UPPSALA
UNIVERSITET

IT 12 034
Examensarbete 15 hp
Juli 2012

Zone-Based Reachability Analysis of Dense-Timed Pushdown Automata

Kristiina Ausmees

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Zone-Based Reachability Analysis of Dense-Timed Pushdown Automata

Kristiina Ausmees

Proving that programs behave correctly is a matter of both great theoretical interest as well as practical use. One way to do this is by analyzing a model of the system in question in order to determine if it meets a given specification. Real-time recursive systems can be modeled by dense-timed pushdown automata, a model which combines the behaviours of classical timed automata and pushdown automata. The problem of reachability has been proven to be decidable for this model. The algorithm that solves this problem relies on constructing a classical pushdown automaton that mimics the behaviour of a given timed pushdown automaton by means of an abstraction that uses regions as a symbolic representation of states. The drawback of this approach is that the untimed automaton produced generally contains a very large number of states. This report proposes a method of generalizing this abstraction by using zones instead of regions, in order to minimize the number of states in the untimed automaton.

Handledare: Mohamed Faouzi Atig
Ämnesgranskare: Parosh Abdulla
Examinator: Olle Gällmo
IT 12 034
Tryckt av: Reprocentralen ITC

Table of Contents

1 Introduction

Model checking is a formal verification technique for asserting that a program fulfils its specification. It requires a model of the system in question and a specific property expressed as a mathematical formula in some formal logic. The verification process consists of systematically checking if the model satisfies the given property. This is done by exploring all possible system states, e.g. all the states that can be reached in the model.

The model should be able to express the behaviour of the system in an accurate and precise way. One common way to model systems is using finite-state automata. The states in the automaton then describe the possible values of the variables in the system, and the transitions describe how the system moves from one state to another. When using automata as models the desired property to check for can often be reduced to a reachability problem e.g., "Is it possible to reach a state such that ϕ holds?", where ϕ is a formula expressing some attribute of variables in the system. However, some models have the property that they have an infinite number of possible system states, in which case the reachability analysis is not trivial. An algorithm that explores all the states would not necessarily terminate.

In the case of timed automata this is solved by creating a finite automaton whose behaviour mimics the behaviour of the timed automaton, so that a state is reachable in the timed model iff it is reachable in the finite model. This approach, described in [1], is called the region abstraction and the resulting finite automaton is called the region automaton. This approach has the drawback that the region automaton generally has a large number of states, and is therefore not suitable for implementation. One method of solving this problem is to create a more general finite automaton that abstracts the behaviour of the timed automaton in a courser manner. This approach relies on zones as a symbolic representation of possible system states rather than regions, and produces the so called zone automaton, [5],[4]

For timed pushdown automata, TPDA, there exists a corresponding region abstraction, described in [3]. The aim of this report is to propose a method for generalizing the region abstraction for TPDA in the same manner as it is done in the case with timed automata, i.e. by creating a zone automaton that mimics the behaviour of the TPDA.

2 Background

2.1 Timed Automata

Timed automata are an extension of classical finite automata introduced by Alur and Dill in [2], [1] . The classical model is enriched with a set of clock-variables that can be used to simulate the passage of time. By adding constraints on the transitions so that they can only be executed if the clocks have certain values, and the possibility to reset clocks, time-related properties such as "does event a occur within x time units?" can be expressed. This makes timed automata a suitable model for many real-time systems.

Definition 1. *A* **clock constraint** g *over a set of clocks* X *is an expression on the form* $x \preceq c$ *where* $x \in X$, $\preceq \in \{<, \leq, =, \geq, >\}$ *and* $c \in \mathbb{N}$.
$C(X)$ *denotes the set of clock constraints over* X.

Definition 2. *A* **clock valuation** $\mathtt{X}$ *over a set* X *of clocks is a total function* $\mathtt{X} : X \to \mathbb{R}^{\geq 0}$ *that associates a positive real value with every clock in* X.
$\mathbb{R}^X$ *denotes the set of clock valuations over* X.
For $d \in \mathbb{R}^{\geq 0}$, *define* $\mathtt{X}+d$ *as the valuation* $(\mathtt{X}+d)(x) = \mathtt{X}(x)+d$ *for all* $x \in X$ *and* $[r \leftarrow d]\mathtt{X}$ *as the valuation that satisfies the following conditions*

- $[r \leftarrow d]\mathtt{X}(x) = \mathtt{X}(x)$ *if* $x \notin r$
- $[r \leftarrow d]\mathtt{X}(x) = d$ *if* $x \in r$

The interpretation of clock valuations and constraints is that a valuation satisfies a constraint on the clock x if the value given to x by the valuation satisfies the constraint.

Definition 3. *Given a constraint* g *on the form* $x \preceq c$ *and a valuation* $\mathtt{X}$, $\mathtt{X}$ **satisfies** g *, denoted* $\mathtt{X} \models g$ *, if* $\mathtt{X}(x) \preceq c$.

Definition 4. *A* **timed automaton** $\mathcal{A}$ *is a tuple* $\langle \Sigma, S, T, Init, F, X \rangle$ *where*

- Σ *is a finite alphabet of actions*
- S *is a finite set of states*
- $T \subseteq S \times (C(X) \times \Sigma \times 2^X) \times S$ *is a finite set of transitions*
- $Init \subseteq S$ *is the set of initial states*
- $F \subseteq S$ *is the set of final states*
- X *is a finite set of clocks*

Definition 5. *A* **configuration** *of a timed automaton is a pair* $\langle s, \mathtt{X} \rangle \in S \times \mathbb{R}^X$ *where* s *is the current location and* $\mathtt{X}$ *gives the current values of the clocks.*

Definition 6. *The* **transition relation** $\longrightarrow$ *for timed automata is composed of two kinds of transitions:*

- *delay transitions:* $\langle s, \mathtt{X}\rangle \xrightarrow{d} \langle s, \mathtt{X} + d\rangle$ *for* $d \in \mathbb{R}^{\geq 0}$
- *action transitions:* $\langle s, \mathtt{X}\rangle \xrightarrow{a} \langle s', \mathtt{X}'\rangle$ *if and only if there exists* $s \xrightarrow{g,a,r} s' \in T$ *such that*

$$\begin{cases} \mathtt{X} \models g \\ \mathtt{X}' = [r \leftarrow 0]\mathtt{X} \end{cases}$$

From a configuration it is possible to delay some time, in which case all clocks are increased by the same amount of time, or take an action transition if the current clock valuation satisfies the guard. This is a discrete transition in the sense that the clock values are not increased. Some clocks may also be reset to zero. Thus, the transition relation defines the manner in which the automaton can move between configurations.

