BTP-Report-II

Emptiness of TRA using graph semantics

Mohan Sai Teja Dantam

Guide: Prof. S. Akshay and Prof. Krishna

IIT BOMBAY

Contents

Chapter 1

Timed Register Automata

1.1 Introduction

Timed Register Automata (TRA) introduced in [3] are register automata with time as data. Throughout this report we will assume that a special register t is present for every TRA which contains the value of current time. In this section we define few models of TRA. But before defining them we will define set of guards and updates possible on a set X.

The set $\mathcal{G}(X)$ is given by the following grammar.
$\psi := x - y \bowtie c \mid \psi \wedge \psi \mid True$
where x,$y \in X \cup \{t\}$, $\bowtie \in \{<, \leq\}$, $c \in \mathbb{Z}$

We call each guard of the form $x - y \bowtie c$ a simple guard (sg). From the above definition it is clear that each guard is either $True$ or conjunction of sg's.

We define two types of updates given a set X
$\mathcal{U}(X) := \{f \mid f : X \to \{0, t\}\ \}$
$\mathcal{U}'(X) := \{f \mid f : X \to \mathbb{Z} \cup \{t\}\ \}$
We can see that $\mathcal{U}(X) \subset \mathcal{U}'(X)$

Definition 1.1 (Model 1a (resp. 1b) TRA)**.** A Model 1a (resp. 1b) TRA is a tuple $\mathcal{A} = (Q, \Sigma, X, I, F, Lab, \Delta)$ where
- Q is a finite set of states, Σ is a finite alphabet
- I, $F \subseteq Q$ are initial and final states
- X is a finite set of registers
- Lab is finite set of labels (for naming transitions)
- $\Delta : Lab \to Q \times \Sigma \times Q \times \mathcal{G}(X) \times \mathcal{U}(X)$ (resp. $\mathcal{U}'(X)$) is a function that defines each transition

Notation

We consider Q to be $\{q_1, q_2, \ldots, q_n\}$ and $Lab = \{t_1, t_2, \ldots, t_m\}$ and $X = \{x_1, x_2, \ldots, x_r\}$. Let $\Delta(t) = (q, a, q', g, u)$. We define $src(t) = q$, $dst(t) = q'$,$\sigma(t) = a$, $guard(t) = g$ and $up(t) = u$.

1.2 Semantics of TRA

Any model of TRA runs on $(\Sigma \times \mathbb{R})^*$. Here we define the semantics for **both** the models.

Definition 1.2 (Valuation)**.** A Valuation V is an assignment of a positive real number to each of the registers in X and to the register t i.e $V : X \cup \{t\} \to \mathbb{R}^+$

A valuation V is said to satisfy a sg $x - y \bowtie c$ (written as $V \models x - y \bowtie c$) iff the statement $V(x) - V(y) \bowtie c$ is true.
A valuation V is said to satisfy a guard g ($V \models g$) iff either g is $True$ or V satisfies all sg's in g.

Definition 1.3 (Acceptance condition)**.** A non-empty word
$tw = (w_1, t_1)\ (w_2, t_2)\ \ldots\ (w_p, t_p)$ is said to be accepted by $\mathcal{A}$ ($tw \models \mathcal{A}$)
iff $\exists$ labels $s_1, s_2, \ldots, s_p$ and Valuations $V_0, V_1, \ldots, V_p$ such that
- $V_0(x) = t_1\ \forall\ x \in X \cup \{t\}$
- $V_{i-1}(t) = t_i\ \forall\ 1 \leq i \leq p$
- $V_i(x) = V_{i-1}(x) + up(s_i)(x)$ if $up(s_i)(x) \neq t$ else $V_i(x) = t_i\ \forall\ 1 \leq i \leq p, \forall\ x \in X$
- $V_{i-1} \models guard(s_i)\ \forall\ 1 \leq i \leq p$
- $dst(s_i) = src(s_{i+1})\ \forall\ 1 \leq i \leq p-1$
- $\sigma(s_i) = w_i\ \forall\ 1 \leq i \leq p$ and $src(s_1) \in I$ and $dst(s_p) \in F$

we say $\epsilon \models \mathcal{A}$ iff $I \cap F \neq \phi$.
Consequently, Language of a TRA $\mathcal{A}$ is defined to be $\mathcal{L}(\mathcal{A}) = \{w \mid w \models \mathcal{A}\}$
The *Emptiness* problem on TRA is given a TRA $\mathcal{A}$ to decide if $\mathcal{L}(\mathcal{A}) = \phi$.

It turns out that the **emptiness problem for Model 1b TRA is undecidable** in general ([3]) as we can obtain a reduction from Halting problem of two counter machines to this problem and it can be done with three registers.

1.3 Bounded span TRA

The undecidability of the emptiness problem for Model 1b is due to the fact that a counter can be modeled as difference between two registers. But if we constraint that difference between any two registers should be bounded by some constant at all times during the run and accept only then, it turns out that the emptiness problem for this model is decidable. This leads us to the following definition

Definition 1.4 (Model 2a (2b) TRA)**.** A Model 2a (2b) TRA is a Model 1a (1b) TRA $\mathcal{A}$ along with an integer $k > 0$.

Semantics

The semantics of the new models are same as that of their respective old models with the following additional constraint.

- $\forall\ 0 \leq i \leq p, \forall x, y \in X\ |V_i(x) - V_i(y)| \leq k$

That is a run is accepting only if at any point during the run the difference between registers is at most k.

1.4 TRA with stack

Till now, we saw the syntax and semantics for various models of timed register automata. It is easy to extend each of these models to include a stack. In this subsection, we see the syntax and semantics for Model 2b Pushdown Timed Register Automata (PDTRA).

