Better abstractions for timed automata

F. Herbreteau, B. Srivathsan and I. Walukiewicz
Univ. Bordeaux, CNRS, LaBRI, UMR 5800
F-33400 Talence, France
Email: {fh, sri, igw}@labri.fr

***Abstract*—We consider 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 parametrized by the maximal lower and upper bounds (LU-bounds) occurring in the guards of the automaton.**

We consider the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction defined by Behrmann et al. Since this abstraction can potentially yield non-convex sets, it has not been used in implementations. We prove that $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is the biggest abstraction with respect to LU-bounds that is sound and complete for reachability. We also provide an efficient technique to use the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction to solve the reachability problem.

I. Introduction

Timed automata are finite automata extended with clocks whose values can be compared with constants and set to 0. 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 that 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 show a rather unexpected fact that $\mathfrak{a}_{\preccurlyeq LU}$ approximation defined by Behrmann et al. [3] is the biggest sound and complete approximation. We also present a method of constructing abstracted zone graph using $\mathfrak{a}_{\preccurlyeq LU}$ approximation. Even though this approximation 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 [3], [11], [14], [17]. Apart from being interesting by itself, the advances on this problem may allow to give new methods for verification of more complicated models, like priced timed-automata [7], or probabilistic timed automata [6], [8], [12].

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 the automaton. The first solution has been based on regions: equivalence classes of clock valuations [1]. 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) [10]. 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, 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. Bigger approximation makes the abstracted zone graph smaller. So potentially it gives a more efficient algorithm.

An important observation made in [3] is that if reachability is concerned then we can consider simulation instead of bisimulation. Indeed, it is safe to add configurations that are 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 [13]. A pragmatic approach is to abstract some part of the structure of the automaton and define simulation based on this information. The most relevant information are the bounds with which clocks are compared in guards of the automaton. Since lower and upper bounds are considered separately, they are called LU-bounds. In [3] 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 biggest 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 bigger (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

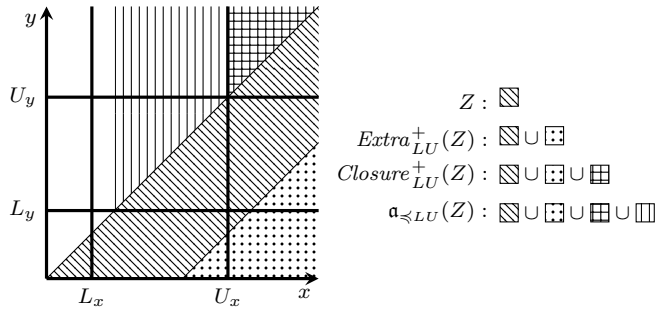

Fig. 1. A comparison of abstraction operators for zones.

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.

A. 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 [9]. In recent years, coarser abstractions have been introduced to improve efficiency of the analysis [3]. An approximation method based on LU-bounds, called $Extra^+_{LU}$, is used in the current implementation of UPPAAL [4]. In [11] 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 approximation. In comparison, $\mathfrak{a}_{\preccurlyeq_{LU}}$ approximation has a well-motivated semantics, it is also region closed, and the resulting inclusion test is even simpler than that of $Closure^+_{LU}$. A comparison of these abstractions is depicted in Fig. 1.

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 [14] 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 [8], [16]. For reachability properties, this approach is not a priori competitive with respect to forward exploration as it requires to construct the whole state space of the automaton. It is more adapted to checking branching time properties.

B. Organization of the paper

In the next section, we present preliminary definitions, introduce the notion of sound and complete abstractions and explain how these abstractions could be used to solve the reachability problem. In Section III, we introduce the concept of LU-bounds putting limits on the constants that can be used in guards of automata. In the same section, we propose an abstraction abs_{LU} and prove that it is the coarsest sound and complete abstraction for all automata with given LU-bounds. Subsequently, in Section IV we show that the $\mathfrak{a}_{\preccurlyeq_{LU}}$ abstraction actually coincides with this biggest abstraction abs_{LU}. Section V then presents the efficient inclusion test for $\mathfrak{a}_{\preccurlyeq_{LU}}$ abstraction which allows for its use in implementations.

II. 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.

A. 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 $\mathbb{R}^X_{\geq 0}$ the set of clock valuations over X, and $\mathbf{0}$ 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: $(q, v) \to^\alpha (q, v')$ for some transition $(q, g, R, q') \in T$ such that $v \vDash g$ and $v' = [R]v$.

In this paper we are interested in the *reachability problem* that asks if there exists a configuration (q, v) with accepting state $q \in Acc$ that is reachable from $(q_0, \mathbf{0})$ by any 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.

B. 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. The transitions are then defined by:

$$(q, W) \Rightarrow^{\alpha} (q', W')$$

where $W' = \{v' : \exists v \in W.\ v \rightarrow^{\alpha} v'\}$, and

$$(q, W) \Rightarrow^{\tau} (q', W')$$

where $W' = \{v' : \exists v \in W.\ \exists \delta \in \mathbb{R}_{\geq 0}\ v \rightarrow^{\delta} v'\}$.

So $\Rightarrow^{\alpha}$ transition is the existential lifting of $\rightarrow^{\alpha}$ transition to sets, similarly for $\Rightarrow^{\tau}$ transition but it moreover permits any delay. We will write $\Rightarrow$ without superscript to denote the union of the two relations.

An *abstraction operation* [2] is a convenient way of expressing a grouping of valuations. It is a function $\mathfrak{a} : \mathcal{P}(\mathbb{R}_{\geq 0}^{|X|}) \rightarrow \mathcal{P}(\mathbb{R}_{\geq 0}^{|X|})$ such that $W \subseteq \mathfrak{a}(W)$ and $\mathfrak{a}(\mathfrak{a}(W)) = \mathfrak{a}(W)$. An abstraction operator defines an abstract semantics:

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

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

If $\mathfrak{a}$ has a finite range then this abstraction is finite. Analogously we define $\Rightarrow_{\mathfrak{a}}^{\alpha}$ and $\Rightarrow_{\mathfrak{a}}^{\tau}$. We write $\Rightarrow^{*}$ for the transitive closure of $\Rightarrow$, similarly for $\rightarrow^{*}$.

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 (where $\rightarrow$ denotes the union of $\rightarrow^{\alpha}$ and $\rightarrow^{\delta}$):

Soundness: if $(q_0, \{v_0\}) \Rightarrow_{\mathfrak{a}}^{*} (q, W)$ then there is $v \in W$ such that $(q_0, v_0) \rightarrow^{*} (q, v)$.

Completeness: if $(q_0, v_0) \rightarrow^{*} (q, v)$ then there is W such that $v \in W$ and $(q_0, \{v_0\}) \Rightarrow_{\mathfrak{a}}^{*} (q, W)$.

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 difference 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) [10]. This suggests that one should consider abstractions that give zones. This is an important restriction: zones are convex, and abstractions based on regions are usually not convex.

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 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 on this representation. We also need to know when two 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.

The first condition is quite easy to satisfy. Every abstraction relation coming from time-abstract simulation [15] is transition compatible (cf. Appendix A) This paper is essentially about how to satisfy the second condition and get as good abstraction as possible at the same time.

III. The biggest LU abstraction

We introduce the concept of LU bounds: maximal constants used in lower and upper bounds. These can be used to define simulations and abstractions independently of automata. The goal of this section is to come up with the coarsest possible abstraction if the only a priori knowledge we have about an automaton is LU-information. To this regard, we propose an abstraction operation abs_{LU} and prove that it is the biggest such (Theorem 7).

One way to obtain abstractions is to group together valuations that are not distinguishable by an automaton, i.e. consider a bisimulation relation. If we are after reachability properties then one can even consider (time abstract) simulation relation [15]. For a given automaton it can be computed if two configurations are in a simulation relation. It should be noted though that computing the coarsest simulation relation is EXPTIME-hard [13]. 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 coarsest 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 [2]. 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 1 (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.

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 2 (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 $v_1 = [R](v+\delta)$.

One can check that $\sqsubseteq_{LU}$ is the biggest relation that is a time-abstract simulation for all automata with given LU bounds.

Simulation relation permits to define an abstraction operator. Basically, to the abstraction of Z we can add all valuations that can be simulated by a valuation in Z. This way we guarantee soundness of the abstraction as the added valuations cannot do more than the valuations already present in Z.

Definition 3 (Abstraction based on LU-simulation) For a zone Z we define: $abs_{LU}(Z) = \{v : \exists v' \in Z.\ v \sqsubseteq_{LU} v'\}$.

The definition of LU-simulation is sometimes difficult to work with since it talks about infinite sequences of actions. In the next lemma we 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 6). For this discussion let us fix some L and U functions. We start with a preparatory definition.

Definition 4 For a valuation v we define its *LU-region*, denoted $r_{LU}(v)$, to be the set of valuations v' such that:

- v' satisfies the same LU-guards as v.
- 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 $\{v(x)\} < \{v(y)\}$ then $\{v'(x)\} < \{v'(y)\}$.
  - if $\{v(x)\} = \{v(y)\}$ then $\{v'(x)\} \leq \{v'(y)\}$.

The first condition roughly says that the integer parts of the two valuations are the same. Observe that we cannot require that they are exactly the same for values between L and U bounds. The second part says that the order of fractional parts should be the same, but once again we restrict only to inequalities that we can express within our LU-bounds. Notice that if $L_x = U_x = M$, for some M and all clocks x, then we get just the usual definition of regions with respect to M.

