On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap

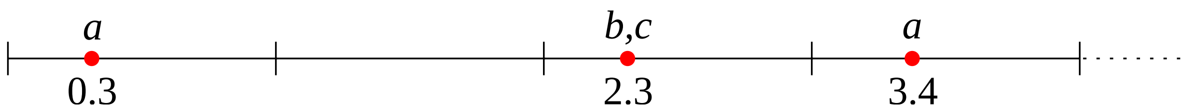
Joël Ouaknine
Computer Science Department, Carnegie Mellon University

James Worrell
Department of Mathematics, Tulane University

SVC seminar
Sept. 2, 2003

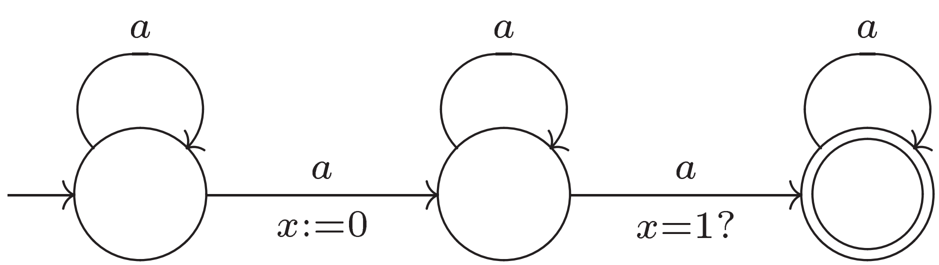
Timed Automata

- Standard modeling formalism for real-time.
- Finite-state automata with clocks.
- Timed trace semantics: sequences of events with real-valued delay timestamps. E.g., $u = \langle 0.3, a, 2, b, 0, c, 1.1, a \rangle$.



$L(A) :=$ set of timed traces accepted by A.

A :



Known Facts about Timed Automata

- Emptiness problem, $L(A) \stackrel{?}{=} \emptyset$: PSPACE-complete [Alur *et al.* 90]
- Universality problem, $L(A) \stackrel{?}{=} \mathbf{TT}$: Undecidable [Alur-Dill 94]
- Language inclusion problem, $L(B) \stackrel{?}{\subseteq} L(A)$: Undecidable [*Idem*]

Our Main Contribution

Theorem. If A has at most one clock, the language inclusion problem

$$L(B) \overset{?}{\subseteq} L(A)$$

is decidable.

Our Main Contribution

Theorem. If A has at most one clock, the language inclusion problem

$$L(B) \stackrel{?}{\subseteq} L(A)$$

is decidable.

This result is unexpected: in all other known computational models, deciding language inclusion always uses

$$L(B) \subseteq L(A) \iff L(B) \cap \overline{L(A)} = \emptyset.$$

However, one-clock timed automata cannot be complemented...

Timed Automata Language Inclusion: Related Work

- **Digitization techniques:**
  [Henzinger-Manna-Pnueli 92], [Bošnački 99],
  [Ouaknine-Worrell 03a]

- **Determinizable classes of timed automata:**
  [Alur-Fix-Henzinger 94], [Wilke 96], [Raskin 99]

- **Fuzzy semantics / noise-based techniques:**
  [Maass-Orponen 96], [Gupta-Henzinger-Jagadeesan 97],
  [Fränzle 99], [Henzinger-Raskin 00], [Puri 00],
  [Asarin-Bouajjani 01], [Ouaknine-Worrell 03b]