In a PDTRA, stack contains data values. These data values pushed can either be from registers used by the automata or the special register t. For PDTRA, along with the special register t, we will also have another special register s which keeps track of the value at the top of the stack at each point. When a value is popped out of the stack, that value can be used in defining guard for that particular transition.

Recall that $\mathcal{G}(X)$ denotes the set of all finite conjunctions of difference constraints from $X \cup \{t\}$.

Definition 1.5 (Model 2b PDTRA)**.** A Model 2b PDTRA is a tuple $\mathcal{A} = (Q, \Sigma, X, I, F, Lab, \Delta, k)$ where

- Q is a finite set of states, Σ is a finite alphabet
- I, $F \subseteq Q$ are initial and final states
- X is a finite set of registers
- Lab is finite set of labels (for naming transitions)
- k is the bounded span constant
- $\Delta : Lab \to Q \times \Sigma \times Q \times \mathcal{G}'(X, s) \times \mathcal{U}'(X) \times \{\text{push(.)},\text{pop()},\text{nop}\}$ is a function that defines each transition where last co-ordinate is a stack operation, push(.) takes an argument from $X \cup \{t\}$ as input and $\mathcal{G}'(X, s) = \mathcal{G}(X)$ if the stack operation is push(.) or nop and $\mathcal{G}'(X, s) = \mathcal{G}(X \cup \{s\})$ otherwise

Definition 1.6 (stack valuation)**.** A stack valuation V_s is a Valuation along with assignment of a real number or $\perp$ to the stack register s i.e $V_s : X \cup \{s, t\} \to \mathbb{R}^+ \cup \{\perp\}$ such that V_s restricted to $X \cup \{t\}$ is a valuation.

The notion of a stack valuation satisfying a guard is similar to that of a valuation satisfying guard.

The condition of acceptance of a word by PDTRA inherits all the conditions required by a Model 2b PDTRA along with the following two additional conditions

1. At the end, the stack must be empty i.e the number of pushes must be equal to the number of pops
2. At any point in between, number of pushes must be greater than or equal to the number of pops

Chapter 2

Semantics of TRA as weighted Graphs

In the Introduction we saw the behaviors of TRA as essentially timed words over Σ. In this chapter we show how to visualize a given run as a weighted graph. Motivation for viewing the semantics as graphs comes from ([2]) in which they used a simliar technique for deciding emptiness of timed pushdown automata

Definition 2.1 (Weighted graph)**.** A weighted graph G is a tuple (V, E) where
- V is the set of vertices
- $E \subseteq V \times \{<, \leq\} \times \mathbb{Z} \times V$ is the set of edges

Given an edge $e = (u, \bowtie, \alpha, v)$ (denoted as $u \xrightarrow{\bowtie \alpha} v$) we define $src(e) = u$, $cmp(e) = \bowtie$, $wt(e) = \alpha$, $dst(e) = v$. We will also assume that $src(e) \neq dst(e)$ $\forall\ e \in E$. For an integer M, graph G is said to *M-weight bounded* iff $|wt(e)|$ $< M\ \forall\ e \in E$. A path is a non-empty sequence of edges $P = e_1 e_2 \cdots e_n$ in G such that $dst(e_i) = src(e_{i+1})$ for all $1 \leq i \leq n$. Sometimes we also denote a path P as $v_1 v_2 \cdots v_{n+1} = e_1 e_2 \cdots e_n$. Similar to edges we define $src(P) = src(e_1)$, $dst(P) = dst(e_n)$, $wt(P) = \sum_{1 \leq i \leq n} wt(e_i)$. A path P is *cyclic* if $src(P) = dst(P)$. A cyclic path C is said to be simple if no vertex occurs twice as a source of an edge in C.

We essentially show graph behaviors for Model 2b. Graph behaviors for other models can be easily seen once we 2b's graphs. To convert a run into a graph, we first convert a run into a linear constraint problem which is then seen as a weighted graph.

2.1 Transforming a run into set of difference constraints

Consider a Model 2b TRA $\mathcal{A} = (Q, \Sigma, X, I, F, Lab, \Delta, k)$.View the TRA as a graph with Q vertices and consider any walk $w = s_1, s_2, s_3, \ldots, s_\alpha$ such that $src(s_1) \in I$

and $dst(s_\alpha) \in F$. Define $\sigma(w)$ to be $\sigma(s_1)\ \sigma(s_2)\ \sigma(s_3)\ \dots\ \sigma(s_\alpha)$
Then $\sigma(w)$ is accepting along this path $\iff \exists$ Valuations $V_0, V_1, \cdots, V_\alpha$ such that
- $V_0(t) = V_0(x)\ \forall x \in X$
- $V_i(x) = V_{i-1}(x) + up(s_i)(x)$ if $up(s_i)(x) \neq t$ else $V_i(x) = V_{i-1}(t)\ \forall\ 1 \leq i \leq \alpha, \forall\ x \in X$
- $V_{i-1} \models guard(s_i)\ \forall\ 1 \leq i \leq \alpha$
- $\forall\ 0 \leq i \leq \alpha, \forall x, y \in X\ |V_i(x) - V_i(y)| \leq k$