Lemma 5 For every two valuations v and v':

$$v \sqsubseteq_{LU} v' \quad \text{iff} \quad \text{there is } \delta' \in \mathbb{R}_{\geq 0} \text{ with } v' + \delta' \in r_{LU}(v).$$

Proof: First let us take v and define a sequence of abstract transitions that reflect the definition of $r_{LU}(v)$. We define some guards. Let g_{int} be the conjunction of all LU guards that v satisfies. For every pair of clocks x, y such that $v(x) \leq U_x$, $v(y) \leq L_y$ we consider guards:

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

Finally for every y with $v(y) < L_y$ we put $g_y = \bigwedge\{g_{xy} : v(x) \leq U_x\}$. Note that the guards that are defined are consistent with the LU bounds.

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: $\{v(y_1)\} \leq \cdots \leq \{v(y_k)\}$. Let $seq(v)$ be the sequence of transitions $\xrightarrow{g_{int}} \xrightarrow{g_{y_k}} \ \ldots \ \xrightarrow{g_{y_1}}$; since the resets are empty we have not represented them in the labels of the sequence.

The sequence $seq(v)$ can be performed from v:

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

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. It is also easy to check that if it is possible to do this sequence of transitions from some valuation v' then there is $\delta' \in \mathbb{R}_{\geq 0}$ such that $v' + \delta' \in r_{LU}(v)$. This shows left to right implication.

For the right to left implication we show that the relation $S = \{(v, v') : v' \in r_{LU}(v)\}$ 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 argument is very similar to the one for standard regions. ∎

The sequence $seq(v)$ introduced in the above proof will be quite useful. In particular the proof shows the following.

Corollary 6 For two valuations v, v':

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

We are now ready to prove the first main result of this section showing that $abs_{LU}(Z)$ is the biggest sound and complete simulation that uses solely LU information

Theorem 7 *The abs_{LU} abstraction is the biggest abstraction that is sound and complete for all LU-automata.*

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

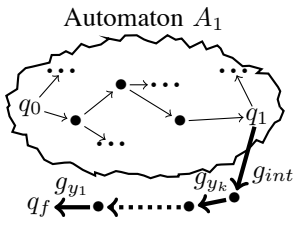

Fig. 2. Adding the sequence $seq(v)$ to A_1.

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 6. 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 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. ■

IV. The $\mathfrak{a}_{\preccurlyeq_{LU}}$ abstraction

Since abs_{LU} is the biggest abstraction, we would like to use it in a reachability algorithm. The definition of abs_{LU}, or even the characterization referring to r_{LU}, are still too complicated to work with. The $\mathfrak{a}_{\preccurlyeq_{LU}}$ abstraction proposed by Behrmann et al. in [3] has much simpler definition. It turns out that in the context of reachability analysis the two abstractions coincide (Theorem 12).

We begin by recalling the definition of an LU-preorder defined in [3]. We use a different but equivalent formulation.

Definition 8 (LU-preorder [3]) Let $L, U : X \to \mathbb{N}$ 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 $v(x) > U_x$.

Definition 9 (LU-abstraction [3]) For L, U as above. For a set of valuations W we define:

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

A. 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 $r_{LU}(v)$ and valuations that simulate v with respect to $\preccurlyeq_{LU}$. The following lemma says that if $v' \in r_{LU}(v)$ 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.

Definition 10 A valuation v_1 is said to be in the *neighbourhood* of v, written $v_1 \in \mathrm{nbd}(v)$ if for all clocks x, y:

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

Notice that the neighbourhood of v is the same as the region of v with respect to the classical region definition [1] with maximal bound being ∞.

Lemma 11 (Adjustment) Let v be a valuation and let $v' \in r_{LU}(v)$. Then, there exists a $v'_1 \in \mathrm{nbd}(v')$ such that $v \preccurlyeq_{LU} v'_1$.

Proof: Let $v' \in r_{LU}(v)$. The goal is to construct a valuation $v'_1 \in \mathrm{nbd}(v')$ that satisfies $v \preccurlyeq_{LU} v'_1$. To be in the neighbourhood, 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 8 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 neighbourhood, 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 $v' \in r_{LU}(v)$ 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 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 $v' \in r_{LU}(v)$, which we restate here:

$$\begin{aligned} \forall x, y \in X_f \text{ such that } v(x) \leq U_x \text{ and } v(y) \leq L_y \quad (1)\\ \{v(x)\} < \{v(y)\} \Rightarrow \{v'(x)\} < \{v'(y)\} \\ \{v(x)\} = \{v(y)\} \Rightarrow \{v'(x)\} \leq \{v'(y)\} \end{aligned}$$

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 k^{th} 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

$$v(y) \leq L_y \Rightarrow \{v(y)\} < \sigma_{n-k+1} \quad (2)$$

Let us now perform the $k+1^{th}$ step and show that the 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:

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

We claim that $l \leq u$. 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 $v(x) \leq U_x$ and $v(y) \leq L_y$ such that $\{v(x)\} < \{v(y)\}$. From Equation 1, this would imply that $\{v'(x)\} < \{v'(y)\}$. But this leads to a contraction 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 $\{v(y)\} \geq \sigma_{n-k}$, it would mean that $\{v(y)\} \geq u$ and from Equation 1 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 1, 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 are now ready to prove the second main result of this section. We write $\overrightarrow{Z}$ for the closure of Z under time-successors: $\overrightarrow{Z} = \{v + \delta : v \in Z, \delta \in \mathbb{R}_{\geq 0}\}$. We say that a zone Z is *time-elapsed* if $Z = \overrightarrow{Z}$.

Theorem 12 *If Z is time-elapsed then*

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

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 Lemma 5, this implies there exists a δ' such that $v' + \delta' \in r_{LU}(v)$. As Z is time-elapsed, we get $v' + \delta' \in Z$. Moreover, from Lemma 11, we know that there is a valuation $v'_1 \in \text{nbd}(v' + \delta')$ such that $v \preccurlyeq_{LU} v'_1$. Every valuation in the neighbourhood 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 $v \in \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$. ■

B. 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. 3). Observe that the algorithm should not take two consecutive action transitions. Indeed, instead of doing $(q_1, Z_1) \Rightarrow^{\alpha} (q_2, Z_2) \Rightarrow^{\alpha} (q_3, Z_3)$, it is preferable to do $(q_1, Z_1) \Rightarrow^{\alpha} (q_2, Z_2) \Rightarrow^{\tau} (q_2, \overrightarrow{Z_2}) \Rightarrow^{\alpha} (q_3, Z'_3)$ since $Z_2 \subseteq \overrightarrow{Z_2}$ and $\Rightarrow$ is monotone with respect to zone inclusion. For this reason the algorithm can start in time-elapsed initial node $(q_0, \overrightarrow{Z_0})$, and for every node (q, Z) consider its successors $(q, Z) \Rightarrow^{\alpha} \Rightarrow^{\tau} (q', Z')$ disregarding the intermediate node. So all nodes visited by the algorithm have time-elapsed zones.

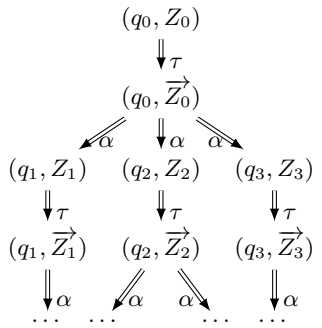

Fig. 3. A reachability tree in a zone graph alternating τ and α edges.

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 fintary abstraction [3] (see also Proposition 14).

Since the reachability algorithm refers to only time-elapsed zones, Theorems 7 and 12 show that $\mathfrak{a}_{\preccurlyeq_{LU}}$ is the biggest sound and complete abstraction provided the only thing we know about the structure of the automaton are its L and U bounds. Recall that bigger abstractions make abstract graph smaller, so the exploration algorithm can finish faster.

The refined forward exploration algorithms calculate LU information for each state of the automaton separately [2], or even on-the-fly during exploration [11]. The maximality argument in favour 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.

V. 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 $Z \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$ (Theorem 24). Since a lot of tests of this kind need to be performed during exploration of the zone graph, it is essential to have a very 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 $Z \subseteq Z'$ or $Z \subseteq \mathit{Closure}(\mathit{Extra}^{+}_{LU}(Z'))$ [11]. It is well known that all the other operations needed for forward exploration, can be done in at most quadratic time [18]. All missing proofs can be found in Appendix B.

We solve the inclusion problem in two steps. We first concentrate on the question: given a region R and a zone Z, when $R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ holds. We show the crucial point that this can be decided by verifying if the projection on every pair of variables satisfies this inclusion. Since $\mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ is not convex we need to find a way to work with Z instead. It turns out that one can define $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$ in such a way that $R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ is equivalent to $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R) \cap Z \neq \emptyset$. We show moreover that $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$ is a zone. This gets us already half way to the result, the rest being examination of the structure of the intersection.

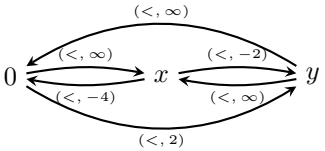

Fig. 4. Distance graph for the zone $(x - y \geq 1 \wedge y < 2 \wedge x > 4)$. Observe that it is in canonical form.

