The Dark Side of Timed Opacity

Franck Cassez
http://www.irccyn.fr/franck

National ICT Australia & CNRS

Work supported by a Marie Curie International Outgoing Fellowship

7th European Community Framework Programme

ISA 2009, Seoul, Korea

June 25th, 2009

Context

Need for Security in Transactional Systems

- Web-services: e-banking, online transactions
- id documents: biometric passport, Medicare Card
- e-voting systems

Different Types of Security

- Integrity: illegal actions cannot be performed by an unauthorized user
  Bank account management cannot be managed by a third party
- Availability: some actions must be available
  Withdrawing money from your bank account
- Privacy: information should remain hidden from some users
  PIN code

introduced in [Mazaré (WITS'2004), Bryans et al. (FAST'2005)]

Context

Need for Security in Transactional Systems

- Web-services: e-banking, online transactions
- id documents: biometric passport, Medicare Card
- e-voting systems

Different Types of Security

- Integrity: illegal actions cannot be performed by an unauthorized user
  Bank account management cannot be managed by a third party
- Availability: some actions must be available
  Withdrawing money from your bank account
- Privacy: information should remain hidden from some users
  PIN code

In this paper: Opacity

introduced in [Mazaré (WITS'2004), Bryans et al. (FAST'2005)]

Context

Need for Security in Transactional Systems

- Web-services: e-banking, online transactions
- id documents: biometric passport, Medicare Card
- e-voting systems

Different Types of Security

- Integrity: illegal actions cannot be performed by an unauthorized user
  Bank account management cannot be managed by a third party
- Availability: some actions must be available
  Withdrawing money from your bank account
- Privacy: information should remain hidden from some users
  PIN code

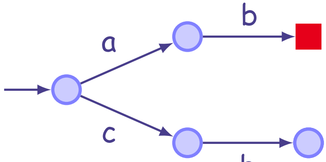
In this paper: Opacity

introduced in [Mazaré (WITS'2004), Bryans et al. (FAST'2005)]

Formal Specification and Verification of Opacity

System S

Secret F



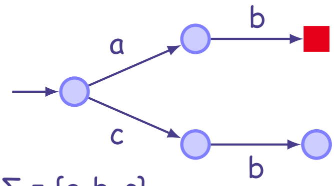
$\Sigma = \{a, b, c\}$

Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

Opacity: an external observer should never know F-states

Formal Specification and Verification of Opacity

System S



$\Sigma = \{a, b, c\}$

Secret F

Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

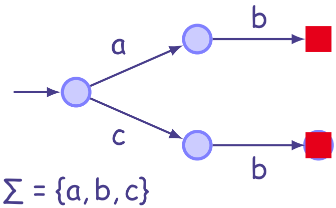
Secret F is opaque

Opacity: an external observer should never know F-states

Formal Specification and Verification of Opacity

System S

Secret F



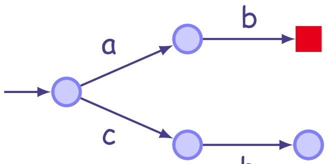
Secret = set of states
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

Secret F is not opaque

Opacity: an external observer should never know F-states

Formal Specification and Verification of Opacity

System S



$\Sigma = \{a, b, c\}$

Secret F

Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{a, b\}$

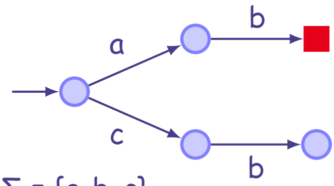
Secret F is not opaque

Opacity: an external observer should never know F-states

Formal Specification and Verification of Opacity

System S

Secret F



$\Sigma = \{a, b, c\}$

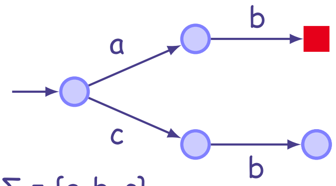
Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{a, b\}$

Opacity Verification Problem: Is F opaque wrt (S, Σ_o) ?

