). Hence, as a result, R′ |= D ⇒ R |= D.

ii. Assume that maxinv(A) > 0 and A∗ is not under ⊗. By induction on the structure of R, it can be seen easily that for any subexpression R1 of R if there exists a sequence of models σi ∈ ℳ(Ri), i ≥ 1 such that lim inv(σi) = ∞, then there exists a sequence of models σi ∈ ℳ(R), i ≥ 1 such that lim inv(σi) = ∞.

Let \( w_0 \) be the optimal solution of the linear programming problem: max \( \sum_{x \in S} c_x d_x(A) \) subject to C(A). Let \( σ_0 \in ℳ(A) \) be the model corresponding to \( w_0 \) then inv(σ0) = maxinv(A) > 0. Hence, for the sequence \( σ_i = σ_0 \supseteq σ_1 \supseteq \cdots \supseteq σ_i \) (i times), i ≥ 1, we have for all i, σi ∈ ℳ(A∗) and inv(σi) = i × inv(σ0) → ∞ when i → ∞. Since A∗ is a sub-expression of R, we can construct a sequence of models σi, i ≥ 1 such that σi ∈ ℳ(R) and inv(σi) → ∞ (when i → ∞). Hence, we can find a model σ ∈ R satisfying that b ≤ d(σ) ≤ ∞ and inv(σ) > M. In the other words, R ≠ D.

By Theorems 4.1, 4.2 and 4.3, we can remove a star in a TRE R without introducing a new star, without increasing the number of stars under ⊗ for the following cases:

i. a star of the form A∗, where A is a simple TRE with \( m(A) = 0 \),

ii. a star of the form A∗, where A is a simple TRE with \( m(A) = 0 \) for the case that either \( M(A∗, R) \) or c is finite.

iii. a star of the form A∗, where A is a simple TRE with \( m(A) = 0 \) for the case that c is infinite and A∗ is not under ⊗.

Therefore, if the linear duration D is of the form \( b ≤ c ≤ \epsilon \Rightarrow \sum_{i=1}^{n} c_i s_i ≤ M \) for which \( \epsilon < ∞ \) or if the TRE R has no sub-expression of the form A∗, where A is a simple TRE with \( m(A) > 0 \) and \( M(A, R) = ∞ \), then checking R |= D can be reduced to solving a finite number of linear programming problems.

**Example** Let us verify that the railroad crossing monitor satisfies the requirement. That is to check RCM |= D, where D = 0 ≤ t ≤ ∞ ⇒ ∫ C − ∫ Dn ≤ 0, and

\[
RCM = (A \otimes U)^n = ((B, [a, ∞)) \otimes (MD, [b, c]) \otimes Dn)^n
\]

Because the subexpression under * is not a simple one, in order to use Theorem 4.3, we transform RCM into the following expression RCM1 using Theorem 3.2:

\[
RCM1 = (A \otimes U)^n = ((B, [a, ∞)) \otimes (MD, [b, c]) \otimes Dn)^n
\]

Let

\[
R_1 = (((B, [a, ∞)) \otimes (MD, [b, c]) \otimes Dn)^n
\]

\[
R_2 = (((B, [a, ∞)) \otimes (MD, [b, c]) \otimes Dn)^n
\]

By Definition 4.1, m(R1) > 0 and m(R2) > 0. Furthermore,

\[
\text{maxinv}(R_1) = \max \{ \text{inv}(σ) | σ \in ℳ(R_1) \}
\]

\[
= \max \{ t_2 - t_4 \} \text{ subject to:a ≤ t_1, 0 ≤ t_2, 0 ≤ t_3 ≤ c, 0 ≤ t_4, t_1 + t_2 = t_3 + t_4, b ≤ t_5, 0 ≤ t_6, 0 ≤ t_7, 0 ≤ t_8, t_5 + t_6 = t_7 + t_8, c = a
\]
Similarly, we have \( \maxin(R_2) = c - a \) as well. Since \( c \leq a \), we have \( \maxin(R_1) \leq 0 \) and \( \maxin(R_2) \leq 0 \). By applying Theorem 4.3 twice with noticing that \( k = 1 \), we have that \( RCM \models D \) is now equivalent to \( RCM_2 \models D \), where