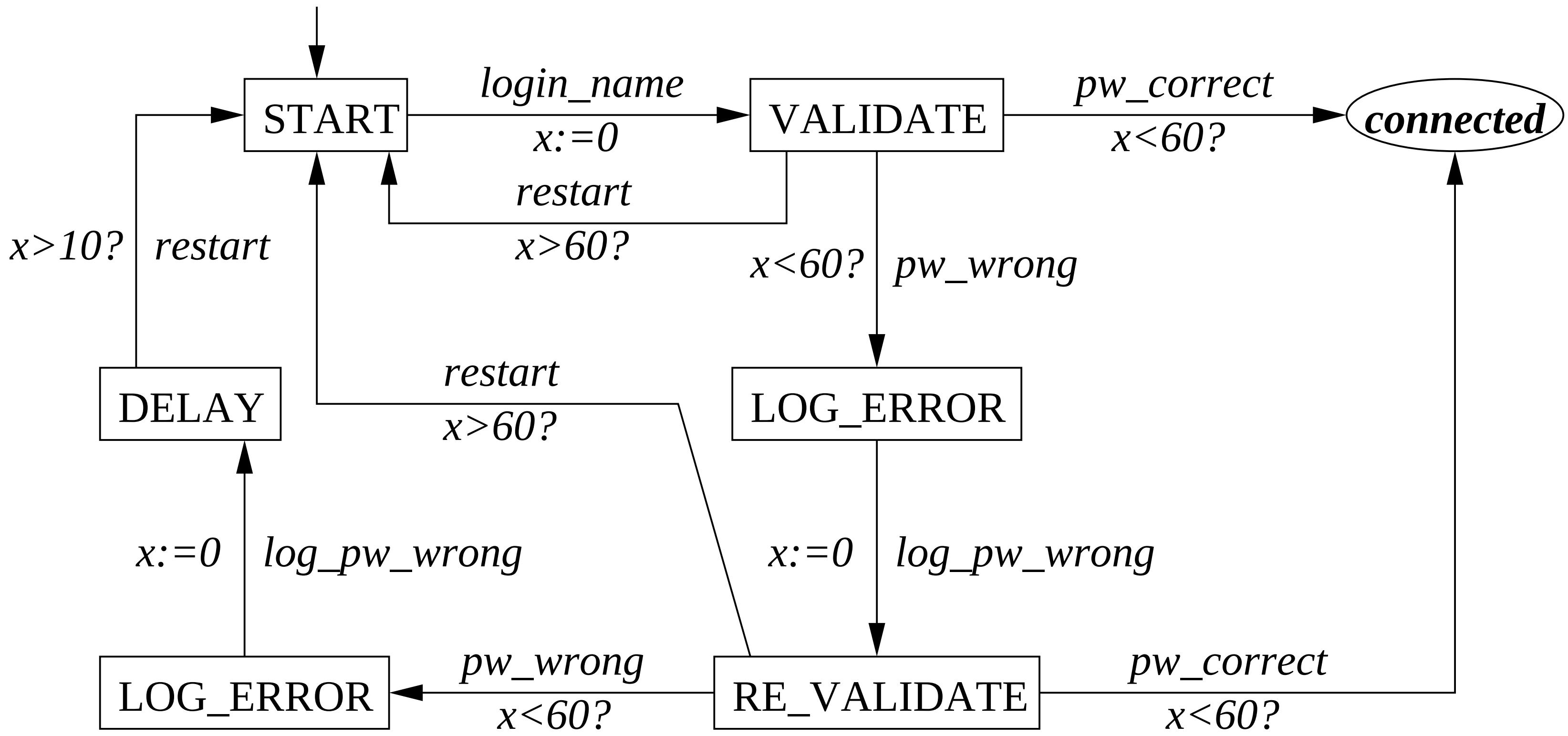
Some Applications

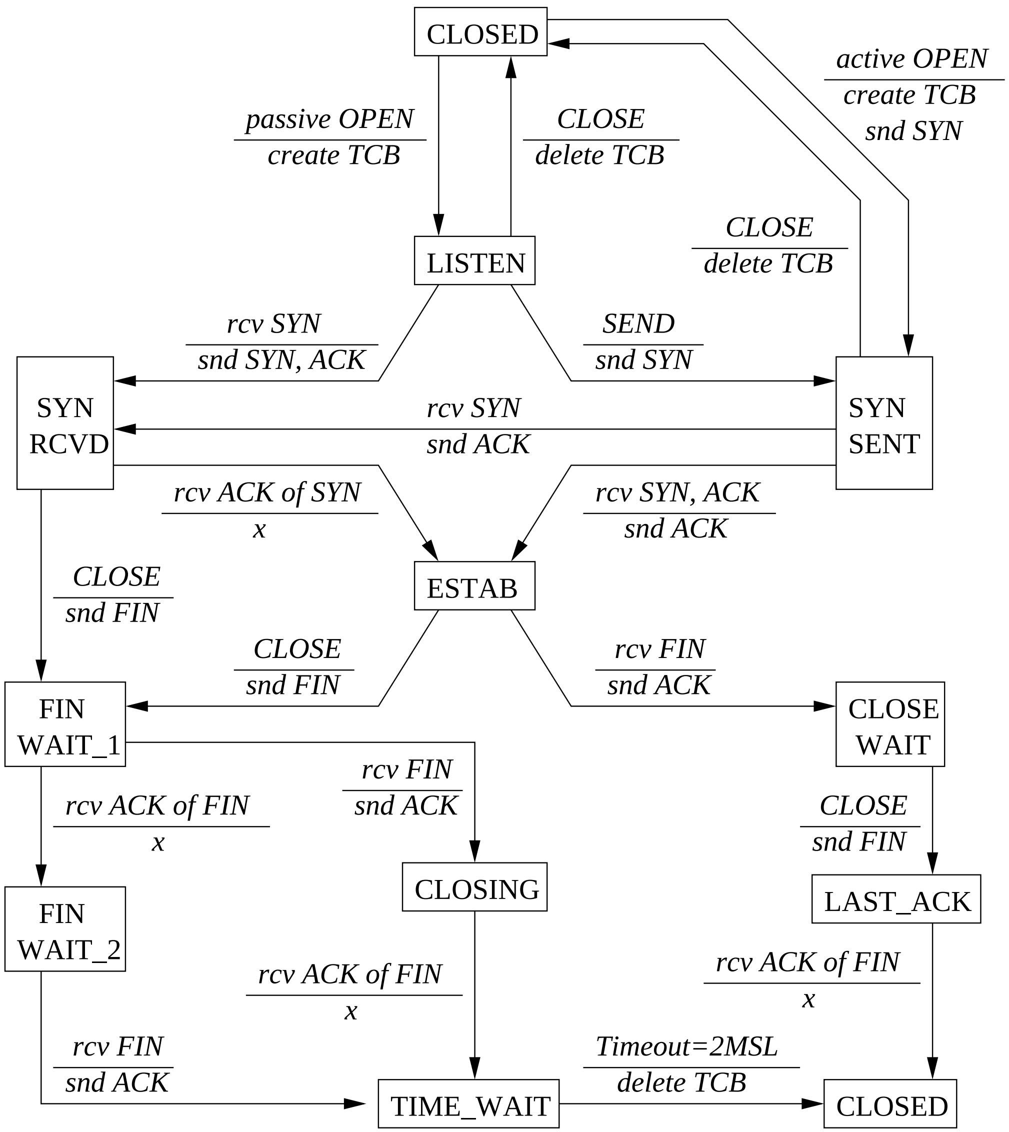
- Hardware and software systems are often described via high-level **specifications**, describing their intended functional behavior.
- Specifications are often given as **finite-state machines**.
  A proposed implementation *IMP* meets its specification *SPEC* iff

$$L(IMP) \subseteq L(SPEC).$$

- Our work enables us to describe and handle **timed** specifications: **timed automata with a single clock**.

Example: A Login Protocol





TCP spec. (DARPA) `www.faqs.org/rfcs/rfc793/html`

Some Applications (ctd.)

Compositional and assume-guarantee verification:

- Let $NET = B_1||B_2||\ldots||B_n$ be a network of timed automata.

Some Applications (ctd.)

Compositional and assume-guarantee verification:

- Let $NET = B_1||B_2||\ldots||B_n$ be a network of timed automata.
- Suppose you want to verify that $L(NET) = \emptyset$.

