Information and Computation 251 (2016) 67–90

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Better abstractions for timed automata ☆

Frédéric Herbreteau [a], B. Srivathsan [b,*], Igor Walukiewicz [a]

[a] Université de Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR 5800, Talence, France
[b] Chennai Mathematical Institute, Chennai, India

ARTICLE INFO

Article history:
Received 20 May 2015
Available online 27 July 2016

Keywords:
Timed automata
Reachability problem
Non-convex abstractions

ABSTRACT

We study the reachability problem for timed automata. A standard solution to this problem involves computing a search tree whose nodes are abstractions of zones. These abstractions preserve underlying simulation relations on the state space of the automaton. For both effectiveness and efficiency reasons, they are parameterized by the maximal lower and upper bounds (LU-bounds) occurring in the guards of the automaton.

One such abstraction is the $\alpha_{\preccurlyeq LU}$ abstraction defined by Behrmann et al. Since this abstraction can potentially yield non-convex sets, it has not been used in implementations. Firstly, we prove that $\alpha_{\preccurlyeq LU}$ abstraction is the coarsest abstraction with respect to LU-bounds that is sound and complete for reachability. Secondly, we provide an efficient technique to use the $\alpha_{\preccurlyeq LU}$ abstraction to solve the reachability problem.

1. Introduction

Timed automata are finite automata extended with clocks whose values can be compared with constants and set to zero. The clocks measure delays between different steps of execution of the automaton. The reachability problem for timed automata asks if there exists a path from its initial state to a given target state. This problem cannot be solved by a simple state exploration since clocks are real-valued variables. The standard solution to this problem involves computing the zone graph of the automaton which in principle could be infinite. In order to make it finite, zones are approximated using an abstraction operator. Till recently it has been generally assumed that for reasons of efficiency an abstraction of a zone should always be a zone. Here we avoid this assumption. We first show that $\alpha_{\preccurlyeq LU}$ abstraction defined by Behrmann et al. [4] is the coarsest sound and complete abstraction. We then present a method of constructing the abstracted zone graph using the $\alpha_{\preccurlyeq LU}$ abstraction. Even though this abstraction can yield non-convex sets, we show that our method is at least as efficient as any other currently known method based on abstractions.

The reachability problem is a basic problem in verification. It is historically the first problem that has been considered for timed-automata, and it is still a lively subject of research [4,24,20,15]. Apart from being interesting by itself, the advances on this problem may give new methods for verification of more complicated models, like priced timed-automata [11], or probabilistic timed automata [14,9,18].

All approaches to solving the reachability problem for timed automata should ensure termination. To tackle this, most of them use abstractions to group together bisimilar valuations of clock variables, that is, valuations not distinguishable by

☆ This work has been supported by project AVeRTS – CEFIPRA – Indo–French Program in ICST – DST/CNRS ref. 218093. Author B. Srivathsan is partially funded by a grant from Infosys Foundation.

* Corresponding author.
E-mail addresses: fh@labri.fr (F. Herbreteau), sri@cmi.ac.in (B. Srivathsan), igw@labri.fr (I. Walukiewicz).

http://dx.doi.org/10.1016/j.ic.2016.07.004
0890-5401/

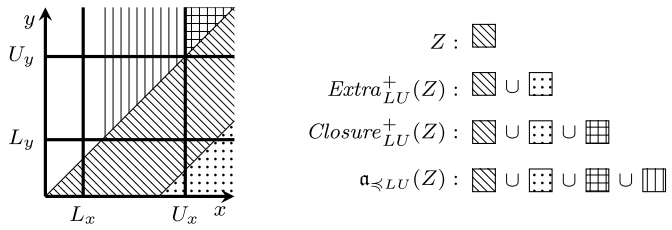


Fig. 1. A comparison of abstraction operators for zones.

the automaton as far as reachability to the final state is concerned. The first solution has been based on regions, which are certain equivalence classes of clock valuations [2]. Their definition is parameterized by a threshold up to which the clock values should be considered. A great improvement in efficiency has been obtained by adopting zones instead of regions. These are sets of valuations defined by conjunctions of differences between pairs of clocks. They can be efficiently implemented using difference bound matrices (DBMs) [7,13]. A challenge with zone based approach is that they are not totally compatible with regions, and moreover a forward exploration algorithm can produce infinitely many zones. The union of regions intersecting a zone is a natural candidate for a finitary abstraction. Indeed this abstraction would make the forward exploration algorithm terminate. However such an union of regions is not necessarily a zone and so it is not clear how to represent it. For this reason a number of abstraction operators have been proposed that give an approximation of the union of regions intersecting a zone. A coarser approximation would make the abstracted zone graph smaller. So potentially it would give a more efficient algorithm.

An important observation made in [4] is that if reachability is concerned then we can consider simulation instead of bisimulation. Indeed, it is safe to add configurations that can be simulated by those that we have already reached. Simulation relations in question depend on the given automaton, and it is EXPTIME-hard to calculate the biggest one [19]. A pragmatic approach is to abstract some part of the structure of the automaton and define a simulation based on this information. The most relevant information is the set of bounds with which clocks are compared to in the guards of the automaton. Since lower and upper bounds are considered separately, they are called LU-bounds. In [4] the authors define an abstraction based on simulation with respect to LU-bounds; it is denoted $\mathfrak{a}_{\preccurlyeq LU}$. Theoretically $\mathfrak{a}_{\preccurlyeq LU}$ is very attractive: it has clear semantics and, as we show here, it is always a union of regions. The problem is that $\mathfrak{a}_{\preccurlyeq LU}$ abstraction of a zone is seldom a convex set, so one cannot represent the result as a zone.

In this paper we give another very good reason to consider $\mathfrak{a}_{\preccurlyeq LU}$ abstraction. We show that it is actually the coarsest abstraction that is sound and complete with respect to reachability for all automata with the same LU-bounds. In other words, it means that in order to get coarser (that is better) abstractions one would need to look at some other structural properties of automata than just LU-bounds. Our main technical result is an effective algorithm for dealing with $\mathfrak{a}_{\preccurlyeq LU}$ abstraction. It allows to manipulate this abstraction as efficiently as purely zone based ones. We propose a forward exploration algorithm working with zones that constructs the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction of the transition graph of the automaton. This algorithm uses standard operations on zones, plus a new test of inclusion of a zone in the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction of another zone. The test is quadratic in the number of clocks and not more complex than that for just testing an inclusion between two zones. Since $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is the coarsest sound and complete abstraction, it can potentially give smallest abstract systems.

1.1. Related work

Forward analysis is the main approach for the reachability testing of real-time systems. The use of zone-based abstractions for termination has been introduced in [12]. In recent years, coarser abstractions have been introduced to improve efficiency of the analysis [4]. An approximation method based on LU-bounds, called $Extra^+_{LU}$, is used in the current implementation of UPPAAL [5]. In [15] it has been shown that it is possible to efficiently use the region closure of $Extra^+_{LU}$, denoted $Closure^+_{LU}$. This has been the first efficient use of a non-convex abstraction. In comparison, $\mathfrak{a}_{\preccurlyeq LU}$ approximation has a well-motivated semantics, it is also region closed, coarser than $Closure^+_{LU}$ and the resulting inclusion test is even simpler than that of $Closure^+_{LU}$. A comparison of these abstractions is depicted in Fig. 1. The $\mathfrak{a}_{\preccurlyeq LU}$ inclusion test (restricted to the case when $L = U$) has recently been adapted to weighted timed automata and priced zones [10].

Let us mention that abstractions are not needed in backward exploration of timed systems. Nevertheless, any feasible backward analysis approach needs to simplify constraints. For example [20] does not use approximations and relies on an SMT solver instead. Clearly this approach is very difficult to compare with the forward analysis approach we study here.

Another related approach to verification of timed automata is to build a quotient graph of the semantic graph of the automaton with respect to some bisimulation relation [23,1,25]. For reachability properties, this approach is not a priori competitive with respect to using the simulation-based abstraction $\mathfrak{a}_{\preccurlyeq LU}$. It is more adapted to checking branching time properties.

1.2. Organization of the paper

The paper is organized as follows.

Section 2 presents the preliminary definitions. It introduces the notion of sound and complete abstractions parameterized by LU-bounds. It also explains how these abstractions could be used to solve the reachability problem.
Section 3 proposes abs_{LU} abstraction, and proves that it is the coarsest sound and complete abstraction for all automata with given LU-bounds.
Section 4 shows that the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction actually coincides with this coarsest abstraction abs_{LU}.
Section 5 presents an efficient inclusion test for $\mathfrak{a}_{\preccurlyeq LU}$ abstraction, which allows for its use in implementations.

A preliminary version of this paper appeared in the conference on Logic in Computer Science [16]. This version includes all missing proofs and gives a more elaborate discussion about the inclusion test (Section 5).

2. Preliminaries

After recalling some preliminary notions, we introduce a concept of abstraction as a means to reduce the reachability problem for timed-systems to the one for finite systems. We then observe that simulation relation is a convenient way of obtaining abstractions with good properties.

2.1. Timed automata and the reachability problem

Let X be a set of clocks, i.e., variables that range over $\mathbb{R}_{\geq 0}$, the set of non-negative real numbers. A *clock constraint* is a conjunction of constraints $x\#c$ for $x \in X$, $\# \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$, e.g. $(x \leq 3 \wedge y > 0)$. Let $\Phi(X)$ denote the set of clock constraints over clock variables X. A *clock valuation* over X is a function $v : X \to \mathbb{R}_{\geq 0}$. We denote by $\mathbb{R}^X_{\geq 0}$ the set of clock valuations over X, and by $\mathbf{0}$ the valuation that associates 0 to every clock in X. We write $v \vDash \phi$ when v satisfies $\phi \in \Phi(X)$, i.e. when every constraint in ϕ holds after replacing every x by $v(x)$. For $\delta \in \mathbb{R}_{\geq 0}$, let $v + \delta$ be the valuation that associates $v(x) + \delta$ to every clock x. For $R \subseteq X$, let $[R]v$ be the valuation that sets x to 0 if $x \in R$, and that sets x to $v(x)$ otherwise.

A *Timed Automaton (TA)* is a tuple $\mathcal{A} = (Q, q_0, X, T, Acc)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, X is a finite set of clocks, $Acc \subseteq Q$ is a set of accepting states, and $T \subseteq Q \times \Phi(X) \times 2^X \times Q$ is a finite set of transitions (q, g, R, q') where g is a *guard*, and R is the set of clocks that are *reset* on the transition.

The semantics of $\mathcal{A}$ is a transition system of its configurations. A *configuration* of $\mathcal{A}$ is a pair $(q, v) \in Q \times \mathbb{R}^X_{\geq 0}$ and $(q_0, \mathbf{0})$ is the *initial configuration*. We have two kinds of transitions:

Delay: $(q, v) \to^{\delta} (q, v + \delta)$ for some $\delta \in \mathbb{R}_{\geq 0}$;

Action: $(q, v) \to^{t} (q', v')$ for some transition $t = (q, g, R, q') \in T$ such that $v \vDash g$ and $v' = [R]v$.

We will denote by $\mathcal{S}_{\mathcal{A}}$ the transition system describing the semantics of a timed automaton $\mathcal{A}$. In this paper we are interested in the *reachability problem*: does there exist a configuration (q, v) with accepting state $q \in Acc$ that is reachable from $(q_0, \mathbf{0})$ by some finite sequence of delay and action transitions?

The class of TA we consider is usually known as diagonal-free TA since clock comparisons like $x - y \leq 1$ are disallowed. Notice that if we are interested in state reachability, considering timed automata without state invariants does not entail any loss of generality as the invariants can be added to the guards. For state reachability, we can also consider automata without transition labels.

2.2. Abstractions

Since the transition system determined by the automaton is infinite, we usually try to find a finite approximation of it by grouping valuations together. In consequence we work with configurations consisting of a state and a set of valuations. For every transition t, we have a transition:

$$(q, W) \Rightarrow^{t} (q', W') \quad \text{where } W' = \{v' : \exists v \in W,\ \exists \delta \in \mathbb{R}_{\geq 0} \text{ s.t. } v \to^{t} \to^{\delta} v'\}$$

We will write $\Rightarrow$ without superscript to denote the union of all $\Rightarrow^t$ relations. The transition relation defined above considers each valuation $v \in W$ that can take the transition t, obtains the valuation after the transition, and then collects the time-successors from this obtained valuation. Therefore the symbolic transition $\Rightarrow$ always yields sets closed under time-successors.

The initial configuration of the automaton is $(q_0, \mathbf{0})$. Starting from the initial valuation $\mathbf{0}$ the set of valuations reachable by a delay at the initial state is given by $\{\mathbf{0} + \delta \mid \delta \in \mathbb{R}_{\geq 0}\}$. Call this W_0.

From (q_0, W_0) as the initial node, computing the symbolic transition relation $\Rightarrow$ leads to different nodes (q, W) wherein the sets W are closed under time-successors. Although the transition relation $\Rightarrow$ talks about sets of valuations and not valuations themselves, it could still be potentially infinite. A further grouping of valuations is necessary to get finiteness.

An *abstraction operation* [3] is a convenient way of expressing a grouping of valuations. It is a function $\mathfrak{a} : \mathcal{P}(\mathbb{R}_{\geq 0}^{|X|}) \to \mathcal{P}(\mathbb{R}_{\geq 0}^{|X|})$ such that $W \subseteq \mathfrak{a}(W)$ and $\mathfrak{a}(\mathfrak{a}(W)) = \mathfrak{a}(W)$. An abstraction operator defines an abstract semantics:

$$(q, W) \Rightarrow_{\mathfrak{a}} (q', \mathfrak{a}(W'))$$

where $\mathfrak{a}(W) = W$ and $(q, W) \Rightarrow (q', W')$.

If $\mathfrak{a}$ has a finite range then this abstraction is said to be finite. We write $\Rightarrow_{\mathfrak{a}}^*$ for the transitive closure of $\Rightarrow_{\mathfrak{a}}$, similarly we write $\Rightarrow^*$ and $\to^*$ respectively for the transitive closure of $\Rightarrow$ and $\to$ (where $\to$ denotes the union of $\to^t$ and $\to^\delta$).

Of course we want this abstraction to reflect some properties of the original system. In order to preserve reachability properties we can require the following two properties (recall that $\mathbf{0} \in W_0$):

Soundness: if $(q_0, \mathfrak{a}(W_0)) \Rightarrow_{\mathfrak{a}}^* (q, W)$ then there is a $v \in W$ such that $(q_0, \mathbf{0}) \to^* (q, v)$.

Completeness: if $(q_0, \mathbf{0}) \to^* (q, v)$ then there is a W such that $v \in W$ and $(q_0, \mathfrak{a}(W_0)) \Rightarrow_{\mathfrak{a}}^* (q, W)$.

It can be easily verified that if an abstraction satisfies $W \subseteq \mathfrak{a}(W)$ then the abstracted system is complete. However soundness is more delicate to obtain.

Naturally, it is important to be able to efficiently compute the abstract transition system. A standard way to do this is to use zones. A *zone* is a set of valuations defined by a conjunction of two kinds of constraints: comparison of differences between two clocks with an integer like $x - y \# c$, or comparison of a single clock with an integer like $x \# c$, where $\# \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$. For instance $(x - y \geq 1) \wedge (y < 2)$ is a zone. Zones can be efficiently represented using difference bound matrices (DBMs) [7,13]. This suggests that one should consider abstractions that give zones. As zones are convex, abstractions that range over zones are called convex abstractions. This is an important restriction as abstractions based on regions are usually not convex [8].

We propose a way to use non-convex abstractions and zone representations at the same time. We will only consider sets W of the form $\mathfrak{a}(Z)$ and represent them simply by the zone Z. This way we can represent states of an abstract transition system efficiently: we need just to store a zone. In order for this to work we need to be able to compute the transition relation $\Rightarrow_{\mathfrak{a}}$ on this representation. We also need to know when two zone representations stand for the same node in the abstract system. This is summarized in the following two requirements:

Transition compatibility: for every transition $(q, \mathfrak{a}(Z)) \Rightarrow_{\mathfrak{a}} (q', W')$ and the matching transition $(q, Z) \Rightarrow (q', Z')$ we have $W' = \mathfrak{a}(Z')$.

Efficient inclusion test: for every two zones Z, Z', the test $Z' \subseteq \mathfrak{a}(Z)$ is efficient. Ideally, it has the same complexity $\mathcal{O}(|X|^2)$ as the inclusion $Z' \subseteq Z$.