The initial configuration α_{init} is defined as $\langle s_{init}, \mathtt{X}_{init}\rangle$, where $s_{init} \in Init$ and $\mathtt{X}_{init}(x) = 0$ for all $x \in X$. The run starts at a configuration in which all clocks have the value 0, and the current state is an initial state. A configuration $\alpha = \langle s, \mathtt{X}\rangle$ is said to be reachable if there is a sequence of configurations $\alpha_1 \cdots \alpha_n$ such that $\alpha_{init} \longrightarrow \alpha_1 \longrightarrow \cdots \longrightarrow \alpha_n \longrightarrow \alpha$. A state s is reachable if there exists a configuration $\alpha = \langle s, \mathtt{X}\rangle$ such that α is reachable for some $\mathtt{X}$. The reachability problem for timed automata consists of determining whether or not a state s is reachable.

2.2 Dense-Timed Pushdown Automata

Dense-timed pushdown automata are an extension of classical pushdown automata. In addition to the behaviour of a pushdown automaton, namely the existence of a stack, the automaton also operates on a set of real-valued clocks. Furthermore, each symbol in the stack is associated to a real value representing its age. As in the case with timed automata, the transitions can be conditioned by the values of the clocks. In this sense, the behaviour of dense-timed pushdown automata combines that of timed automata and pushdown automata. This makes them a suitable model for real-timed recursive systems, as timed automata can model the real-time aspect, pushdown automata can model procedure calls, and neither can model the behaviour of both.

The set of stack operations O that can be performed in a TPDA with a set X of clocks contains the following operations:

- The empty operation *nop* which does not change the stack content.
- The test operation $x \in I?$ where $x \in X$ is a clock and I is an interval, which may only be fired if the value of x lies in I.
- The assignment operation $x \leftarrow I$ where $x \in X$ is a clock and I is an interval, which assigns a new value in the interval I to the clock x.
- The pop operation $pop(a, I)$ where $a \in \Gamma$ and I is an interval, which pops the topmost stack symbol in the case that it is a and its value lies in I.

– The push operation $push(a, I)$ where $a \in \Gamma$ and I is an interval, which pushes a to the top of the stack and gives it an age that lies in I.

Definition 7. *Let X be a finite set of clocks. A* **timed pushdown autmaton** *$\mathcal{T}$ is a tuple $\langle S, s_{init}, \Gamma, \Delta \rangle$ where*

– *S is a finite set of states*
– *$s_{init} \in S$ is the initial state*
– *Γ is a finite set of stack symbols*
– *$\Delta \subseteq S \times O \times S$ is a finite set of transitions*

Definition 8. *A* **stack content** *is a word $q \in (\Gamma \times \mathbb{R}^{\geq 0})^*$ of tuples each containing a stack symbol and its age. For a stack content $q = \langle a_1, v_1 \rangle \cdots \langle a_n, v_n \rangle$ and $d \in \mathbb{R}^{\geq 0}$, define $q + d$ to be the stack content $\langle a_1, v_1 + d \rangle \cdots \langle a_n, v_n + d \rangle$.*

Definition 9. *A* **configuration** *γ of a TPDA is a tuple $\langle s, \mathtt{X}, q \rangle$ where $s \in S$ is the current control state, $\mathtt{X}$ is the clock valuation that defines the values of the clocks, and q is the content of the stack.*

Definition 10. *The* **transition relation** *for TPDA $\rightsquigarrow$ consists of two kinds of transitions:*

– *Timed transitions: $\langle s, \mathtt{X}, q \rangle \overset{d}{\rightsquigarrow}_{Time} \langle s, \mathtt{X} + d, q + d \rangle$, for $d \in \mathbb{R}^{\geq 0}$*
– *Discrete transitions:$\langle s, \mathtt{X}, q \rangle \overset{t}{\rightsquigarrow}_{Disc} \langle s', \mathtt{X}', q' \rangle$ iff there exists a $t = \langle s, op, s' \rangle \in \Delta$ such that one of the following holds:*

1. $\begin{cases} op = nop \\ q' = q \\ \mathtt{X}' = \mathtt{X} \end{cases}$
2. $\begin{cases} op = x \in I? \\ q' = q \\ \mathtt{X}' = \mathtt{X} \\ \mathtt{X}(x) \in I \end{cases}$
3. $\begin{cases} op = x \leftarrow I \\ q' = q \\ \mathtt{X}' = [x \leftarrow d]\mathtt{X} \text{ where } d \in I \end{cases}$
4. $\begin{cases} op = pop(a, I) \\ \mathtt{X}' = \mathtt{X} \\ q = q' \langle a, v \rangle \text{ where } v \in I \end{cases}$
5. $\begin{cases} op = push(a, I) \\ \mathtt{X}' = \mathtt{X} \\ q' = q \langle a, v \rangle \text{ where } v \in I \end{cases}$

Figure 1 shows an example of a run in a TPDA. The values of the clocks and ages of the stack symbols are given at each configuration γ. The current state at each configuration is not shown.

As in the case with timed automata, the transition relation describes the manner in which the system can move between configurations. The initial configuration for TPDA γ_{init} is defined as $\langle s_{init}, \mathtt{X}_{init}, \epsilon \rangle$. The reachability of configurations and states for TPDA are defined in the same manner as for timed automata. Thus, the reachability problem for TPDA also consists of determining whether or not a state s is reachable.

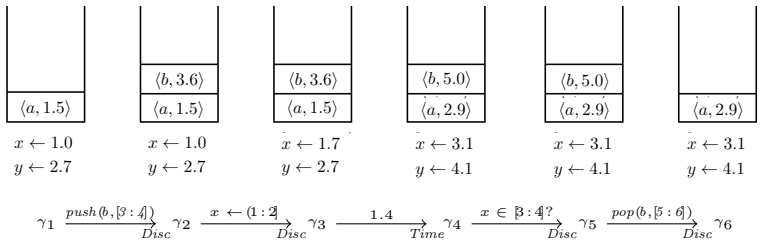

Fig. 1. Example of how a TPDA can move between configurations.

2.3 Region-Based Reachability Analysis of Timed Automata

The reachability problem for timed automata is decidable. The technique used in the proof relies on a construction proposed in [1]. This construction is the so called region abstraction in which the behaviour of a timed automaton is abstracted to that of a finite (untimed) automaton, for which the reachability problem is decidable. The reason why the methods that can be used to prove reachability in finite automatons cannot directly be applied to timed automatons is that the number of configurations is infinite in the case with timed automatons, due to the real-valued clocks. The region abstraction relies on creating an equivalence relation for the set of configurations. This equivalence relation contains a finite number of equivalence classes, and has the property that if two configurations are equivalent then the configurations that can be reached from them by enabling transitions are also equivalent. Using this one can, given a timed automaton $\mathcal{A}$, create a finite automaton $\mathcal{R}(\mathcal{A})$ which mimics the behaviour of $\mathcal{A}$.