The other conditions for acceptance (such as $dst(s_i) = src(s_{i+1})$) are trivially satisfied as we considered a walk. Each V_i is a vector of $|X|+1$ variables. Fix a walk w. The first condition $V_0(t) = V_0(x)\ \forall x \in X$ can be rewritten as $V_0(t) - V_0(x) \leq 0$ and $V_0(x) - V_0(t) \leq 0\ \forall x \in X$. The second condition also contains only equality constraints (which depend on walk w but since we are looking at a particular run these constraints are well defined) and those can be similarly rewritten. Now, in the third condition if $guard(s_i)$ is true,
the constraint $V_{i-1} \models guard(s_i)$ is trivially true else
$V_{i-1} \models guard(s_i) \iff V_{i-1}(x) - V_{i-1}(y) \bowtie c\ \forall\ x - y \bowtie c \in guard(s_i)$. In the fourth condition, removing the modulus we get that $\forall\ 0 \leq i \leq \alpha, \forall x, y \in X$ such that $x \neq y$, $V_i(x) - V_i(y) \leq k$.
Hence, we can say that the **run w is accepting $\iff \exists$ Valuations $V_0, V_1, \cdots, V_\alpha$ such that the following set of difference constraints are satisfied**.
- $V_0(t) - V_0(x) \leq 0$, $V_0(x) - V_0(t) \leq 0\ \forall x \in X$
- $\forall\ 1 \leq i \leq \alpha, \forall\ x \in X\ V_i(x) - V_{i-1}(x) \leq up(s_i)(x)$, $V_{i-1}(x) - V_i(x) \leq -up(s_i)(x)$ if $up(s_i)(x) \neq t$ else $V_i(x) - V_{i-1}(t) \leq 0$, $V_{i-1}(t) - V_i(x) \leq 0$
- $(V_{i-1}(x) - V_{i-1}(y) \bowtie c\ \forall\ x - y \bowtie c \in guard(s_i))\ \forall\ 1 \leq i \leq \alpha \wedge guard(s_i) \neq True$
- $\forall\ 0 \leq i \leq \alpha, \forall x, y \in X$ such that $x \neq y$, $V_i(x) - V_i(y) \leq k$

In effect knowing whether the path is accepting or not is equivalent to solving the above set of difference constraints in reals. This set of difference constraints can be represented as a directed weighted "layered" graph as well with each layer having $|X| + 1$ vertices (Each layer essentially representing a Valuation). Note that in that graph **edges are present only between the adjacent layers**.

2.2 Representation of above linear constraints as weighted Graph

We will have $\alpha + 1$ layers, each layer corresponding to an integer in $[0, \alpha]$. Each layer will have $|X| + 1$ vertices (one for each register in X and one for special register t). Formally, let w be a walk as described above. Define a weighted graph $G_w^{\mathcal{A}} = (V_w^{\mathcal{A}}, E_w^{\mathcal{A}})$ corresponding to the walk w as follows.
- $V_w^{\mathcal{A}} = \{(i, x) \mid 0 \leq i \leq |w|,\ x \in X \cup \{t\}\}$
- $(0, x) \xrightarrow{\leq 0} (0, t)$, $(0, t) \xrightarrow{\leq 0} (0, x) \in E_w^{\mathcal{A}}$ for all $x \in X$ (Type-1 Edges)
- $(i, x) \xrightarrow{\leq up(s_i)(x)} (i-1, x)$, $(i-1, x) \xrightarrow{\leq -up(s_i)(x)} (i, x) \in E_w^{\mathcal{A}}$ if $up(s_i)(x) \neq t$ else $(i, x) \xrightarrow{\leq 0} (i-1, t)$, $(i-1, t) \xrightarrow{\leq 0} (i, x) \in E_w^{\mathcal{A}}$ $\forall\ 1 \leq i \leq |w|, \forall\ x \in X$ (Type-2 Edges)

- $(i-1, x) \xrightarrow{\bowtie c} (i-1, y) \in E_w^{\mathcal{A}} \forall\, x - y \bowtie c \in guard(s_i)$ for all $1 \leq i \leq |w|$ and $guard(s_i) \neq True$ (Type-3 Edges)
- $\forall\, x,\, y \in X$ such that $x \neq y$, $\forall\, 0 \leq i \leq |w|$, $(i, x) \xrightarrow{\leq k} (i, y) \in E_w^{\mathcal{A}}$ (Type-4 Edges)

Define the graph language of the TRA $\mathcal{A}$ to be

$$L_{\mathcal{G}}(\mathcal{A}) = \{G_w^{\mathcal{A}} \mid src(w) \in I \text{ and } dst(w) \in F\}$$

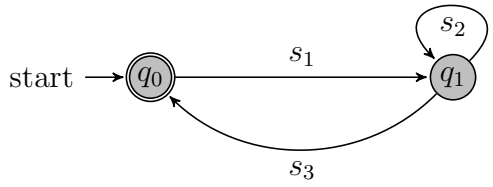
Now that we have specified $G_w^{\mathcal{A}}$ we introduce some notation and state some observations regarding these graphs. Let $v = (i, x) \in V_w^{\mathcal{A}}$, we define $\#(v) = i$ and we say that v is a *register point* if $x \in X$. Also for any non-empty $V \subseteq V_w^{\mathcal{A}}$ define $max(V) = \max\{\#(v) \mid v \in V\}$. $min(V)$ is defined correspondingly.
For any $g = (V_g, E_g) \in L_{\mathcal{G}}(\mathcal{A})$ the following statements hold

Observation 2.1. If $u \xrightarrow{\bowtie c} v \in E_g$ then $|\#(u) - \#(v)| \leq 1$

Observation 2.2. If $u \xrightarrow{\bowtie c} v \in E_g$ and $\#(u) > \#(v)$ then u is a register point. Also if $\#(u) < \#(v)$ then v is a register point

Observation 2.3. If $u_1 \xrightarrow{\bowtie c_1} v \in E_g$ and $u_2 \xrightarrow{\bowtie c_2} v \in E_g$ and $\#(v) > \#(u_1)$ and $\#(v) > \#(u_2)$ then $u_1 = u_2$

Observations 2 and 3 will be useful in chapter 3 where we try to prove the realizability of these graphs.
We will see an example to make things clear.