Once the inclusion question is solved with respect to regions, we extend the solution to zones thanks to a method allowing us to quickly tell which regions intersect a given zone.

For the rest of the section, we assume a given automaton $\mathcal{A}$ with LU bounds. Before we begin we will need to recall some standard notions. Let us consider a *bound function* associating to each clock x of $\mathcal{A}$ a bound $\alpha_x \in \mathbb{N}$ (that is the maximum of L and U bounds). A *region* [1] with respect to α is the set of valuations specified as follows:

1) for each clock $x \in X$, one constraint from the set: $\{x = c \mid c = 0, \ldots, \alpha_x\} \cup \{c - 1 < x < c \mid c = 1, \ldots, \alpha_x\} \cup \{x > \alpha_x\}$
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 $\{y\}$.

One can check that the set of regions finitely partitions $\mathbb{R}^X_{\geq 0}$.

A notion of a zone has already been recalled on page 3. Every region is a zone but not vice-versa.

It will be very convenient to represent zones by *distance graphs*. Such a graph has clocks as vertices, with an additional special clock x_0 representing the constant 0. For readability, we will often write 0 instead of x_0. Between every two vertices there is an edge with a weight of the form $(\preccurlyeq, c)$ where $c \in \mathbb{Z} \cup \{\infty\}$ and $\preccurlyeq$ is either $\leq$ or $<$. An edge $x \xrightarrow{\preccurlyeq c} y$ represents a constraint $y - x \preccurlyeq c$: or in words, the distance from x to y is bounded by c. An example of a distance graph is depicted in Fig. 4.

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 denote a distance graph G by the set of its weights: $(\preccurlyeq_{ij}, c_{ij})_{i,j \in X}$.

An arithmetic over the weights $(\preccurlyeq, c)$ can be defined as follows [5].

Equality $(\preccurlyeq_1, c_1) = (\preccurlyeq_2, c_2)$ if $c_1 = c_2$ and $\preccurlyeq_1 = \preccurlyeq_2$.
Addition $(\preccurlyeq_1, c_1) + (\preccurlyeq_2, c_2) = (\preccurlyeq, c_1 + c_2)$ where $\preccurlyeq = <$ iff either $\preccurlyeq_1$ or $\preccurlyeq_2$ is $<$.
Minus $-(\preccurlyeq, c) = (\preccurlyeq, -c)$.
Order $(\preccurlyeq_1, c_1) < (\preccurlyeq_2, c_2)$ if either $c_1 < c_2$ or ($c_1 = c_2$ and $\preccurlyeq_1 = <$ and $\preccurlyeq_2 = \leq$).

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 useful lemma is folklore.

Lemma 13 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. A *distance graph of a region* R, denoted G_R, is the canonical graph representing all the constraints defining R. Similarly G_Z for a zone Z. 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 G_2.

The first result says that for every zone Z, the set $\mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ is a union of regions.

Proposition 14 Let Z be a zone: every region that has a nonempty intersection with $\mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ is included in $\mathfrak{a}_{\preccurlyeq_{LU}}(Z)$.

A. When is $R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$?

We will first transform the question about the inclusion $R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ into one about an intersection. We begin by defining an operator $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}$.

Definition 15 ($\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}$ abstraction) Let W be a set of valuations. Then, $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(W)$ is the set of valuations defined as follows:

$$\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(W) = \{v' \mid \exists v \in W \text{ with } v \preccurlyeq_{LU} v'\}.$$

Next lemma says that deciding if $R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ can be reduced to checking if $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$ intersects with Z.

Lemma 16 Given a region R and a zone Z, we have

$$R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z) \text{ iff } \mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R) \cap Z \neq \emptyset$$

Proof: Suppose $R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ and let $v \in R$. As $v \in \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ too, there exists a valuation $v' \in Z$ such that $v \preccurlyeq_{LU} v'$. Now by Definition 15, we get $v' \in \mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$ showing that v' belongs to both Z and $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$. Hence $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R) \cap Z \neq \emptyset$.

Suppose $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R) \cap Z \neq \emptyset$ and let $v' \in \mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R) \cap Z$. This shows that $v' \in Z$ and $v \preccurlyeq_{LU} v'$ for some valuation $v \in R$. Now from the definition of $\preccurlyeq_{LU}$, we get $v \in \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$. Therefore, we have a valuation v such that $v \in R$ and $v \in \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$. From Lemma 14, this means $R \subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$. ■

We will now focus on the intersection question: when is $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R) \cap Z$ empty. Given the canonical distance graphs G_R and G_Z for R and Z respectively, the idea is to represent $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$ as a distance graph G^*_R and check when $\min(G^*_R, G_Z)$ has negative cycles. We first partition the set of clocks X into four sets based on the region R and then define the distance graph G^*_R for $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$ based on these sets.

Definition 17 (Partitioning clocks based on R) Let R be a region and let $G_R = (\lessdot_{ij}, c_{ij})_{i,j \in X}$ be its distance graph in canonical form. Then, we partition the set of clocks X into four sets: $\mathcal{B}_R, \mathcal{L}_R, \mathcal{U}_R$ and $\mathcal{M}_R$ as follows:

$$\begin{aligned}
\mathcal{B}_R &= \{x \in X \mid c_{0x} \leq \min(L_x, U_x)\} \cup x_0 \\
\mathcal{L}_R &= \{x \in X \mid L_x < c_{0x} \leq U_x\} \\
\mathcal{U}_R &= \{x \in X \mid U_x < c_{0x} \leq L_x\} \\
\mathcal{M}_R &= \{x \in X \mid \max(L_x, U_x) < c_{0x}\}
\end{aligned}$$

Definition 18 (Distance graph for $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$) Given a region R and its associated distance graph in canonical form $G_R = (\lessdot_{ij}, c_{ij})_{i,j \in X}$, the distance graph G^*_R is given by $(\lessdot'_{ij}, c'_{ij})_{i,j \in X}$ where:

$$(\lessdot'_{ij}, c'_{ij}) = \begin{cases} (<, \infty) & \text{if } j \in \mathcal{M}_R \cup \mathcal{U}_R \\ (<, \infty) & \text{if } i \in \mathcal{M}_R \cup \mathcal{L}_R \text{ and } j \neq 0 \\ (<, -L_i) & \text{if } i \in \mathcal{M}_R \cup \mathcal{L}_R \text{ and } j = 0 \\ (\lessdot_{ij}, c_{ij}) & \text{otherwise} \end{cases}$$

The following lemma confirms that the distance graph defined above indeed represents $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$.

Lemma 19 Let G_R be the canonical distance graph of a region R. Then $[\![G^*_R]\!] = \mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$.

We now have two distance graphs G^*_R, G_Z corresponding to $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$ and Z respectively. Therefore, checking if $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R) \cap Z$ is empty reduces to checking if the distance graph $\min(G^*_R, G_Z)$ has a negative cycle. To get G^*_R, we took G_R and modified some edges to $(<, \infty)$ and some edges of the form $x \to 0$ to $(<, -L_x)$. So graph G^*_R need not necessarily be in canonical form and we want to find negative cycles without canonizing it as this can be algorithmically expensive.

We will now state a necessary and sufficient condition for the graph $\min(G^*_R, G_Z)$ to have a negative cycle. We denote by Z_{xy} the weight of the edge $x \xrightarrow{\lessdot_{xy} c_{xy}} y$ in the canonical distance graph representing Z. Similarly for R. When a variable x represents the special clock x_0, we define R_{0x} to be $(\leq, 0)$. Since by convention x_0 is always 0, this is consistent.

Proposition 20 Let G_R, G_Z be the canonical distance graphs for a region R and a zone Z respectively. Then, $\min(G^*_R, G_Z)$ has a negative cycle iff there exists a variable $x \in \mathcal{B}_R \cup \mathcal{L}_R$ and a variable $y \in X$ such that one of the following conditions is true:

1) either $y \in \mathcal{B}_R \cup \mathcal{U}_R$ and $Z_{xy} + R_{yx} < (\leq, 0)$,
2) or $y \in \mathcal{L}_R \cup \mathcal{M}_R$ and $R_{0x} + Z_{xy} + (<, -L_y) < (\leq, 0)$.

The proof of Lemma 20 follows from Lemmas 22 and 23 below whose proofs in turn rely on an important observation made in Lemma 21. We say that a variable x is *bounded* in R if a constraint $x \leq c$ holds in R for some constant c.

Lemma 21 Let x, y be bounded variables of R appearing in some negative cycle N of $\min(G^*_R, G_Z)$. Let the edge weights be $x \xrightarrow{\preccurlyeq c_{xy}} y$ and $y \xrightarrow{\preccurlyeq c_{yx}} x$ in G_R. If the value of the path $x \to \ldots \to y$ in N is strictly less than $(\preccurlyeq_{xy}, c_{xy})$, then $x \to \ldots \to y \xrightarrow{\preccurlyeq_{yx} c_{yx}} x$ is a negative cycle.

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

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

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

Lemma 22 Suppose there exists a negative cycle in $\min(G^*_R, G_Z)$ containing no edges of the form $x \xrightarrow{<-L_x} 0$. Then, there exist variables $x \in \mathcal{B}_R \cup \mathcal{L}_R$ and $y \in \mathcal{B}_R \cup \mathcal{U}_R$ such that $Z_{xy} + R_{yx} < (\leq, 0)$.