Some Applications (ctd.)

Compositional and assume-guarantee verification:

- Let $NET = B_1||B_2||\ldots||B_n$ be a network of timed automata.
- Suppose you want to verify that $L(NET) = \emptyset$.
- Find one-clock automata A_1, A_2, ... , A_n such that $L(B_i) \subseteq L(A_i)$.

Some Applications (ctd.)

Compositional and assume-guarantee verification:

- Let $NET = B_1||B_2||\ldots||B_n$ be a network of timed automata.
- Suppose you want to verify that $L(NET) = \emptyset$.
- Find one-clock automata A_1, A_2, ... , A_n such that $L(B_i) \subseteq L(A_i)$.
- Write $ABS = A_1||A_2||\ldots||A_n$.

Some Applications (ctd.)

Compositional and assume-guarantee verification:

- Let $NET = B_1 || B_2 || \ldots || B_n$ be a network of timed automata.
- Suppose you want to verify that $L(NET) = \emptyset$.
- Find one-clock automata A_1, A_2, ... , A_n such that $L(B_i) \subseteq L(A_i)$.
- Write $ABS = A_1 || A_2 || \ldots || A_n$.
- **Theorem:** If $L(ABS) = \emptyset$, then $L(NET) = \emptyset$.

Sketch of the Algorithm

- Reduce the language inclusion question $L(B) \stackrel{?}{\subseteq} L(A)$ to a **reachability** question on an infinite graph $\mathcal{H}$.

Sketch of the Algorithm

- Reduce the language inclusion question $L(B) \overset{?}{\subseteq} L(A)$ to a **reachability** question on an infinite graph $\mathcal{H}$.
- Construct a compatible **well-quasi-order** $\preccurlyeq$ on $\mathcal{H}$:
  - Whenever $W \preccurlyeq W'$: if W is safe, then W' is safe.
  - Any infinite sequence $W_1, W_2, W_3, \ldots$ eventually saturates: there exists $i < j$ such that $W_i \preccurlyeq W_j$.

Sketch of the Algorithm

- Reduce the language inclusion question $L(B) \stackrel{?}{\subseteq} L(A)$ to a **reachability** question on an infinite graph $\mathcal{H}$.
- Construct a compatible **well-quasi-order** $\preccurlyeq$ on $\mathcal{H}$:
  - Whenever $W \preccurlyeq W'$: if W is safe, then W' is safe.
  - Any infinite sequence $W_1, W_2, W_3, \ldots$ eventually saturates: there exists $i < j$ such that $W_i \preccurlyeq W_j$.
- Explore $\mathcal{H}$, looking for bad nodes. The search must eventually terminate.

Sketch of the Algorithm

- Reduce the language inclusion question $L(B) \overset{?}{\subseteq} L(A)$ to a **reachability** question on an infinite graph $\mathcal{H}$.
- Construct a compatible **well-quasi-order** $\preccurlyeq$ on $\mathcal{H}$:
  - Whenever $W \preccurlyeq W'$: if W is safe, then W' is safe.
  - Any infinite sequence $W_1, W_2, W_3, \ldots$ eventually saturates: there exists $i < j$ such that $W_i \preccurlyeq W_j$.
- Explore $\mathcal{H}$, looking for bad nodes. The search must eventually terminate.
- For simplicity, we will focus on **universality**: $A \overset{?}{=} \mathbf{TT}$.

Higman's Lemma

Let $\Lambda = \{a_1, a_2, \dots, a_n\}$ be an alphabet.
Let $\preccurlyeq$ be the **subword order** on Λ^*, the set of finite words over Λ.

Ex.: HIGMAN $\preccurlyeq$ HIGHMOUNTAIN

Then $\preccurlyeq$ is a **well-quasi-order** on Λ^*:

Any infinite sequence of words $W_1, W_2, W_3, \dots$ must eventually have two words $W_i \preccurlyeq W_j$, with $i < j$.

Higman's Lemma

Let $\Lambda = \{a_1, a_2, \ldots, a_n\}$ be an alphabet.
Let $\preccurlyeq$ be the **subword order** on Λ^*, the set of finite words over Λ.

Ex.: HIGMAN $\preccurlyeq$ HIGHMOUNTAIN

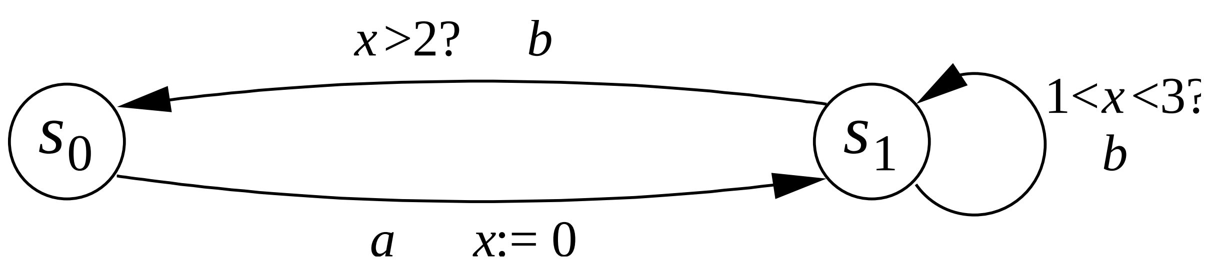
Then $\preccurlyeq$ is a **well-quasi-order** on Λ^*:

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words $W_i \preccurlyeq W_j$, with $i < j$.

Timed Automata Configurations

Let A be a timed automaton with a single clock x,
and discrete **locations** $s_0, s_1, \ldots, s_n$.

- A **state** of A is a pair (s, v):
  - s is a location.
  - $v \in \mathbb{R}^+$ is the value of clock x.
- A **configuration** of A is a finite set of states.