The first condition is quite easy to satisfy: We will show quickly below that every abstraction relation coming from time-abstract simulation [22] is transition compatible. The second condition is the main topic of the paper.

Definition 1 *(Time-abstract simulation).* A *(state based) time-abstract simulation* between two states of a transition system $\mathcal{S}_{\mathcal{A}}$ is a relation $(q, v) \preceq_{t.a.} (q', v')$ such that:

- $q = q'$,
- if $(q, v) \to^\delta (q, v + \delta) \to^t (q_1, v_1)$, then there exists a $\delta' \in \mathbb{R}_{\geq 0}$ such that $(q, v') \to^{\delta'} (q, v' + \delta') \to^t (q_1, v_1')$ satisfying $(q_1, v_1) \preceq_{t.a.} (q_1, v_1')$.

For two valuations v, v', we say that $v \preceq_{t.a.} v'$ if for every state q of the automaton, we have $(q, v) \preceq_{t.a.} (q', v')$. An abstraction $\mathfrak{a}_{\preceq_{t.a.}}$ based on a simulation $\preceq_{t.a.}$ can be defined as follows:

Definition 2 *(Abstraction based on simulation).* Given a set W, we define $\mathfrak{a}_{\preceq_{t.a.}}(W) = \{v \mid \exists v' \in W.\ v \preceq_{t.a.} v'\}$.

Definition 3 *(Timed-elapsed zone).* A zone Z is said to be *time-elapsed* if it is closed under time-successors: that is $Z = \{v + \delta \mid v \in Z, \delta \in \mathbb{R}_{\geq 0}\}$.

We had previously noted that all nodes (q, W) reachable through $\Rightarrow^*$ from the initial node (q_0, W_0) are all time elapsed. We can now show that transition relations coming from abstractions based on time-abstract simulations satisfy the transition compatibility condition.

Lemma 4. *Let $\mathfrak{a}_{\preceq_{t.a.}}$ be an abstraction based on a time-abstract simulation relation and let Z be a time-elapsed zone. For every transition $(q, \mathfrak{a}_{\preceq_{t.a.}}(Z)) \Rightarrow_{\mathfrak{a}_{\preceq_{t.a.}}} (q', W')$ and the matching transition $(q, Z) \Rightarrow (q', Z')$, we have $W' = \mathfrak{a}_{\preceq_{t.a.}}(Z')$.*

Proof. Let t be the transition corresponding to $\Rightarrow_{\mathfrak{a}_{\preceq_{t.a.}}}$ and $\Rightarrow$. We first prove that $W' \subseteq \mathfrak{a}_{\preceq_{t.a.}}(Z')$. Let $v' \in W'$. We will show that there exists a valuation in Z' that simulates v' with respect to $\preceq_{t.a.}$.

By the definition of the abstract symbolic transition $(q, \mathfrak{a}_{\preceq_{t.a.}}(Z)) \Rightarrow_{\mathfrak{a}_{\preceq_{t.a.}}} (q', W')$, there are a valuation $v_1 \in \mathfrak{a}_{\preceq_{t.a.}}(Z)$ and a time elapse $\delta_1 \in \mathbb{R}_{\geq 0}$ such that:

$$(q, v_1) \rightarrow^t \rightarrow^{\delta_1} (q', v'_1) \text{ and } v' \preceq_{t.a.} v'_1$$

Firstly consider the intermediate configuration obtained after the $\rightarrow^t$ transition from (q, v_1). Call it $(q', \overline{v}'_1)$. We know that $\overline{v}'_1 \in W'$. This valuation $\overline{v}'_1$ can elapse a time δ_1 and become v'_1. Given that $\preceq_{t.a.}$ is a time-abstract simulation, this intermediate valuation $\overline{v}'_1$ can simulate v' too:

$$(q, v_1) \rightarrow^t (q', \overline{v}'_1) \text{ and } v' \preceq_{t.a.} \overline{v}'_1 \tag{1}$$

Recall that $v_1 \in \mathfrak{a}_{\preceq_{t.a.}}(Z)$. Therefore, there exists a valuation $v_2 \in Z$ such that $v_1 \preceq_{t.a.} v_2$. As (q, v_1) can take the transition $\rightarrow^t$, by definition of time-abstract simulation, there exists a time elapse δ_2 such that (q, v_2) can take the transition after the time elapse δ_2:

$$(q, v_2) \rightarrow^{\delta_2} \rightarrow^t (q', \overline{v}'_2) \text{ and } \overline{v}'_1 \preceq_{t.a.} \overline{v}'_2 \tag{2}$$

From (1) and (2), we see that $v' \preceq_{t.a.} \overline{v}'_2$. Note that as Z is a time-elapsed zone and since $v_2 \in Z$, we also have $v_2 + \delta_2 \in Z$ and this in turn implies that $\overline{v}'_2 \in Z'$. This shows that $v' \in \mathfrak{a}_{\preceq_{t.a.}}(Z')$ and hence $W \subseteq \mathfrak{a}_{\preceq_{t.a.}}(Z')$.

We will now show the converse: $\mathfrak{a}_{\preceq_{t.a.}}(Z') \subseteq W'$. Let $v \in \mathfrak{a}_{\preceq_{t.a.}}(Z')$. Then, there exist $v_1 \in Z$ and a $\delta_1 \in \mathbb{R}_{\geq 0}$ such that $(q, v_1) \rightarrow^t \rightarrow^{\delta_1} (q', v'_1)$ and $v \preceq_{t.a.} v'_1$. By the property of an abstraction operator, we will have $v_1 \in \mathfrak{a}_{\preceq_{t.a.}}(Z)$ too. Now, directly by the definition of $(q, \mathfrak{a}_{\preceq_{t.a.}}(Z)) \Rightarrow^{\mathfrak{a}_{\preceq_{t.a.}}} (q', W')$, we get that $v \in W'$ and hence $\mathfrak{a}_{\preceq_{t.a.}}(Z') \subseteq W'$. □

This paper is essentially about how to satisfy the "efficient inclusion test" condition and get as good abstraction as possible at the same time. In this context, let us highlight an important remark that describes when an abstraction is better than another.

Remark 5. If $\mathfrak{a}$ and $\mathfrak{b}$ are two abstractions such that for every set of valuations W, we have $\mathfrak{a}(W) \subseteq \mathfrak{b}(W)$, we prefer to use $\mathfrak{b}$ since the graph induced by $\mathfrak{b}$ is *a priori* smaller than the one induced by $\mathfrak{a}$ (sic [4]). In such a case, the abstraction $\mathfrak{b}$ is said to be *coarser* than abstraction $\mathfrak{a}$.

2.3. Bounds as parameters for abstraction

Remark 5 suggests to make use of the coarsest possible abstraction. For a given automaton it can be computed if two configurations are in a simulation relation. It should be noted though that computing the biggest simulation relation is EXPTIME-hard [19]. Since the reachability problem can be solved in PSPACE, this suggests that it may not be reasonable to try to solve it using the abstraction based on the biggest simulation.

We can get simulation relations that are computationally easier if we consider only a part of the structure of the automaton. The simplest is to take a simulation based on the maximal constant that appears in guards. More refined is to take the maximum separately over constants from lower bound constraints, that is in guards of the form $x > c$ or $x \geq c$, and those from upper bound constraints, that is in guards $x < c$ or $x \leq c$. If one moreover does this for every clock x separately, one gets for each clock two integers L_x and U_x. The abstraction that is currently most used is a refinement of this method by calculating L_x and U_x for every state of the automaton separately [3]. For simplicity of notation we will not consider this optimization but it can be incorporated with no real difficulty in everything that follows. We summarize this presentation in the following definition.

Definition 6 *(LU-bounds).* The L bound for an automaton $\mathcal{A}$ is the function assigning to every clock a maximal constant that appears in a lower bound guard for x in $\mathcal{A}$. Similarly U but for upper bound guards. An *LU-guard* is a guard where lower bound guards use only constants bounded by L and upper bound guards use only constants bounded by U. An *LU-automaton* is an automaton using only LU-guards.

In the rest of the paper, we try to find good abstractions parameterized by LU-bounds that also have an efficient inclusion test. Section 3 defines an abstraction abs_{LU} and proves that this is the optimal sound and complete abstraction that is based on LU-bounds. Section 4 then shows that the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is the same as abs_{LU} when zones closed under time successors are considered. We then give an efficient test $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ in Section 5, which enables the use of $\mathfrak{a}_{\preccurlyeq LU}$ in implementations.

3. The coarsest LU abstraction

We call an abstraction that is based on LU bounds an *LU abstraction*. A natural question is to know what is the coarsest LU-abstraction sound and complete for reachability testing. Given L and U bounds, we know that the automata under consideration have guards only of the following form (with $\prec \in \{<, \leq\}$ and $\succ \in \{>, \geq\}$):

$$x \gg 0,\ x \gg 1, \ldots,\ x \gg L_x$$

$$x \ll 0,\ x \ll 1, \ldots,\ x \ll U_x$$

However, we do not know the shape of the automata, in particular, the order in which the above guards appear in the paths of the automata.

An abstraction $\mathfrak{a}$ is sound if for every possible path using the above guards that a valuation $v \in \mathfrak{a}(W)$ can execute, there is a representative $v' \in W$ that can execute the same path. If this rule is not followed, there is one possible automaton with guards respecting the given LU-bounds for which this abstraction is not sound. Hence our question can be reworded as follows:

Given L and U bounds, what is the coarsest abstraction that is sound and complete for all LU-automata?

We answer this question in four steps:

Step 1. We define a generic simulation relation $\sqsubseteq_{LU}$ (Definition 7) which is a union over all time-abstract simulation relations on LU-automata. Roughly the simulation relation says that $v \sqsubseteq_{LU} v'$ if all paths, using LU-guards, executed by v can be executed by v'. We define an abstraction abs_{LU} that is based on LU-simulation (Definition 8). The definition of LU-simulation is difficult to work with as it talks about infinite sequences of transitions.

Step 2. The next aim is to characterize this LU-simulation using a finite sequence of transitions (Definition 11). We want to come up with a sequence of LU-guards $seq(v)$ executed by a valuation v for which we can say $v \sqsubseteq_{LU} v'$ iff v' executes this characteristic sequence $seq(v)$. To achieve this, we go through an intermediate definition of what we call LU-regions (Definition 9). We define this sequence in Definition 11.

Step 3. Steps 1 and 2 have defined the necessary notions. We now observe that the following are equivalent (Proposition 12, Corollary 13):
- $v \sqsubseteq_{LU} v'$,
- v' can execute $seq(v)$.

Step 4. The previous step gives a finite characterization of the generic LU-simulation $\sqsubseteq_{LU}$. We use this to prove that every sound abstraction should be contained in abs_{LU}, in other words abs_{LU} is the coarsest abstraction sound and complete for all LU-automata (Theorem 14).

Section 3.1 handles Step 1; Section 3.2 defines the LU-regions as mentioned in Step 2; Sections 3.3 and 3.4 handle Steps 3 and 4 respectively.

3.1. LU-simulation

Using LU-bounds we define a simulation relation on valuations without referring to any particular automaton; or to put it differently, by considering all LU-automata at the same time.

Definition 7 *(LU-simulation).* Let L, U be two functions giving an integer bound for every clock. The *LU-simulation relation* between valuations is the biggest relation $\sqsubseteq_{LU}$ such that if $v \sqsubseteq_{LU} v'$ then for every LU-guard g, and set of clocks $R \subseteq X$ we have

- if $v \xrightarrow{g,R} v_1$ for some v_1 then $v' \xrightarrow{g,R} v'_1$ for v'_1 such that $v_1 \sqsubseteq_{LU} v'_1$,

where $v \xrightarrow{g,R} v_1$ means that for some $\delta \in \mathbb{R}_{\geq 0}$ we have $v + \delta \models g$ and $v_1 = [R](v + \delta)$.

Note that in the above definition, the time elapse δ' required for v' to satisfy the guard g could be different from the time elapse δ required for v to satisfy the guard g. It is immediate that $\sqsubseteq_{LU}$ is the biggest relation that is a time-abstract simulation for all automata with given LU bounds. We define abstraction operator abs_{LU} to be the abstraction based on this LU-simulation.

Definition 8 *(Abstraction based on LU-simulation).* For a zone Z we define: $abs_{LU}(Z) = \{v \mid \exists v' \in Z.\ v \sqsubseteq_{LU} v'\}$.

The definition of LU-simulation is sometimes difficult to work with since it talks about infinite sequences of actions. We will present a useful characterization implying that actually we need to consider only very particular sequences of transitions that are of length bounded by the number of clocks (Corollary 13). Essentially, we are interested in the following question: given a valuation v, when does a valuation v' LU-simulate it, that is, when is $v \sqsubseteq_{LU} v'$. We start with a preparatory definition of what we call LU-regions.

3.2. LU-regions

We introduce the notion of LU-regions. The classical notion of regions [2] depends on the maximum bounds function M. Given only the maximum bounds M, we know that there could be guards $x \lessdot c$ and $x \gtrdot c$ for $c \in \{0, \ldots, M_x\}$ in the automaton. Let us call them the *M-guards*. However, with the LU-bounds, there is more information and consequently fewer guards: $x \lessdot c$ for $c \in \{0, \ldots, U_x\}$, and $x \gtrdot c$ for $c \in \{0, \ldots, L_x\}$. Note that for each x, we have $M_x = \max(L_x, U_x)$.

Let us denote the region of a valuation v by $[v]^M$. A valuation v' belongs to the region $[v]^M$ if two properties are satisfied:

Invariance by guards: v' satisfies all M-guards that v satisfies.
Invariance by time-elapse: for every time elapse $\delta \in \mathbb{R}_{\geq 0}$, there is a $\delta' \in \mathbb{R}_{\geq 0}$ such that $v' + \delta' \in [v + \delta]^M$.

We would like to define a notion of LU-regions in the same spirit, now with the additional information on the guards. For this discussion let us fix some L and U functions.

Definition 9 *(LU-region).* For a valuation v we define its *LU-region*, denoted $\langle v \rangle^{LU}$, to be the set of valuations v' such that:

- v' satisfies all LU-guards that v satisfies.
- For every pair of clocks x, y with $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$, $\lfloor v(y) \rfloor = \lfloor v'(y) \rfloor$, $v(x) \leq U_x$ and $v(y) \leq L_y$ we have:
  - if $0 < \{v(x)\} < \{v(y)\}$ then $\{v'(x)\} < \{v'(y)\}$;
  - if $0 < \{v(x)\} = \{v(y)\}$ then $\{v'(x)\} \leq \{v'(y)\}$.

The first invariance with respect to guards has been directly incorporated in the first condition of the definition. The second condition in the definition of LU-regions has been added in order to obtain the invariance by time-elapse property mentioned. Note that it is possible to have $v' \in \langle v \rangle^{LU}$ but $v \notin \langle v' \rangle^{LU}$. The following lemma will now show that with the two conditions specified in the definition, one can achieve the invariance with respect to time-elapse.

Lemma 10. *Let v, v' be valuations such that $v' \in \langle v \rangle^{LU}$. For all $\delta \in \mathbb{R}_{\geq 0}$, there exists a $\delta' \in \mathbb{R}_{\geq 0}$ such that $v' + \delta' \in \langle v + \delta \rangle^{LU}$.*

Proof. We are given valuations v and v' such that $v' \in \langle v \rangle^{LU}$. Therefore, v' satisfies all the LU-guards that v satisfies, and the property given by second condition of Definition 9 is true for the ordering of fractional parts. Let us call it the *order property*. Additionally, we are given a time elapse $\delta \in \mathbb{R}_{\geq 0}$ from the valuation v. We need to construct a value $\delta' \in \mathbb{R}_{\geq 0}$ such that $v' + \delta' \in \langle v + \delta \rangle^{LU}$.

Assume $\delta < 1$. Without loss of generality, we can assume that $\delta < 1$. If $\delta \geq 1$, then we can put $\lfloor \delta' \rfloor = \lfloor \delta \rfloor$ and consider the valuations $v + \lfloor \delta \rfloor$ and $v' + \lfloor \delta' \rfloor$. As we are not altering the fractional parts in these valuations, the order property is true for $v + \lfloor \delta \rfloor$ and $v' + \lfloor \delta' \rfloor$. It is also easy to see that as v' satisfies all LU-guards that v does, the valuation $v' + \lfloor \delta' \rfloor$ satisfies all LU-guards that $v + \lfloor \delta \rfloor$ does. This gives us $v' + \lfloor \delta' \rfloor \in \langle v + \lfloor \delta \rfloor \rangle^{LU}$ and we need to consider the time elapse $\{\delta\}$ from $v + \lfloor \delta \rfloor$. Therefore, in the rest of the proof, without loss of generality, we can assume that $\delta < 1$.