Formal Specification and Verification of Opacity

System S

Secret F



$\Sigma = \{a, b, c\}$

Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{a, b\}$

Opacity Verification Problem: Is F opaque wrt (S, Σ_o) ?

To check opacity: use your favorite Formal Method:

- Model-checking
- Theorem proving
- Tools to support automatic analysis of systems

Results for Checking Opacity of Finite Systems

Inputs:

- S is finite automaton over alphabet Σ
- $\Sigma_o \subseteq \Sigma$, set of observable events
- a secret F, given by a subset of the set of states of S

Theorem ([Cassez et al. (ATVA'09)])

Checking wether F is opaque wrt (S, Σ_o) is PSPACE-complete.

What if an external observer can measure time ?

Results for Checking Opacity of Finite Systems

Inputs:

- S is finite automaton over alphabet Σ
- $\Sigma_o \subseteq \Sigma$, set of observable events
- a secret F, given by a subset of the set of states of S

Theorem ([Cassez et al. (ATVA'09)])

Checking wether F is opaque wrt (S, Σ_o) is PSPACE-complete.

What if an external observer can measure time ?

Opacity for Timed Systems

Inputs:

- S is timed automaton over alphabet Σ
- $\Sigma_o \subseteq \Sigma$, set of observable events
- a secret F, given by a subset of the set of S

Secret = ■
b observable + time

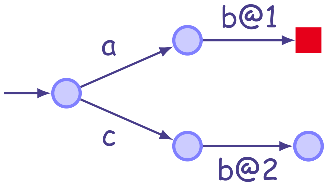
System is not opaque

This paper: checking opacity for timed systems

Opacity for Timed Systems

Inputs:

- S is timed automaton over alphabet Σ
- $\Sigma_o \subseteq \Sigma$, set of observable events
- a secret F, given by a subset of the set of S



Secret =
b observable + time

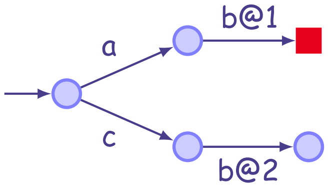
System is not opaque

This paper: checking opacity for timed systems

Opacity for Timed Systems

Inputs:

- S is timed automaton over alphabet Σ
- $\Sigma_o \subseteq \Sigma$, set of observable events
- a secret F, given by a subset of the set of S



Secret = ■
b observable + time

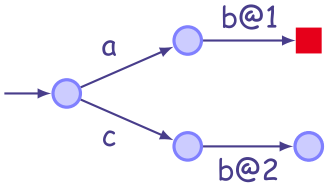
System is not opaque

This paper: checking opacity for timed systems

Opacity for Timed Systems

Inputs:

- S is timed automaton over alphabet Σ
- $\Sigma_o \subseteq \Sigma$, set of observable events
- a secret F, given by a subset of the set of S



Secret = ■
b observable + time

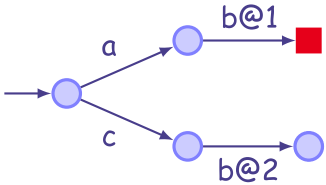
System is not opaque

This paper: checking opacity for timed systems

Opacity for Timed Systems

Inputs:

- S is timed automaton over alphabet Σ
- $\Sigma_o \subseteq \Sigma$, set of observable events
- a secret F, given by a subset of the set of S



Secret = ■
b observable + time

System is not opaque

This paper: checking opacity for timed systems

Outline of the Talk

- Modelling Timed Systems
  - Timed Words and Languages
  - Timed Automata
  - Verification of Timed Automata
- Timed Opacity
  - Timed Opacity Problem
  - Timed Opacity is Undecidable for TA
- Conclusion

Timed Words and Languages

A finite timed word over Σ is a word in $(\Sigma \times \mathbb{R}_{\geq 0})^*$
(a, 1)(c, 2.34)(a, 2.986)(b, 3.146)(c, 4.16)