- s_1: a, $t - x \leq 5$, $x - y \leq 3$, $x := t$
- s_2: a, $t - y < 1$, $y := t$
- s_3: a, $t - y \geq 1$, $t - x > 7$, $x := t$, $y := y + 2$

$k = 4$

Figure 2.1: An example automata A

In the above example, q_0 is both initial and the final state and x and y are registers. s_1, s_2, s_3 are labels of the transitions. Assume $\Sigma = \{a\}$. Now consider the walk $w = (s_1, s_2, s_3)$. We will draw the graph corresponding to this walk.
The linear program corresponding to above walk would be
$\exists\, V_0, V_1, V_2, V_3$ such that

- $V_0(t) = V_0(x) = V_0(y) \wedge$
- $V_0(t) - V_0(x) \leq 5 \wedge V_0(x) - V_0(y) \leq 3 \wedge V_1(x) = V_0(t) \wedge V_1(y) = V_0(y) \wedge$
- $V_1(t) - V_1(y) < 1 \wedge V_2(y) = V_1(t) \wedge V_2(x) = V_1(x) \wedge$
- $V_2(t) - V_2(y) \geq 1 \wedge V_2(t) - V_2(x) > 7 \wedge V_3(x) = V_2(t) \wedge V_3(y) = V_2(y) + 2$
- $\bigwedge_{0 \leq i \leq 3} (V(x) - V(y) \leq 4 \wedge V(y) - V(x) \leq 4)$

The corresponding graph will look as follows

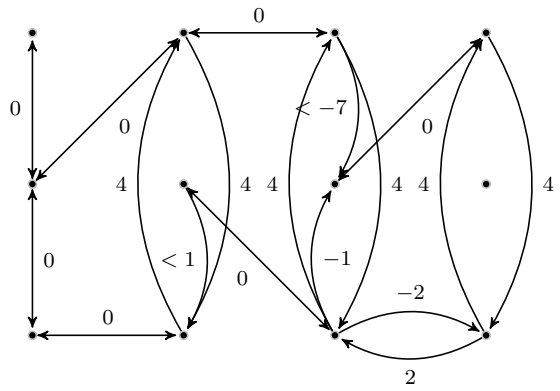


Figure 2.2: graph for the walk s_1, s_2, s_3 in Fig.2.1

In the above graph some redundant edges arising from conditions like $V_0(x) - V_0(y) \leq 3$ are not drawn and each vertical cross section represents a layer. In every layer top vertex represents register x and bottom vertex represents register y and middle vertex represents the special register t.

One interesting property of the above graph is that it can decomposed using $6 = 2 * (2 + 1)$ colours.In fact any graph generated by the TRA considered in Fig.2.1 can be decomposed using 6 colours which we will prove shortly.

Let $\beta \in \Sigma^+$ be any non-empty word. Then define $\mathbf{G}_\beta^{\mathcal{A}} = \{G_w^{\mathcal{A}} \mid \sigma(w) = \beta\}$

Lemma 2.1. Let $\mathcal{A} = (Q, \Sigma, X, I, F, Lab, \Delta, k)$ be a TRA.Then all the elements in $L_{\mathcal{G}}(\mathcal{A})$ can be decomposed using $2|X| + 2$ colours.

Proof. We prove an alternate statement which helps us in proving the above lemma.

Lemma 2.2. Let G be a graph such that it's vertex set can be partitioned into some k partitions such that each partition has same number of vertices(l) and the partitions can be numbered from 1 to k (k is the number of partitions) such that
- vertices in P_1 have edges to vertices in P_1 or P_2
- $\forall i \in \mathbb{N}, 2 \leq i < k$ vertices in P_i have edges to vertices in P_{i-1} or P_i or P_{i+1}
- vertices in P_k have edges to vertices in P_{k-1} or P_k

Now,if all the vertices in P_k are coloured(for decomposing the graph),then we additionally require only l extra colours for decomposing the whole graph

Proof of the above statement is by the induction on number of partitions.
Base case: $k = 1$, which means we have only a single layer and we coloured all vertices. Hence our statement is trivially true.
Induction step:
- Assume above statement to be true for $k = n$.We will now prove for $k = n + 1$.
- We now have $n+1$ partitions and we coloured all the vertices in the last partition.
- Now colour all the vertices present in partition P_n using l different colours.
- Remove all the edges from the graph whose both endpoints are coloured.

- Any vertex in P_{n+1} will now be disconnected from the original graph because if at all a vertex is connected we can remove that edge as both of it's endpoints are coloured.
- As a result we obtain a graph satisfying our hypothesis but having only n partitions.
- By Induction hypothesis we require at most l colours to decompose the above graph.
- But those l colours can be the ones we used for colouring vertices in P_{n+1} as they are not part of our graph anymore and we can choose the colours.
- Hence, the number of additional colours we used are l (for colouring the vertices in the penultimate partition)

From the previous lemma, we can see that for such a graph the tree width is bounded by $2l$. Now, observe that for any walk w, $G_w^{\mathcal{A}}$ satisfies the hypothesis of above lemma with $l = |X| + 1$.
$\implies$ the set of graphs represented by $\mathcal{A}$ have bounded tree width □

2.2.1 Decomposition of Fig.2.2 using 6 colours

We will now show the decomposition of Fig.2.2 using 6 colours. First colour the last 3 vertices.