Assume $\lfloor v(z) \rfloor = \lfloor v'(z) \rfloor$ for all clocks z. Suppose for a clock z, we have $\lfloor v(z) \rfloor < \lfloor v'(z) \rfloor$. Then, as v' satisfies all guards which v does, it should be the case that $U_z < \lfloor v(z) \rfloor$. All guards having constants in between $\lfloor v(z) \rfloor$ and $\lfloor v'(z) \rfloor$ would be lower bound guards. So, irrespective of what we choose for δ', the value $v'(z) + \delta'$ will satisfy all the LU-guards with respect to z that $v(z) + \delta$ satisfies. Also, z does not concern the order property at all. Similarly, if $\lfloor v'(z) \rfloor < \lfloor v(z) \rfloor$, as v' satisfies all LU-guards that v satisfies, it should be the case that $L_z < \lfloor v'(z) \rfloor < \lfloor v(z) \rfloor$. In both the cases, we can safely ignore the clock z.

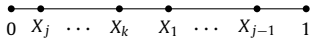
Assume $\lfloor v(z) \rfloor \leq \max(L_z, U_z)$ for all clocks z. For a clock z such that $\lfloor v(z) \rfloor$ is greater than both L_z and U_z, we know that $v'(z)$ should be greater than L_z in order to satisfy the same LU-guards. Hence any amount of time elapse would maintain this property and additionally such clocks do not concern order property. Hence, we assume without loss of generality that all clocks are less than at least one of the bounds.

Constructing δ'. We now have v and v' such that for all clocks z, the integral parts match, that is $\lfloor v(z) \rfloor = \lfloor v'(z) \rfloor$ and $\lfloor v(z) \rfloor \leq \max(L_z, U_z)$. Moreover, the delay is $\delta < 1$.

Let $0 \leq \lambda_1 < \lambda_2 < \cdots < \lambda_k < 1$ be the fractional parts of clocks in v. Let us denote by X_i the set of clocks z that have $\{v(z)\} = \lambda_i$. Similarly, let $0 \leq \lambda'_1 < \lambda'_2 < \cdots < \lambda'_{k'} < 1$ denote the fractional parts in v' and we define the set X'_i to be the set of clocks z such that $\{v'(z)\} = \lambda'_i$. This is pictorially illustrated below.

In v In v'

After a delay δ from v, some of the clocks cross the next integer, whereas some of them do not. Let us say that clocks in $X_j \cup \cdots \cup X_k$ have crossed the integer. Now the fractional parts of these clocks would be smaller than those of $X_1 \cup \cdots \cup X_{j-1}$ as shown below:



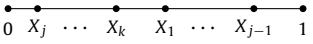
In $v+\delta$

We need to choose a value δ' so that for all clocks $y \in X_j \cup \cdots \cup X_k$ such that $(v+\delta)(y) \leq L_y$, the time elapse δ' takes v' to the next integer. Correspondingly for all the clocks $x \in X_1 \cup \cdots \cup X_{j-1}$ such that $(v+\delta)(x) \leq U_x$, the time elapse δ' still keeps v' within the same integer. Clearly we need this property to be satisfied so that $v'+\delta'$ satisfies the same LU-guards as $v+\delta$. To this regard, we define the following two values:

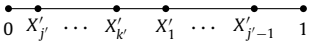
$$l = \min\{\, \{v'(y)\} \mid (v+\delta)(y) \leq L_y \text{ and } y \in X_j \cup \cdots \cup X_k\}$$
$$u = \max\{\, \{v'(x)\} \mid (v+\delta)(x) \leq U_x \text{ and } x \in X_1 \cup \cdots \cup X_{j-1}\}$$

Firstly note that $u < l$. If not, there exist clocks y and x such that $v(y) \leq L_y$ and $v(x) \leq U_x$, for which the order property is not true, thus giving a contradiction. Let $\overline{\delta}$ be a value between u and l. Set $\delta' = 1 - \overline{\delta}$. By construction, $v'+\delta'$ satisfies the same LU-guards as $v+\delta$.

We will now see that this choice of δ' also satisfies order property for $v'+\delta'$ and $v+\delta$. Due to δ' some clocks in v' would have crossed the next integer. Let us say that clocks in $X_{j'} \cup \cdots \cup X_{k'}$ have crossed and the others stay within the same integer. We pictorially depict the scenario with the two valuations $v+\delta$ and $v'+\delta'$ below.



In $v+\delta$



In $v'+\delta'$

Pick two clocks x, y such that:

$$\lfloor (v+\delta)(x) \rfloor = \lfloor (v'+\delta')(x) \rfloor \text{ and } \lfloor (v+\delta)(y) \rfloor = \lfloor (v'+\delta')(y) \rfloor \tag{3}$$
$$(v+\delta)(x) \leq U_x \text{ and } (v+\delta)(y) \leq L_y$$
$$\{(v+\delta)(x)\} < \{(v+\delta)(y)\}$$

Consider the case when both $x, y \in X_1 \cup \cdots \cup X_{j-1}$. As they have not crossed integer in v, they should not have crossed integer in v' too because of (3). Therefore both $x, y \in X'_1 \cup \cdots \cup X'_{j'-1}$. We know from order property that $\{v'(x)\} < \{v'(y)\}$. Clearly the time elapse of δ' has not changed this ordering for these clocks and hence $\{(v'+\delta')(x)\} < \{(v'+\delta')(y)\}$. We can prove similarly when both $x, y \in X_j \cup \cdots \cup X_k$.

Let us now consider the case when $y \in X_1 \cup \cdots \cup X_{j-1}$ and $x \in X_j \cup \cdots \cup X_k$. As x has crossed integer in v, it should have crossed integer in v' too by the hypothesis (3). Therefore $x \in X'_{j'} \cup \cdots \cup X'_{k'}$. Again by hypothesis (3) the clock y should not have crossed integer and hence $y \in X'_1 \cup \cdots \cup X'_{j'-1}$. Hence we get that $\{(v'+\delta')(x)\} < \{(v'+\delta')(y)\}$.

This way we have proved the order property for $v+\delta$ and $v'+\delta'$ for the case of the strict inequality. The case of the equality can be handled in a similar way. □

3.3. Finite paths characterizing LU-simulation

The previous section took a digression to define the notion of LU-regions. Now, we are in a position to answer the question: given two valuations v, v', when is $v \sqsubseteq_{LU} v'$. This section is devoted to show the link between this question and the definition of LU-regions. For valuations v, v', we will show that $v \sqsubseteq_{LU} v'$ if and only if v' can elapse some amount of time and fall into the LU-region of v (Proposition 12). Before that, we will define a sequence of guards that succinctly describes the LU-region $\langle v \rangle^{LU}$.

Definition 11 *(LU-sequence).* For a valuation v, let g_{int} be the conjunction of all LU guards that v satisfies. For every pair of clocks x, y such that $v(x) \leq U_x$, $v(y) \leq L_y$, consider guards:

- if $0 < \{v(x)\} < \{v(y)\}$ then we take a guard $g_{xy} \equiv (x < \lfloor v(x) \rfloor + 1) \wedge (y > \lfloor v(y) \rfloor + 1)$;
- if $0 < \{v(x)\} = \{v(y)\}$ then we take a guard $g_{xy} \equiv (x \leq \lfloor v(x) \rfloor + 1) \wedge (y \geq \lfloor v(y) \rfloor + 1)$.

For every y with $v(y) \leq L_y$ put $g_y = \bigwedge\{g_{xy} : v(x) \leq U_x\}$. Consider all the clocks y with $v(y) \leq L_y$ and suppose that $y_1, \ldots, y_k$ is the ordering of these clocks with respect to the value of their fractional parts: $0 \leq \{v(y_1)\} \leq \cdots \leq \{v(y_k)\}$. The LU-sequence $seq(v)$ is defined to be the sequence of transitions $\xrightarrow{g_{int}} \xrightarrow{g_{y_k}} \cdots \xrightarrow{g_{y_1}}$.

Proposition 12. *For every two valuations v and v':*

$$v \sqsubseteq_{LU} v' \quad \textit{iff} \quad \textit{there is } \delta' \in \mathbb{R}_{\geq 0} \textit{ with } v' + \delta' \in \langle v \rangle^{LU}.$$

Proof. First let us take v and consider its LU-sequence $seq(v)$. The sequence $seq(v)$ can be performed from v (the symbol τ denotes a time elapse):

$$v \xrightarrow{g_{int}} v \xrightarrow{\tau} v + \delta_k \xrightarrow{g_{y_k}} v + \delta_k \xrightarrow{\tau} v + \delta_{k-1} \xrightarrow{g_{y_{k-1}}} \dots \\ \dots \xrightarrow{\tau} v + \delta_1 \xrightarrow{g_{y_1}} v + \delta_1$$

when choosing $\delta_i = (1 - \{v(y_i)\})$ or $\delta_i = (1 - \{v(y_i)\}) + \varepsilon$ for some sufficiently small $\varepsilon > 0$; depending on whether we test for non-strict or strict inequality in g_{y_i}. Delay δ_i makes the value of y_i integer or just above integer.

If $v \sqsubseteq_{LU} v'$, then there exists a δ' such that $v' + \delta'$ can do the sequence of transitions given by $seq(v)$. The guard g_{int} ensures that $v' + \delta'$ satisfies the same LU-guards as v. Note that in particular, this entails that for every pair of clocks x, y such that $v(x) \leq U_x$, $v(y) \leq L_y$ and $\{v(x)\} > 0$, we have:

- $\lfloor (v' + \delta')(x) \rfloor < \lfloor v(x) \rfloor + 1$, and
- $\lfloor (v' + \delta')(y) \rfloor \geq \lfloor v(y) \rfloor$.

Following transition g_{int}, valuation $v' + \delta'$ can satisfy guards g_{y_k} to g_{y_1} by letting some time elapse:

$$v' + \delta' \xrightarrow{g_{int}} v' + \delta' \xrightarrow{\tau} v + \delta'_k \xrightarrow{g_{y_k}} v + \delta'_k \xrightarrow{\tau} v + \delta'_{k-1} \xrightarrow{g_{y_{k-1}}} \dots \\ \dots \xrightarrow{\tau} v' + \delta_1 \xrightarrow{g_{y_1}} v' + \delta'_1$$

Consider the clock y_i. If the integral part $\lfloor (v' + \delta')(y_i) \rfloor$ is strictly greater than $\lfloor v(y_i) \rfloor$, time elapse is not necessary to cross the guard g_{y_i}. On the other hand, if $\lfloor (v' + \delta')(y_i) \rfloor = \lfloor v(y_i) \rfloor$, then for the guard g_{y_i} to be crossed, δ'_i should be sufficiently large to make the value of $(v' + \delta'_i)(y_i)$ integer or just above integer. But at the same time, the guard g_{xy_i} is satisfied, which entails that for all x such that $v(x) \leq U_x$, $\lfloor v(x) \rfloor = \lfloor (v' + \delta')(x) \rfloor$, we get:

- if $0 < \{v(x)\} < \{v(y_i)\}$, then $\{(v' + \delta')(x)\} < \{(v' + \delta')(y_i)\}$ and
- if $0 < \{v(x)\} = \{v(y_i)\}$, then $\{(v' + \delta')(x)\} \leq \{(v' + \delta')(y_i)\}$

Therefore, from the definition of LU-regions, we get that $v' + \delta' \in \langle v \rangle^{LU}$. This shows left to right implication.

For the right to left implication we show that the relation $S = \{(v, v') : v' \in \langle v \rangle^{LU}\}$ is an LU-simulation relation. For this we take any $(v, v') \in S$, any LU guard g, and any reset R such that $v \xrightarrow{g,R} v_1$. We show that $v' \xrightarrow{g,R} v'_1$ for some v'_1 with $(v_1, v'_1) \in S$. The only non-trivial part in this is to show that if $v + \delta \models g$ for some δ, then there exists a δ' such that $(v + \delta, v' + \delta') \in S$ and $v' + \delta' \models g$. But this is exactly given by Lemma 10. □

In particular the proof shows the following.

Corollary 13. *For two valuations v, v':*

$$v \sqsubseteq_{LU} v' \textit{ iff } v' \textit{ can execute the sequence } seq(v).$$

3.4. Proof of optimality

We are now ready to show that $abs_{LU}(Z)$ (Definition 8), the abstraction based on $\sqsubseteq_{LU}$ simulation, is the coarsest sound and complete abstraction that uses solely the LU information.

Theorem 14. *The abs_{LU} abstraction is the coarsest abstraction that is sound and complete for all LU-automata. It is also finite.*

Proof. Suppose that we have some other abstraction $\mathfrak{a}'$ that is not included in abs_{LU} on at least one LU-automaton. This means that there is some LU automaton $\mathcal{A}_1$ and its reachable configuration (q_1, Z) such that $\mathfrak{a}'(Z) \setminus abs_{LU}(Z)$ is not empty. We suppose that $\mathfrak{a}'$ is complete and show that it is not sound.

Take $v \in \mathfrak{a}'(Z) \setminus abs_{LU}(Z)$. Consider the test sequence $seq(v)$ as in Corollary 13. From this corollary we know that it is possible to execute this sequence from v but it is not possible to do it from any valuation in Z since otherwise we would get $v \in abs_{LU}(Z)$.

As illustrated in Fig. 2 we add to $\mathcal{A}_1$ a new sequence of transitions constructed from the sequence $seq(v)$. We start this sequence from q_1, and let q_f be the final state of this new sequence. The modified automaton $\mathcal{A}_1$ started in the initial

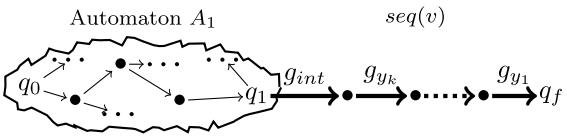


Fig. 2. Adding the sequence $seq(v)$ to $\mathcal{A}_1$.

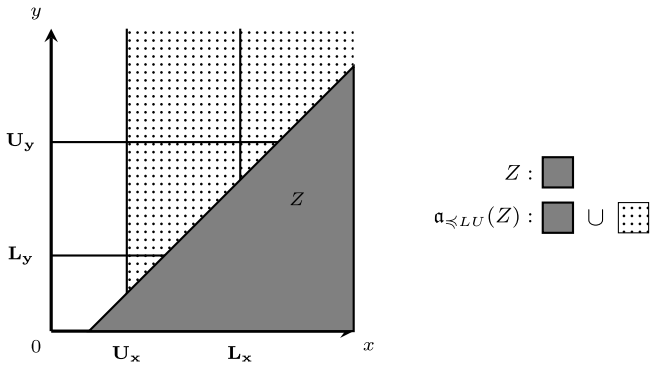


Fig. 3. Zone Z is given by the gray area. Abstraction $\mathfrak{a}_{\preccurlyeq LU}(Z)$ is given by the gray area along with the dotted area.

configuration arrives with (q_1, Z) in q_1 and then it can try to execute the sequence we have added. From what we have observed above, it will not manage to reach q_f. On the other hand from (q_1, v) it will manage to complete the sequence. But then by completeness of the abstraction $(q_1, \mathfrak{a}'(Z)) \xrightarrow{seq(v)} (q_f, W)$ for a nonempty W. So $\mathfrak{a}'$ is not a sound abstraction.

That abs_{LU} is finite is easy to see. The set $abs_{LU}(Z)$ is a union of classical regions. Recall that we denote by M the bound function that assigns to each clock x, the maximum of L_x and U_x. Let v' be a valuation in Z. If $v' \in [v]^M$ then it is easy to see that $v' \in \langle v \rangle^{LU}$ and by definition $v' \in abs_{LU}(Z)$. □

4. The $\mathfrak{a}_{\preccurlyeq LU}$ abstraction

Since abs_{LU} is the coarsest abstraction, we would like to use it in a reachability algorithm. The definition of abs_{LU}, or even the characterization referring to LU-regions, is still too complicated to work with. It turns out though that there is a close link to an existing abstraction.