Proof: Let N be a negative cycle of $\min(G^*_R, G_Z)$ containing no edges of the form $x \xrightarrow{<-L_x} 0$. Therefore the value of every edge in N comes from either G_Z or G_R. Since both these graphs are canonical, we can assume without loss of generality that no two consecutive edges in N come from the same graph.

Suppose N has two edges $x_1 \to x_2$ and $y_1 \to y_2$ with edge values coming from G_R. From the definition of G^*_R we get:

$$\begin{aligned} x_1, y_1 &\notin \mathcal{L}_R \cup \mathcal{M}_R \\ x_2, y_2 &\notin \mathcal{U}_R \cup \mathcal{M}_R \end{aligned} \tag{3}$$

This condition implies that all the four variables are bounded. Hence there exist finite valued edges $x_1 \xrightarrow{\preccurlyeq c} y_2$ and $y_2 \xrightarrow{\preccurlyeq' c'} x_1$ in G_R.

Suppose $(\preccurlyeq, c)$ is lesser than or equal to the value of the path $x_1 \to \ldots \to y_2$ of N. Then, we could replace this path by the edge $x_1 \xrightarrow{\preccurlyeq c} y_2$ to get a smaller negative cycle N_1. From condition (3) and from the definition of G^*_R, we get that the edge $x_1 \xrightarrow{\preccurlyeq c} y_2$ remains in G^*_R and hence N_1 is a negative cycle of $\min(G^*_R, G_Z)$.

Suppose $(\preccurlyeq, c)$ is greater than the value of the path $x_1 \to \ldots \to y_2$. Then, by Lemma 21, we get $x_1 \to x_2 \to \ldots y_1 \to y_2 \xrightarrow{\preccurlyeq' c'} x_1$ to be a negative cycle. Since G_R is canonical, we can replace $y_1 \to y_2 \to x_1 \to x_2$ by the edge $y_1 \to x_2$ to get a smaller negative cycle N_2. Again from condition (3), we get that $y_1 \to x_2$ remains in G^*_R and N_2 is a negative cycle of $\min(G^*_R, G_Z)$.

In both cases, we have eliminated two edges with value coming from G_R to get a smaller cycle with a single edge instead. Continuing this further, we would get a negative cycle containing only one edge coming from G_R. Moreover, we have seen that this edge would be retained in G^*_R too. Since G_Z is canonical, there would be only one edge coming from G_Z, which gives a negative cycle of the form $x \to y \to x$ with $x \to y$ coming from G_Z and $y \to x$ coming from G^*_R. From the definition of G^*_R, we see that $x \in \mathcal{B}_R \cup \mathcal{L}_R$ and $y \in \mathcal{B}_R \cup \mathcal{U}_R$. ∎

Lemma 23 Suppose there exists a negative cycle in $\min(G^*_R, G_Z)$ containing an edge $y \xrightarrow{<-L_y} 0$ with $y \in \mathcal{L}_R \cup \mathcal{M}_R$. Then, either there is a smaller negative cycle with no edge of the form $y \xrightarrow{<-L_y} 0$, or there exists $x \in \mathcal{B}_R \cup \mathcal{U}_R$ such that $R_{0x} + Z_{xy} + (<, -L_y) < (\leq, 0)$.

Proof: Let N be a negative cycle in $\min(G^*_R, G_Z)$ that contains the edge $y \xrightarrow{<-L_y} 0$ with $y \in \mathcal{L}_R \cup \mathcal{M}_R$. If the vertex 0 occurs once again in N, we could obtain a smaller negative cycle containing only one occurrence of 0. Hence without loss of of generality, we can assume that 0 occurs only once in N, with the incoming edge $y \xrightarrow{<-L_y} 0$. Consequently, every other edge value in N comes from either G_R or G_Z and since both these graphs are canonical, without loss of generality, we can assume that no two consecutive edges come from the same graph in the path from 0 to y.

Consider the variable y with its predecessor: $x \xrightarrow{\preccurlyeq d} y$. Suppose the value $(\preccurlyeq, d)$ comes from G_R. We can first infer from the definition of G^*_R that $x \notin \mathcal{L}_R \cup \mathcal{M}_R$. Now suppose we have the edge $y \xrightarrow{\preccurlyeq' -d'} 0$ in G_R. This means that $d' \preccurlyeq' y$ in R and since $y \in \mathcal{L}_R \cup \mathcal{M}_R$, we can see that $d' \geq L_y$. This gives $(<, -L_y) \geq (\preccurlyeq', -d')$ and hence we can replace $x \xrightarrow{\preccurlyeq d} y \xrightarrow{<-L_y} 0$ by the edge $x \to 0$ coming from G_R. As we have already seen that $x \notin \mathcal{L}_R \cup \mathcal{M}_R$, the edge $x \to 0$ from G_R remains in G^*_R too. Replacing by the edge $x \to 0$ gives a negative cycle without an edge of the form $y \xrightarrow{<-L_y} 0$. Therefore, without loss of generality let us consider the value $(\preccurlyeq, d)$ to come from G_Z.

Consider an edge $x_1 \to x_2$ that is part of N with edge value coming from G_R. Firstly, we can infer that $x_2 \notin \mathcal{U}_R \cup \mathcal{M}_R$. Now consider the edges $0 \xrightarrow{\preccurlyeq c} x_2$ and $x_2 \xrightarrow{\preccurlyeq' c'} 0$ of G_R. If $(\preccurlyeq, c)$ is smaller than the value of the path $0 \to \ldots \to x_2$ in N, we can replace the path by the edge $0 \to x_2$ that we know remains in G^*_R since $x_2 \notin \mathcal{U}_R \cup \mathcal{M}_R$. Otherwise, from Lemma 21, we get $0 \to \ldots x_1 \to x_2 \xrightarrow{\preccurlyeq' c'} 0$ to be a negative cycle. This cycle does not contain the edge $y \xrightarrow{<-L_y} 0$ and it is indeed smaller than N since we have assumed the edge $x \to y$ to come from G_Z and so x_2 is not y.

From the above paragraphs, we get that we can reduce N either to a smaller negative cycle without $y \xrightarrow{<-L_y} 0$ edge or to a negative cycle with $y \xrightarrow{<-L_y} 0$ that satisfies the following properties:

- if the predecessor to y is x, the edge $x \to y$ should come from G_Z,
- the only edge coming from G_R is of the form $0 \to x'$, with $x' \in \mathcal{B}_R \cup \mathcal{L}_R$.

Hence, along with the fact that G_Z is canonical, we get this negative cycle to be of the form $0 \to x \to y \xrightarrow{<-L_y} 0$ where the value of $0 \to x$ comes from G_R and the value of $x \to y$ comes from G_Z with $x \in \mathcal{B}_R \cup \mathcal{L}_R$ and $y \in \mathcal{L}_R \cup \mathcal{M}_R$. ■

B. Efficient inclusion testing

We briefly present the remaining steps for constructing an efficient algorithm to check if $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$. Recall that we are aiming at an $\mathcal{O}(|X|^2)$ complexity. Lemma 20 can be used to efficiently determine if a region $R \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$. The task is to find the regions intersecting Z and to consider all the possible cases.

For two variables x, y, Lemma 34 in Appendix C gives the minimum value of R_{yx} among the regions R intersecting a zone Z. To be able to use Lemma 20 we additionally require the variables x, y to be in appropriate sets $\mathcal{B}_R, \mathcal{L}_R, \mathcal{U}_R$ or $\mathcal{M}_R$ with respect to R. To achieve this, one needs to consider the relevant part of the zone that has regions with x and y in appropriate sets and then apply Lemma 34. We get the following theorem that can be directly transformed into an algorithm. The proof of this Theorem appears in Appendix C

Theorem 24 *Let Z, Z' be non-empty zones. Then, $Z \not\subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ iff there exists a variable x with $Z_{x0} \geq (\leq, -U_x)$ and a variable y such that one of the following conditions is true:*

- *$Z_{y0} > (<, -L_y)$ and $Z'_{xy} < Z_{xy}$ and $Z'_{xy} + (\leq, -L_y) < Z_{x0}$*
- *$Z_{y0} \leq (<, -L_y)$ and $Z_{xy} + (<, U_x - L_y) \geq (\leq, 0)$ and $Z'_{xy} + (<, -L_y) < Z_{x0}$*

VI. Conclusions

We have shown how one can use non-convex abstractions while still working with zones. This works as soon as the abstraction satisfies the transition compatibility condition. For the construction to be efficient though, one needs an efficient inclusion test. We have given such a test for $\mathfrak{a}_{\preccurlyeq LU}$ abstraction. In [11] we have shown an efficient inclusion test for $Closure^+_{LU}$ abstraction. The test presented here is conceptually more difficult to obtain. In the case of $Closure^+_{LU}$ we were looking which regions intersect a closure of a zone. For this it has been of course enough to look at the zone itself. Since $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is not defined as a closure of a zone, the task here has been substantially more complicated. It is even surprising that the inclusion test with respect to such a big abstraction can be done by simply looking at projections on two variables.