Definition 11. *Given a timed automaton $\mathcal{A} = \langle \Sigma, S, T, Init, F, X \rangle$, let M be the maximal constant which appears in the clock-constraints in $\mathcal{A}$. For $d \in \mathbb{R}^{\geq 0}$ let $fract(d)$ denote the fractional part of d and $\lfloor d \rfloor$ the integral part of d. Two clock valuations $\mathtt{X}$ and $\mathtt{X}'$ are said to be equivalent, denoted $\mathtt{X} \sim \mathtt{X}'$ if the following conditions hold:*

- *For all $x \in X$, either $\lfloor \mathtt{X}(x) \rfloor = \lfloor \mathtt{X}'(x) \rfloor$ or both $\mathtt{X}(x)$ and $\mathtt{X}'(x)$ are greater than M*
- *For all $x, y \in X$ such that $\mathtt{X}(x) \leq M$ and $\mathtt{X}(y) \leq M$:*
  $fract(\mathtt{X}(x)) \leq fract(\mathtt{X}(y)) \iff fract(\mathtt{X}'(x)) \leq fract(\mathtt{X}'(y))$
- *For all $x \in X$ such that $\mathtt{X}(x) \leq M$:*
  $fract(\mathtt{X}(x)) = 0 \iff fract(\mathtt{X}'(x)) = 0)$

An equivalence class of the relation $\sim$ is called a **region**. *Two configurations $\alpha = \langle s, \mathtt{X} \rangle$ and $\alpha' = \langle s', \mathtt{X}' \rangle$ are said to be equivalent if $s = s'$ and $\mathtt{X} \sim \mathtt{X}'$.*

For two clocks x, y and a maximal constant $M = 3$, the resulting regions are illustrated in Figure 2. In the figure, each line, dot and open space corresponds to a region.

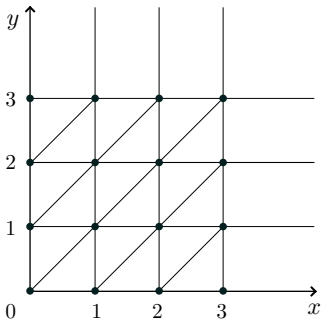

Fig. 2. Example of region partitioning

Definition 12. *Given a timed automaton $\mathcal{A} = \langle \Sigma, S, T, Init, F, X \rangle$, the corresponding* **region automaton**, *$\mathcal{R}(\mathcal{A})$, is a finite automaton defined as follows.*

- *The alphabet of $\mathcal{R}(\mathcal{A})$ is Σ*
- *The states of $\mathcal{R}(\mathcal{A})$ are on the form $\langle s, R \rangle$ where $s \in S$ and R is a region*
- *The initial states of $\mathcal{R}(\mathcal{A})$ are on the form $\langle s_{init}, R_{init} \rangle$ where $s_{init} \in Init$ and R_{init} is the region containing $\mathtt{X}_{init}$*
- *$\mathcal{R}(\mathcal{A})$ has a transition $\langle s, R \rangle \xrightarrow{a} \langle s', R' \rangle$ iff there is a transition $s \xrightarrow{g,a,r} s' \in T$, a valuation $\mathtt{X} \in R$ and $t \geq 0$ such that $[r \leftarrow 0](\mathtt{X} + t) \in R'$ and $(\mathtt{X} + t) \models g$*

Since the region automaton $\mathcal{R}(\mathcal{A})$ simulates the behaviour of $\mathcal{A}$, the problem of deciding if a state is reachable in $\mathcal{A}$ reduces to the reachability problem in $\mathcal{R}(\mathcal{A})$.

2.4 Zone-Based Reachability Analysis of Timed Automata

The drawback of the region abstraction is that the region automaton contains a large number of states, and it is therefore not often used in practice. The most common technique for analyzing reachability properties in timed automata instead relies on manipulating sets of regions, or zones, to explore the reachable configurations.

Definition 13. *A* **zone constraint** *c is on the form $x - y \preceq n$ where x, y are variables, $n \in \mathbb{Z}$ and $\preceq \in \{\leq, <\}$. A* **zone** *Z over Y is a finite conjunction of zone constraints over variables in $Y \cup \{0\}$, where 0 which is a special variable whose value is always 0 in order to express upper and lower bounds for the other variables.*

The solution set of a zone Z is the set of valuations that satisfy the constraints in Z. This set is denoted by $sol(Z)$. One solution set can be represented by several different sets of constraints. This is because one constraint can be a more general

form of another. Consider the constraints $0 - x \leq 0$, $0 - y \leq -3$, $y - x \leq 2$. These define the same solution set as the constraints $0 - x \leq -1$, $0 - y \leq -3$, $y - x \leq 2$.

However, each solution set can be described by a set of constraints on a canonical form. This is the set which contains the tightest constraint on each difference. It is assumed that every zone is on this form, and thus $c \in Z$ is used to denote that c is one of the constraints which occurs in Z.

Definition 14. *A valuation* $\mathtt{X}$ *is said to* **satisfy** Z *if* $\mathtt{X} \in sol(Z)$*. This is denoted by* $\mathtt{X} \in Z$*.*

For a zone Z, define the following operations:

- Letting time pass, defined by $\overrightarrow{Z} = \{\mathtt{X} + t \mid \mathtt{X} \in Z \wedge t \in \mathbb{R}^{\geq 0}\}$
- Adding a constraint, defined by $Z \cap g = \{\mathtt{X} \mid \mathtt{X} \in Z \wedge \mathtt{X} \models g\}$
- Reset to zero r of Z, defined by $[r \leftarrow 0]Z = \{[r \leftarrow 0]\mathtt{X} \mid \mathtt{X} \in Z\}$

Figure 3 gives an example of these operations on a zone Z containing two variables x and y.

Definition 15. *Given a timed automaton* $\mathcal{A} = \langle \Sigma, S, T, Init, F, X \rangle$*, the corresponding* **zone automaton***,* $\mathcal{Z}(\mathcal{A})$*, is an automaton defined as follows.*

- *The alphabet of* $\mathcal{Z}(\mathcal{A})$ *is* Σ
- *The states of* $\mathcal{Z}(\mathcal{A})$ *are on the form* $\langle s, Z \rangle$ *where* $s \in S$ *and* Z *is a zone*
- *The initial states of* $\mathcal{Z}(\mathcal{A})$ *are on the form* $\langle s_{init}, Z_{init} \rangle$ *where* $s_{init} \in Init$ *and* Z_{init} *is the zone containing* $\mathtt{X}_{init}$
- *For every transition* $s \xrightarrow{g,a,r} s' \in T$ *and zone* Z*,* $\mathcal{Z}(\mathcal{A})$ *has a transition* $\langle s, Z \rangle \xrightarrow{a} \langle s', Z' \rangle$*, where* $Z' = [r \leftarrow 0](\overrightarrow{Z} \cap g)$*.*

Like the region automaton, the zone automaton has the property that a state is reachable in $\mathcal{A}$ iff it is reachable in $\mathcal{Z}(\mathcal{A})$. Furthermore, since zones are unions of regions they constitute a more compact representation and give rise to a smaller automaton. A problem with the zone automaton is that it can in some cases be infinite, which means that the algorithm for determining reachability may not terminate. There is however a method for solving this problem, called normalization, described in [4].