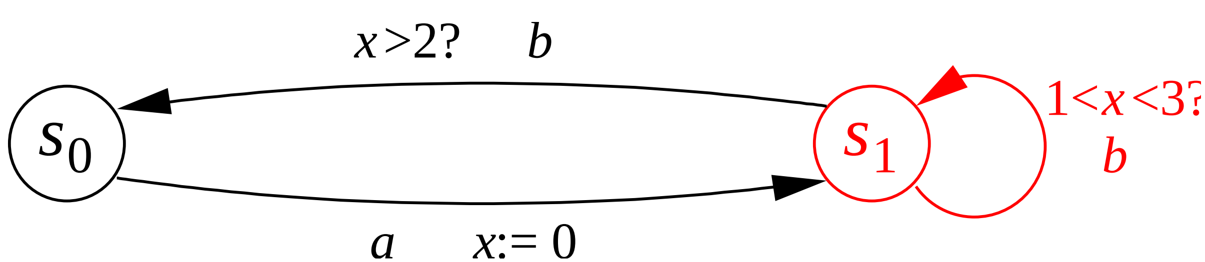


Timed Automata Configurations

Let A be a timed automaton with a single clock x, and discrete **locations** $s_0, s_1, \ldots, s_n$.

- A **state** of A is a pair (s, v):
  - s is a location.
  - $v \in \mathbb{R}^+$ is the value of clock x.
- A **configuration** of A is a finite set of states.

$(s_1, 1.4)$

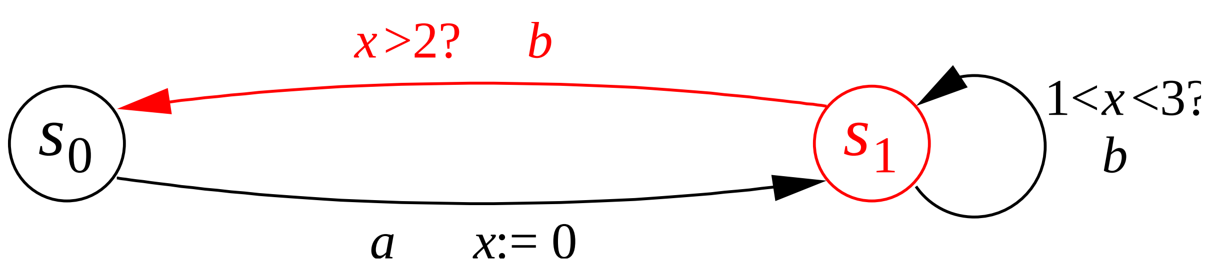


Timed Automata Configurations

Let A be a timed automaton with a single clock x, and discrete **locations** $s_0, s_1, \ldots, s_n$.

- A **state** of A is a pair (s, v):
  - s is a location.
  - $v \in \mathbb{R}^+$ is the value of clock x.
- A **configuration** of A is a finite set of states.

$(s_1, 4.5)$

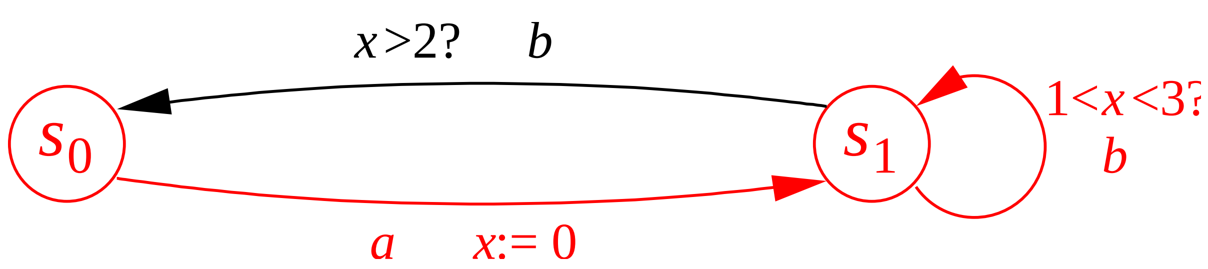


Timed Automata Configurations

Let A be a timed automaton with a single clock x, and discrete **locations** $s_0, s_1, \ldots, s_n$.

- A **state** of A is a pair (s, v):
  - s is a location.
  - $v \in \mathbb{R}^+$ is the value of clock x.
- A **configuration** of A is a finite set of states.

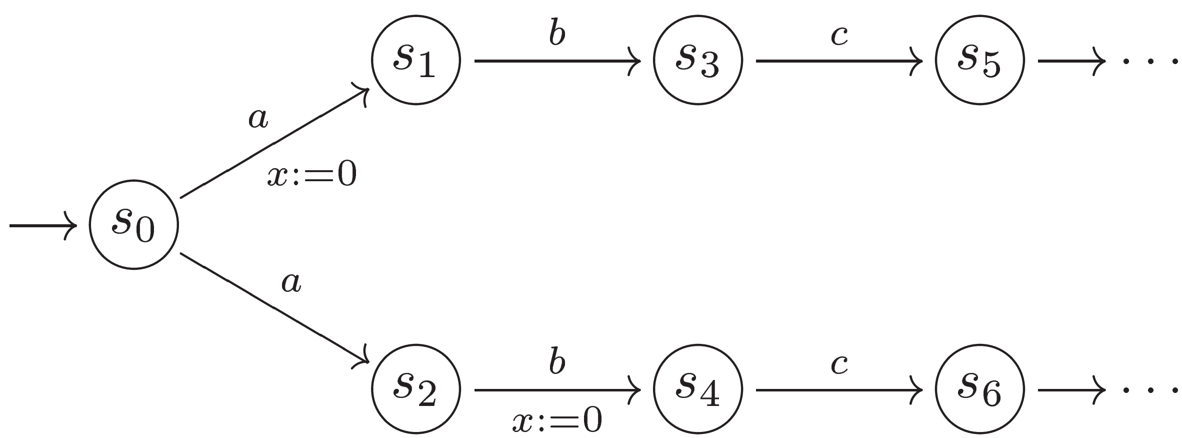
$\{(s_0, 2), (s_1, 1.4)\}$



Timed Automata Configurations (ctd.)