The result showing that $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is the biggest 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 [3] there have been no new abstraction operators [6]. 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 inclusion test for $\mathfrak{a}_{\preccurlyeq LU}$ turns out to be even simpler than for $Closure^+_{LU}$, the latter in turn subsumes $Extra^+_{LU}$ test. Hence by all criteria it is preferable to use $\mathfrak{a}_{\preccurlyeq LU}$ to the other two.

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.

References

[1] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] G. Behrmann, P. Bouyer, E. Fleury, and K. G. Larsen. Static guard analysis in timed automata verification. In *TACAS*, volume 2619 of *LNCS*, pages 254–270. Springer, 2003.

[3] G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelanek. Lower and upper bounds in zone-based abstractions of timed automata. *Int. J. on Software Tools for Technology Transfer*, 8(3):204–215, 2006.

[4] G. Behrmann, A. David, K. G Larsen, J. Haakansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *QEST*, pages 125–126. IEEE Computer Society, 2006.

[5] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 87–124. Springer, 2004.

[6] P. Bouyer. *From Qualitative to Quantitative Analysis of Timed Systems*. Mémoire d'habilitation, Université Paris 7, Paris, France, 2009.

[7] P. Bouyer, U. Fahrenberg, K.G.. Larsen, and N. Markey. Quantitative analysis of real-time systems using priced timed automata. *Communications of the ACM*, 54:78–87, 2011.

[8] T. Chen, T. Han, and J.-P. Katoen. Time-abstracting bisimulation for probabilistic timed automata. In *TASE*, pages 177–184. IEEE Computer Society, 2008.

[9] C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In *TACAS*, volume 1384 of *LNCS*, pages 313–329. Springer, 1998.

[10] D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *AVMFSS*, volume 407 of *LNCS*, pages 197–212. Springer, 1989.

[11] F. Herbreteau, D. Kini, B. Srivathsan, and I. Walukiewicz. Using non-convex approximations for efficient analysis of timed automata. In *FSTTCS*, volume 13 of *LIPIcs*, pages 78–89. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[12] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[13] F. Laroussinie and Ph. Schnoebelen. The state-explosion problem from trace to bisimulation equivalence. In *FoSSaCS*, volume 1784 of *LNCS*, pages 192–207. Springer, 2000.

[14] G. Morbé, F. Pigorsch, and C. Scholl. Fully symbolic model checking for timed automata. In *CAV*, volume 6806 of *LNCS*, pages 616–632. Springer, 2011.

[15] S. Tasiran, R. Alur, R. P. Kurshan, and R. K. Brayton. Verifying abstractions of timed systems. In *CONCUR*, volume 1119 of *LNCS*, pages 546–562. Springer, 1996.

[16] S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Form. Methods Syst. Des.*, 18:25–68, 2001.

[17] F. Wang. Efficient verification of timed automata with BDD-like data structures. *Int. J. on Software Tools for Technology Transfer*, 6:77–97, 2004.

[18] J. Zhao, X. Li, and G. Zheng. A quadratic-time DBM-based successor algorithm for checking timed automata. *Inf. Process. Lett.*, 96(3):101–105, 2005.

APPENDIX

A. Compatibility of abstractions

We show that every abstraction defined based on a time-abstract simulation is transition compatible. We assume that we are given an automaton $\mathcal{A}$.

Definition 25 (Time-abstract simulation) A *(state based) time-abstract simulation* between two states of a transition system is a relation $(q, v) \preceq_{t.a.} (q', v')$ such that:

- $q = q'$,
- if $(q, v) \to^{\delta} (q, v + \delta) \to^{\alpha} (q_1, v_1)$, then there exists a $\delta' \in \mathbb{R}_{\geq 0}$ such that $(q', v') \to^{\delta'} (q', v' + \delta') \to^{\alpha} (q'_1, v'_1)$ 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 $\mathfrak{a}$ based on a simulation $\preceq_{t.a.}$ can be defined as follows:

Definition 26 (Abstraction based on simulation) Given a zone Z, we define $\mathfrak{a}(Z) = \{v : \exists v' \in Z.\ v \preceq_{t.a.} v'\}$.

For a given automaton this abstraction defines an abstract transition system. Our goal is to efficiently construct this system, or a relevant part of it if we are checking a reachability property. As explained in Section II, for nodes of this system we can use pairs of the form (q, Z), i.e., pairs consisting of a state and a zone. Such a pair will represent a configuration $(q, \mathfrak{a}(Z))$. Transition relation will be computed on zones. This is possible since the abstraction is defined using a simulation so it is automatically transition compatible.

Lemma 27 Let $\mathfrak{a}$ be an abstraction based on a simulation relation. For every transition $(q, \mathfrak{a}(Z)) \Rightarrow_{\mathfrak{a}} (q', W')$ and the matching transition $(q, Z) \Rightarrow (q', Z')$, we have $W' = \mathfrak{a}(Z')$.

Proof: Let $\mathfrak{a}$ be based on a simulation relation $\preceq_{t.a.}$, that is, for a set W, we have $\mathfrak{a}(W) = \{v : \exists v' \in W.\ v \preceq_{t.a.} v'\}$. Without loss of generality, assume that $\Rightarrow$ denotes a time-transition followed by an action: $\to^{\delta}\to^{a}$.

Let $v \in W'$. Then, by definition of $(q, \mathfrak{a}(Z)) \Rightarrow_{\mathfrak{a}} (q', W')$, there exists $v_1 \in \mathfrak{a}(Z)$ and a $\delta_1 \in \mathbb{R}_{\geq 0}$ such that $(q, v_1) \to^{\delta_1}\to^{a} (q', v'_1)$ and $v \preceq_{t.a.} v'_1$. Now, since $v_1 \in \mathfrak{a}(Z)$, we can find $v_2 \in Z$ satisfying $v_1 \preceq_{t.a.} v_2$. Therefore by definition of simulation relation, there exists a $\delta_2 \in \mathbb{R}_{\geq 0}$ which enables the transition: $(q, v_2) \to^{\delta_2}\to^{a} (q', v'_2)$ and yields $v'_1 \preceq_{t.a.} v'_2$. As we have seen before we have $v \preceq_{t.a.} v'_1$ and so we can infer that $v \preceq_{t.a.} v'_2$. By completeness of $\Rightarrow$, we will have $v'_2 \in Z'$ and hence $v \in \mathfrak{a}(Z')$. This shows that $W' \subseteq \mathfrak{a}(Z')$.

Let $v \in \mathfrak{a}(Z')$. Then, there exists $v_1 \in Z$ and a $\delta_1 \in \mathbb{R}_{\geq 0}$ such that $v_1 \to^{\delta_1}\to^{a} v'_1$ and $v \preceq_{t.a.} v'_1$. By the property of an abstraction operator, we will have $v_1 \in \mathfrak{a}(Z)$ too. Now, directly by the definition of $(q, \mathfrak{a}(Z)) \Rightarrow^{\mathfrak{a}} (q', W')$, we get that $v \in W'$ and this shows $\mathfrak{a}(Z') \subseteq W'$. ■

B. Proofs for Section V

1) Proof of Proposition 14: Proposition 14 states that for every zone Z, the abstraction $\mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ is always a union of regions. Before proving the proposition, we begin with a lemma that relates the simulation $v \preccurlyeq_{LU} v'$ and the containment $v' \in r_{LU}(v)$ defined in page 4.

Lemma 28 Let v, v' be valuations such that $v \preccurlyeq_{LU} v'$. Then, $v' \in r_{LU}(v)$.

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

Let x, y be clocks such that $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ and $v(x) \leq U_x$, $v(y) \leq L_y$. 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 $v'(x) \leq v(x)$. Similarly, for y, one can conclude that $v'(y) \geq v(y)$. As the integer parts are the same in v and v', we get $\{v'(x)\} < \{v'(y)\}$ or $\{v'(x)\} \leq \{v'(y)\}$ depending on whether $\lessdot$ is $<$ or $=$. ■

►**PROPOSITION 14.** Let Z be a zone: every region that has a nonempty intersection with $\mathfrak{a}_{\preccurlyeq_{LU}}(Z)$ is included in $\mathfrak{a}_{\preccurlyeq_{LU}}(Z)$.

Proof: Let v and w be valuations belonging to the same region. Assume that $v \in \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$. So there exists a valuation $v' \in Z$ such that $v \preccurlyeq_{LU} v'$. From Lemma 28, we get $v' \in r_{LU}(v)$. Since w belongs to the same region as v, one also has $v' \in r_{LU}(w)$. From the adjustment lemma, there exists $w' \in \mathrm{nbd}(v')$ such that $w \preccurlyeq_{LU} w'$. But values in the same neighbourhood satisfy the same difference constraints and should hence belong to the same zones. This gives that $w' \in Z$ and hence $w \in \mathfrak{a}_{\preccurlyeq_{LU}}(Z)$. ■

2) Proof of Lemma 19: Lemma 19 states that the distance graph G^*_R defined in Definition 18 captures exactly the set $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$.

►**LEMMA 19.** Let G_R be the canonical distance graph of a region R. Then $[\![G^*_R]\!] = \mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$.

We begin with the following lemma that shows one side of the implication.

Lemma 29 Let v' be a valuation in $\mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$. Then, $v' \in [\![G^*_R]\!]$.

Proof: Let G_R be given by $(\lessdot_{ij}, c_{ij})_{i,j \in X}$ and let $G^*_R = (\lessdot'_{ij}, c'_{ij})_{i,j \in X}$ be the graph obtained from Definition 18.