2.5 Region-Based Reachability Analysis of Dense-Timed Pushdown Automata

The method for constructing a region automaton for a TPDA is described in [3]. Below is an informal explanation of the intuition behind the construction.

In a TPDA, the existence of an unbounded stack containing symbols with real ages that change continuously due to time elapsing gives rise to a more complicated behaviour than in the case with timed automata. Furthermore, the relationships between clocks and stack symbols that lie arbitrarily deep in the

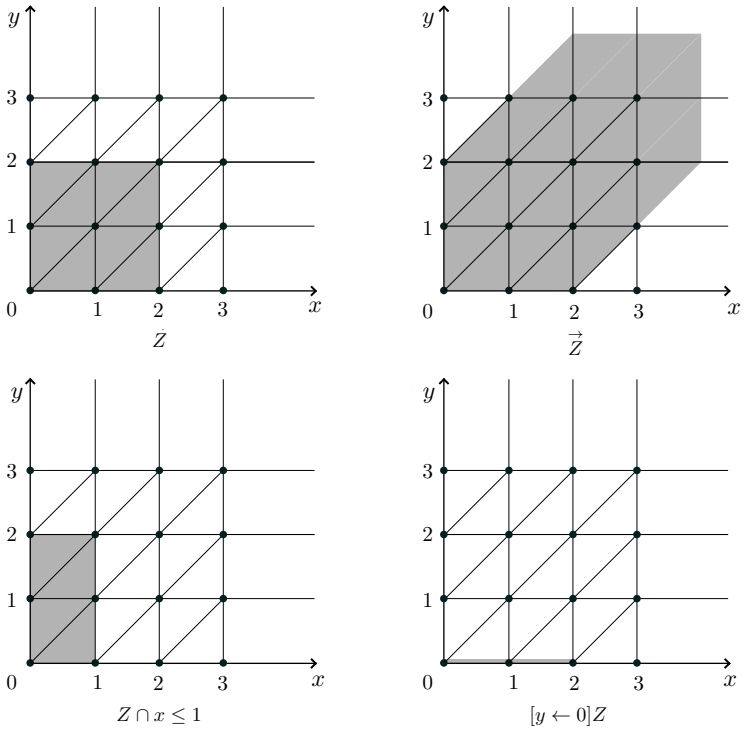

Fig. 3. Examples of operations on zones

stack can be modified during the run due to resets to the clocks. These in combination with the fact that in traditional pushdown automata, the symbols in the stack do not change makes the translation of a TPDA to an untimed pushdown automaton quite complicated. The symbolic representation used to describe the configurations of a TPDA needs to contain additional information that is used to register these relationships.

Given a TPDA $\mathcal{T}$ operating on a set X of clocks, a region in the symbolic automaton contains the following information:

- The values of every clock $x \in X$.
- One symbol from the alphabet of $\mathcal{T}$ and its age.
- The relationships between these items.

The restrictions of what values the items in the region can contain is the same as in the region abstraction for timed automata, i.e. they give rise to the same partitioning given the same items.

Furthermore, there are some additional items in the regions:

- The symbol $\vdash$, which is used to update the next topmost region when performing a *pop* transition.
- The symbol $\vdash^\bullet$, which records the amount of time that has passed since the region was pushed to the stack.
- For each clock $x \in X$, there is a shadow clock $x^\bullet$ which is used to record the value of x when the region was pushed to the stack.
- One shadow stack symbol. This is the shadow of the stack symbol in the region below the current region, and is used to record the value of the next topmost stack symbol.

The idea of the region automaton for a TPDA $\mathcal{T}$ is a pushdown automaton $\mathcal{R}(\mathcal{T})$ whose stack-alphabet is the set of regions. For each symbol that is pushed to the stack in $\mathcal{T}$, a region containing the same symbol is pushed to the stack in $\mathcal{R}(\mathcal{T})$, with the same age. The values of the clock items in the topmost region in the stack reflect the values of the clocks in the simulated automaton $\mathcal{T}$. Whenever performing a clock transition, i.e. $x \leftarrow I$ or $x \in I?$, this is simulated by changing the topmost region in the stack accordingly. Note that this can temporarily make the information in the other regions incorrect, it is possible that some clock value or relationship between stack symbols and clocks was changed by the transition. If a *pop* transition is simulated, it is thus not enough to remove the topmost region in the stack, but is indeed necessary to update the next topmost region so that it contains the correct information. This is what the additional items in the region are used for.

The region automaton $\mathcal{R}(\mathcal{T})$ of a TPDA $\mathcal{T}$ has the property that a state is reachable in $\mathcal{T}$ iff it is reachable in $\mathcal{R}(\mathcal{T})$. Thus, the problem of analyzing reachability in $\mathcal{T}$ is reduced to that of reachability in the untimed pushdown automaton $\mathcal{R}(\mathcal{T})$, which is decidable.

3 Overview

This section contains an informal overview of the proposed zone-based construction of a pushdown automaton $\mathcal{P}$ to simulate a TPDA $\mathcal{T}$. The idea is to create a corresponding abstraction of configurations for a TPDA as the zone automaton is for timed automata. The symbolic representation used and the manner in which $\mathcal{P}$ is created is based on the construction of the region automaton described in [3].

3.1 Symbolic representation

The symbolic automaton $\mathcal{P}$ is a classical pushdown automaton whose stack alphabet consists of a set of zones containing certain variables. These variables include one copy of every clock of the automaton, and one symbol from the alphabet of $\mathcal{T}$. As in the case with regions, the zones also need to contain additional items so as not to lose information due to the fact that the symbols in the stack of $\mathcal{P}$ do not change during the run. For this reason, every zone also contains the symbol $\vdash$ and $\vdash^\bullet$. The value of $\vdash$ is always 0, unless when performing a *pop* transition, and $\vdash^\bullet$ records the amount of time that has passed since the zone was pushed to the stack. These two items are used when a zone is popped from the stack to update the constraints in the next topmost zone.