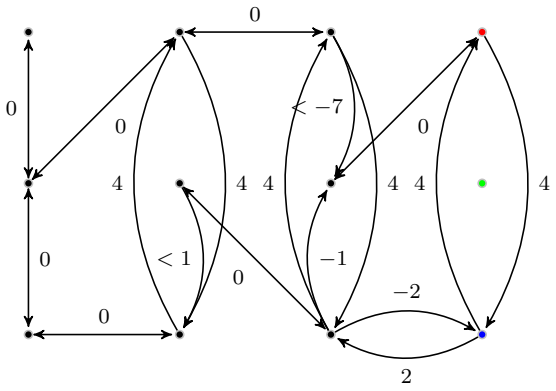


Figure 2.3: step 1

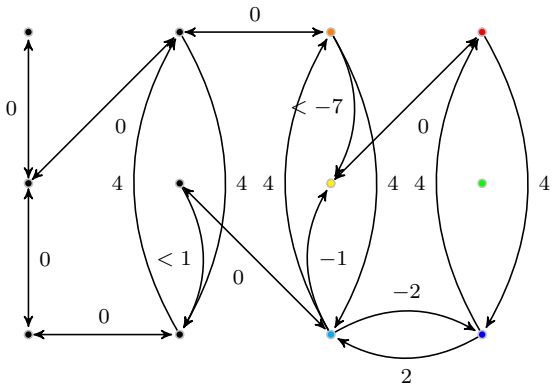


Figure 2.4: step 2

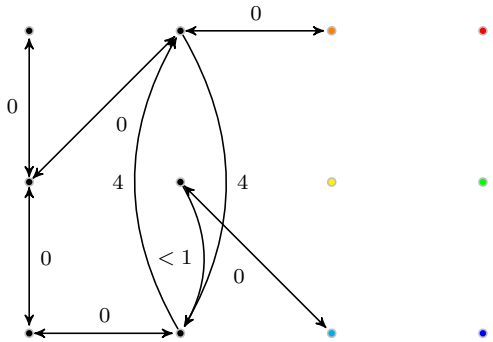


Figure 2.5: step 3

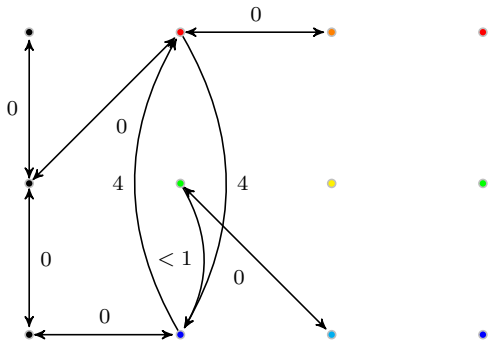


Figure 2.6: step 4

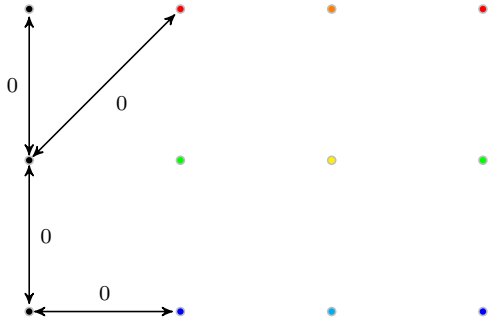


Figure 2.7: step 5

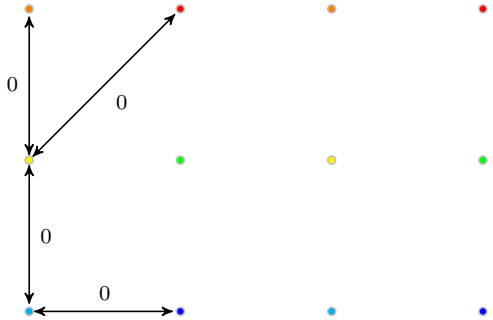


Figure 2.8: step 6

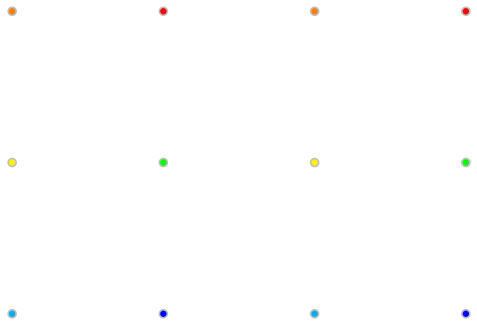


Figure 2.9: step 7

2.3 Graph behaviors for other models

We saw the graph behaviors for Model 2b TRA. The graph behaviors for Model 1 TRA's can be easily obtained by removing the Type-4 Edges from each graph. The graph behaviors for Model 2a TRA do not change much as any Model 2a TRA is also a Model 2b TRA. Also notice that all the observations made for Model 2b graphs hold for these model graphs as well.

2.4 Graph behaviors for PDTRA

By the same method, we can also construct the graph behaviors for a PDTRA. In these graphs, in each layer there will be $|X|+2$ points and also there may be edges between layers which are not adjacent as a push and a matching pop may not be adjacent. Hence, the above decomposition won't work for Model 2b PDTRA.

Chapter 3

L-boundedness

3.1 Definitions

We begin with the definition of Size of a path in a weighted graph and some other useful definitions.