We will show that valuation v' has to satisfy the constraints given by G^*_R. That is, we will now show that for every $i, j \in X$, we get $v'_j - v'_i \lessdot'_{ij} c'_{ij}$. From the definition of G^*_R finite weights occur only in edges of the form $i \to j$ and $j \xrightarrow{<-L_j} 0$ with $i \in \mathcal{B}_R \cup \mathcal{U}_R$ and $j \in \mathcal{B}_R \cup \mathcal{L}_R$. In the former case, the finite values are in fact $(\lessdot_{ij}, c_{ij})$. It is enough to consider these edges.

Now, as $v' \in \mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$, there exists a valuation $v \in R$ such that $v \preccurlyeq_{LU} v'$. The valuation v satisfies the constraints of G_R, that is $v_j - v_i \lessdot_{ij} c_{ij}$. Consider two variables, $i \in \mathcal{B}_R \cup \mathcal{U}_R$ and $j \in \mathcal{B}_R \cup \mathcal{L}_R$. Since $v \preccurlyeq_{LU} v'$, we will have $v'_i \geq v_i$ and $v'_j \leq v_j$. This clearly gives $v'_i - v'_j \lessdot_{ij} c_{ij}$ too. Also since $j \in \mathcal{L}_R \cup \mathcal{M}_R$, we will have $L_j < v'_j \leq v_j$ which shows that the constraint $j \xrightarrow{<-L_j} 0$ is satisfied. ■

The rest of the section is devoted to prove that if $v' \in G^*_R$ then $v' \in \mathfrak{a}^{-1}_{\preccurlyeq_{LU}}(R)$. Let v be an arbitrary valuation such that $v \in R$. We will first show that $v' \in r_{LU}(v)$. We will then give a reverse-adjustment lemma below which will entail there exists a valuation $v_1 \in \mathrm{nbd}(v)$ such that $v_1 \preccurlyeq_{LU} v'$. Since $v_1 \in \mathrm{nbd}(v)$, it would also belong to R.

Lemma 30 Let R be a region and let $v' \in G^*_R$. Then, for every valuation $v \in R$, $v' \in r_{LU}(v)$.

Proof: Let v be a valuation in R. From the definition of G^*_R, it can be easily seen that both v and v' satisfy the same LU-guards. It is the second property about the fractional parts for clocks with the same integer parts that needs to be checked.

Let x, y be clocks such that $\lfloor v'(x) \rfloor = \lfloor v(x) \rfloor$, $\lfloor v'(y) \rfloor = \lfloor v(y) \rfloor$ and $v(x) \leq U_x$ and $v(y) \leq L_y$. By the partition of clocks this means that $x \notin \mathcal{U}_R$ and $y \notin \mathcal{L}_R$. From Definition 18, the edge $y \to x$ carries the same weight as that of G_R in G^*_R.

Let $\lfloor v(x) \rfloor = c_x$, $\lfloor v(y) \rfloor = c_y$ and let $y \xrightarrow{\lessdot d} x$ be the edge in G_R. This entails that all valuations in R satisfy $x - y \lessdot d$. Hence their fractional parts satisfy:

$$\{x\} - \{y\} \lessdot d - (c_x - c_y)$$

Suppose $\{x\} < \{y\}$ for all valuations and since G_R is canonical, we can infer $d - (c_x - c_y) \leq 0$ and if it is 0 then $\lessdot$ is $<$.

Now consider the graph G^*_R. Since the edge $y \xrightarrow{\lessdot d} x$ remains in G^*_R, and since $v' \in G^*_R$, the valuation v' should satisfy $v'_x - v'_y \lessdot d$ and as $\lfloor v'_y \rfloor = c_y$ and $\lfloor v'_x \rfloor = c_x$, we get:

$$\begin{aligned} & \{v'_x\} - \{v'_y\} \lessdot d - (\lfloor v'_x \rfloor - \lfloor v'_y \rfloor) \\ \Rightarrow\ & \{v'_x\} - \{v'_y\} \lessdot d - (c_x - c_y) \end{aligned}$$

We saw before that either $d - (c_x - c_y) < 0$ or if it is 0, then $\lessdot$ is $<$. This shows that $\{v'(x)\} < \{v'(y)\}$.

The other case when $\{v(x)\} = \{v(y)\}$ can be shown exactly in the same manner. ■

Lemma 31 (Reverse-adjustment) Let v, v' be valuations such that $v' \in r_{LU}(v)$. Then there exists a valuation $v_1 \in \mathrm{nbd}(v)$ such that $v_1 \preccurlyeq_{LU} v'$.

Proof: The task is to pick a valuation v_1 that has the same integral parts as v and agrees to the ordering of fractional parts as in v. Similar to the proof of the adjustment lemma, it is enough to choose fractional parts for the clocks X_f that have:

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

Again, as $v' \in r_{LU}(v)$, we have the following property:

$$\begin{aligned} \forall x, y \in X_f \text{ such that } v(x) \leq U_x \text{ and } v(y) \leq L_y \qquad (4) \\ \{v(x)\} < \{v(y)\} \Rightarrow \{v'(x)\} < \{v'(y)\} \\ \{v(x)\} = \{v(y)\} \Rightarrow \{v'(x)\} \leq \{v'(y)\} \end{aligned}$$

Let $0\delta_1 < \delta_2 < \cdots < \delta_n < 1$ be the fractional parts taken by clocks of X_f in v and let X_i be defined as follows:

$$X_i = \{x \in X_f \mid \{v(x)\} = \delta_i\}$$

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

We will now select n values $0 < \sigma_1 < \sigma_2 < \cdots < \sigma_n < 1$ and set for all clocks $x_i \in X_i$, the $\{v_1(x_i)\}$ to be δ_i. We perform an induction involving n steps.

After the k^{th} 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

$$v'(y) \leq U_y \Rightarrow \{v'(y)\} < \sigma_{n-k+1} \qquad (5)$$

Let us now perform the $k + 1^{th}$ step and show that the 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:

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

It can be shown that $u' \leq l'$. The rest of the proof follows in exactly the same lines as that of the adjustment lemma. ■

C. Proof of Theorem 24

For ease of reading, we make use of the following notations in this section.

Remark 32 (Notations) For a clock x and a valuation v, we denote $v(x)$ by v_x.

We begin with a few definitions. For a weight $(\lessdot, c)$ we define $-(\lessdot, c)$ as $(\lessdot, -c)$. We now define a *ceiling* function $\lceil \cdot \rceil$ for weights.

Definition 33 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 $\lceil (\lessdot, c) \rceil$ for a weight $(\lessdot, c)$ depending on whether $\lessdot$ 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 lemma is the core for the proof of the main theorem. It gives the least value of R_{xy} from among the regions R that intersect Z.

Lemma 34 Let Z be a non-empty zone and let x, y be variables. Then, from among the regions R that intersect Z, the least value of R_{xy} is given by

$$\begin{cases} (<, \infty) & \text{if } Z_{y0} < (\leq, -\alpha_y) \\ \max\{\lceil -Z_{yx} \rceil, \lceil -Z_{y0} \rceil - (\leq, \alpha_x)\} & \text{otherwise} \end{cases}$$

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 $(\preccurlyeq_{ij}, c_{ij})$. Recall that this means $Z_{ij} = (\preccurlyeq_{ij}, c_{ij})$. For clarity, for a valuation v, we write v_x for $v(x)$.

We are interested in computing the smallest value of the $y - x$ constraint defining a region belonging to $Closure_\alpha(Z)$, that is, we need to find $\min\{[v]_{xy} \mid v \in Z\}$. Call this β. By definition of regions, we have for a valuation v:

$$[v]_{xy} = \begin{cases} (<, \infty) & \text{if } v_y > \alpha_y \\ \lceil (\leq, v_y - v_x) \rceil & \text{if } v_y \leq \alpha_y \text{ and } v_x \leq \alpha_x \\ (<, \lceil v_y \rceil - \alpha_x) & \text{if } v_y \leq \alpha_y \text{ and } v_x > \alpha_x \end{cases} \quad (6)$$

We now consider the first of the two cases from the statement of the lemma. Namely, $Z_{y0} < (\leq, -\alpha_y)$. This means that $0 - v_y \preccurlyeq_{y0} c_{y0}$ and $c_{y0} \leq -\alpha_y$; moreover $\preccurlyeq_{y0}$ is the strict inequality if $c_{y0} = -\alpha_y$. In consequence, all valuations $v \in Z$, satisfy $v_y > \alpha_y$. Whence $\beta = (<, \infty)$.