The $\mathfrak{a}_{\preccurlyeq LU}$ abstraction proposed by Behrmann et al. in [4] has a much simpler definition. Quite surprisingly, in the context of reachability analysis the two abstractions coincide (Theorem 19). The $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is based on a simulation relation $\preccurlyeq_{LU}$.

Definition 15 *(LU-preorder [4]).* Let $L, U : X \to \mathbb{N} \cup \{-\infty\}$ be two bound functions. For a pair of valuations we set $v \preccurlyeq_{LU} v'$ if for every clock x:

- if $v'(x) < v(x)$ then $v'(x) > L_x$, and
- if $v'(x) > v(x)$ then $v(x) > U_x$.

It has been shown in [4] that $\preccurlyeq_{LU}$ is a simulation relation. The $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is based on this relation.

Definition 16 *($\mathfrak{a}_{\preccurlyeq LU}$-abstraction [4]).* Given L and U bound functions, for a set of valuations W we define:

$$\mathfrak{a}_{\preccurlyeq LU}(W) = \{v \mid \exists v' \in W.\ v \preccurlyeq_{LU} v'\}.$$

Fig. 3 gives an example of a zone Z and its abstraction $\mathfrak{a}_{\preccurlyeq LU}(Z)$. It can be seen that $\mathfrak{a}_{\preccurlyeq LU}(Z)$ is not a convex set.

4.1. Abstractions abs_{LU} and $\mathfrak{a}_{\preccurlyeq LU}$ coincide

Our goal is to show that when we consider zones closed under time-successors, $\mathfrak{a}_{\preccurlyeq LU}$ and abs_{LU} coincide. To prove this, we would first show that there is a very close connection between valuations in $\langle v \rangle^{LU}$ and valuations that simulate v with respect to $\preccurlyeq_{LU}$. The following lemma says that if $v' \in \langle v \rangle^{LU}$ then by slightly adjusting the fractional parts of v' we can get a valuation v'_1 such that $v \preccurlyeq_{LU} v'_1$. We start with a preliminary definition.

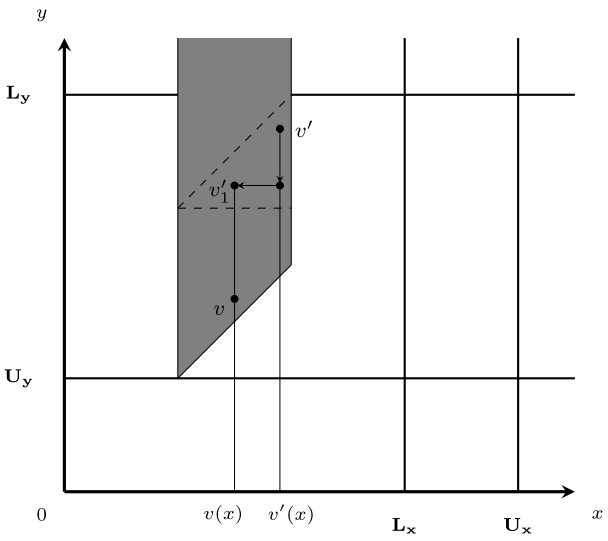


Fig. 4. Intuition for the adjustment lemma.

Definition 17. A valuation v_1 is said to be in the *neighborhood* of v, written $v_1 \in \mathrm{nbd}(v)$ if for all clocks x, y:

- $\lfloor v(x) \rfloor = \lfloor v_1(x) \rfloor$,
- $\{v(x)\} = 0$ iff $\{v_1(x)\} = 0$,
- $\{v(x)\} \lessdot \{v(y)\}$ implies $\{v_1(x)\} \lessdot \{v_1(y)\}$ where $\lessdot$ is either $<$ or $=$.

Notice that the neighborhood of v is the same as the region of v with respect to the classical region definition with maximal bound being ∞.

We give a brief intuition before proving the following lemma which gives the relation between LU-regions and $\preccurlyeq_{LU}$ simulation. Consider a valuation v shown in Fig. 4. Its LU-region $\langle v \rangle^{LU}$ is given by the shaded portion. Pick a valuation v' that belongs to $\langle v \rangle^{LU}$ and let us see if it satisfies $v \preccurlyeq_{LU} v'$.

As we can see in Fig. 4, the value of $v'(x) > v(x)$ and additionally $v'(x) \leq U_x$. This shows that $v \not\preccurlyeq_{LU} v'$ due to its x-coordinate. However, by slightly adjusting the fractional parts: that is, reducing $\{v'(y)\}$ to move it down a bit and then reducing $\{v'(x)\}$ to make $v'(x)$ equal to $v(x)$ leads us to a valuation v'_1 which is in the neighborhood of v' but now $v \preccurlyeq_{LU} v'_1$. The adjustment is depicted in Fig. 4. Essentially, the following lemma claims that the LU-region $\langle v \rangle^{LU}$ can be obtained as the downward closure of $\preccurlyeq_{LU}$ over the set $\mathrm{nbd}(v)$, in other words, $\mathfrak{a}_{\preccurlyeq_{LU}}(\mathrm{nbd}(v))$.

Lemma 18 *(Adjustment). Let v be a valuation and let $v' \in \langle v \rangle^{LU}$. Then, there exists a $v'_1 \in \mathrm{nbd}(v')$ such that $v \preccurlyeq_{LU} v'_1$.*

Proof. Let $v' \in \langle v \rangle^{LU}$. The goal is to construct a valuation $v'_1 \in \mathrm{nbd}(v')$ that satisfies $v \preccurlyeq_{LU} v'_1$. To be in the neighborhood, the valuation v'_1 should have the same integral parts as that of v' and should agree on the ordering of fractional parts. So for all x, we put $\lfloor v'_1(x) \rfloor = \lfloor v'(x) \rfloor$. It remains to choose the fractional parts for v'_1. But before, we will first see that there are clocks for which irrespective of what the fractional part is, the two conditions in Definition 15 would be true.

Consider a clock x that has $\lfloor v'(x) \rfloor < \lfloor v(x) \rfloor$. Since v' satisfies all LU-guards as v, we should have $v'(x) > L_x$. The first condition of $\preccurlyeq_{LU}$ for x becomes true and the second condition is vacuously true. Similarly, when $\lfloor v'(x) \rfloor > \lfloor v(x) \rfloor$, we should have $v(x) > U_x$ and the second condition of $\preccurlyeq_{LU}$ becomes true and the first condition is vacuously true. Therefore, clocks x that do not have the same integral part in v and v' satisfy the $\preccurlyeq_{LU}$ condition directly thanks to the different integral parts. Whatever the fractional parts of v'_1 are, the $\preccurlyeq_{LU}$ condition for these clocks would still be true.

Let us therefore now consider only the clocks that have the same integral parts: $\lfloor v'(x) \rfloor = \lfloor v(x) \rfloor$. If this integer is strictly greater than both L_x and U_x, the two conditions of $\preccurlyeq_{LU}$ would clearly be satisfied, again irrespective of the fractional parts. So we consider only the clocks x that have the same integral part in both v and v' and additionally either $\lfloor v(x) \rfloor \leq U_x$ or $\lfloor v(x) \rfloor \leq L_x$.

We prune further from among these clocks. Suppose there is such a clock that has $\{v'(x)\} = 0$. To be in the neighborhood, we need to set $\{v'_1(x)\} = 0$. If $\{v(x)\}$ is 0 too, we are done as the $\preccurlyeq_{LU}$ condition becomes vacuously true. Otherwise, we would have $v'(x) = v'_1(x) < v(x)$. But recall that $v' \in \langle v \rangle^{LU}$ and so it satisfies the same LU-guards as v does. This entails that $v'_1(x) > L_x$ and we get the first condition of $\preccurlyeq_{LU}$ to be true. Once again, the other condition is trivial. So we eliminate clocks that have zero fractional parts in v'. A similar argument can be used to eliminate clocks that have zero fractional parts in v.

So finally, we end up with the set of clocks x that have:

- $\lfloor v'(x) \rfloor = \lfloor v(x) \rfloor$,
- $\{v'(x)\} > 0$ and $\{v(x)\} > 0$,
- $v(x) < \max(U_x, L_x)$.

Call this set X_f. The task is to select non-zero fractional values $\{v'_1(x)\}$ for all clocks x in X_f so that they match with the order in v'. This is the main challenge and this is where we would be using the second property in the definition of $v' \in \langle v \rangle^{LU}$, which we restate here:

$$\begin{aligned} &\forall x, y \in X_f \text{ such that } v(x) \le U_x \text{ and } v(y) \le L_y \\ &\quad 0 < \{v(x)\} < \{v(y)\} \Rightarrow \{v'(x)\} < \{v'(y)\} \\ &\quad 0 < \{v(x)\} = \{v(y)\} \Rightarrow \{v'(x)\} \le \{v'(y)\} \end{aligned} \tag{4}$$

Let $0 < \lambda'_1 < \lambda'_2 < \cdots < \lambda'_n < 1$ be the fractional values taken by clocks of X_f in v', that is, for every clock $x \in X_f$, the fractional value $\{v'(x)\} = \lambda'_i$ for some $i \in \{1, \ldots, n\}$. Let X_i be the set of clocks $x \in X_f$ that have the fractional value as λ'_i:

$$X_i = \{x \in X_f \mid \{v'(x)\} = \lambda'_i\}$$

for $i \in \{1, \ldots, n\}$.

In order to match with the ordering of v', one can see that for all clocks x_i in some X_i, the value of $\{v'_1(x_i)\}$ should be the same, and if $x_j \in X_j$ with $i \neq j$, then we need to choose $\{v'_1(x_i)\}$ and $\{v'_1(x_j)\}$ depending on the order between λ'_i and λ'_j.

Therefore, we need to pick n values $0 < \sigma_1 < \sigma_2 < \cdots < \sigma_n < 1$ and assign for all $x_i \in X_i$, the fractional part $\{v'_1(x_i)\} = \sigma_i$. We show that it can be done by an induction involving n steps.

After the kth step of the induction we assume the following hypothesis:

- we have picked values $0 < \sigma_{n-k+1} < \sigma_{n-k+2} < \cdots < \sigma_n < 1$,
- for all clocks $x \in X_{n-k+1} \cup X_{n-k+2} \cdots \cup X_n$, the $\preccurlyeq_{LU}$ condition is satisfied,
- for all clocks $y \in X_1 \cup X_2 \cdots \cup X_{n-k}$, we have

$$v(y) \le L_y \Rightarrow \{v(y)\} < \sigma_{n-k+1} \tag{5}$$

Let us now perform the $k+1$th step and show that the induction hypothesis is true for $k+1$. The task is to pick σ_{n-k}. We first define two values $0 < l < 1$ and $0 < u < 1$ as follows:

$$\begin{aligned} l &= \max\left\{ \{v(z)\} \mid z \in X_{n-k} \text{ and } v(z) \le L_z \right\} \\ u &= \min\left\{ \{ \{v(z)\} \mid z \in X_{n-k} \text{ and } v(z) \le U_z \} \cup \sigma_{n-k+1} \right\} \end{aligned}$$

We claim that $l \le u$. Firstly, $l < \sigma_{n-k+1}$ from the third part of the induction hypothesis. So if u is σ_{n-k+1} we are done. If not, suppose $l > u$, this means that there are clocks $x, y \in X^{n-k}$ with $v(x) \le U_x$ and $v(y) \le L_y$ such that $\{v(x)\} < \{v(y)\}$. From Equation (4), this would imply that $\{v'(x)\} < \{v'(y)\}$. But this leads to a contradiction since we know they both equal λ'_{n-k} in v'.

This leaves us with two cases, either $l = u$ or $l < u$. When $l = u$, we pick $\sigma_{n-k} = l = u$. Firstly, from the third part of the hypothesis, we should have $l < \sigma_{n-k+1}$ and so $\sigma_{n-k} < \sigma_{n-k+1}$. Secondly for all $z \in X_{n-k}$, if $v'_1(z) < v(z)$, then z should not contribute to l and so $v(z) > L_z$, which is equivalent to saying, $v'_1(z) > L_z$. Similarly, if $v'_1(z) > v(z)$, then z should not contribute to u and so $v(z) > U_z$, thus satisfying the $\preccurlyeq_{LU}$ condition for z. Finally, we should show the third hypothesis. Consider a clock $y \in X_1 \cup \cdots \cup X_{n-k-1}$ with $v(y) < L_y$. If $\{v(y)\} \ge \sigma_{n-k}$, it would mean that $\{v(y)\} \ge u$ and from Equation (4) gives a contradiction. So the three requirements of the induction assumption are satisfied after this step in this case.

Now suppose $l < u$. Consider a clock $y \in X_1 \cup \cdots \cup X_{n-k-1}$ such that $v(y) < L_y$. From Equation (4), we should have $\{v(y)\} < u$. Take the maximum of $\{v(y)\}$ over all such clocks:

$$\lambda = \max\{\{v(y)\} \mid y \in X_1 \cup \cdots \cup X_{n-k-1} \text{ and } v(y) < L_y\}$$

Choose σ_{n-k} in the interval (λ, u). We can see that all the three assumptions of the induction hold after this step. □

We can now prove the main result of this section. Recall Definition 3 of time-elapsed zones.

Theorem 19. *If Z is time-elapsed then*

$$abs_{LU}(Z) = \mathfrak{a}_{\preccurlyeq LU}(Z)$$

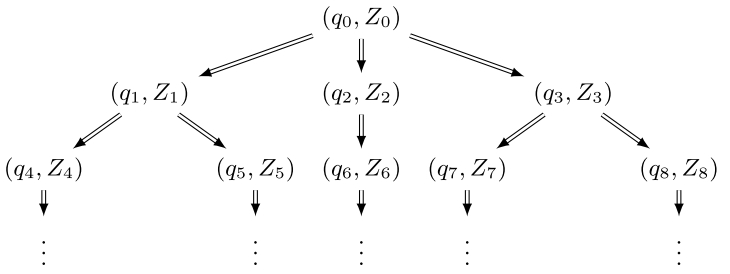


Fig. 5. A reachability tree of zones computed by a forward exploration.

Proof. Suppose $v \in \mathfrak{a}_{\preccurlyeq LU}(Z)$. There exists a $v' \in Z$ such that $v \preccurlyeq_{LU} v'$. It can be easily verified that $\preccurlyeq_{LU}$ is a LU-simulation relation. Since $\sqsubseteq_{LU}$ is the biggest LU-simulation, we get that $v \sqsubseteq_{LU} v'$. Hence $v \in abs_{LU}(Z)$.

Suppose $v \in abs_{LU}(Z)$. There exists $v' \in Z$ such that $v \sqsubseteq_{LU} v'$. From Proposition 12, this implies that there exists a δ' such that $v' + \delta' \in \langle v \rangle^{LU}$. As Z is time-elapsed, we get $v' + \delta' \in Z$. Moreover, from Lemma 18, we know that there is a valuation $v'_1 \in \mathrm{nbd}(v' + \delta')$ such that $v \preccurlyeq_{LU} v'_1$. Every valuation in the neighborhood of $v' + \delta'$ satisfies the same constraints of the form $y - x \prec c$ with respect to all clocks x, y and hence v'_1 belongs to Z too. Therefore, we have a valuation $v'_1 \in Z$ such that $v \preccurlyeq_{LU} v'_1$ and hence $v \in \mathfrak{a}_{\preccurlyeq LU}(Z)$. □

4.2. Using $\mathfrak{a}_{\preccurlyeq LU}$ to solve the reachability problem

A forward exploration algorithm for solving the reachability problem constructs the reachability tree starting from the initial node (q_0, Z_0) (cf. Fig. 5), with $Z_0 = \{\mathbf{0} + \delta \mid \delta \in \mathbb{R}_{\geq 0}\}$. The successor with respect to $\Rightarrow$ can be computed in time $\mathcal{O}(|X|^2)$ where X is the number of clocks [26]. By definition $\Rightarrow$ computes a time-elapsed zone. Therefore, all nodes that are explored by the algorithm have time-elapsed zones.

Before continuing exploration from a node (q, Z), the algorithm first checks if q is accepting. If not, the algorithm checks if for some visited node (q, Z'), we have $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$. If this is the case, (q, Z) need not be explored. Otherwise, the successors of (q, Z) are computed as stated above. This way we ensure termination of the algorithm since $\mathfrak{a}_{\preccurlyeq LU}$ is a finite abstraction [4].