$TW^*(\Sigma)$ = set of timed words over Σ

Operations on timed words

- untiming: Unt(a, 1)(c, 2.34)(a, 2.986)(b, 3.146)(c, 4.16) = a.c.a.b.c
- Projection:
  $\pi_{\{a,b\}}((a, 1)(c, 2.34)(a, 2.986)(b, 3.146)(c, 4.16)) =$
  (a, 1)(a, 2.986)(b, 3.146)
- Inverse Projection: $\pi^{-1}_{\Sigma}(w) = \{w' \in TW^*(\Sigma) \mid \pi_{\Sigma'}(w') = w\}$

A timed language is a set of timed words

Operations on timed words extend to timed languages

Timed Words and Languages

A finite timed word over Σ is a word in $(\Sigma \times \mathbb{R}_{\geq 0})^*$
$(a,1)(c,2.34)(a,2.986)(b,3.146)(c,4.16)$

$TW^*(\Sigma)$ = set of timed words over Σ

Operations on timed words

- untiming: $Unt(a,1)(c,2.34)(a,2.986)(b,3.146)(c,4.16) = a.c.a.b.c$
- Projection:
  $\pi_{\{a,b\}}((a,1)(c,2.34)(a,2.986)(b,3.146)(c,4.16)) =$
  $(a,1)(a,2.986)(b,3.146)$
- Inverse Projection: $\pi^{-1}_{\Sigma}(w) = \{w' \in TW^*(\Sigma) \mid \pi_{\Sigma'}(w') = w\}$

A timed language is a set of timed words

Operations on timed words extend to timed languages

Timed Words and Languages

A finite timed word over Σ is a word in $(\Sigma \times \mathbb{R}_{\geq 0})^*$
$(a, 1)(c, 2.34)(a, 2.986)(b, 3.146)(c, 4.16)$

$TW^*(\Sigma)$ = set of timed words over Σ

Operations on timed words

- untiming: $Unt(a, 1)(c, 2.34)(a, 2.986)(b, 3.146)(c, 4.16) = a.c.a.b.c$
- Projection:
  $\pi_{\{a,b\}}((a, 1)(c, 2.34)(a, 2.986)(b, 3.146)(c, 4.16)) =$
  $(a, 1)(a, 2.986)(b, 3.146)$
- Inverse Projection: $\pi_{\Sigma}^{-1}(w) = \{w' \in TW^*(\Sigma) \mid \pi_{\Sigma'}(w') = w\}$

A timed language is a set of timed words

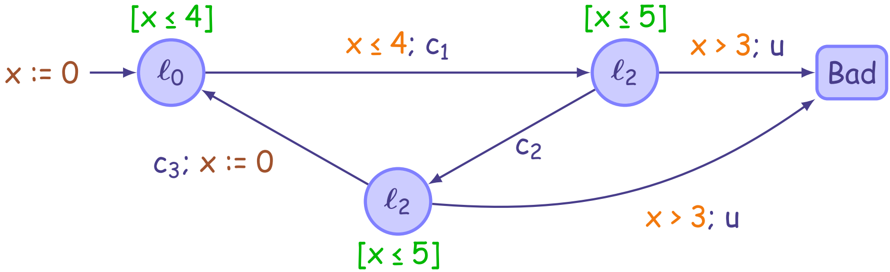
Operations on timed words extend to timed languages

Timed Automata [Alur and Dill (TCS 94)]

- Timed Automaton = Finite Automaton + clock variables
  All clocks evolve at the same speed
- Clocks take their values in a dense-time domain
- Transitions are guarded by clocks constraints

- g: guard of the form $g ::= x \sim c \mid g \wedge g$
  where x is a clock and $c \in \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$
- R : the set of clocks to be reset when firing the transition
- Inv(ℓ) is an invariant to ensure (some sort of) liveness