Definition 3.1. (Size of a path) The size of a path P is defined to be

$$size(P) \;=\; \max\{wt(P') \mid P' \text{ is a subpath of } P\}$$

Definition 3.2. (Cycle decomposition) A cycle C in a weighted graph G is said to be decomposable into cycles $C_1, C_2, \cdots, C_n \in G \iff$ there are integers $a_1, a_2, \cdots, a_n$ such that

$$|C|_e \;=\; \sum_{i=1}^{n} a_i \, |C_i|_e \; \forall \, e \in E$$

Definition 3.3. (L-coverable) For $L \in \mathbb{N}$, a cycle C is said to be L-coverable iff it can be decomposed into cycles $C_1, C_2, \cdots, C_n$ such that $size(C_i) < L \forall 1 \leq i \leq n$

Definition 3.4. (L-cycle bounded) A weighted graph G is said to be L-cycle bounded if every cycle $C \in G$ is L-coverable

Definition 3.5. (L-bounded) Let $\mathcal{G}$ be a set of weighted graphs. Then $\mathcal{G}$ is said to be L-bounded if every graph $G \in \mathcal{G}$ is L-cycle bounded and L-weight bounded

Observation 3.1. If a cycle C can be decomposed into cycles $C_1, C_2, \cdots, C_n$ where each of C_i is L-coverable then C is L-coverable

Observation 3.2. G is L-cycle bounded $\iff$ all simple cyclic paths C are L-coverable

3.2 L-boundedness for TRA

Theorem 3.1. Let $\mathcal{A} = (Q, \Sigma, X, I, F, Lab, \Delta, k)$ be a TRA, then $\exists\, L \in \mathbb{N}$ such that $L_{\mathcal{G}}(\mathcal{A})$ is L-bounded

Proof. Let M be the max constant in the TRA. Let $L = 5 * \max(M, k) + 1$. We will show that for any graph $g = (V_g, E_g) \in L_{\mathcal{G}}(\mathcal{A})$, g is L-cycle bounded and L-weight bounded.

Proof of g being L-cycle bounded

Let $C = v_1 v_2 \cdots v_\beta v_{\beta+1} (= v_1) = e_1 e_2 \cdots e_\beta$ be a simple cycle in g. Let $V_C = \{v_i | 1 \leq i \leq \beta\}$. We show that C is L-coverable by induction on $\vartheta = max(V_C) - min(V_C)$.
Base case ($\vartheta = 0$)
We will prove that C is L-coverable by induction on $|V_C|$
If $|V_C| = 2 \implies C = v_1 v_2 v_1 = e_1 e_2$.
$\implies size(C) < 2 * \max(M, k) + 1 \leq L$.
similarly $|V_C| = 3 \implies size(C) < 3 * \max(M, k) + 1 \leq L$.
Let $|V_C| = \beta > 3$. Assume that for $\vartheta = 0$ and $|V_C| < \beta$, C is L-coverable
Since cycle C is simple and every vertex of the cycle has same $\# \implies$ at most one vertex of the cycle is not a register point. W.l.o.g, we can assume it to be v_1 $\implies$ v_2 and v_β are register points. Also $v_2 \neq v_\beta$ as C is simple and $\beta > 2$.
$\implies v_2 \xrightarrow{\leq k} v_\beta\ (e')$ and $v_\beta \xrightarrow{\leq k} v_2\ (e'') \in E_g$ and these edges are not part of the cycle as $\beta > 3$
$\implies C = v_1 v_2 \cdots v_\beta v_1 = e_1 e_2 \cdots e_\beta = C_1 + C_2 - C_3$ where
$C_1 = v_1 v_2 \xrightarrow{\leq k} v_\beta v_1 = e_1 e' e_\beta$
$C_2 = v_2 \cdots v_\beta \xrightarrow{\leq k} v_2 = e_2 \cdots e_{\beta-1} e''$
$C_3 = v_2 \xrightarrow{\leq k} v_\beta \xrightarrow{\leq k} v_2 = e' e''$
C_1 and C_3 are L-coverable as $|V_{C_1}| = 3$ and $|V_{C_3}| = 2$.
Also $|V_{C_2}| = \beta - 1 < \beta \implies C_2$ is also L-coverable $\implies C$ is L-coverable

Before doing the induction step we consider a particular set of simple cycles and show that they are L-coverable. Consider a simple cycle $C = v_1 v_2 \cdots v_\beta v_{\beta+1} (= v_1) = e_1 e_2 \cdots e_\beta$ such that $\#(v_1) = \#(v_i)\ \forall\ 4 \leq i \leq \beta$ and $\#(v_2) = \#(v_3) = \#(v_1) + 1$. Let us call such cycles as *special cycles*. If $\beta \leq 5$, then $size(C) < 5 * \max(M, k) + 1 \leq L$. So, assume $\beta > 5$.
$\implies v_1, v_4, \cdots, v_\beta$ have same $\# \implies$ at most one of them is not a register point
Case -1 v_4 and v_β are register points
then $C = C_1 + C_2 - C_3$ where
$C_1 = v_1 v_2 v_3 v_4 \xrightarrow{\leq k} v_\beta v_1$
$C_2 = v_4 \cdots v_\beta \xrightarrow{\leq k} v_4$
$C_3 = v_4 \xrightarrow{\leq k} v_\beta \xrightarrow{\leq k} v_4$
$size(C_1) < 5 * \max(M, k) + 1 \leq L$ and for C_2 we have $\tau = 0$ and also $size(C_3) < L \implies C$ is L-coverable
Case-2 v_1 and $v_{\beta-1}$ are register points