\[
RCM_2 = (A \otimes U)^\sim \\
(\epsilon \oplus \\
((((B, [a, \infty]))^\sim C \otimes (MD, [0, c])^\sim Dn)^\sim \\
((P, [b, \infty])^\sim A \otimes (MU^\sim U)))^\sim \\
(\epsilon \oplus \\
((((B, [a, \infty]))^\sim C \otimes (MD, [0, c])^\sim Dn)^\sim \\
((P, [b, \infty])^\sim A \otimes (MU^\sim U)))^\sim \\
((B \oplus (B, [a, \infty]))^\sim C \otimes (MD, [0, c])^\sim Dn) \oplus \\
((P \otimes P^\sim A) \otimes (MU \otimes MU^\sim U)))
\]

\( RCM_2 \) is a finite TRE, and checking \( RCM_2 \models D \) is so simple for this case. For the left case, it seems that the problem is difficult. In the next section, we propose some techniques to solve the problem for some special subcases and show how to check the emptiness of a TRE.

5. Checking emptiness and reducing star occurrences under \( \otimes \)

As mentioned earlier, checking the emptiness of a TRE in which the star does not occur under \( \otimes \) (in the operands of a \( \otimes \)) is so simple. However, the problem becomes difficult when the star occurs in the operands of a \( \otimes \). Let for example

\[
R = ((s_1, [a_1, b_1])^* \otimes (s_2, [a_2, b_2])^*) \otimes \\
((s_3, [a_3, b_3])^* \otimes (s_4, [a_4, b_4])^*)
\]

Replacing each star * with an integral variable, we get

\[
R = \bigoplus_{m_1, m_2, m_3, m_4 \geq 0} ((s_1, [a_1, b_1])^{m_1} \otimes (s_2, [a_2, b_2])^{m_2}) \otimes \\
((s_3, [a_3, b_3])^{m_3} \otimes (s_4, [a_4, b_4])^{m_4})
\]

Thus, \( R \) is not empty iff the inequalities

\[
m_1a_1 + m_2a_2 \leq m_3b_2 + m_4b_4 \\
m_3b_4 + m_4a_4 \leq m_1b_1 + m_2b_2 \quad (1)
\]

has an integral solution. In order to make the problem easier, we assume in this section that all the real constants occurring in a TRE are rational. Thus, checking the emptiness of TREs leads to checking the emptiness of \( \{Ax \leq b | x \text{ integral} \} \), where \( A \) is a matrix, \( b \) is a vector, and \( A, b \) are rational, which is an integer linear programming problem and can be solved in polynomial time.

To reduce the star occurring under \( \otimes \) is difficult for the case that \( e = \infty \). We show that the problem can be solved by the mixed integer linear programming techniques. For simplicity, we present our idea via the previous example, i.e. to decide whether

\[
R \models D \quad (2)
\]

where \( R \) is given as above, \( D \) is of the form \( b \leq \ell \Rightarrow c_1 \int s_1 + c_2 \int s_2 + c_3 \int s_3 + c_4 \int s_4 \). Then, the problem is equivalent to

\[
\forall m_1, m_2, m_3, m_4 : \\
((s_1, [a_1, b_1])^\sim (s_2, [a_2, b_2])^\sim) \otimes \\
((s_3, [a_3, b_3])^\sim (s_4, [a_4, b_4])^\sim) \models D
\]

By Theorem 3.3, the problem is now equivalent to \( K \leq M \), where \( K \) is the result of the following mixed integer linear programming problem \( \max(c_1t_1 + c_2t_2 + c_3t_3 + c_4t_4) \) subject to \( m_1a_1 \leq t_1 \leq m_1b_1, m_3 \leq 0, m_i \text{ integral}, i = 1, 2, 3, 4 \).

This technique can be easily generalised to the case when \( R \) has no nested stars occurring under \( \otimes \).

In some cases, we can use the following technique to reduce a star.