Every timed trace u gives rise to a configuration of A.

Ex.: $u = \langle 0.5, a, 0.2, b, 0.4, c \rangle$ leads to $\{(s_5, 0.6), (s_6, 0.4)\}$.



Bisimilar Configurations

If C is a configuration, let $A[C]$ be A 'started' in configuration C.

Definition. A relation $\mathcal{R}$ on configurations is a **bisimulation** if, whenever $C_1 \; \mathcal{R} \; C_2$, then

- $\forall a \in \Sigma$, $\forall t_1 \in \mathbb{R}^+$, $\exists t_2 \in \mathbb{R}^+$ such that if $A[C_1] \xrightarrow{t_1, a} A[C_1']$, then $A[C_2] \xrightarrow{t_2, a} A[C_2']$, and $C_1' \; \mathcal{R} \; C_2'$.
- Vice-versa.

Bisimilar Configurations

If C is a configuration, let $A[C]$ be A 'started' in configuration C.

Definition. A relation $\mathcal{R}$ on configurations is a **bisimulation** if, whenever $C_1 \; \mathcal{R} \; C_2$, then

- $\forall a \in \Sigma$, $\forall t_1 \in \mathbb{R}^+$, $\exists t_2 \in \mathbb{R}^+$ such that if $A[C_1] \stackrel{t_1,a}{\longrightarrow} A[C_1']$, then $A[C_2] \stackrel{t_2,a}{\longrightarrow} A[C_2']$, and $C_1' \; \mathcal{R} \; C_2'$.
- Vice-versa.

We say that C_1 and C_2 are **bisimilar**, written $C_1 \sim C_2$, if there exists some bisimulation relating them.

Bisimilar Configurations

If C is a configuration, let $A[C]$ be A 'started' in configuration C.

Definition. A relation $\mathcal{R}$ on configurations is a **bisimulation** if, whenever $C_1 \; \mathcal{R} \; C_2$, then

- $\forall a \in \Sigma$, $\forall t_1 \in \mathbb{R}^+$, $\exists t_2 \in \mathbb{R}^+$ such that if $A[C_1] \xrightarrow{t_1,a} A[C_1']$, then $A[C_2] \xrightarrow{t_2,a} A[C_2']$, and $C_1' \; \mathcal{R} \; C_2'$.
- Vice-versa.

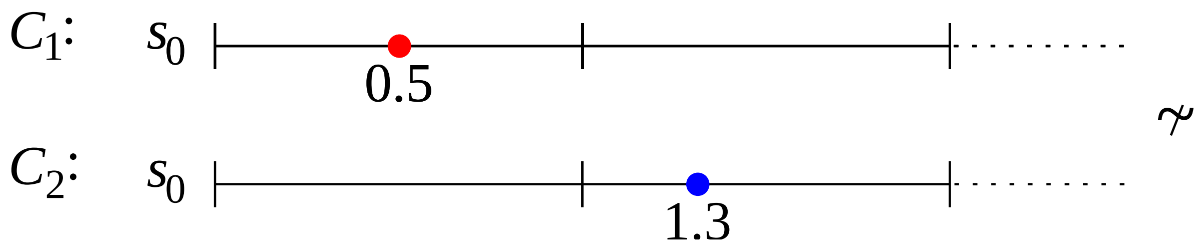
We say that C_1 and C_2 are **bisimilar**, written $C_1 \sim C_2$, if there exists some bisimulation relating them.

Theorem. If $C_1 \sim C_2$, then

$$A[C_1] \text{ is universal} \iff A[C_2] \text{ is universal.}$$

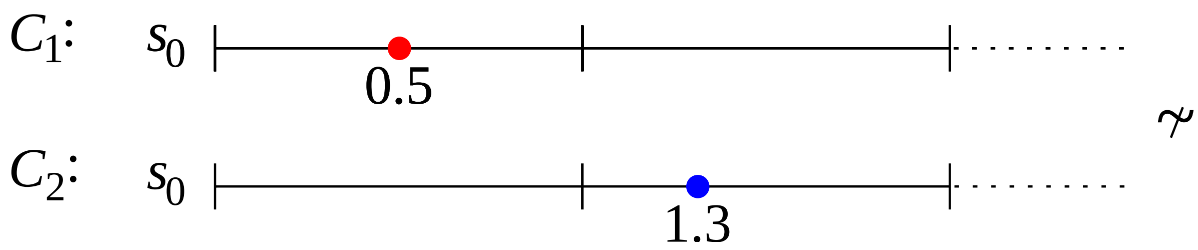
Bisimilar Configurations: Examples

$$C_1 = \{(s_0, 0.5)\} \not\approx C_2 = \{(s_0, 1.3)\}.$$

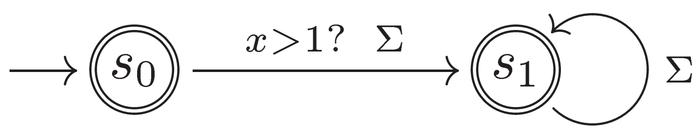


Bisimilar Configurations: Examples

$$C_1 = \{(s_0, 0.5)\} \not\approx C_2 = \{(s_0, 1.3)\}.$$

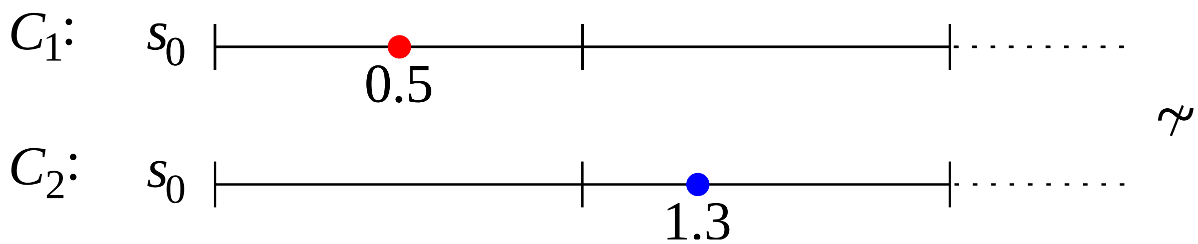


A :