then $C = C_1 + C_2 - C_3$ where
$C_1 = v_1 \xrightarrow{\leq k} v_{\beta-1}v_\beta v_1$
$C_2 = v_1v_2v_3v_4\cdots v_{\beta-1} \xrightarrow{\leq k} v_1$
$C_3 = v_1 \xrightarrow{\leq k} v_{\beta-1} \xrightarrow{\leq k} v_1$
C_1 and C_3 are L-coverable and observe that C_2 is similar to C but with a smaller length. Hence with an inductive argument we can show that C is L-coverable
Induction step
Assume that if $max(V_C) - min(V_C) \leq \tau$ then C is L-coverable
Consider a simple cycle C such that $max(V_C) - min(V_C) = \tau + 1$
Let $C = v_1v_2\cdots v_\beta v_{\beta+1}(= v_1) = e_1e_2\cdots e_\beta$
Also if $\beta \leq 5 \implies size(C) < L$ so assume $\beta > 5$
W.l.o.g, assume $\#(v_1) = min(V_C)$ and $\#(v_2) = min(V_C) + 1$
Let $Y_C = \{i \mid \#(v_i) = min(V_C) \wedge 1 < i \leq \beta\}$ and
$Z_C = \{i \mid \#(v_i) = min(V_C) + 1 \wedge 2 < i \leq \beta\}$
$Z_C \neq \phi$ as $Z_C = \phi \implies v_3 \in Y_C$ but from **Observation 2.3** this would mean $v_1 = v_3$ contradicting $\beta > 5$
Case-1 $Y_C = \phi$
$\implies \#(v_i) > min(V_C) \,\forall\, 1 < i \leq \beta$
$\implies$ v_2 and v_β are register points such that $\#(v_2) = \#(v_\beta)$
$\implies C = C_1 + C_2 - C_3$ where
$C_1 = v_1v_2 \xrightarrow{\leq k} v_\beta v_1$
$C_2 = v_2\cdots v_\beta \xrightarrow{\leq k} v_2$
$C_3 = v_2 \xrightarrow{\leq k} v_\beta \xrightarrow{\leq k} v_2$
C_1 and C_3 are L-coverable and $max(V_{C_2}) - min(V_{C_2}) = \tau \implies C_2$ is also L-coverable
$\implies$ C is L-coverable
Case-2 $Y_C \neq \phi$
Consider $y = min(Y_C)$ and $z = max(Z_C)$

sub case-1 $z \neq y - 1$
then $z > y$ as $y - 1 \in Z_C$ and $y \notin Z_C$
Also observe that v_{y-1} and v_z are register points with same #
$\implies C = v_1v_2\cdots v_{y-1}v_y\cdots v_z\cdots v_\beta v_1 = C_1 + C_2 - C_3$ where
$C_1 = v_1v_2\cdots v_{y-1} \xrightarrow{\leq k} v_z\cdots v_\beta v_1$
$C_2 = v_{y-1}v_y\cdots v_z \xrightarrow{\leq k} v_{y-1}$
$C_3 = v_{y-1} \xrightarrow{\leq k} v_z \xrightarrow{\leq k} v_{y-1}$
C_3 is L-coverable and also observe that $|Y_{C_1}| + |Y_{C_2}| = |Y_C|$ and also $|Y_{C_1}| < |Y_C|$ as $v_y \in Y_C$ but $\notin Y_{C_1} \implies |Y_{C_2}| < |Y_C|$. Since we already proved that C is L-coverable when $Y_C = \phi$ by inductive argument we can say that C_1 and C_2 are L-coverable and hence C is also L-coverable

sub case-2 $z = y - 1$
$\implies \#(v_i) = min(V_C)$ if $i \geq y$ and $\#(v_i) > min(V_C)$ if $1 < i < y$.
Also v_2 and v_{y-1} are register points with same #.

$\implies C = C_1 + C_2 - C_3$ where
$C_1 = v_1 v_2 \xrightarrow{\leq k} v_{y-1} v_y \cdots v_\beta v_1$
$C_2 = v_2 \cdots v_{y-1} \xrightarrow{\leq k} v_2$
$C_3 = v_{y-1} \xrightarrow{\leq k} v_2 \xrightarrow{\leq k} v_{y-1}$
Observe that C_1 is a special cycle hence it is L-coverable. Also $max(V_{C_2}) - min(V_{C_2}) = \tau \implies C_2$ is L-coverable, also C_3 is L-coverable. Hence C is L-coverable
This proves that g is L-cycle bounded

Proof of g being L-weight bounded

For any $g \in L_{\mathcal{G}}(A)$ g is $max(M,k)$-weight bounded. As $L \geq max(M,k) \implies g$ is L-weight bounded.
$\implies L_{\mathcal{G}}(A)$ is L-bounded □

The set of graphs being L-bounded gives us a MSO formula for realizability of graphs in that set.

3.3 L-boundedness for PDTRA

In this section, we will see that there are examples of Model 2b PDTRA whose graph behaviors are not L-bounded. Hence we cannot use L-boundedness for realizability of PDTRA.

Consider the following PDTRA

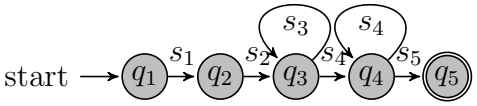


Figure 3.1: Counter Example $\mathcal{A}'$

- s_1: $x - t \leq c_1$, push(t)
- s_2: $t - x \leq c_2$, $x := t$
- s_3: $t - x \leq c_3$, $x := t$
- s_4: $x - t \geq c_4$, $x := t$
- s_5: $x - t \geq c_4$, $s - t \leq c_5$, pop()

Assume $\Sigma = \{a\}$. Also x is the only register so the bounded span constant is not required. For simplicity, assume $c_1 = c_2 = c_3 = c_4 = c_5 = 1$

Theorem 3.2. $L_{\mathcal{G}}(\mathcal{A}')$ is not L-cycle bounded for any L

Proof. consider any number $m \in \mathbb{N}$. We will show a walk w_m whose corresponding graph $G^{\mathcal{A}'}_{w_m}$ in $L_{\mathcal{G}}(\mathcal{A}')$ is not m-cycle bounded.