Since the reachability algorithm refers to only time-elapsed zones, Theorems 14 and 19 show that $\mathfrak{a}_{\preccurlyeq LU}$ is the coarsest sound and complete abstraction provided the only thing we know about the structure of the automaton are its L and U bounds. Recall that coarser abstractions make abstract graph smaller, so the exploration algorithm can finish faster.

In Definition 6, we introduced LU-bounds that associate an L bound and a U bound to every clock in an automaton $\mathcal{A}$. Those bounds are the same in every state of the automaton. Instead, state-of-the-art algorithms calculate LU-bounds for each state of the automaton separately [3], or even on-the-fly during exploration [15,17]. The maximality argument in favor of $\mathfrak{a}_{\preccurlyeq LU}$ is of course true also in this case.

The last missing piece is an efficient inclusion test $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$. This is the main technical contribution of this paper.

5. An $\mathcal{O}(|X|^2)$ algorithm for $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$

In this section, we present an efficient algorithm for the inclusion test $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ (Theorem 32). In the algorithm for the reachability problem as explained in Section 4.2, each time a new node (q, Z) is seen, it is checked if there exists an already visited node (q, Z') such that $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$. This means that a lot of such inclusion tests need to be performed during the course of the algorithm. Hence it is essential to have a low complexity for this inclusion procedure. We are aiming at quadratic complexity as this is the complexity incurred in the existing algorithms for inclusions of the form $Z \subseteq Z'$ used in the standard reachability algorithm. It is well known that all the other operations needed for forward exploration, can be done in at most quadratic time [26].

We will now characterize when $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ holds. The main steps are outlined below.

Step 1. As a first step, we reduce the inclusion problem to a problem of intersection in Section 5.1. The question $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ boils down to asking if there exists a valuation $v \in Z$ such that its LU-region $\langle v \rangle^{LU}$ does not intersect Z' (Proposition 21).

Step 2. As a next step, we consider the intersection $\langle v \rangle^{LU} \cap Z'$. We aim to show that this intersection can be decided by looking at projections on every pair of clocks (Proposition 28). This is the most difficult step in the way to the inclusion test and spans three sections. We first describe a convenient graph representation of zones in Section 5.2. We call this the distance graph and will use it to represent Z'. Subsequently, in Section 5.3, we see how we can represent LU-regions as distance graphs. This gives a distance graph representation for $\langle v \rangle^{LU}$. Finally, in Section 5.4, we analyze the graph representations of $\langle v \rangle^{LU}$ and Z' to see when the intersection $\langle v \rangle^{LU} \cap Z'$ is empty. We show that for this it is enough to look at two clocks at a time.

Step 3. The previous step gives the condition for $\langle v\rangle^{LU} \cap Z'$ to be empty. We now look at the zone Z to find out quickly the valuation $v \in Z$ that can potentially satisfy this condition (Proposition 31).

Step 4. We substitute the valuation obtained from Step 3 to the condition of Step 2 to give the efficient test for inclusion (Theorem 32). Both steps 3 and 4 appear in Section 5.5.

Notation. For notational convenience, we denote $v(x)$ by v_x for a valuation v and clock x.

5.1. Reducing inclusion to intersection

The aim of this chapter is to reduce the question of inclusion to a question of intersection. The adjustment lemma (Lemma 18) shows a close connection between LU-regions and $\preccurlyeq_{LU}$-simulation in one direction: that is, if $v' \in \langle v\rangle^{LU}$ then we can find a valuation v'_1 in the neighborhood of v' such that $v \preccurlyeq_{LU} v'_1$. We show below a connection in the other direction too.

Lemma 20. *Let v, v' be valuations. If $v \preccurlyeq_{LU} v'$, then $v' \in \langle v\rangle^{LU}$.*

Proof. It is not difficult to see from the definition of $\preccurlyeq_{LU}$ (Definition 15) that both v and v' satisfy the same LU-guards. It remains to show the second property for v' to be in $\langle v\rangle^{LU}$.

Let x, y be clocks such that $\lfloor v_x\rfloor = \lfloor v'_x\rfloor$, $\lfloor v_y\rfloor = \lfloor v'_y\rfloor$, and $v_x \le U_x$, $v_y \le L_y$. Suppose $\{v_x\} \lessdot \{v_y\}$, for $\lessdot$ being either $<$ or $=$. As $v \preccurlyeq_{LU} v'$, if $v'_x > v_x$, we need $v_x > U_x$ which is not true. Hence we can conclude that $v'_x \le v_x$. Similarly, for y, one can conclude that $v'_y \ge v_y$. As the integer parts are the same in v and v', we get $\{v'_x\} < \{v'_y\}$ or $\{v'_x\} \le \{v'_y\}$ depending on whether $\lessdot$ is $<$ or $=$. □

The above along with the adjustment lemma help us to reduce the question of inclusion as a question of intersection.

Proposition 21. *Let Z, Z' be zones. Then, $Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$ iff there exists a valuation $v \in Z$ such that $\langle v\rangle^{LU} \cap Z'$ is empty.*

Proof. Consider the left-to-right direction. Suppose $Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$. Then there exists a valuation $v \in Z$ such that for every valuation $v' \in Z'$ we have $v \not\preccurlyeq_{LU} v'$. Pick an arbitrary $v' \in Z'$. In particular, every valuation $v'_1 \in \mathrm{nbd}(v')$ satisfies $v \not\preccurlyeq_{LU} v'_1$. From the Adjustment Lemma 18, we get that $v' \notin \langle v\rangle^{LU}$. Since v' is arbitrary, we get that $\langle v\rangle^{LU} \cap Z'$ is empty.

Now for the right-to-left direction. Suppose $\langle v\rangle^{LU} \cap Z'$ is empty. Then by Lemma 20, we get that $v \not\preccurlyeq_{LU} v'$ for every valuation $v' \in Z'$. This shows that $Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$. □

5.2. Distance graphs

Thanks to Proposition 21, we know that to solve $Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$, we need to check if there exists a valuation $v \in Z$ such that its LU-region $\langle v\rangle^{LU}$ does not intersect with the zone Z'. The focus now is to study this intersection.

We will begin with a convenient representation of zones that we use to solve this intersection question. The standard way to represent a zone is using a DBM. An equivalent representation is in terms of graphs [21] which we call here *distance graphs*.

Definition 22 *(Distance graph).* A *distance graph* G has clocks as vertices, with an additional special vertex x_0 representing constant 0. Between every two vertices there is an edge with a *weight* of the form $(\lessdot, c)$ where $c \in \mathbb{Z}$ and $\lessdot \in \{\le, <\}$ or $(\lessdot, c) = (<, \infty)$. An edge $x \xrightarrow{\lessdot c} y$ represents a constraint $y - x \lessdot c$: or in words, the distance from x to y is bounded by c. We let $[\![G]\!]$ be the set of valuations of clock variables satisfying all the constraints given by the edges of G with the restriction that the value of x_0 is 0. We also say that the $y - x$*-constraint* of $[\![G]\!]$ is $\lessdot c$ if the weight of the $x \to y$ edge in G is $\lessdot c$.

For readability, we will often write 0 instead of x_0. Fig. 6 illustrates a zone with its constraints and the corresponding distance graph.

An arithmetic over the weights $(\lessdot, c)$ can be defined as follows [6].

Equality $(\lessdot_1, c_1) = (\lessdot_2, c_2)$ if $c_1 = c_2$ and $\lessdot_1 = \lessdot_2$.
Addition $(\lessdot_1, c_1) + (\lessdot_2, c_2) = (\lessdot, c_1 + c_2)$ where $\lessdot = <$ iff either $\lessdot_1$ or $\lessdot_2$ is $<$.
Minus $-(\lessdot, c) = (\lessdot, -c)$.
Order $(\lessdot_1, c_1) < (\lessdot_2, c_2)$ if either $c_1 < c_2$ or $(c_1 = c_2$ and $\lessdot_1 = <$ and $\lessdot_2 = \le)$.

This arithmetic lets us talk about the weight of a path as a weight of the sum of its edges.

A cycle in a distance graph G is said to be *negative* if the sum of the weights of its edges is at most $(<, 0)$; otherwise the cycle is *positive*. The following proposition is folklore.

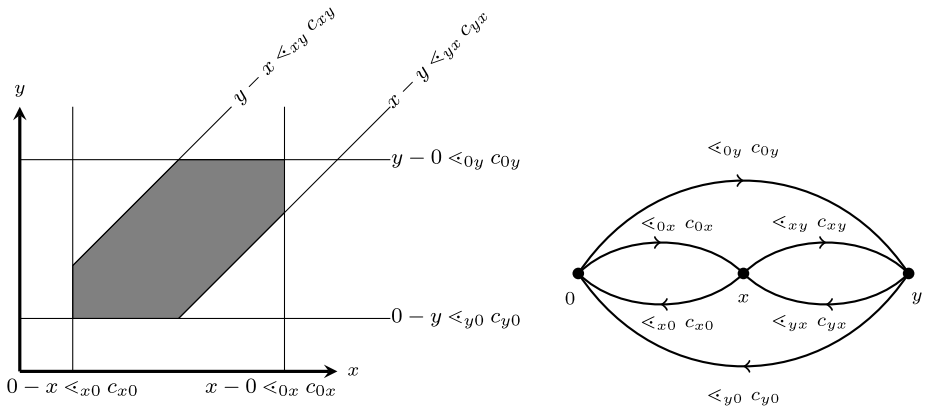


Fig. 6. An arbitrary zone with its constraints and distance graph.

Proposition 23. *A distance graph G has only positive cycles iff $[\![G]\!] \neq \emptyset$.*

A distance graph is in *canonical form* if the weight of the edge from x to y is the lower bound of the weights of paths from x to y. The canonical form only exists for graphs without negative cycle. Given a distance graph, its canonical form can be computed by using an all-pairs shortest paths algorithm like Floyd–Warshall's [6] in time $\mathcal{O}(|X|^3)$ where $|X|$ is the number of clocks. Note that the number of vertices in the distance graph is $|X|+1$. For computing the successors of a node in the zone graph, the most complex operation is the computation of $Z \wedge g$ which involves a canonicalization operation. However, since g has diagonal free constraints, the canonicalization procedure involved to compute $Z \wedge g$ is easier and costs only $\mathcal{O}(|X|^2)$ [26].

Recall that we have reduced the problem $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ to checking if there exists v such that $\langle v \rangle^{LU} \cap Z'$ is empty. For this, we need to know when the intersection of an LU-region and a zone is empty. In the next section we will see that an LU-region is a zone and can be represented using a distance graph. Therefore, it boils down to asking given two distance graphs G_1 and G_2 when is $[\![G_1]\!] \cap [\![G_2]\!]$ empty.

For two distance graphs G_1, G_2 which are not necessarily in canonical form, we denote by $\min(G_1, G_2)$ the distance graph where each edge has the weight equal to the minimum of the corresponding weights in G_1 and G_2. Even though this graph may be not in canonical form, it should be clear that it represents intersection of the two arguments, that is, $[\![\min(G_1, G_2)]\!] = [\![G_1]\!] \cap [\![G_2]\!]$; in other words, the valuations satisfying the constraints given by $\min(G_1, G_2)$ are exactly those satisfying all the constraints from G_1 as well as from G_2.

Proposition 23 tells us that the intersection $[\![G_1]\!] \cap [\![G_2]\!]$ is empty iff the distance graph $\min(G_1, G_2)$ has a negative cycle.

5.3. LU-regions as distance graphs

Our aim is to check when the intersection of $\langle v \rangle^{LU}$ and Z' is empty. We saw in the previous subsection that zones can be conveniently and canonically represented by distance graphs. Here, we will see how we can canonically represent an LU-region of a valuation as a distance graph.

We will first recall a constructive definition of Alur–Dill regions.

Definition 24 *(Regions: constructive definition).* A region with respect to bound function M is the set of valuations specified as follows:

1. for each clock $x \in X$, one constraint from the set:
   $\{x = c \mid c = 0, \ldots, M_x\} \cup \{c - 1 < x < c \mid c = 1, \ldots, M_x\} \cup \{x > M_x\}$
2. for each pair of clocks x, y having interval constraints: $c - 1 < x < c$ and $d - 1 < y < d$, it is specified if $\{x\}$ is less than, equal to or greater than $\{y\}$.

The distance graph representing a region can be constructed using the above constructive definition of a region. For a valuation v, let G_v^M denote the canonical distance graph representing the region $[v]^M$. We are now interested in getting the LU-region of v, that is, $\langle v \rangle^{LU}$ as a distance graph. For convenience, we will recall below the definition of LU-regions.

▶ **Definition 9. (LU-region)**
For a valuation v we define its *LU-region*, denoted $\langle v \rangle^{LU}$, to be the set of valuations v' such that:

- v' satisfies the same LU-guards as v.

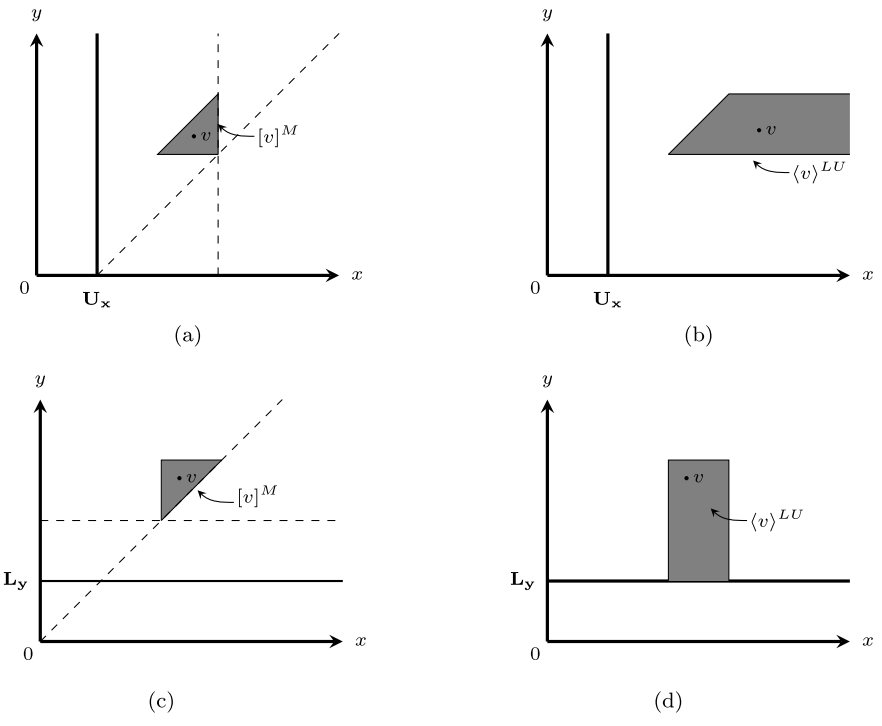


Fig. 7. $\langle v \rangle^{LU}$ can be thought of as a transformation of $[v]^M$ by altering select constraints. Pictures (a) and (b) handle the case when $v_x > U_x$; pictures (c) and (d) handle the case when $v_y > L_y$.

- For every pair of clocks x, y with $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$, $\lfloor v(y) \rfloor = \lfloor v'(y) \rfloor$, $v(x) \leq U_x$ and $v(y) \leq L_y$ we have:
  - if $0 < \{v(x)\} < \{v(y)\}$ then $\{v'(x)\} < \{v'(y)\}$;
  - if $0 < \{v(x)\} = \{v(y)\}$ then $\{v'(x)\} \leq \{v'(y)\}$.

For a valuation v, we need to collect all valuations v' satisfying the above two conditions to get $\langle v \rangle^{LU}$. We begin with a motivating example in Fig. 7. In the (a) part of the figure, we consider a valuation v such that $v_x > U_x$. The shaded portion in Fig. 7 (a) shows the region $[v]^M$ and the finite valued $x - y$ and $x - 0$ constraints bounding this region (cf. Definition 22 for definition of $x - y$-constraints and Fig. 6 for an illustration). The LU-region $\langle v \rangle^{LU}$ is shown in Fig. 7 (b). Observe that it matches with the definition given above. But more importantly, note that it can be seen as a transformation of $[v]^M$ by moving its $x - y$ and $x - 0$ constraints to infinity and keeping the rest same.