We now consider the case when $Z_{y0} \geq (\leq, -\alpha_y)$. Let G' be the graph in which the edge $0 \to y$ has weight $\min\{(\leq, \alpha_y), (\preccurlyeq_{0y}, c_{0y})\}$ and the rest of the edges are the same as that of G. This graph G' represents the valuations of Z that have $v_y \leq \alpha_y$: $[\![G']\!] = \{v \in Z \mid v_y \leq \alpha_y\}$. 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 $0 \to y$. Note that the shortest path value from y to 0 does not change due to this modified edge. So the only possible negative cycle in G' is $0 \to y \to 0$. But then we are considering the case when $Z_{y0} \geq (\leq, -\alpha_y)$, and so $Z_{y0} + (\leq, \alpha_y) \geq (\leq, 0)$. 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 6, among the valuations in $[\![G']\!]$, we need to differentiate between those with $v_x \leq \alpha_x$ and the ones with $v_x > \alpha_x$. We proceed as follows. We first compute $\min\{[v]_{xy} \mid v \in [\![G']\!] \text{ and } v_x \leq \alpha_x\}$. Call this β_1. Next, we compute $\min\{[v]_{xy} \mid v \in [\![G']\!] \text{ and } v_x > \alpha_x\}$ 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 $0 \to x$ to $\min\{(\leq, \alpha_x), (\preccurlyeq_{0x}, c_{0x})\}$ and keeping the remaining edges the same as in G'. The set of valuations $[\![G'_1]\!]$ equals $\{v \in [\![G']\!] \mid v_x \leq \alpha_x\}$. If $[\![G'_1]\!] = \emptyset$, we set β_1 to $(<, \infty)$ and proceed to calculate β_2. If not, we see that from Equation 6, for every $v \in [\![G'_1]\!]$, $[v]_{xy}$ is given by $\lceil (\leq, v_y - v_x) \rceil$. Let $(\preccurlyeq_1, w_1)$ be the shortest path from y to x in the graph G'_1. Then, we have for all $v \in [\![G'_1]\!]$, $v_x - v_y \preccurlyeq_1 w_1$. If $\preccurlyeq_1$ is $\leq$, then the least value of $[v]_{xy}$ would be $(\leq, -w_1)$ and if $\preccurlyeq_1$ is $<$, one can see that the least value of $[v]_{xy}$ is $(<, -w_1 + 1)$. This shows that $\beta_1 = \lceil (\preccurlyeq_1, -w_1) \rceil$. It now remains to calculate $(\preccurlyeq_1, w_1)$.

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 y to x has changed in G'_1, then clearly it should be due to one of the above two edges. However note that the edge $0 \to y$ cannot belong to the shortest path from y to x since it would contain a cycle $y \to \ldots 0 \to y \to \ldots x$ that can be removed to give shorter path. Therefore, only the edge $0 \to x$ can potentially yield a shorter path: $y \to \ldots \to 0 \to x$. However, the shortest path from y 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 y to 0 is the direct edge $y \to 0$, and the shortest path from y to x is the minimum of the direct edge $y \to x$ and the path $y \to 0 \to x$. We get: $(\preccurlyeq_1, w_1) = \min\{(\preccurlyeq_{yx}, c_{yx}), (\preccurlyeq_{y0}, c_{y0}) + (\leq, \alpha_x)\}$ which equals $\min\{Z_{yx}, Z_{y0} + (\leq, \alpha_x)\}$. Finally, from the argument in the above two paragraphs, we get:

$$\beta_1 = \begin{cases} (<, \infty) & \text{if } [\![G'_1]\!] = \emptyset \\ \lceil -Z_{yx} \rceil & \text{if } [\![G'_1]\!] \neq \emptyset \text{ and} \\ & \quad Z_{yx} \leq Z_{y0} + (\leq, \alpha_x) \\ \lceil -Z_{y0} \rceil + (\leq, -\alpha_x) & \text{if } [\![G'_1]\!] \neq \emptyset \text{ and} \\ & \quad Z_{yx} > Z_{y0} + (\leq, \alpha_x) \end{cases} \quad (7)$$

We now proceed to compute $\beta_2 = \min\{[v]_{xy} \mid v \in [\![G']\!] \text{ and } v_x > \alpha_x\}$. Let G'_2 be the graph which is obtained from G' by modifying the edge $x \to 0$ to $\min\{Z_{x0}, (<, -\alpha_x)\}$ and keeping the rest of the edges the same as in G'. Clearly $[\![G'_2]\!] = \min\{v \in [\![G']\!] \mid v_x > \alpha_x\}$.

Again, if $[\![G'_2]\!]$ is empty, we set β_2 to $(<, \infty)$. Otherwise, from Equation 6, for each valuation $v \in [\![G'_2]\!]$, the value of $[v]_{xy}$ is given by $(<, \lceil v_y \rceil - \alpha_x)$. For the minimum value, we need the least value of v_y from $v \in [\![G'_2]\!]$. Let $(\preccurlyeq_2, w_2)$ be the shortest path from y to 0 in G'_2. Then, since $-v_y \preccurlyeq_2 w_2$, the least value of $\lceil v_y \rceil$ would be $-w_2$ if $\preccurlyeq_2$ is $\leq$ and equal to $\lceil -w_2 \rceil$ if $\preccurlyeq_2$ is $<$ and β_2 would respectively be $(<, -w_2 - \alpha_x)$ or $(<, -w_2 + 1 - \alpha_x)$. It now remains to calculate $(\preccurlyeq_2, w_2)$.

Recall that G'_2 is G with $0 \to y$ and $x \to 0$ modified. The shortest path from y to 0 cannot include the edge $0 \to y$ since it would need to contain a cycle, for the same reasons as in the β_1 case. So we get $(\preccurlyeq_2, w_2) = \min\{Z_{y0}, Z_{yx} + (<, -\alpha_x)\}$. If $Z_{y0} \leq Z_{yx} + (<, -\alpha_x)$, then we take $(\preccurlyeq_2, w_2)$ as Z_{y0}, otherwise we take it to be $Z_{yx} + (<, -\alpha_x)$. So, we get β_2 as the following:

$$\beta_2 = \begin{cases} (<,\infty) & \text{if } [\![G'_2]\!] = \emptyset \\ -Z_{yx} + (<,1) & \text{if } [\![G'_2]\!] \neq \emptyset \text{ and} \\ & \quad Z_{y0} \geq Z_{yx} + (<,-\alpha_x) \\ \lceil -Z_{y0} \rceil + (<,-\alpha_x) & \text{if } [\![G'_2]\!] \neq \emptyset \text{ and} \\ & \quad Z_{y0} < Z_{yx} + (<,-\alpha_x) \end{cases} \tag{8}$$

However, we would like to write β_2 in terms of the cases used for β_1 in Equation 7 so that we can write β, which equals $\min\{\beta_1, \beta_2\}$, conveniently.

Let ψ_1 be the inequation: $Z_{yx} \leq Z_{y0} + (\leq, \alpha_x)$. From Equation 7, 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: $Z_{y0} \geq Z_{yx} + (<, -\alpha_x)$. From Equation 8 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 α_x: in the former the inequality associated with α_x 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 $(\preccurlyeq_{yx}, c_{yx}) \leq (\preccurlyeq_{y0}, c_{y0} + \alpha_x)$. This implies: $c_{yx} \leq c_{y0} + \alpha_x$. Therefore, $c_{y0} \geq c_{yx} - \alpha_x$. When $c_{y0} > c_{yx} - \alpha_x$, ψ_2 is clearly true. For the case when $c_{y0} = c_{yx} - \alpha_x$, note that in ψ_2 the right hand side is always of the form $(<, c_{yx} - \alpha_x)$, irrespective of the inequality in Z_{yx} 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 $(\preccurlyeq_{yx}, c_{yx}) > (\preccurlyeq_{y0}, c_{y0}+\alpha_x)$. If $c_{yx} > c_{y0} + \alpha_x$, then clearly $c_{y0} < c_{yx} - \alpha_x$ implying that $\neg\psi_2$ holds. If $c_{yx} = c_{y0} + \alpha_x$, then we need to have $\preccurlyeq_{yx} = \leq$ and $\preccurlyeq_{y0} = <$. Although $\neg\psi_2$ does not hold now, we can safely take β_2 to be $\lceil -Z_{y0} \rceil + (<, -\alpha_x)$ as its value is in fact equal to $-Z_{yx} + (<, 1)$ in this case. Summarizing the above two paragraphs, we can rewrite β_2 as follows:

$$\beta_2 = \begin{cases} (<,\infty) & \text{if } [\![G'_2]\!] = \emptyset \\ -Z_{yx} + (<,1) & \text{if } [\![G'_2]\!] \neq \emptyset \text{ and} \\ & \quad Z_{xy} \leq Z_{y0} + (\leq,\alpha_x) \\ \lceil -Z_{y0} \rceil + (<,-\alpha_x) & \text{if } [\![G'_2]\!] \neq \emptyset \text{ and} \\ & \quad Z_{xy} > Z_{y0} + (\leq,\alpha_x) \end{cases} \tag{9}$$

We are now in a position to determine β as $\min\{\beta_1, \beta_2\}$. Recall that we are in the case where $Z_{y0} \leq (\leq, -\alpha_y)$ 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 7 and 9, we get β, the $\min\{\beta_1, \beta_2\}$ as:

$$\beta_= \begin{cases} \lceil -Z_{yx} \rceil & \text{if } Z_{xy} \leq Z_{y0} + (\leq,\alpha_x) \\ \lceil -Z_{y0} \rceil + (<,-\alpha_x) & \text{if } Z_{xy} > Z_{y0} + (\leq,\alpha_x) \end{cases} \tag{10}$$

There remains one last reasoning. To prove the lemma, we need to show that $\beta = \max\{\lceil -Z_{yx} \rceil, \lceil -Z_{y0} \rceil + (<, -\alpha_x)\}$. For this it is enough to show the following two implications:

$$Z_{yx} \leq Z_{y0} + (\leq,\alpha_x) \Rightarrow \lceil -Z_{yx} \rceil \geq \lceil -Z_{y0} \rceil + (<,-\alpha_x)$$
$$Z_{yx} > Z_{y0} + (\leq,\alpha_x) \Rightarrow \lceil -Z_{yx} \rceil \leq \lceil -Z_{y0} \rceil + (<,-\alpha_x)$$