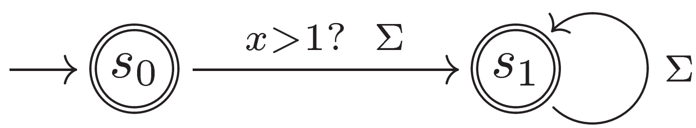


Bisimilar Configurations: Examples

$$C_1 = \{(s_0, 0.5)\} \not\sim C_2 = \{(s_0, 1.3)\}.$$

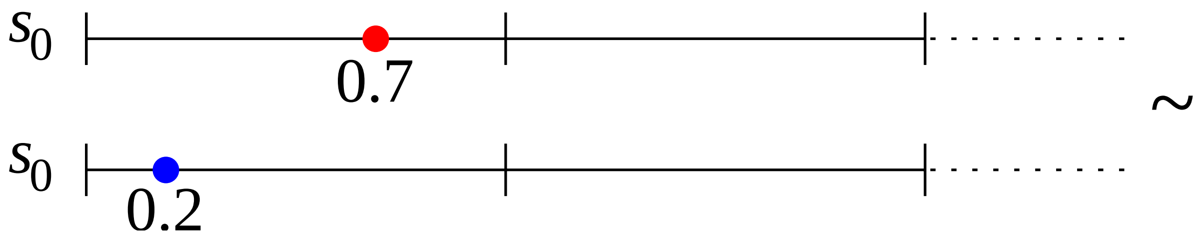


A :

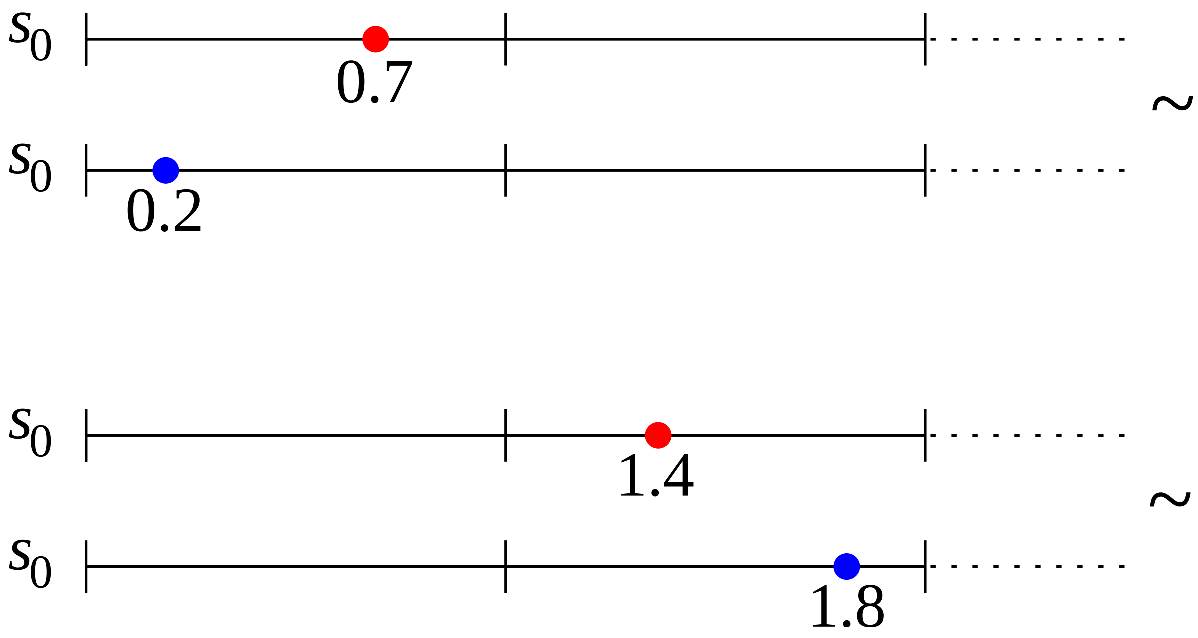


$A[C_2]$ is universal, but $A[C_1]$ rejects $\langle 0, a \rangle$.