We now consider a valuation v with $v_y > L_y$ in Fig. 7 (c). Once again, the shaded portion shows the region $[v]^M$ and the relevant boundary constraints. We depict the LU-region $\langle v \rangle^{LU}$ in Fig. 7 (d) matching the definition given above. Note that it can be seen as a transformation of $[v]^M$ by moving the $x - y$ constraint to infinity and the $0 - y$ constraint up to L_y. However, when we move $x - y$ to infinity, the graph that we get would no longer be canonical. We could then consider the canonicalization of the transformed graph.

These examples lead us to the definition of the distance graph G_v^{LU} for $\langle v \rangle^{LU}$ as a transformation of the distance graph G_v^M.

Definition 25 *(Distance graph G_v^{LU}).* Let v be valuation. Given the distance graph of the region $[v]^M$ in canonical form $G_v^M = (\lessdot_{xy}, c_{xy})_{x,y \in X}$, first define the distance graph $G' = (\lessdot'_{xy}, c'_{xy})_{x,y \in X}$ as follows:

$$(\lessdot'_{yx}, c'_{yx}) = \begin{cases} (<, \infty) & \text{if } v_x > U_x \\ (<, \infty) & \text{if } v_y > L_y \text{ and } x \neq 0 \\ (<, -L_y) & \text{if } v_y > L_y \text{ and } x = 0 \\ (\lessdot_{yx}, c_{yx}) & \text{otherwise} \end{cases}$$

Then, the distance graph G_v^{LU} is defined to be the canonical form of G'.

The following lemma confirms that the distance graph defined above indeed represents $\langle v \rangle^{LU}$.

Lemma 26. *Let v be a valuation. Let G_v^{LU} be the graph obtained by Definition 25. Then the sets $\langle v \rangle^{LU}$ and $[\![G_v^{LU}]\!]$ are equal.*

Proof. Let $v' \in \langle v \rangle^{LU}$. We will now show that $v' \in [\![G_v^{LU}]\!]$. First observe that $[\![G_v^{LU}]\!] = [\![G']\!]$ where G' is the graph as in Definition 25. Therefore it is sufficient to show that v' satisfies the constraints given by G'. From the definition, it is clear that an edge $y \to x$ is finite valued in G' only if $v_x \leq U_x$. Additionally when $v_y \leq L_y$, the value of the edge $y \to x$ is the same as that in G_v^M. Otherwise if $v_y > L_y$, the only finite value is $(<, -L_y)$ for the edge $y \to x_0$.

Since $v' \in \langle v \rangle^{LU}$, it satisfies all the LU-guards that v satisfies. If y is a clock such that $v_y > L_y$ then $v'_y > L_y$ too. So v' satisfies constraints of the form $y \xrightarrow{<-L_y} x_0$. It now remains to look at edges $y \xrightarrow{\preccurlyeq d} x$ with $v_y \leq L_y$, $v_x \leq U_x$ and the weight $(\preccurlyeq, d)$ coming from G_v^M. Let $\lfloor v_x \rfloor$ and $\lfloor v_y \rfloor$ be denoted as c_x and c_y respectively. As v' satisfies the same LU-guards as v, we have:

$$\begin{aligned} v'_x &< c_x + 1 \\ v'_y &\geq c_y \end{aligned} \tag{6}$$

Therefore $v'_x - v'_y < c_x + 1 - c_y$. Since G_v^M represents the region containing v, by definition of regions, the constant in the weight $(\preccurlyeq, d)$ is either $c_x - c_y + 1$ or $c_x - c_y$. If it is the former, then clearly, v' also satisfies $x - y \preccurlyeq d$. We need to consider the latter case, that is, d is $c_x - c_y$. Thanks to (6) above, if either $v'_x < c_x$ or $v'_y \geq c_y + 1$, we are done. We are left with considering the case when $\lfloor v'_x \rfloor = c_x$ and $\lfloor v'_y \rfloor = c_y$. We have:

$$\begin{aligned} & & v_x - v_y &\preccurlyeq c_x - c_y \\ \Rightarrow & & \{v_x\} - \{v_y\} &\preccurlyeq 0 \\ \Rightarrow & & \{v'_x\} - \{v'_y\} &\preccurlyeq 0 \qquad (\text{as } v' \in \langle v \rangle^{LU}) \\ \Rightarrow & & \lfloor v'_x \rfloor + \{v'_x\} - (\lfloor v'_y \rfloor + \{v'_y\}) &\preccurlyeq c_x - c_y \\ \Rightarrow & & v'_x - v'_y &\preccurlyeq d \end{aligned}$$

This proves that if $v' \in \langle v \rangle^{LU}$, then v' satisfies the constraints of G' and hence $v' \in [\![G']\!]$.

Now for the other direction, assume $v' \in [\![G']\!]$. We will show that $v' \in \langle v \rangle^{LU}$. Let x, y be clocks such that $v_x \leq U_x$ and $v_y \leq L_y$. From the definition of $[\![G']\!]$, edges of the form $y \to x_0$ and $x_0 \to x$ are retained as in G_v^M. Since $v' \in [\![G']\!]$, it is clear that v' satisfies the same LU-guards as v. We now consider the order property for LU-regions. From Definition 9, the order property only need to be considered when $\lfloor v'_x \rfloor = \lfloor v_x \rfloor$ and $\lfloor v'_y \rfloor = \lfloor v_y \rfloor$. Let us denote $c_x = \lfloor v'_x \rfloor = \lfloor v_x \rfloor$ and $c_y = \lfloor v'_y \rfloor = \lfloor v_y \rfloor$ By definition of G', the edge $y \to x$ in G' has the same weight as that in G_v^M. Let the edge weight $y \to x$ be $(\preccurlyeq, d)$. We have:

$$\begin{aligned} & & v_x - v_y &\preccurlyeq d \\ \Rightarrow & & \{v_x\} - \{v_y\} &\preccurlyeq d - (c_x - c_y) \end{aligned}$$

If $\{v_x\} < \{v_y\}$ then either $d - (c_x - c_y) < 0$ or if it is 0 then $\preccurlyeq$ is the strict inequality. As this edge remains in G', the valuation v' satisfies $v'_x - v'_y \preccurlyeq d$. Moreover, since the integral parts of v' match, we get $\{v'_x\} - \{v'_y\} \preccurlyeq d - (c_x - c_y)$. By the aforementioned property, we get $\{v'_x\} < \{v'_y\}$. A similar argument follows for the case when $\{v_x\} = \{v_y\}$. □

Before we use the distance graph G_v^{LU} for further analysis, recall that we first defined a graph G' in Definition 25 and then obtained G_v^{LU} by canonicalizing it. We will now observe some properties of G_v^{LU} that are either retained from G' or obtained thanks to canonicalization. These observations would be important in the next section when we do the analysis on the distance graph representing LU-region $\langle v \rangle^{LU}$ and zone Z'.

Lemma 27. *Let v be a valuation. Let G_v^M, G_v^{LU} be the canonical distance graphs of $[v]^M$ and $\langle v \rangle^{LU}$ respectively. For variables x, y, if the edge $y \to x$ has a finite value in G_v^{LU}, then:*

1. *$v_x \leq U_x$,*
2. *if $v_y \leq L_y$, the value of $y \to x$ in G_v^{LU} and G_v^M are equal,*
3. *if $v_y > L_y$, the value of $y \to x$ in G_v^{LU} equals the value of the path $y \to x_0 \to x$ in G_v^{LU}.*

Proof. The graph G_v^{LU} is the canonical form of the graph G' defined in Definition 25. By definition, if $v_x > U_x$, all incoming edges to x in G' have weight $(<, \infty)$. So, the shortest path in this graph G' from a variable y to a variable x such that $v_x > U_x$ is $(<, \infty)$. Therefore, if in the canonical form G_v^{LU}, the edge $y \to x$ is finite valued, we should have $v_x \leq U_x$. This gives the first part of the lemma.

Consider the second part of the lemma. We know that $v_y \leq L_y$ and from the first part of the lemma, we know that $v_x \leq U_x$. The weight of $y \to x$ in G' is the same as that of G_v^M according to Definition 25. Note that the finite values in the graph G' are either the same as that of G_v^M or of the form $(<, -L_z)$ for some edges $z \to 0$. In the latter case, we also know by definition that $v_z > L_z$. Therefore the value $(<, -L_z)$ is greater than the corresponding value in G_v^M. As G_v^M is canonical,

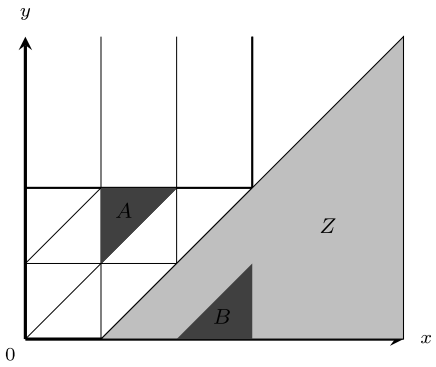


Fig. 8. A bounded region is either totally contained in a zone Z or totally disjoint from it. It can never intersect partially. For example, the bounded region B is totally inside Z, however the bounded region A is totally disjoint from Z.

the shortest path from y to x in G' cannot reduce from its value in G_v^M and hence equals just the edge value $y \to x$. This gives the second part of the lemma.

Finally consider the third part. Assume $v_y > L_y$. From Part 1, we know that $v_x \leq U_x$. By Definition 25, the weight of $y \to x$ equals $(<, \infty)$ if x is not x_0. The only finite valued outgoing edge from y is $y \to x_0$. Therefore, we can infer two things: the shortest path from y to x_0 is given by the edge $y \to x_0$; and the shortest path from y to x should contain this edge $y \to x_0$. Secondly, note that variable x_0 has $v_{x_0} \leq L_{x_0}$ ($v_{x_0} = 0 = L_{x_0}$). By definition, the value of $x_0 \to x$ in G' is given by the corresponding value in G_v^M and by Part 2, we know that this value stays in G_v^{LU}, that is, the shortest path from x_0 to x in G' is given by the direct edge. Summing up, the shortest path from y to x in G' is given by $y \to x_0 \to x$, where both $y \to x_0$ and $x_0 \to x$ are values coming from G_v^{LU}. □

5.4. When does an LU-region intersect a zone

We are now in a position to characterize the intersection $\langle v \rangle^{LU} \cap Z'$. Let G_v^{LU} as defined in the previous section be the canonical distance graph of $\langle v \rangle^{LU}$ and let $G_{Z'}$ be the canonical distance graph of Z'. By Proposition 23, the intersection $\langle v \rangle^{LU} \cap Z'$ is empty iff $\min(G_v^{LU}, G_{Z'})$ has a negative cycle.

We will now state a necessary and sufficient condition for the distance graph $\min(G_v^{LU}, G_{Z'})$ to have a negative cycle. We denote by Z'_{xy} the weight of the edge $x \to y$ in $G_{Z'}$. Similarly we denote $\langle v \rangle_{xy}^{LU}$ for the weight of $x \to y$ in G_v^{LU}. When a variable x represents the special clock x_0, we define $\langle v \rangle_{0x}^{LU}$ and $\langle v \rangle_{x0}^{LU}$ to be $(\leq, 0)$. Since by convention x_0 is always 0, this is consistent. We also denote $[v]_{xy}^M$ for the weight of $x \to y$ in G_v^M with the same convention when $x = x_0$.

The next proposition is the most important observation used in getting the final inclusion test.

Proposition 28. *Let v be a valuation and Z' a zone. The intersection $\langle v \rangle^{LU} \cap Z'$ is empty iff there exist two variables x, y such that $v_x \leq U_x$ and $Z'_{xy} + \langle v \rangle_{yx}^{LU} < (\leq, 0)$.*

To prove the above proposition, we need a small but a crucial observation that exploits the special structure of regions. A variable x is said to be bounded in valuation v if $v_x \leq \max(L_x, U_x)$. If x, y are bounded in v, then the projection of the region $[v]^M$ onto x, y has very specific boundaries. Call it a bounded region. The following lemma makes use of the fact that a bounded region is either fully contained in a zone or is totally disjoint from it, that is, there cannot be a partial intersection of the bounded region and zone, as illustrated in Fig. 8.

Lemma 29. *Let x, y be bounded variables of v appearing in some negative cycle N of $\min(G_v^{LU}, G_{Z'})$. Let the edge weights be $x \xrightarrow{\prec_{xy} c_{xy}} y$ and $y \xrightarrow{\prec_{yx} c_{yx}} x$ in G_v^M. If the value of the path $x \to \ldots \to y$ in N is strictly less than $(\prec_{xy}, c_{xy})$, then $x \to \ldots \to y \xrightarrow{\prec_{yx} c_{yx}} x$ is a negative cycle.*

Proof. Let the path $x \to \ldots \to y$ in N have weight $(\prec, c)$. Now, since x and y are bounded variables in v, we can have either $y - x = d$ or $d - 1 < y - x < d$ for some integer d in G_v^M.

In the first case, we have edges $x \xrightarrow{\leq d} y$ and $y \xrightarrow{\leq -d} x$ in G_v^M, that is $(\prec_{xy}, c_{xy}) = (\leq, d)$ and $(\prec_{yx}, c_{yx}) = (\leq, -d)$. Since by hypothesis $(\prec, c)$ is strictly less than $(\leq, d)$, we have either $c < d$ or $c = d$ and $\prec$ is the strict inequality. Hence $(\prec, c) + (\leq, -d) < (\leq, 0)$ showing that $x \to \ldots \to y \xrightarrow{\prec_{yx} c_{yx}} x$ is a negative cycle.

In the second case, we have edges $x \xrightarrow{<d} y$ and $y \xrightarrow{<-d+1} x$ in G_v^M, that is, $(\prec_{xy}, c_{xy}) = (<, d)$ and $(\prec_{yx}, c_{yx}) = (<, -d)$. Here $c < d$ and again $x \to \ldots \to y \xrightarrow{\prec_{yx} c_{yx}} x$ gives a negative cycle. □

We can now prove Proposition 28.

Proof of Proposition 28. The distance graph $\min(G_v^{LU}, G_{Z'})$ represents the set $\langle v \rangle^{LU} \cap Z'$. By Proposition 23, the intersection is empty iff $\min(G_v^{LU}, G_{Z'})$ has a negative cycle. If there exist variables x, y such that $Z'_{xy} + \langle v \rangle^{LU}_{yx} < (\leq, 0)$, then there is a negative cycle $x \to y \to x$ in $\min(G_v^{LU}, G_{Z'})$ and hence $\langle v \rangle^{LU} \cap Z'$ is empty. This shows the right-to-left direction.

The left-to-right direction is less trivial. Assume that $\langle v \rangle^{LU} \cap Z'$ is empty. Then, there is a negative cycle N in $\min(G_v^{LU}, G_{Z'})$. To prove the proposition, we aim to show the following.

Aim. We show that the negative cycle N of $\min(G_v^{LU}, G_{Z'})$ can be reduced to the form:

$$x \xrightarrow{Z'_{xy}} y \xrightarrow{\langle v \rangle^{LU}_{yx}} x \tag{7}$$

Firstly, since both G_v^{LU} and $G_{Z'}$ are canonical, we can assume without loss of generality that no two consecutive edges in N come from the same graph.

Suppose there are two edges $y_1 \to x_1$ and $y_2 \to x_2$ in N with weights coming from G_v^{LU}:

$$y_1 \xrightarrow{\langle v \rangle^{LU}_{y_1 x_1}} x_1 \to \cdots \to y_2 \xrightarrow{\langle v \rangle^{LU}_{y_2 x_2}} x_2 \to \cdots \to y_1 \tag{8}$$

Since they are part of a negative cycle, their edge weights should be a finite value and by Part 1 of Lemma 27, this means:

$$v_{x_1} \leq U_{x_1} \text{ and } v_{x_2} \leq U_{x_2}$$

1. Suppose $v_{y_1} \leq L_{y_1}$ and $v_{y_2} \leq L_{y_2}$. By Part 2 of Lemma 27, the edge values $y_1 \to x_1$ and $y_2 \to x_2$ are the same as in G_v^M. Consider the edge:

$$y_1 \to x_2 \text{ in } G_v^{LU}$$

Again, from the same lemma, this edge value comes from G_v^M too.

If the value of this edge $y_1 \to x_2$ is smaller than the value of the path $y_1 \to x_1 \to \cdots \to y_2 \to x_2$ in N, then this path can be replaced by the single edge $y_1 \to x_2$ to get a smaller negative cycle in $\min(G_v^{LU}, G_Z)$.