3.2 Simulation

The computation in figure 1, starting from γ_1, is simulated in $\mathcal{P}$ as follows. The contents of the stack in $\mathcal{P}$ is shown in figure 4. The constraints containing 0 are not shown, since the value of $\vdash$ is 0 in all zones except the temporay one when performing a pop transition. To distinguish between the items in different zones Z_1 and Z_2, the notation $\vdash_1, \vdash_2$ is used. The zones below Z_1 in the stack are not shown in the example.

Firstly, note which configurations Z_1 contains. There are no upper bounds on the ages of the items. This is because every zone Z contains the time closure of the configurations that it can be located at after the series of transitions that led to Z being pushed to the stack, i.e. all the configurations that can be reached by delaying an arbitrary amount of time. Recall that a timed transition of any length can be taken from a state in $\mathcal{T}$ at any time. For this reason, timed transitions in $\mathcal{T}$ are not simulated as separate transition in $\mathcal{P}$.

The *push* transition that leads to γ_2 is simulated in $\mathcal{P}$ as follows. A new zone Z_2 is pushed to the stack. Z_2 is derived from Z_1 by copying the relationships of the clocks, removing the stack symbol a, and adding the symbol b in the interval $[3:4]$. Also, the symbol $\vdash_2^\bullet$ is reset to 0, as this symbol represents the time that has passed since Z_2 was pushed to the stack. Finally, the passage of an arbitrary amount of time is simulated. This is done by removing all the upper bounds on the items (except the symbol $\vdash$, whose value is always 0 unless when performing a *pop* transition).

The assignment transition $x \leftarrow [1:2]$ is simulated by changing the topmost zone Z_2 so that is shows the new relationships of the clocks. First, the item x is freed, i.e. all constraints on it are removed. Secondly, new constraints are added so that x lies in the specified interval. Finally, the passage of time is simulated.

The transition $x \in [3:4]$? changes Z_2 in the following way. Constraints are added so that x lies in the specified interval. In this case, this will not change the relationships between x and any other items. The new constraint will however change the lower bounds for some items. Intuitively, since x was pushed in the interval $[1:2]$, it cannot have gotten the value 3 unless at least 1 unit of time has passed. This is why the lower bound of $\vdash_2^\bullet$ is changed to 1 in the updated zone. For the same reason the lower bound of y is updated to 1. The time closure of this updated zone finally replaces the old Z_2 as the topmost stack symbol.

When simulating the *pop* transition the goal is to remove the zone containing b from the top of the stack, so that Z_1 is at the top again. The problem is that Z_1 does not contain the correct constraints on the items. This is due to the fact that Z_1 has not been changed since Z_2 was pushed to the stack, and the transitions that have occurred since that have had an effect on the items in Z_1. For example, Z_1 contains the constraint $x - a \leq 0$, but the assignment transition $x \leftarrow [1:2]$ that has occurred since has changed that relationship to $x - a \leq 1$. This relationship will be recovered using the special items.

A new zone Z_1', is created by combining Z_1 and Z_2, and using the special items $\vdash$ and $\vdash^\bullet$ to obtain the updated constraints. Z_1' then replaces both Z_2 and Z_1 on the stack. Z_1' is created by starting from Z_2 and adding constraints in the following way. First, constraints are added so that b lies in the specified interval. This is because the *pop* transition can only be taken from configurations that fulfil this condition. Secondly, the relationships between y and $\vdash_1^\bullet$ are added to Z_2. As there has not been a reset to y since Z_2 was pushed, these old relationships need to be preserved. There has, however, been a reset to x (the assignment transition), so old relationships of x are not added. After this, all clock-items are removed from Z_1, and the remaining constraints are added to Z_2. Since Z_2 contains the current constraints on the clocks it is not necessary to keep any of the other clock constraints from Z_1. Now the special items are used to regain the correct constraints for the other items. By adding the constraints $\vdash_2^\bullet = \vdash_1$, the passage of time that has occurred in Z_2 is simulated for the items in Z_1. This also relates the symbol $\vdash_1^\bullet$ to x, and a to the clocks. In this manner the correct relationships between the items are gained. Finally, b, $\vdash_2^\bullet$ and $\vdash_2$ are removed from the zone, and $\vdash_1$ is set to 0 again. As always, the passage of time is simulated in Z_1' before it is pushed to the stack.

4 Symbolic Representation

The symbolic representation used to manipulate configurations is zones, as defined in Definition 13, enriched with special variables that are needed to record the more complicated relationships that can exist between variables in timed pushdown automata. From now on, when writing "zone", the specific kind used