Let $w_m = s_1 s_2 s_3^m s_4^m s_5$. It is a valid walk as it starts in an initial state and ends up in a final state and at s_5, the stack is empty and at any intermediate point, a pop() doesn't happen from an empty stack. The graph corresponding to w_m is

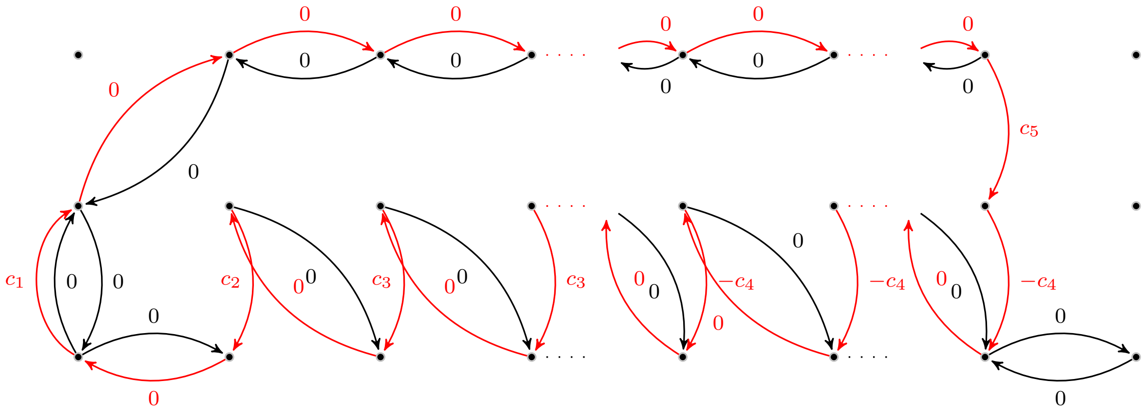


Figure 3.2: graph for the walk $w_m = s_1 s_2 s_3^m s_4^m s_5$ in Fig.3.1

In the above figure the top set of vertices correspond to stack register s during various points of time, the middle ones correspond to special register t and the bottom ones represent register x. Consider the simple cycle whose edges are highlighted by red in Fig.3.2. Observe that the size of the cycle is $c_1 + c_2 + m * c_3 + c_5$ $= m + 4$ and also it cannot decomposed into any cycles with smaller size $\implies$ $G_{w_m}^{\mathcal{A}'}$ is not m-cycle bounded $\implies$ $L_{\mathcal{G}}(\mathcal{A}')$ is not m-cycle bounded for any m □

Chapter 4

Emptiness of TRA

In chapter 2, we saw graph behaviors for a given timed register automata and also that all the graph behaviors of a given automata have a uniform tree width bound. Also in chapter 3, we saw that the graph language of any Model 2b TRA is L-bounded, where the bound depends on the max constant and the bounded span constant. In this chapter, we utilise both these properties of Model 2b TRA to come up with an algorithm for deciding the emptiness problem.

To begin with, we will define realizability for a weighted graph.

Definition 4.1 (Realizabilty)**.** A weighted graph $G = (V, E)$ is realizable iff there exists a map $f : V \to \mathbb{R}$ such that $\forall\ e = (u, v, \bowtie, z) \in E,\ f(v) - f(u) \bowtie z$

Also, we denote the set of all realizable graphs with *Realizable*.
Now observe that for a Model 2b-TRA $\mathcal{A}$

$$L(\mathcal{A}) = \phi \iff (I \cap F = \phi) \wedge (L_{\mathcal{G}}(\mathcal{A}) \cap \mathit{Realizable} = \phi) \tag{4.1}$$

Checking if $I \cap F = \phi$ is trivial and can be done once we are given the automaton. Hence, the emptiness problem boils down to deciding whether $L_{\mathcal{G}}(\mathcal{A}) \cap \mathit{Realizable}$ is empty or not. The following two theorem's from [1] help us in deciding the above expression.

Theorem 4.1. Let $\mathcal{A}$ be a Model-2b TRA. Then $\exists$ a tree automaton $T_{\mathcal{A}}$ such that $\tau \in L(T_{\mathcal{A}}) \iff [[\tau]] \in L_{\mathcal{G}}(\mathcal{A})$

The existence of such a tree automaton could be attributed to the fact that $L_{\mathcal{G}}(\mathcal{A})$ has a tree-width bound which is proved in chapter 2

Theorem 4.2. Let $G = (V, E)$ be an M-bounded weigthed graph. Then $\exists$ an MSO formula ψ_M such that $G \in L(\psi_M) \iff G \in \mathit{Realizable}$

Since we proved that $L_{\mathcal{G}}(\mathcal{A})$ is L-bounded for some finite L, from theorem 4.2, we can say that there is a mso formula ψ which decides realizability in $L_{\mathcal{G}}(\mathcal{A})$. We can construct a tree automaton T_ψ such that it accepts only those tree terms whose associated graph satisfies ψ. Taking $T_{\mathcal{A}} \cap T_\psi$ and checking for emptiness of this tree automata decides our problem

Bibliography

[1] S. Akshay, Paul Gastin, Vincent Jugé, and Shankara Narayanan Krishna. Logically well-behaved timed systems. unpublished.

[2] S. Akshay, Paul Gastin, Shankara Narayanan Krishna, and Ilias Sarkar. Towards an efficient tree automata based technique for timed systems. *CoRR*, abs/1707.02297, 2017.

[3] Mikołaj Bojańczyk and Sławomir Lasota. A machine-independent characterization of timed languages. *Automata, Languages, and Programming*, pages 92–103, 2012.