However, if the value of the path $y_1 \to x_1 \to \cdots \to y_2 \to x_2$ is less than the edge value $y_1 \to x_2$, then by Lemma 29:

$$y_1 \to x_1 \to \ldots \to y_2 \to x_2 \to y_1, \text{ where } x_2 \to y_1 \text{ comes from } G_v^M$$

is a negative cycle. The edge $x_2 \to y_1$ might be infinity in G_v^{LU}. But as G_v^M is canonical, we can replace $y_2 \to x_2 \to y_1 \to x_1$ by $y_2 \to x_1$. From Lemma 27, this edge is retained in G_v^{LU} and hence we get a smaller negative cycle.

Therefore in this case, we can eliminate the two edges $y_1 \to x_1$ and $y_2 \to x_2$ to get a smaller negative cycle containing either $y_1 \to x_2$ or $y_2 \to x_1$. If N does not contain a variable z such that $v_z > L_z$, this elimination can be repeatedly applied and N can be reduced to a negative cycle of the form $y \to x \to y$ with $v_y \leq L_y$, $v_x \leq U_x$ and the edge weights $y \to x$ coming from G_v^{LU} and $x \to y$ coming from $G_{Z'}$, exactly as required by (7).

2. Suppose $v_{y_1} > L_{y_1}$. Consider again the two edges $y_1 \to x_1$ and $y_2 \to x_2$ of (8) and now suppose that $v_{y_1} > L_{y_1}$. By Part 3 of Lemma 27, the edge $y_1 \to x_1$ can be replaced by:

$$y_1 \to x_0 \to x_1 \text{ of } G_v^{LU}$$

If there is another variable in N that is greater than its L bound, then the vertex x_0 would occur twice in the negative cycle. From this negative cycle, we can obtain a smaller negative cycle containing only one occurrence of x_0. Hence, without loss of generality, we can assume that x_0 occurs only once in N. In particular, this gives us that:

$$v_{y_2} \leq L_{y_2}$$

Note that the special variable x_0 has $v_{x_0} \leq L_{x_0}$ as its value is always supposed to be 0 and L_{x_0} is defined to be 0. Now consider the two edges:

$$x_0 \to x_1 \text{ and } y_2 \to x_2$$

This corresponds to Case 1 as $v_{x_0} \leq L_{x_0}$ and $v_{y_2} \leq L_{y_2}$. As we have seen, these two edges can be eliminated to give a smaller negative cycle containing either $x_0 \to x_2$ or $y_2 \to x_1$, with the respective value coming from G_v^{LU}.

If it is the latter edge $y_2 \to x_1$, the smaller negative cycle does not contain y_1 and hence all variables are bounded by L. By Case 1, it can be reduced to a cycle as required by the proposition.

Let us now consider the former edge $x_0 \to x_2$. We have the cycle:

$$y_1 \to x_0 \to x_2 \to \ldots \to y_1$$

All the variables other than y_1 in the path $x_0 \to \ldots \to y_1$ are bounded by their L bound. We can therefore assume that all edges in $x_2 \to \ldots \to y_1$ come from $G_{Z'}$, because if not, we can apply the argument of Case 1 to further reduce the cycle. As $G_{Z'}$ and G_v^{LU} are canonical, this cycle reduces to $y_1 \to x_2 \to y_1$ with $y_1 \to x_2$ coming from G_v^{LU} and $x_2 \to y_1$ coming from Z'. This again conforms to the form of the cycle required by (7). □

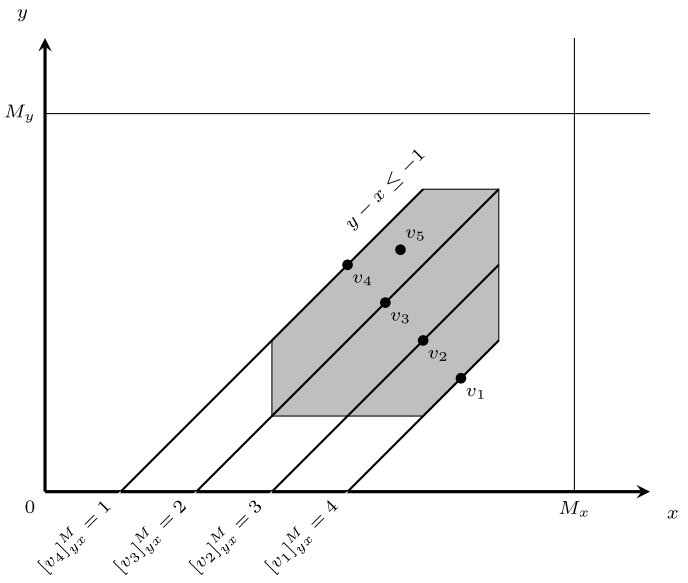


Fig. 9. Value of $[v]^M_{yx}$ decreases as we move left.

5.5. Final steps

Proposition 28 gives a useful characterization of when $\langle v \rangle^{LU} \cap Z'$ is empty. To lift this characterization to $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ and Proposition 21, we need to find the least value of $\langle v \rangle^{LU}_{yx}$ from among the valuations $v \in Z$ and see if this satisfies the condition given in Proposition 28.

For the moment, assume that $L = U$ so that the LU-regions coincide with the classic regions. Consider a zone Z lying within the M bounds, as shown in Fig. 9. The values of $[v]^M_{yx}$ for different valuations in the zone are shown. The value decreases as we move towards the "left boundary". In the figure, since the $y - x$ constraint of Z is given by $y - x \leq -1$, there exists a valuation v_4 on the "boundary" and hence the least value of $[v]^M_{yx}$ among all $v \in Z$ would be given by $[v_4]^M_{yx}$ which is $(\leq, 1)$. If the $y - x$ constraint of Z was $y - x < -1$ with a strict inequality, then the least value of $[v]^M_{yx}$ from $v \in Z$ is no longer $(\leq, 1)$ as there is no valuation attaining this value. In this case, the least value of $[v]^M_{yx}$ would be $(<, 2)$, given by the open region containing v_5 with constraints $y - x < -1$ and $x - y < 2$.

Due to this asymmetry, we need to define the following notion to handle weights in a convenient way. For a weight $(\prec, c)$ we define $-(\prec, c)$ as $(\prec, -c)$ and a *ceiling* function $\lceil \cdot \rceil$ as follows.

Definition 30. For a real c, let $\lceil c \rceil$ denote the smallest integer that is greater than or equal to c. We define the *ceiling* function $\lceil (\prec, c) \rceil$ for a weight $(\prec, c)$ depending on whether $\prec$ equals $\leq$ or $<$, as follows:

$$\lceil (\leq, c) \rceil = \begin{cases} (\leq, c) & \text{if } c \text{ is an integer} \\ (<, \lceil c \rceil) & \text{otherwise} \end{cases}$$

$$\lceil (<, c) \rceil = \begin{cases} (<, c+1) & \text{if } c \text{ is an integer} \\ (<, \lceil c \rceil) & \text{otherwise} \end{cases}$$

The following proposition is one of the two cores for the proof of the main theorem. It gives the least value of $\langle v \rangle^{LU}_{yx}$ from among the valuations v present in a zone Z.

Proposition 31. *Let Z be a non-empty zone, and let x, y be two clocks. The least value of $\langle v \rangle^{LU}_{yx}$ among the valuations $v \in Z$ such that $v_x \leq U_x$ is given by:*

$$\begin{cases} (<, \infty) & \text{if } Z_{x0} < (\leq, -U_x) \\ \max\{\lceil -Z_{xy} \rceil, \lceil -Z_{x0} \rceil - (<, L_y)\} & \text{otherwise} \end{cases}$$

Proof. Let G be the canonical distance graph representing the zone Z. We denote the weight of an edge $i \to j$ in G by $(\prec_{ij}, c_{ij})$. Recall that this means $Z_{ij} = (\prec_{ij}, c_{ij})$.

We are interested in computing the smallest value of the $x-y$ constraint defining an LU-region intersecting Z. Additionally we want to restrict to LU-region in which all its valuations satisfy $x \le U_x$, that is, we need to find:

$$\beta := \min\{\langle v \rangle_{yx}^{LU} \mid v \in Z \text{ and } v_x \le U_x\}$$

Clearly, if $v_x > U_x$ for all valuations $v \in Z$, then β is $(<, \infty)$. When $Z_{x0} < (\le, -U_x)$, it means that all valuations $v \in Z$ satisfy $0 - v_x \prec_{x0} c_{x0}$ and $c_{x0} \le -U_x$. Moreover $\prec_{x0}$ is the strict inequality if $c_{x0} = -U_x$. In consequence, all valuations $v \in Z$ satisfy $v_x > U_x$ when $Z_{x0} < (\le, -U_x)$. Whence $\beta = (<, \infty)$. This corresponds to the first case in the statement of the lemma.

Let us now restrict to the case when $Z_{x0} \ge (\le, -U_x)$. By definition of regions (cf. Definition 24) and Lemma 27, we have for a valuation v:

$$\langle v \rangle_{yx}^{LU} = \begin{cases} \lceil (\le, v_x - v_y) \rceil & \text{if } v_x \le U_x \text{ and } v_y \le L_y \\ (<, -L_y) + \lceil (\le, v_x) \rceil & \text{if } v_x \le U_x \text{ and } v_y > L_y \end{cases} \tag{9}$$

Let G' be the graph in which the edge $0 \to x$ has weight $\min\{(\le, U_x), (\prec_{0x}, c_{0x})\}$ and the rest of the edges are the same as that of G. This graph G' represents the valuations of Z that have $v_x \le U_x$: $[\![G']\!] = \{v \in Z \mid v_x \le U_x\}$. We show that this set is not empty. For this we check that G' does not have negative cycles. Since G does not have negative cycles, every negative cycle in G' should include the newly modified edge $0 \to x$. Note that the shortest path value from x to 0 does not change due to this modified edge. So the only possible negative cycle in G' is $0 \to x \to 0$. But then we are considering the case when $Z_{x0} \ge (\le, -U_x)$, and so $Z_{x0} + (\le, U_x) \ge (\le, 0)$. Hence this cycle cannot be negative either. In consequence all the cycles in G' are positive and $[\![G']\!]$ is not empty.

To find β, it is sufficient to consider only the valuations in $[\![G']\!]$. As seen from Equation (9), among the valuations in $[\![G']\!]$, we need to differentiate between those with $v_y \le L_y$ and the ones with $v_y > L_y$. We proceed as follows. We first compute $\min\{\langle v \rangle_{yx}^{LU} \mid v \in [\![G']\!] \text{ and } v_y \le L_y\}$. Call this β_1. Next, we compute $\min\{[v]_{yx} \mid v \in [\![G']\!] \text{ and } v_y > L_y\}$ and set this as β_2. Our required value β would then equal $\min\{\beta_1, \beta_2\}$.

To compute β_1, consider the following distance graph G'_1 which is obtained from G' by just changing the edge $0 \to y$ to $\min\{(\le, L_y), (\prec_{0y}, c_{0y})\}$ and keeping the remaining edges the same as in G'. The set of valuations $[\![G'_1]\!]$ equals $\{v \in [\![G']\!] \mid v_y \le L_y\}$. If $[\![G'_1]\!] = \emptyset$, we set β_1 to $(<, \infty)$ and proceed to calculate β_2. If not, we see that from Equation (9), for every $v \in [\![G'_1]\!]$, $[v]_{yx}$ is given by $\lceil (\le, v_x - v_y) \rceil$. Let $(\prec_1, w_1)$ be the shortest path from x to y in the graph G'_1. Then, we have for all $v \in [\![G'_1]\!]$, $v_y - v_x \prec_1 w_1$. If $\prec_1$ is $\le$, then the least value of $[v]_{yx}$ would be $(\le, -w_1)$ and if $\prec_1$ is $<$, one can see that the least value of $[v]_{yx}$ is $(<, -w_1 + 1)$. This shows that $\beta_1 = \lceil (\prec_1, -w_1) \rceil$. It now remains to calculate $(\prec_1, w_1)$.

Recall that G'_1 has the same edges as in G except possibly different edges $0 \to x$ and $0 \to y$. If the shortest path from x to y has changed in G'_1, then clearly it should be due to one of the above two edges. However note that the edge $0 \to x$ cannot belong to the shortest path from x to y since it would contain a cycle $x \to \ldots 0 \to x \to \ldots y$ that can be removed to give shorter path. Therefore, only the edge $0 \to y$ can potentially yield a shorter path: $x \to \ldots \to 0 \to y$. However, the shortest path from x to 0 in G'_1 cannot change due to the added edges since that would form a cycle with 0 and we know that all cycles in G'_1 are positive. Therefore the shortest path from x to 0 is the direct edge $x \to 0$, and the shortest path from x to y is the minimum of the direct edge $x \to y$ and the path $x \to 0 \to y$. We get: $(\prec_1, w_1) = \min\{(\prec_{xy}, c_{xy}), (\prec_{x0}, c_{x0}) + (\le, L_y)\}$ which equals $\min\{Z_{xy}, Z_{x0} + (\le, L_y)\}$. Finally, from the argument in the above two paragraphs, we get:

$$\beta_1 = \begin{cases} (<, \infty) & \text{if } [\![G'_1]\!] = \emptyset \\ \lceil -Z_{xy} \rceil & \text{if } [\![G'_1]\!] \ne \emptyset \text{ and } Z_{xy} \le Z_{x0} + (\le, L_y) \\ \lceil -Z_{x0} \rceil + (\le, -L_y) & \text{if } [\![G'_1]\!] \ne \emptyset \text{ and } Z_{xy} > Z_{x0} + (\le, L_y) \end{cases} \tag{10}$$

We now proceed to compute $\beta_2 = \min\{[v]_{yx} \mid v \in [\![G']\!] \text{ and } v_y > L_y\}$. Let G'_2 be the graph which is obtained from G' by modifying the edge $y \to 0$ to $\min\{Z_{y0}, (<, -L_y)\}$ and keeping the rest of the edges the same as in G'. Clearly $[\![G'_2]\!] = \min\{v \in [\![G']\!] \mid v_y > L_y\}$.

Again, if $[\![G'_2]\!]$ is empty, we set β_2 to $(<, \infty)$. Otherwise, from Equation (9), for each valuation $v \in [\![G'_2]\!]$, the value of $[v]_{yx}$ is given by $(<, \lceil v_x \rceil - L_y)$. For the minimum value, we need the least value of v_x from $v \in [\![G'_2]\!]$. Let $(\prec_2, w_2)$ be the shortest path from x to 0 in G'_2. Then, since $-v_x \prec_2 w_2$, the least value of $\lceil v_x \rceil$ would be $-w_2$ if $\prec_2$ is $\le$ and equal to $\lceil -w_2 \rceil$ if $\prec_2 =<$ and β_2 would respectively be $(<, -w_2 - L_y)$ or $(<, -w_2 + 1 - L_y)$. It now remains to calculate $(\prec_2, w_2)$.

Recall that G'_2 is G with $0 \to x$ and $y \to 0$ modified. The shortest path from x to 0 cannot include the edge $0 \to x$ since it would need to contain a cycle, for the same reasons as in the β_1 case. So we get $(\prec_2, w_2) = \min\{Z_{x0}, Z_{xy} + (<, -L_y)\}$. If $Z_{x0} \le Z_{xy} + (<, -L_y)$, then we take $(\prec_2, w_2)$ as Z_{x0}, otherwise we take it to be $Z_{xy} + (<, -L_y)$. So, we get β_2 as the following:

$$\beta_2 = \begin{cases} (<, \infty) & \text{if } [\![G'_2]\!] = \emptyset \\ -Z_{xy} + (<, 1) & \text{if } [\![G'_2]\!] \ne \emptyset \text{ and } Z_{x0} \ge Z_{xy} + (<, -L_y) \\ \lceil -Z_{x0} \rceil + (<, -L_y) & \text{if } [\![G'_2]\!] \ne \emptyset \text{ and } Z_{x0} < Z_{xy} + (<, -L_y) \end{cases} \tag{11}$$

However, we would like to write β_2 in terms of the cases used for β_1 in Equation (10) so that we can write β, which equals $\min\{\beta_1, \beta_2\}$, conveniently.