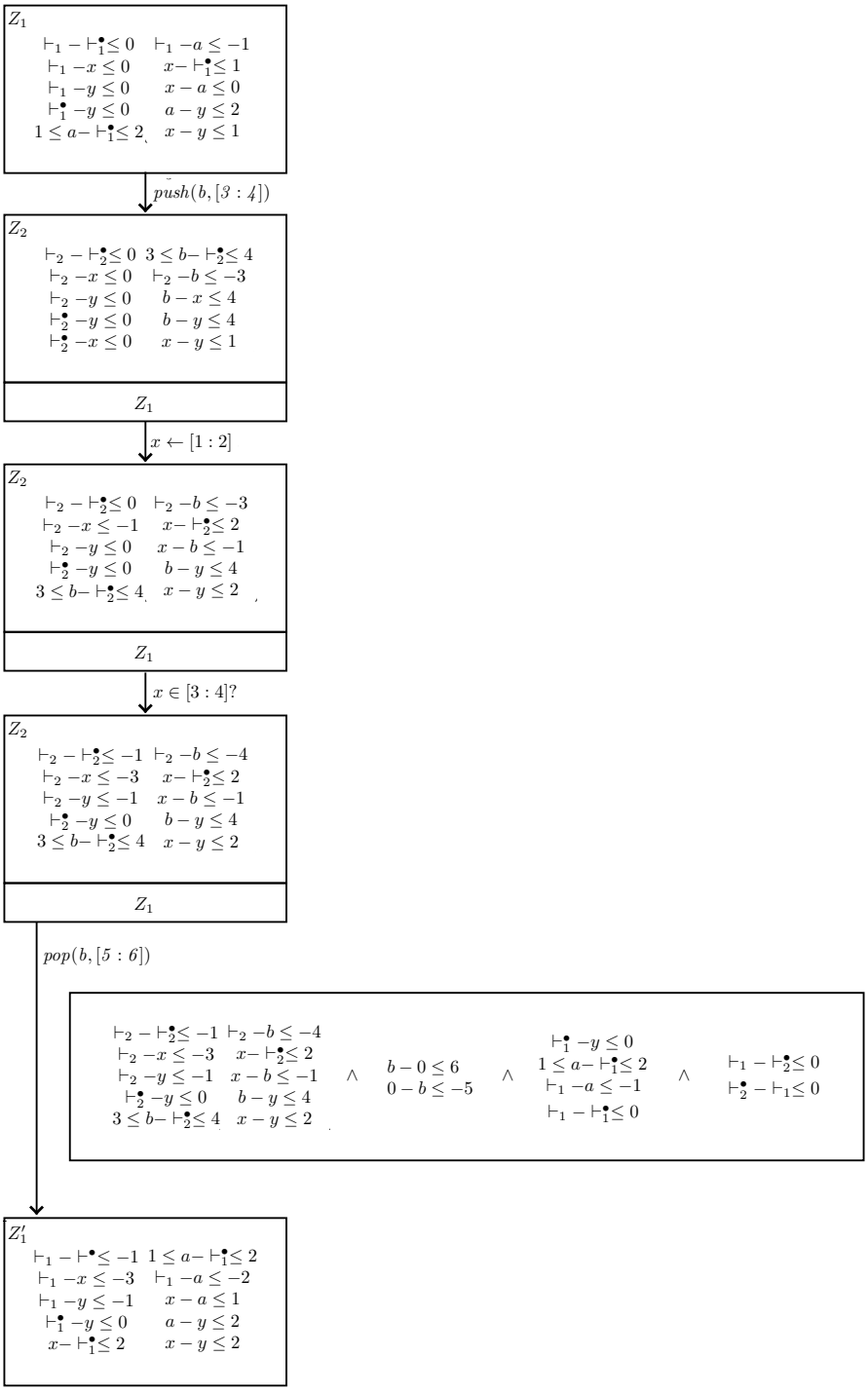

Fig. 4. Stack content during a run in $\mathcal{P}$.

in the symbolic representation of timed pushdown automata are referred to. In this section, these zones and some operations on them are defined.

4.1 Zones

The items define the set of variables in a zone. A zone contains one copy of each clock in $\mathcal{T}$ and exactly one symbol from the stack alphabet of $\mathcal{T}$. In addition to this there is a symbol $\vdash$ whose value is always 0 unless when performing a pop-transition, and a symbol $\vdash^\bullet$ which records the amount of time that has passed since the zone was pushed to the stack.

Definition 16. *Let $\Gamma := \Gamma^{\mathcal{T}} \cup \{\texttt{bottom}\}$, where $\texttt{bottom} \notin \Gamma^{\mathcal{T}}$ is a special symbol indicating the bottom of the stack. The set of* **items** *is defined as $W := X \cup \Gamma \cup \{\vdash^\bullet, \vdash\} \cup \{0\}$. $Z^\top$ is used to denote the set of items that occur in Z.*

Definition 17. *A* **zone** *Z is a finite conjunction of zone constraints over items in W that satisfies the conditions below:*

- $\left|\Gamma \cap Z^\top\right| = 1$, *i.e. there is exactly one occurrence of a stack symbol in the zone.*
- $\left|\{\vdash\} \cap Z^\top\right| = 1$, *i.e. there is exactly one occurrence of the symbol $\vdash$ in the zone.*
- $\left|\{\vdash^\bullet\} \cap Z^\top\right| = 1$, *i.e. there is exactly one occurrence of the symbol $\vdash^\bullet$ in the zone.*
- $\left|\{x\} \cap Z^\top\right| = 1$ *for all $x \in X$, i.e. each clock occurs exactly once in the zone.*
- *For each $w \in Z^\top$, $0 - w \leq k \in Z$ where $k \leq 0$, i.e. negative values for the items are not allowed.*

When dealing with several different zones, s_1 is used to denote the stack symbol that occurs in Z_1, and $\vdash_1$, $\vdash_1^\bullet$ to denote the symbols $\vdash$ and $\vdash^\bullet$ that it contains.

4.2 Operations on zones

The following operations on zones will be needed in the simulation. Let Z be a zone.

***Up*(Z)** This operation simulates the passage of an arbitrary amount of time in the system, i.e. $Up(Z)$ contains the time valuations that can be reached from Z by delay. This operation does not, however, change the value of $\vdash$, which is always 0 except when performing a pop-transition.
Formally this is defined as $Up(Z) = \{[\vdash \leftarrow 0](u + d) \mid u \in Z \wedge d \in \mathbb{R}^{\geq 0}\}$.

***And*$(Z, w - w' \preceq n)$** This operation adds a constraint to a zone. Formally this is defined as $And(Z, w - w' \preceq n) = \{u \mid u \in Z \wedge u(w) - u(w') \preceq n\}$. For a variable w and an interval I the notation $And(Z, w \in I)$ will be used to denote the zone

- $And(And(Z, w - 0 \leq b), 0 - w \leq -a)$ if $I = [a : b]$
- $And(And(Z, w - 0 < b), 0 - w < -a)$ if $I = (a : b)$

And correspondingly for different combinations of closed and open limits for I.

***Free*(Z, w)** This operation removes all constraints on the item w. Formally this is defined as $Free(Z, w) = \{[w \leftarrow d]u \mid u \in Z \wedge d \in \mathbb{R}^{\geq 0}\}$

Resetting This operation is used when simulating push-transitions. A new zone Q is pushed to the stack in $\mathcal{P}$ which is derived from the current topmost zone Z in the following way. The constraints on the item $\vdash^\bullet$ are removed, and its value is set to zero again. Also, the stack symbol in Z removed, and the new stack symbol a that is to be pushed is added, and constraints are added so that it lies in the specified interval. Formally, $Reset(Z)[a \leftarrow I]$ is defined as the zone $(And(And(Free(Z \ominus s_Z \oplus a, \vdash^\bullet), \vdash^\bullet \in [0 : 0]), a \in I)$.

Adding/deleting items and constraints For zones Z and Z' such that ${Z'}^\top \subseteq Z^\top$, and items $w \in Z^\top$, $w' \notin Z^\top$ define the follwing operations:

- $Z \ominus w$ which deletes an item w from Z. All constraints containing the item are removed.
- $Z \oplus w'$ which adds an item to Z. The new zone will contain an extra item which has no constraints except that it is larger than 0.
- $Z \wedge Z'$ which adds all constraints from Z' to Z. Formally, $Z \wedge Z'$ is defined as the zone $And(And(...And(Z, c_1), ...), c_n)$ for all zone constraints $c_i \in Z'$.

Sometimes it is written "constraint $w_1 - w_2$ is copied from Z' to Z". This means that if $w_1 - w_2 \preceq n \in Z'$, then we create the zone $And(Z, w_1 - w_2 \preceq n)$.

Product When performing a pop transition it is required to merge two zones, since next topmost zone in the stack will not contain information about the transitions that have occured since the topmost stack symbol was pushed. If Z_2 is the topmost zone and Z_1 is the next topmost zone, this operation defines a new zone Z_1' which contains the same items as Z_1, but with their current constraints according to the information in Z_2.