Let \( \zeta = (m_1, m_2, m_3, m_4) \) be a solution of (1). From the classical techniques of integer programming ([7]), we can find an integral Hilbert basis \( \zeta_1, \ldots, \zeta_k \), such that \( \zeta \) is a solution of (1) iff \( \zeta = \sum_{i=1}^k \lambda_i \zeta_i \). Let \( \zeta_i = (m_{i1}, m_{i2}, m_{i3}, m_{i4}) \). The mixed integer and linear problem now becomes:

\[
\max(c_1t_1 + c_2t_2 + c_3t_3 + c_4t_4)
\]

subject to

\[
\sum_{j=1}^k \lambda_i m_{ij} a_j \leq t_j \leq \sum_{j=1}^k \lambda_i m_{ij} b_j (j = 1, 2, 3, 4) \\
t_1 + t_2 = t_3 + t_4 \\
\lambda_i \geq 0 (i = 1, \ldots, k) \\
\lambda_i (i = 1, \ldots, k) \text{ integral}
\]

Thus, \( c_1t_1 + c_2t_2 + c_3t_3 + c_4t_4 \) should satisfy:

\[
\sum_{j=1}^k \lambda_i \sum_{j=1}^k \lambda_i m_{ij} c_j u_{ij} \leq \\
c_1t_1 + c_2t_2 + c_3t_3 + c_4t_4 \leq \\
\sum_{j=1}^k \lambda_i \sum_{j=1}^k \lambda_i m_{ij} c_j u_{ij}
\]

where for \( j = 1, 2, 3, 4 \) if \( c_j \geq 0 \) then \( u_{ij} = m_{ij} a_j, u_{ij} = m_{ij} b_j \), and otherwise \( u_{ij} = m_{ij} b_j, u_{ij} = m_{ij} a_j \). Now, if there is i such that \( \sum_{j=1}^k \lambda_i c_j u_{ij} > 0 \), we can conclude that \( R \not\models D \), since \( c_1t_1 + c_2t_2 + c_3t_3 + c_4t_4 \geq M \) when \( \lambda_i \) approaches \( \infty \). Otherwise, if \( \sum_{j=1}^k \lambda_i c_j u_{ij} \leq 0 \) for all i and \( \sum_{j=1}^k \lambda_i c_j u_{ij} < 0 \) for some i, then we can always find
nonnegative integers $\lambda_i^0, i = 1, \ldots, k$ such that for $\lambda_i > \lambda_i^0, i = 1, \ldots, k, \sum_{i=1}^k \lambda_i^0 \sum_{j=1}^4 c_{ij} u_{ij} \leq M$. Then, $R \models D$ iff

$$\max(c_1 t_1 + c_2 t_2 + c_3 t_3 + c_4 t_4)$$

subject to

$$\sum_{i=1}^k \lambda_i m_{ij} a_j \leq t_j \leq \sum_{i=1}^k \lambda_i m_{ij} b_j \quad (j = 1, 2, 3, 4)$$

$$t_1 + t_2 = t_3 + t_4$$

$$\lambda_i \geq \lambda_i^0 \geq 0, \quad (i = 1, \ldots, k)$$

$$\lambda_i \quad (i = 1, \ldots, k) \text{ integral}$$

is not greater than $M$, which can be decide by solving a finite number of linear programming problems.

6. Conclusion

We have introduced the concept of timed regular expressions by a simple extension of the classical regular expressions to describe the behaviour of real-time concurrent systems, and shown that for the class of real time systems whose behaviour can be described by $\text{TREs}$, checking for a linear duration invariant can be done by using mixed integer linear programming techniques in general. We have also presented a technique to transform the problems into simpler ones which works for many cases. Since model-checking for real-time systems is difficult and possesses high complexity, our technique could help to reduce the complexity in the case it can be used. We have seen that checking whether our techniques can be applied is very simple, it is not a time consuming work. Thus, in practice, a procedure can be developed to analyse the problem and choose the suitable algorithms to use for solving the model-checking problem more efficiently.

Acknowledgement The authors would like to thank Professor Zhou Chaochen for his valuable comments and continuous encouragement when writing this paper.

References