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

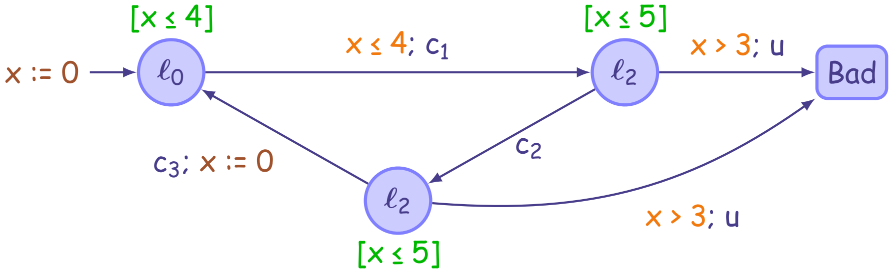
ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

ρ_3 : $(\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



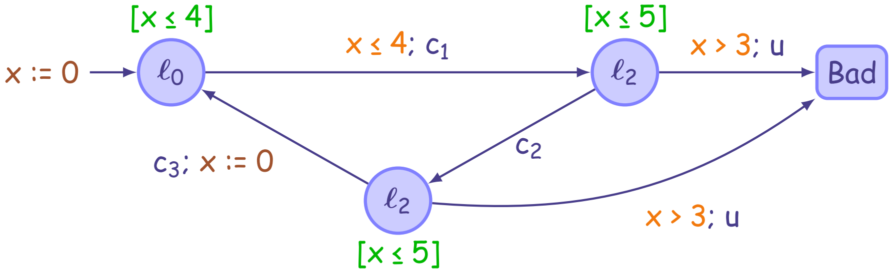
Runs = alternating sequence of discrete and time steps

$\rho_1: \quad (\ell_0,0) \xrightarrow{1.55} (\ell_0,1.55) \xrightarrow{c_1} (\ell_1,1.55) \xrightarrow{1.67} (\ell_1,3.22) \xrightarrow{u} (Bad,3.22)$

$\rho_2: \quad (\ell_0,0) \xrightarrow{1.1} (\ell_0,1.1) \xrightarrow{c_1} (\ell_1,1.1) \xrightarrow{2.1} (\ell_1,3.2) \xrightarrow{c_2} (\ell_2,3.2)$
$\xrightarrow{0.1} (\ell_2,3.3) \xrightarrow{c_3} (\ell_0,0) \quad \cdots\cdots\cdots$

$\rho_3: \quad (\ell_0,0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{2}} (\ell_0,0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{4}} (\ell_0,0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

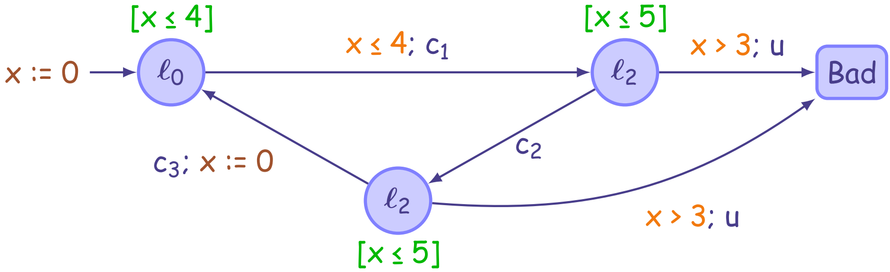
$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



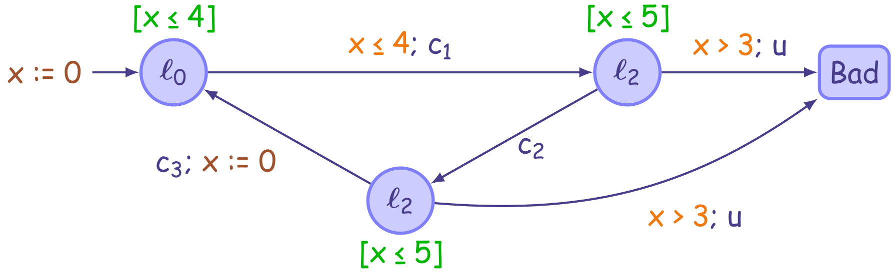
Runs = alternating sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0)$ ·········