We prove only the first implication. The second follows in a similar fashion. Let us consider the notation $(\preccurlyeq_{yx}, c_{yx})$ and $(\preccurlyeq_{y0}, c_{y0})$ for Z_{yx} and Z_{y0} respectively. So we have:

$$\begin{aligned} & (\preccurlyeq_{yx}, c_{yx}) \leq (\preccurlyeq_{y0}, c_{y0}) + (\leq, \alpha_x) \\ \Rightarrow\ & (\preccurlyeq_{yx}, c_{yx}) \leq (\preccurlyeq_{y0}, c_{y0} + \alpha_x) \end{aligned}$$

If the constant $c_{yx} < c_{y0} + \alpha_x$, then $-c_{yx} > -c_{y0} - \alpha_x$ and we clearly get that $\lceil -Z_{yx} \rceil \geq \lceil -Z_{y0} \rceil + (<, -\alpha_x)$. If the constant $c_{yx} = c_{y0} + \alpha_x$ and if $\preccurlyeq_{y0} = \leq$, then the required inequation is trivially true; if $\preccurlyeq_{y0} = <$, it implies that $\preccurlyeq_{yx} = <$ too and clearly $\lceil (<, -c_{yx}) \rceil$ equals $\lceil (<, -c_{y0}) \rceil + (<, -\alpha_x)$.

■

We are now in a position to prove the main theorem.

►**THEOREM 24.** Let Z, Z' be non-empty zones. Then, $Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$ iff there exists a variable x with $Z_{x0} \geq (\leq, -U_x)$ and a variable y such that one of the following conditions is true:

- $Z_{y0} > (<, -L_y)$ and $Z'_{xy} < Z_{xy}$ and $Z'_{xy} + (\leq, -L_y) < Z_{x0}$
- $Z_{y0} \leq (<, -L_y)$ and $Z_{xy} + (<, U_x - L_y) \geq (\leq, 0)$ and $Z'_{xy} + (<, -L_y) < Z_{x0}$

Proof: From Lemma 20, we get that $Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$ iff there exists a region R intersecting Z that satisfies one of the following conditions for variables $x \in \mathcal{B}_R \cup \mathcal{L}_R$ and $y \in X$:

$$\begin{aligned} & y \in \mathcal{B}_R \cup \mathcal{U}_R \text{ and } Z'_{xy} + R_{yx} < (\leq, 0), \text{ or} \\ & y \in \mathcal{L}_R \cup \mathcal{M}_R \text{ and } R_{0x} + Z'_{xy} + (<, -L_y) < (\leq, 0) \end{aligned} \tag{11}$$

To see if the first of the above two conditions is true, we need the minimum value of R_{yx} from among the regions R intersecting Z and satisfying $R_{0y} \leq L_y$ and $R_{0x} \leq U_x$. The sum of this minimum value of R_{yx} and Z'_{xy} is less than $(\leq, 0)$ iff $Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$. Therefore, we first restrict our attention to the part of Z that gives regions with $R_{0y} \leq L_y$ and $R_{0x} \leq U_x$.

Let G_1 be the graph obtained from G_Z by modifying the edge $0 \to x$ to $\min(Z_{0x}, (\leq, U_x))$ and $0 \to y$ to $\min(Z_{0y}, (\leq, L_y))$. Every valuation $v \in [\![G_1]\!]$ has $v_x \leq U_x$ and $v_y \leq L_y$ and hence gives rise to a region of our required form. Conversely, every valuation $v \in Z$ that is part of a region of the required form has $v_x \leq U_x$, $v_y \leq L_y$ and hence satisfies the constraints of G_1, that is belongs to $[\![G_1]\!]$. We know that Z is non-empty. Therefore, $[\![G_1]\!]$ will be non-empty if the two modified edges do not introduce negative cycles:

$$[\![G_1]\!] \neq \emptyset \ \Leftrightarrow\ Z_{x0} \geq (\leq, -U_x) \text{ and } Z_{y0} \geq (\leq, -L_y) \tag{12}$$

Let us assume that $[\![G_1]\!]$ is non-empty. We will now use Lemma 34 to get the least value of R_{yx} among the regions R that intersect $[\![G_1]\!]$. There are two cases given by Lemma 34. We first need the shortest path from x to 0 in G_1 to find the

correct case. It is given by Z_{x0} itself since the newly modified edges cannot influence it. Therefore $[\![G_1]\!]_{x0}$ is exactly Z_{x0} and since $[\![G_1]\!]$ is non-empty, from Equation (12), $Z_{x0} \geq (\leq, -U_x)$ and in particular this implies $Z_{x0} \geq (<, -\alpha_x)$. So we need to consider the second case of the equation given in Lemma 34. The shortest path from x to y in G_1 is given by $\min(Z_{xy}, Z_{x0} + (\leq, L_y))$. From Lemma 34, the minimum value of R_{yx} is given by $\max(\lceil -Z_{xy} \rceil, \lceil -Z_{x0} \rceil + (\leq, -L_y), \lceil -Z_{x0} \rceil + (\leq, -\alpha_y))$. Since $(\leq, \alpha_y) \leq (\leq, L_y)$, we can safely discard the last component. Substituting in Condition 1, we get:

$$\begin{aligned} & Z'_{xy} + \max(\lceil -Z_{xy} \rceil, \lceil -Z_{x0} \rceil + (\leq, -L_y)) < (\leq, 0) \\ \Leftrightarrow\ & Z'_{xy} + \lceil -Z_{xy} \rceil < (\leq, 0) \text{ and} \\ & Z'_{xy} + \lceil -Z_{x0} \rceil + (\leq, -L_y) < (\leq, 0) \\ \Leftrightarrow\ & Z'_{xy} < Z_{xy} \text{ and} \\ & Z'_{xy} + (\leq, -L_y) < Z_{x0} \end{aligned} \tag{13}$$

Let us follow a similar procedure to now see if the second of our required conditions is true. Let G_2 be the graph obtained from G_Z by modifying the edge $0 \to x$ to $\min(Z_{0x}, (\leq, U_x))$ and the edge $y \to 0$ to $\min(Z_{y0}, (<, -L_y))$. The set $[\![G_2]\!]$ represents the set of valuations $v \in Z$ that have $v_x \leq U_x$ and $v_y > L_y$. As Z is non-empty, for $[\![G_2]\!]$ to be non-empty, the newly modified edges should not introduce a negative cycle:

$$\begin{aligned} [\![G_2]\!] \neq \emptyset \ \Leftrightarrow\ & Z_{x0} + (\leq, U_x) \geq (\leq, 0) \text{ and} \\ & Z_{0y} + (<, -L_y) \geq (\leq, 0) \text{ and} \\ & (\leq, U_x) + Z_{xy} + (<, -L_y) \geq (\leq, 0) \end{aligned} \tag{14}$$

Assume $[\![G_2]\!]$ is empty. We will again use Lemma 34 to get the least value of R_{0x} among the regions that intersect $[\![G_2]\!]$. The shortest path from x to 0 in G_2 is given by $\min(Z_{x0}, Z_{xy} + (< , -L_y))$. Call it δ. From Equation (14), we get both $Z_{x0} \geq (\leq, -U_x)$ and $Z_{xy} + (<, -L_y) \geq (\leq, -U_x)$ and hence in particular greater than or equal to $(\leq, -\alpha_x)$. Therefore $\delta \geq (\leq, -\alpha_x)$ and from Lemma 34, the least value of R_{0x} is given by $\lceil -\delta \rceil$. We now consider two separate cases:

a) When $Z_{y0} \leq (<, -L_y)$.: In this case, $\delta = Z_{x0}$ and substituting the least value of R_{0x} in Condition 2 of (11) gives:

$$Z'_{xy} + (<, -L_y) < Z_{x0} \tag{15}$$

So, when $Z_{y0} \leq (<, -L_y)$ checking for Condition 2 is equivalent to checking the conditions given by Equations (14) and (15). Also note that since $Z_{0y} + Z_{y0} \geq (\leq, 0)$, in this case we directly get $Z_{0y} + (<, -L_y) \geq (\leq, 0)$ and thus we get the second part in the statement of the theorem.

b) When $Z_{y0} > (<, -L_y)$.: Now, δ is $\min(Z_{x0}, Z_{xy} + (<, -L_y))$ and $\lceil -\delta \rceil$ equals $\max(\lceil -Z_{x0} \rceil, -Z_{xy} + (<, L_y) + (<, 1))$. Substituting this in Condition 2 gives:

$$\begin{aligned} & Z'_{xy} + (<, -L_y) < Z_{x0} \text{ and} \\ & Z_{xy} + Z'_{xy} + (<, 1) < (\leq, 0) \end{aligned} \tag{16}$$

So, when $Z_{y0} > (<, -L_y)$, checking for Condition 2 of (11) is equivalent to checking the conditions given by Equations (14) and (16). However this would imply that the conjunction of (12) and (13) is true. Therefore when $Z_{y0} > (<, -L_y)$ it is sufficient to check for Condition 1 of (11). ■