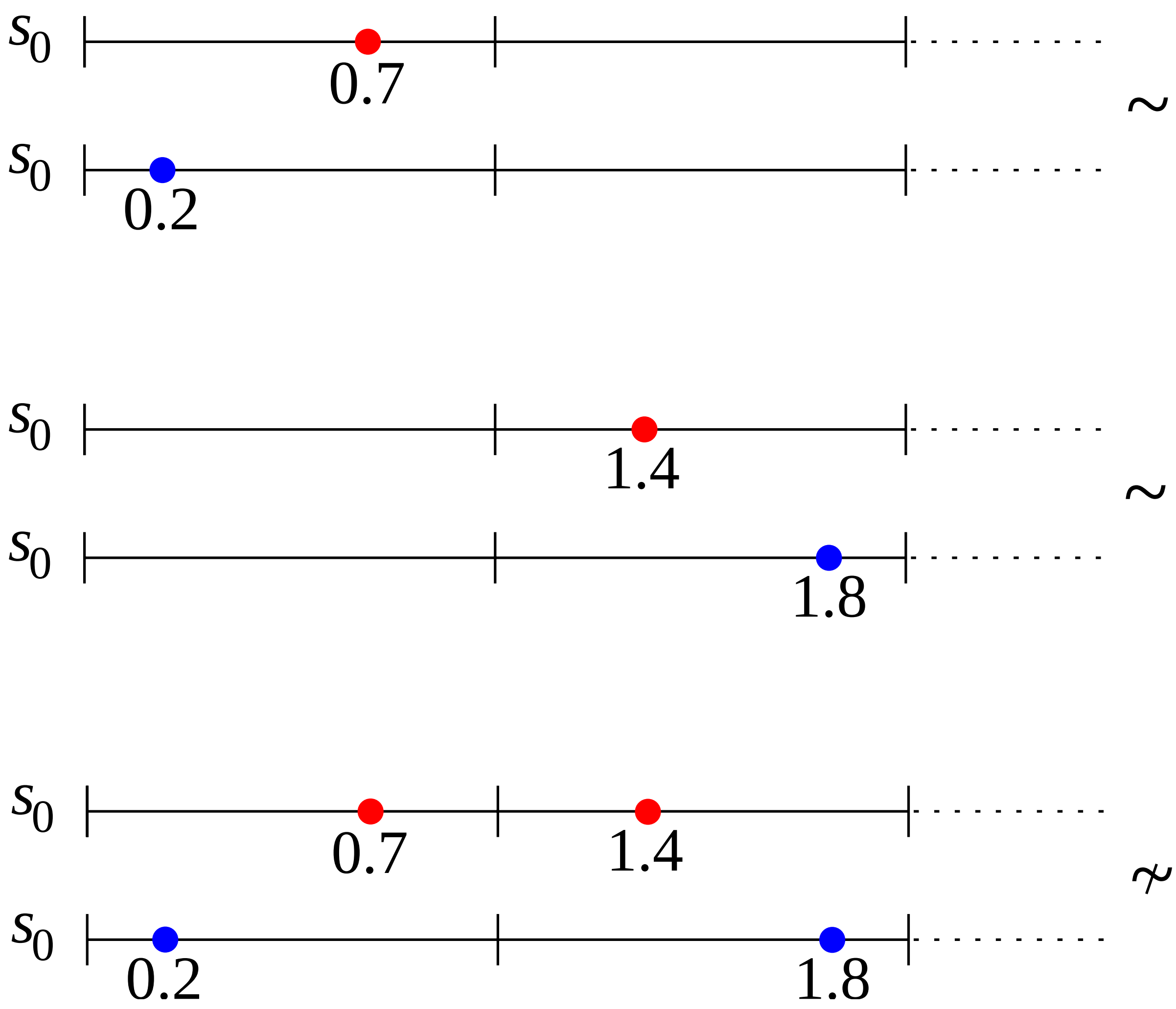
Bisimilar Configurations: Examples (ctd.)



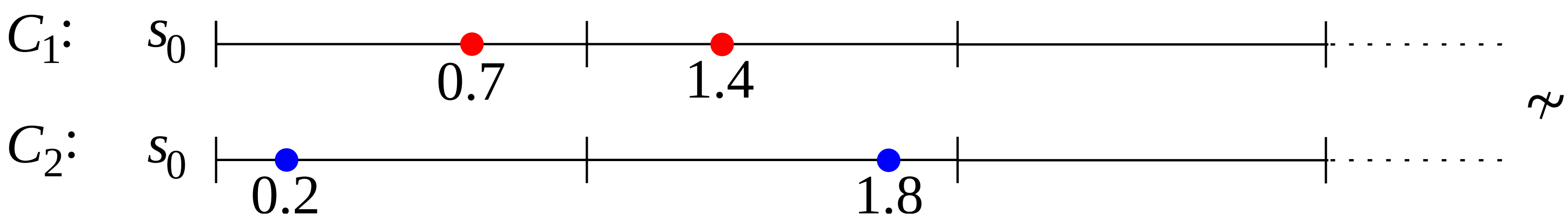
Bisimilar Configurations: Examples (ctd.)



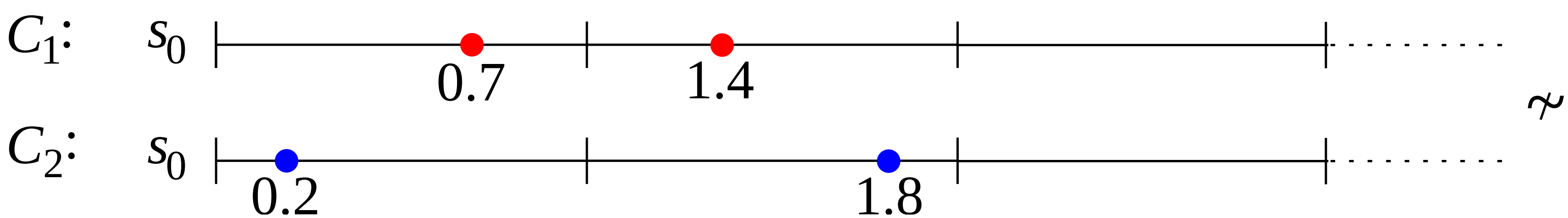
Bisimilar Configurations: Examples (ctd.)

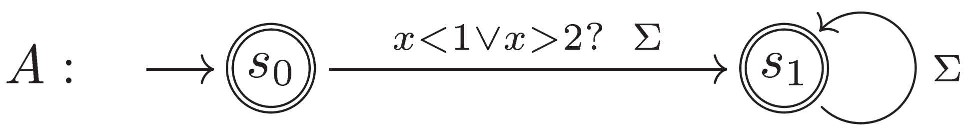


Bisimilar Configurations: Examples (ctd.)