Let ψ_1 be the inequation: $Z_{xy} \leq Z_{x0} + (\leq, L_y)$. From Equation (10), note that β_1 has been classified according to ψ_1 and $\neg\psi_1$ when $[\![G'_1]\!]$ is not empty. Similarly, let ψ_2 be the inequation: $Z_{x0} \geq Z_{xy} + (<, -L_y)$. From Equation (11) we see that β_2 has been classified in terms of ψ_2 and $\neg\psi_2$ when $[\![G'_2]\!]$ is not empty. Notice the subtle difference between ψ_1 and ψ_2 in the weight component involving L_y: in the former the inequality associated with L_y is $\leq$ and in the latter it is $<$. This necessitates a bit more of analysis before we can write β_2 in terms of ψ_1 and $\neg\psi_1$.

Suppose ψ_1 is true. So we have $(\prec_{xy}, c_{xy}) \leq (\prec_{x0}, c_{x0} + L_y)$. This implies: $c_{xy} \leq c_{x0} + L_y$. Therefore, $c_{x0} \geq c_{xy} - L_y$. When $c_{x0} > c_{xy} - L_y$, ψ_2 is clearly true. For the case when $c_{x0} = c_{xy} - L_y$, note that in ψ_2 the right hand side is always of the form $(<, c_{xy} - L_y)$, irrespective of the inequality in Z_{xy} and so yet again, ψ_2 is true. We have thus shown that ψ_1 implies ψ_2.

Suppose $\neg\psi_1$ is true. We have $(\prec_{xy}, c_{xy}) > (\prec_{x0}, c_{x0} + L_y)$. If $c_{xy} > c_{x0} + L_y$, then clearly $c_{x0} < c_{xy} - L_y$ implying that $\neg\psi_2$ holds. If $c_{xy} = c_{x0} + L_y$, then we need to have $\prec_{xy}$ equal to $\leq$ and $\prec_{x0}$ equal to $<$. Although $\neg\psi_2$ does not hold now, we can safely take β_2 to be $\lceil -Z_{x0} \rceil + (<, -L_y)$ as its value is in fact equal to $-Z_{xy} + (<, 1)$ in this case. Summarizing the above two paragraphs, we can rewrite β_2 as follows:

$$\beta_2 = \begin{cases} (<, \infty) & \text{if } [\![G'_2]\!] = \emptyset \\ -Z_{xy} + (<, 1) & \text{if } [\![G'_2]\!] \neq \emptyset \text{ and } Z_{xy} \leq Z_{x0} + (\leq, L_y) \\ \lceil -Z_{x0} \rceil + (<, -L_y) & \text{if } [\![G'_2]\!] \neq \emptyset \text{ and } Z_{xy} > Z_{x0} + (\leq, L_y) \end{cases} \tag{12}$$

We are now in a position to determine β as $\min\{\beta_1, \beta_2\}$. Recall that we are in the case where $Z_{x0} \leq (\leq, -U_x)$ and we have established that $[\![G']\!]$ is non-empty. Now since $[\![G']\!] = [\![G'_1]\!] \cup [\![G'_2]\!]$ by construction, both of them cannot be simultaneously empty. Hence from Equations (10) and (12), we get β, the $\min\{\beta_1, \beta_2\}$ as:

$$\beta_= \begin{cases} \lceil -Z_{xy} \rceil & \text{if } Z_{xy} \leq Z_{x0} + (\leq, L_y) \\ \lceil -Z_{x0} \rceil + (<, -L_y) & \text{if } Z_{xy} > Z_{x0} + (\leq, L_y) \end{cases} \tag{13}$$

There remains one last reasoning. To prove the lemma, we need to show that $\beta = \max\{\lceil -Z_{xy} \rceil, \lceil -Z_{x0} \rceil + (<, -L_y)\}$. For this it is enough to show the following two implications:

$$Z_{xy} \leq Z_{x0} + (\leq, L_y) \Rightarrow \lceil -Z_{xy} \rceil \geq \lceil -Z_{x0} \rceil + (<, -L_y)$$
$$Z_{xy} > Z_{x0} + (\leq, L_y) \Rightarrow \lceil -Z_{xy} \rceil \leq \lceil -Z_{x0} \rceil + (<, -L_y)$$

We prove only the first implication. The second follows in a similar fashion. Let us consider the notation $(\prec_{xy}, c_{xy})$ and $(\prec_{x0}, c_{x0})$ for Z_{xy} and Z_{x0} respectively. So we have:

$$\begin{aligned} & (\prec_{xy}, c_{xy}) \leq (\prec_{x0}, c_{x0}) + (\leq, L_y) \\ \Rightarrow\ & (\prec_{xy}, c_{xy}) \leq (\prec_{x0}, c_{x0} + L_y) \end{aligned}$$

If the constant $c_{xy} < c_{x0} + L_y$, then $-c_{xy} > -c_{x0} - L_y$ and we clearly get that $\lceil -Z_{xy} \rceil \geq \lceil -Z_{x0} \rceil + (<, -L_y)$. If the constant $c_{xy} = c_{x0} + L_y$ and if $\prec_{x0}$ is $\leq$, then the required inequation is trivially true; if $\prec_{x0}$ is $<$, it implies that $\prec_{xy}$ is $<$ too and clearly $\lceil (<, -c_{xy}) \rceil$ equals $\lceil (<, -c_{x0}) \rceil + (<, -L_y)$. □

We have a simple method that tells us when an LU-region $\langle v \rangle^{LU}$ does not intersect a zone Z' (Proposition 28). We have also characterized the potential valuation v from Z that could satisfy the non-intersection condition with Z' (Proposition 31). This gives the necessary tools to solve inclusion $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$. The following theorem presents the efficient inclusion test.

Theorem 32. *Let Z, Z' be non-empty zones. Then, $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ iff there exist two variables x, y such that:*

$$Z_{x0} \geq (\leq, -U_x) \text{ and } Z'_{xy} < Z_{xy} \text{ and } Z'_{xy} + (<, -Ly) < Z_{x0}$$

Proof. From Proposition 21, we know that $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ iff there exists a valuation $v \in Z$ such that $\langle v \rangle^{LU}$ does not intersect Z'.

From Proposition 28, we know that $\langle v \rangle^{LU} \cap Z'$ is empty iff there exists a variable x such that $v_x \leq U_x$ and a variable y such that:

$$Z'_{xy} + \langle v \rangle^{LU}_{yx} < (\leq, 0) \tag{14}$$

This is possible for variables x, y iff the least value of $\langle v \rangle^{LU}_{yx}$ from among the valuations in Z satisfies the inequation (14) with Z'_{xy}.

This is where we use Proposition 31. According to this proposition, for (14) to be true for some valuation $v \in Z$, we would need $Z_{x0} \geq (\leq, -U_x)$ and:

$$Z'_{xy} + \lceil -Z_{xy} \rceil < (\leq, 0) \text{ and } Z'_{xy} + \lceil -Z_{x0} \rceil - (<, L_y) < (\leq, 0) \tag{15}$$

Consider the first inequality: $Z'_{xy} + \lceil -Z_{xy} \rceil < (\leq, 0)$. Let Z_{xy} be $(\prec_{xy}, c_{xy})$. If $\prec_{xy}$ is the weak inequality $\leq$, then $\lceil -Z_{xy} \rceil$ is $(\leq, -c_{xy})$ and hence the condition becomes: $Z'_{xy} + (\leq, -c_{xy}) < (\leq, 0)$. This is equivalent to saying $Z'_{xy} < (\leq, c_{xy})$, that is, $Z'_{xy} < Z_{xy}$. Now, if $\prec_{xy}$ is the strict inequality $<$, then $\lceil -Z_{xy} \rceil$ becomes $(<, -c_{xy} + 1)$ and hence the condition becomes: $Z'_{xy} + (<, -c_{xy} + 1) < (\leq, 0)$. This is equivalent to saying $Z'_{xy} < (<, c_{xy})$. In both cases, the first inequality of Equation (15) becomes $Z'_{xy} < Z_{xy}$.

By a similar reasoning, the second inequality of Equation (15) can be seen to correspond to $Z'_{xy} + (<, -L_y) < Z_{x0}$. This proves the theorem. □

The $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ test involves a comparison of corresponding edges in the distance graphs G_Z and $G_{Z'}$, so it takes in the worst case a $\mathcal{O}(|X|^2)$ number of steps. Notice that in fact the test requires only two tests for every pair of clocks.

6. Conclusions

Traditional methods for timed automata reachability *store abstractions* of zones for termination. Therefore, only convex abstractions have been used in implementations. We have proposed to store zones and use abstractions indirectly by means of inclusion tests $Z \subseteq \mathfrak{a}(Z')$. This allows us to use non-convex abstractions while still working with zones. The coarser the abstraction $\mathfrak{a}$, the higher is the possibility of inclusion, and hence smaller would be the reachability tree that is explored. For this construction to work efficiently, one also needs an efficient inclusion test.

In this paper, we have given a complete solution to using the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction for the reachability algorithm. Firstly, we have given an $\mathcal{O}(|X|^2)$ test for $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ inclusion. This is the cornerstone of our approach since this test is used in the inner loop of the algorithm. Our test has the same complexity as $Z \subseteq Z'$ test used in the traditional algorithm. We have also shown that the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is the coarsest abstraction that is sound and complete with respect to reachability for all automata with the same LU-bounds. The result showing that $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is the coarsest possible is quite unexpected. It works thanks to the observation that when doing forward exploration it is enough to consider only time-elapsed zones. This result explains why after $Extra^+_{LU}$ from [4] there have been no new abstraction operators [9]. Indeed it is not that easy to find a better zone inside $\mathfrak{a}_{\preccurlyeq LU}$ abstraction than that given by $Extra^+_{LU}$ abstraction.

The maximality result for $\mathfrak{a}_{\preccurlyeq LU}$ shows that to improve reachability testing even further we will need to look at new structural properties of timed automata, or to consider more refined algorithms than forward exploration. A work in this direction is [17].

References

[1] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, David L. Dill, Howard Wong-Toi, Minimization of timed transition systems, in: Rance Cleaveland (Ed.), CONCUR '92, Third International Conference on Concurrency Theory, Proceedings, Stony Brook, NY, USA, August 24–27, 1992, in: Lect. Notes Comput. Sci., vol. 630, Springer, 1992, pp. 340–354.

[2] Rajeev Alur, David L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (2) (1994) 183–235.

[3] Gerd Behrmann, Patricia Bouyer, Emmanuel Fleury, Kim Guldstrand Larsen, Static guard analysis in timed automata verification, in: Hubert Garavel, John Hatcliff (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, 9th International Conference, Proceedings, TACAS 2003, Warsaw, Poland, April 7–11, 2003, in: Lect. Notes Comput. Sci., vol. 2619, Springer, 2003, pp. 254–277.

[4] Gerd Behrmann, Patricia Bouyer, Kim Guldstrand Larsen, Radek Pelánek, Lower and upper bounds in zone-based abstractions of timed automata, Int. J. Softw. Tools Technol. Transf. 8 (3) (2006) 204–215.

[5] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Hakansson, Paul Pettersson, Wang Yi, Martijn Hendriks, UPPAAL 4.0, in: Third International Conference on the Quantitative Evaluation of Systems, (QEST 2006), 11–14 September 2006, Riverside, California, USA, IEEE Computer Society, 2006, pp. 125–126.

[6] Johan Bengtsson, Wang Yi, Timed automata: semantics, algorithms and tools, in: Jörg Desel, Wolfgang Reisig, Grzegorz Rozenberg (Eds.), Lectures on Concurrency and Petri Nets, Advances in Petri Nets, in: Lect. Notes Comput. Sci., vol. 3098, Springer, 2003, pp. 87–124.

[7] Bernard Berthomieu, Miguel Menasche, An enumerative approach for analyzing time petri nets, in: IFIP Congress, 1983, pp. 41–46.

[8] Patricia Bouyer, Forward analysis of updatable timed automata, Form. Methods Syst. Des. 24 (3) (2004) 281–320.

[9] Patricia Bouyer, From qualitative to quantitative analysis of timed systems, Mémoire d'habilitation, Université Paris 7, Paris, France, 2009.

[10] Patricia Bouyer, Maximilien Colange, Nicolas Markey, Symbolic optimal reachability in weighted timed automata, in: Swarat Chaudhuri, Azadeh Farzan (Eds.), Computer Aided Verification – 28th International Conference, Proceedings, Part I, CAV 2016, Toronto, ON, Canada, July 17–23, 2016, in: Lect. Notes Comput. Sci., vol. 9779, Springer, 2016, pp. 513–530.

[11] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, Quantitative analysis of real-time systems using priced timed automata, Commun. ACM 54 (9) (2011) 78–87.

[12] Conrado Daws, Stavros Tripakis, Model checking of real-time reachability properties using abstractions, in: Bernhard Steffen (Ed.), Tools and Algorithms for Construction and Analysis of Systems, 4th International Conference, Proceedings, TACAS '98, Lisbon, Portugal, March 28–April 4, 1998, in: Lect. Notes Comput. Sci., vol. 1384, Springer, 1998, pp. 313–329.

[13] David L. Dill, Timing assumptions and verification of finite-state concurrent systems, in: Joseph Sifakis (Ed.), Automatic Verification Methods for Finite State Systems, International Workshop, Proceedings, Grenoble, France, June 12–14, 1989, in: Lect. Notes Comput. Sci., vol. 407, Springer, 1989, pp. 197–212.

[14] H. Gregersen, H.E. Jensen, Formal design of reliable real time systems, Master's thesis, Department of Mathematics and computer Science, Aalborg University, 1995.

[15] Frédéric Herbreteau, Dileep Kini, B. Srivathsan, Igor Walukiewicz, Using non-convex approximations for efficient analysis of timed automata, in: Supratik Chakraborty, Amit Kumar (Eds.), IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12–14, 2011, Mumbai, India, in: LIPIcs, vol. 13, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2011, pp. 78–89.

[16] Frédéric Herbreteau, B. Srivathsan, Igor Walukiewicz, Better abstractions for timed automata, in: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012, IEEE Computer Society, 2012, pp. 375–384.

[17] Frédéric Herbreteau, B. Srivathsan, Igor Walukiewicz, Lazy abstractions for timed automata, in: Natasha Sharygina, Helmut Veith (Eds.), Computer Aided Verification – 25th International Conference, Proceedings, CAV 2013, Saint Petersburg, Russia, July 13–19, 2013, in: Lect. Notes Comput. Sci., vol. 8044, Springer, 2013, pp. 990–1005.
[18] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, Jeremy Sproston, Automatic verification of real-time systems with discrete probability distributions, Theor. Comput. Sci. 282 (1) (2002) 101–150.
[19] François Laroussinie, Philippe Schnoebelen, The state explosion problem from trace to bisimulation equivalence, in: Jerzy Tiuryn (Ed.), Foundations of Software Science and Computation Structures, Third International Conference, Proceedings, FOSSACS 2000, Berlin, Germany, March 25–April 2, 2000, in: Lect. Notes Comput. Sci., vol. 1784, Springer, 2000, pp. 192–207.
[20] Georges Morbé, Florian Pigorsch, Christoph Scholl, Fully symbolic model checking for timed automata, in: Ganesh Gopalakrishnan, Shaz Qadeer (Eds.), Computer Aided Verification – 23rd International Conference, Proceedings, CAV 2011, Snowbird, UT, USA, July 14–20, 2011, in: Lect. Notes Comput. Sci., vol. 6806, Springer, 2011, pp. 616–632.
[21] Robert E. Shostak, Deciding linear inequalities by computing loop residues, J. ACM 28 (4) (1981) 769–779.
[22] Serdar Tasiran, Rajeev Alur, Robert P. Kurshan, Robert K. Brayton, Verifying abstractions of timed systems, in: Ugo Montanari, Vladimiro Sassone (Eds.), CONCUR '96, Concurrency Theory, 7th International Conference, Proceedings, Pisa, Italy, August 26–29, 1996, in: Lect. Notes Comput. Sci., vol. 1119, Springer, 1996, pp. 546–562.
[23] Stavros Tripakis, Sergio Yovine, Analysis of timed systems using time-abstracting bisimulations, Form. Methods Syst. Des. 18 (1) (2001) 25–68.
[24] Farn Wang, Efficient verification of timed automata with BDD-like data structures, Int. J. Softw. Tools Technol. Transf. 6 (1) (2004) 77–97.
[25] Mihalis Yannakakis, David Lee, An efficient algorithm for minimizing real-time transition systems, Form. Methods Syst. Des. 11 (2) (1997) 113–136.
[26] Jianhua Zhao, Xuandong Li, Guoliang Zheng, A quadratic-time DBM-based successor algorithm for checking timed automata, Inf. Process. Lett. 96 (3) (2005) 101–105.