$Z_1 \odot Z_2$ is defined as the zone Z_1' which is created by the following process, where P_1-P_6 are temporary conjunctions of constraints:

- $P_1 := Z_2 \oplus \vdash_1 \oplus \vdash_1^\bullet \oplus s_1 \ominus s_2$
- P_2 is created by adding the following constraints to P_1: For each clock $x \in X$ such that a $x \leftarrow I$ transition has not occured since Z_2 was pushed to the stack, the constraints $x - \vdash_1^\bullet$ and $\vdash_1^\bullet - x$ are copied from Z_1 to P_1.
- $P_3 := (Z_1 \ominus x)$ for all clocks $x \in X$
- $P_4 := P_2 \wedge P_3$
- $P_5 := And(P_4, \vdash_1 - \vdash_2^\bullet \leq 0)$
- $P_6 := And(P_5, \vdash_2^\bullet - \vdash_1 \leq 0)$
- $P_7 := And(P_6, \vdash_1 \in [0:0])$
- $Z_1' := P_7 \ominus \vdash_2 \ominus \vdash_2^\bullet$

The updated zone Z_1' is gained by creating a new zone from Z_2 by replacing its stack symbol with the stack symbol from Z_1 and adding the items $\vdash$ and $\vdash^\bullet$ from Z_1. The clock-items and their values in Z_2 are not changed, since they represent the current state in the automaton. The clock-items are removed from Z_1, but in the cases where no assignment has been made to a clock x since Z_2 was pushed, the old relationships between x and $\vdash_1^\bullet$ are first copied to the new zone. In the case when an assignment has happened to a clock since Z_2 was pushed, all constraints containing x in Z_1 are obsolete. All constraints not containing clock items are also copied from Z_1 to the new zone. By creating P_3 and P_4, i.e. setting $\vdash_1 = \vdash_2^\bullet$ the new, correct relationships between the stack symbol s_1 and the clocks are obtained. Finally, the symbol $\vdash_1$ is again set to 0, and the symbols $\vdash_2$ and $\vdash_2^\bullet$ are removed (the values of these were only needed to obtain the correct constraints for the other items).

5 Simulation

Fix a TPDA $\mathcal{T} = \langle S^\mathcal{T}, s_{init}^\mathcal{T}, \Gamma^\mathcal{T}, \Delta^\mathcal{T} \rangle$ with a set X of clocks. This section shows how $\mathcal{T}$ can be simulated by an (untimed) PDA $\mathcal{P} = \langle S^\mathcal{P}, s_{init}^\mathcal{P}, \Gamma^\mathcal{P}, \Delta^\mathcal{P} \rangle$.

The states and transitions in $\mathcal{P}$ are described below. The transitions include those of initializing $\mathcal{P}$ and those of simulating the different kinds of transitions in $\mathcal{T}$. Note that the timed transitions in $\mathcal{T}$ are not simulated as a separate transition, instead every zone that is pushed to the stack in $\mathcal{P}$ contains the time closure of the configurations at which the simulated automaton can be located at the specific point in its execution. This means that each zone contains the configurations that can be reached by delaying an arbitrary time. This simulates the behaviour of $\mathcal{T}$ correctly since a timed transition of any length can be taken from every state in $\mathcal{T}$ at any time.

States There are two kinds of states in $S^\mathcal{P}$, *genuine* and *temporary* states. The set of genuine states consists of all the states in $S^\mathcal{T}$. The temporary states are used when a transition of $\mathcal{T}$ is simulated in several steps in $\mathcal{P}$. The temporary

states are written on the form $\texttt{tmp}(\cdot,\cdot)$ where the arguments indicate the transition in $\Delta^{\mathcal{T}}$ that is currently being simulated and the number of steps that have been performed in the simulation. A configuration $\beta = \langle s, w\rangle$ in $\mathcal{P}$ is said to be *genuine* resp. *temporary* if s is *genuine* resp. *temporary*.

The simulation starts form the distinguished initial state $s^{\mathcal{P}}_{init}$ (which is considered to be a temporary state).

Initialization The initial zone Z_{init} is defined as follows. Let P be a zone whose stack symbol is `bottom`, such that $w_1 - w_2 \leq 0 \in P$ for all $w_1, w_2 \in P^{\top}$. Then, $Z_{init} = Up(P)$. In other words, all the clocks, `bottom` and $\vdash^{\bullet}$ are equal and greater than or equal to 0, and $\vdash$ is equal to 0. The zone's stack symbol is `bottom`, which indicates that it is the bottom of the stack. The value of `bottom` will never actually be used in the simulation, but is included to preserve the invariant for zones.

The set $\Delta^{\mathcal{P}}$ contains a transition $\langle s^{\mathcal{P}}_{init}, push(Z_{init}), s^{\mathcal{T}}_{init}\rangle$. This transition pushes the initial zone, and moves from the initial state of $\mathcal{P}$ to the initial state of $\mathcal{T}$, from which the simulation is started.

nop For each transition $\langle s, nop, s'\rangle \in \Delta^{\mathcal{T}}$, the set $\Delta^{\mathcal{P}}$ contains the transition $\langle s, nop, s'\rangle$.

The empty operation only changes the local state of $\mathcal{T}$ and the stack content is unchanged.

***x* ∈ *I*?** For each transition $\langle s, x \in I?, s'\rangle \in \Delta^{\mathcal{T}}$ and zone Z the set $S^{\mathcal{P}}$ contains the state $\texttt{tmp}(t, Z)$, and $\Delta^{\mathcal{P}}$ contains the transitions $\langle s, pop(Z), \texttt{tmp}(t, Z)\rangle$ and $\langle \texttt{tmp}(t, Z), push(Z'), s'\rangle$, where $Z' = Up(And(Z, x \in I))$.
The new topmost zone Z' is the time closure of the set of valuations in Z that satisfy the condition $x \in I$.

x* ← *I For each transition $\langle s, x \leftarrow I, s'\rangle \in \Delta^{\mathcal{T}}$ and zone Z the set $S^{\mathcal{P}}$ contains the state $\texttt{tmp}(t, Z)$, and $\Delta^{\mathcal{P}}$ the transitions $\langle s, pop(Z), \texttt{tmp}(t, Z)\rangle$ and $\langle \texttt{tmp}(t, Z), push(Z'), s'\rangle$, where $Z' = Up(And(Free(Z, x), x \in I))$.
The new topmost zone Z' is gained by first removing all constraints on x, then adding the constraint that it is in the interval I, and finally letting time pass.

***pop*(*a*, *I*)** The topmost zone Z_2 in the stack is removed in the case that its stacksymbol is a, and the information in this zone is used to update the next topmost zone Z_1 to the current values of all items. Since the transition can only be taken from a zone in which a lies in the interval I, the constraint $a \in I$ is added, and then the product Z_1' is computed and used to replace both Z_2 and Z_1.