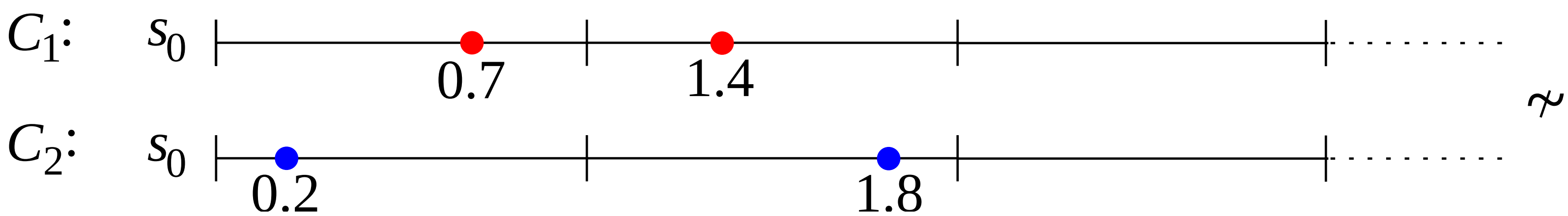


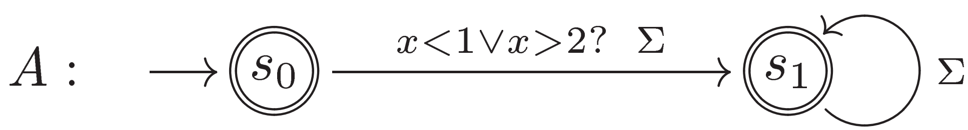
Bisimilar Configurations: Examples (ctd.)





Bisimilar Configurations: Examples (ctd.)





$A[C_2]$ is universal, but $A[C_1]$ rejects $\langle 0.5, a\rangle$.

From Bisimulation to Simulation

Definition. We say that C_1 is **simulated** by C_2, written $C_1 \preccurlyeq C_2$, if there exists $C_2' \subseteq C_2$ such that $C_1 \sim C_2'$.

From Bisimulation to Simulation

Definition. We say that C_1 is **simulated** by C_2, written $C_1 \preccurlyeq C_2$, if there exists $C_2' \subseteq C_2$ such that $C_1 \sim C_2'$.

Theorem. If $C_1 \preccurlyeq C_2$, then

$$A[C_1] \text{ is universal } \implies A[C_2] \text{ is universal.}$$

Constructing a Decidable Bisimulation Relation

Let $K \in \mathbb{N}$ be the largest constant appearing in clock constraints of A.

Theorem. Let C and C' be configurations of A.
If there exists a bijection $f : C \to C'$ that preserves

- locations: $f(s, v) = (s', v') \implies s = s'$,
- integer parts of clock x, up to K:
  $f(s, v) = (s', v') \implies ((\lceil v \rceil = \lceil v' \rceil \ \wedge \ \lfloor v \rfloor = \lfloor v' \rfloor) \ \vee \ v, v' > K)$,
- the ordering of the fractional parts of clock x:
  $f(s_i, v_i) = (s'_i, v'_i) \implies (v_i < v_j \iff v'_i < v'_j)$,

then $C \sim C'$.

Constructing a Decidable Bisimulation Relation (ctd.)

- Let K be the largest constant appearing in clock constraints of A.
- Let $REG = \Big\{\{0\}, (0,1), \{1\}, (1,2), \ldots, \{K\}, (K, \infty)\Big\}$ be the collection of 'one-dimensional regions' of A.
- Let $S = \{s_0, s_1, \ldots, s_n\}$ be the set of locations of A.
- Let $\Lambda = S \times REG$.
- Let C be a configuration of A. For simplicity, assume all the fractional parts of states in C are distinct.
- Note that each state in C has a unique matching letter in Λ.
- Encode C as a word $H(C) \in \Lambda^*$, ordered by increasing fractional parts of states.

Constructing a Decidable Bisimulation Relation (ctd.)

Corollary. For any configurations C and C' of A,

$$H(C) = H(C') \implies C \sim C'.$$

Constructing a Decidable Bisimulation Relation (ctd.)

Corollary. For any configurations C and C' of A,

$$H(C) = H(C') \implies C \sim C'.$$

Theorem. For any configurations C and C' of A,

$$H(C) \preccurlyeq H(C') \implies C \preccurlyeq C'.$$

Constructing a Decidable Bisimulation Relation (ctd.)

Corollary. For any configurations C and C' of A,

$$H(C) = H(C') \implies C \sim C'.$$

Theorem. For any configurations C and C' of A,

$$H(C) \preccurlyeq H(C') \implies C \preccurlyeq C'.$$

Corollary. For configurations C and C' of A, if $H(C) \preccurlyeq H(C')$, then

$$A[C] \text{ is universal} \implies A[C'] \text{ is universal}.$$

The Algorithm: Recapitulation

- Reduce the universality question $L(A) \stackrel{?}{=} \mathbf{TT}$ to a reachability question on the infinite graph Λ^*.
- The subword order $\preccurlyeq$ on Λ^* is a compatible well-quasi-order:
  - Whenever $H(C) \preccurlyeq H(C')$:
    if $A[C]$ is universal, then $A[C']$ is universal.
  - Any infinite sequence $H(C_1), H(C_2), H(C_3), \ldots$ eventually saturates: there exists $i < j$ such that $H(C_i) \preccurlyeq H(C_j)$.
- Explore Λ^*, looking for a word/configuration from which A cannot perform some event. The search must eventually terminate.

Summary

- We have given an algorithm to decide the language inclusion question $L(B) \stackrel{?}{\subseteq} L(A)$, provided A has at most one clock.
- Unexpected and fundamentally new result in the theory of timed automata.
- Interesting potential applications to hardware/software engineering and verification.

Closing the Gap

Our decidability result is for all practical purposes the tightest one can get in terms of restricting the resources of timed automata:

Theorem. For A a timed automaton, the universality question $L(A) \stackrel{?}{=} \mathbf{TT}$ remains undecidable under any of the following restrictions:

- A has two clocks and a one-event alphabet, **or**
- A has two clocks and uses a single constant in clock constraints, **or**
- A has a single location and a one-event alphabet, **or**
- A has a single location and uses a single constant in clock constraints.

Future Work

- Complexity of the algorithm.
- Extend (if possible) to Büchi timed automata.
- Efficient implementation.