For each transition $\langle s, pop(a, I), s'\rangle \in \Delta^{\mathcal{T}}$ the following states and transitions are added. For zones Z_2, Z_1 in which $s_2 = a$, $S^{\mathcal{P}}$ contains the states $\texttt{tmp}(t, Z_2)$

and $\texttt{tmp}(t, Z_2, Z_1)$, and $\Delta^{\mathcal{P}}$ contains the transitions $\langle s, pop(Z_2), \texttt{tmp}(t, Z_2)\rangle$, $\langle \texttt{tmp}(t, Z_2), pop(Z_1), \texttt{tmp}(t, Z_2, Z_1)\rangle$, and $\langle \texttt{tmp}(t, Z_2, Z_1), push(Z'), s'\rangle$, where $Z' = Up(Z_1 \odot And(Z_2, a \in I))$.

***push*(a, I)** For each transition $t = \langle s, push(a, I), s'\rangle \in \Delta^{\mathcal{T}}$ and zone Z_1, the set $S^{\mathcal{P}}$ contains the states $\texttt{tmp}_1(t, Z_1)$ and $\texttt{tmp}_2(t, Z_1)$. The set $\Delta^{\mathcal{P}}$ contains the transitions $\langle s, pop(Z_1), \texttt{tmp}_1(t, Z_1)\rangle$, $\langle \texttt{tmp}_1(t, Z_1), push(Z_1), \texttt{tmp}_2(t, Z_1)\rangle$. Finally, $\Delta^{\mathcal{P}}$ contains the transition $\langle \texttt{tmp}_2(t, Z_1), push(Z_2), s'\rangle$, where $Z_2 = Up(Reset(Z_1)[a \leftarrow I])$.
In other words, a new zone is pushed to the stack which is derived from Z_1 by deleting the stack symbol s_1, adding the item a, setting $\vdash^{\bullet}$ equal to zero, adding the constraint that $a \in I$, and finally letting time pass.

6 Correctness

The correctness of the proposed construction is not showed in this report. Below is an outline of a possible proof, based on that in [3]. In this case, the lemmas remain to be proven.

Given a TPDA $\mathcal{T} = \left\langle S^{\mathcal{T}}, s^{\mathcal{T}}_{init}, \Gamma^{\mathcal{T}}, \Delta^{\mathcal{T}}\right\rangle$ consider the PDA $\mathcal{P} = \left\langle S^{\mathcal{P}}, s^{\mathcal{P}}_{init}, \Gamma^{\mathcal{P}}, \Delta^{\mathcal{P}}\right\rangle$ derived from $\mathcal{T}$ as described in Section 5. Consider a state $s_F \in S^{\mathcal{T}}$. Then:

Theorem 1. *s_F is reachable in $\mathcal{T}$ iff s_F is reachable in $\mathcal{P}$.*

Given a stack-zone $\mathcal{Z} = Z_0 Z_1 \cdots Z_n$ the stack zone $\mathcal{Q} = Q_0 Q_1 \cdots Q_n$ is said to be a *strengthening* of $\mathcal{Z}$ if $Q_n = Z_n$ and $Q_i = Z_i \odot Q_{i+1}$ for all $i : 0 \leq i < n$. The definition of strengthening is extended to configurations in the following manner. Given a configuration $\beta = \langle s, \mathcal{Z}\rangle$, β' is said to be a strengthening of β if $\beta' = \langle s, \mathcal{Q}\rangle$, and $\mathcal{Q}$ is a strenghthening of $\mathcal{Z}$.

Consider a configuration $\beta = \langle s, \mathcal{Z}\rangle$ in $\mathcal{P}$, a strenghtening $\beta' = \langle s, \mathcal{Q}\rangle$ of β and a configuration $\gamma = \langle s', \texttt{X}, w\rangle$ in $\mathcal{T}$. Let $w = \langle a_1, v_1\rangle \cdots \langle a_n, v_n\rangle$ and let $\mathcal{Q} = Q_0 Q_1 \cdots Q_n$.
Define a valuation f_i for each Q_i, $i : 1 \leq i \leq n$, by

- $f_i(x) = \texttt{X}(x)$ for all $x \in X$
- $f_i(a_i) = v_i$

Let $\gamma \models \beta'$ denote that the following conditions hold:

- $s' = s$
- $a_i \in Q_i^{\top}$ for all $i : 1 \leq i \leq n$
- $f_i \in Q_i$ for all $i : 1 \leq i \leq n$

Below is an outline of the proof of Theorem 1 in both directions.

From $\mathcal{P}$ to $\mathcal{T}$

Lemma 1. *For any genuine reachabe configuration β in $\mathcal{P}$, strengthening $\beta' = \langle s, \mathcal{Q} \rangle$ of β, there is a configuration γ in $\mathcal{T}$ such that $\gamma \models \beta'$ and $\gamma_{init} \overset{*}{\rightsquigarrow} \gamma$.*

By definition, if the state s_F is reachable in $\mathcal{P}$ then there exists a genuine configuration $\beta = \langle s_F, \mathtt{X} \rangle$ that is reachable for some $\mathtt{X}$. Let β' be the strengthening of β. Then Lemma 1, applied to β and the strengthening β', results in the existence of a reachable configuration γ in $\mathcal{T}$ whose state is s_F, since $\gamma \models \beta'$.

From $\mathcal{T}$ to $\mathcal{P}$

Lemma 2. *For any reachabe configuration γ in $\mathcal{T}$, there is a configuration β in $\mathcal{P}$ such that the strengthening β' of β satisfies $\gamma \models \beta'$ and $\beta_{init} \xrightarrow{*} \beta$.*

If the state s_F is reachable in $\mathcal{T}$, accordning to Lemma 2 there exists a reachable configuration β in $\mathcal{P}$ whose state is s_F. Thus, s_F is also reachable in $\mathcal{P}$.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science 126:1, pages 183–235, 1994*
2. Rajeev Alur, and David L. Dill. Automata for modeling realtime systems. *Proc. 17th Int. Coll Automata, Languages and Programming (ICALP'90)*
3. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Jari Stenman. Dense-Timed Pushdown Automata. *Proc. 27th Annual IEEE Symposium on Logic in Computer Science (LICS'12), Dubrovnik (Croatia), June 2012*
4. Johan Bengtsson, Wang Yi. Timed Automata: Semantics, Algorithms and Tools. *Lecture Notes in Computer Science, 2004, Volume 3098/2004, pages 87-124*
5. David L. Dill. Timing Assumptions and Verification of Finite-State Concurrent Systems *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems, pages 197 - 212, 1989*