ρ_3 : $(\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

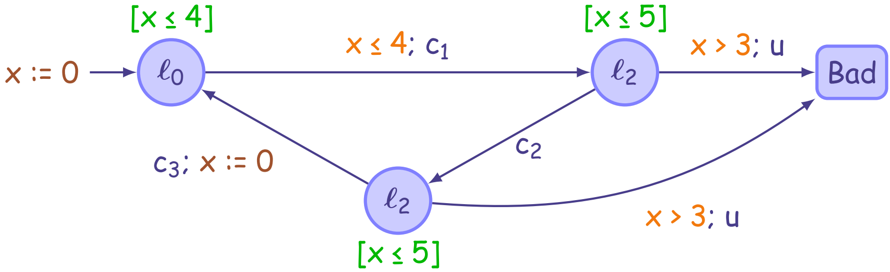
$$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$$

$$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$$

$$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$$

$$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

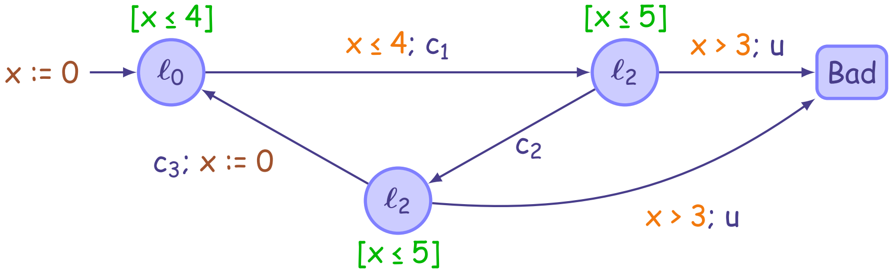
ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0)$ ·········

ρ_3 : $(\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



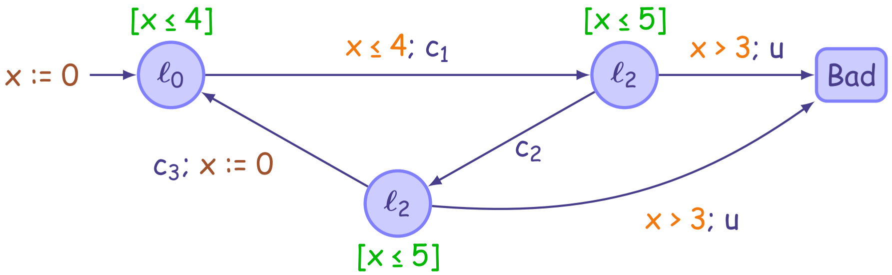
Runs = alternating sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0)$ ·········

ρ_3 : $(\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

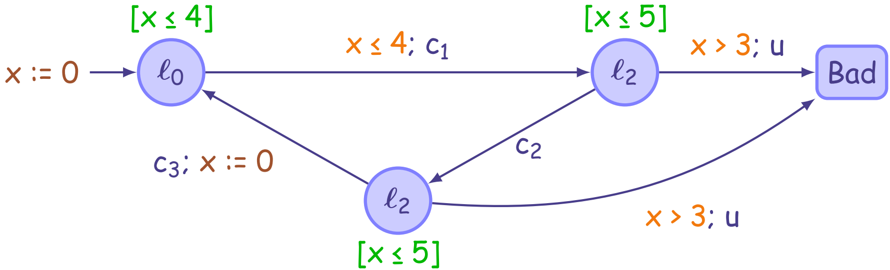
ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0)$ ....

ρ_3 : $(\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

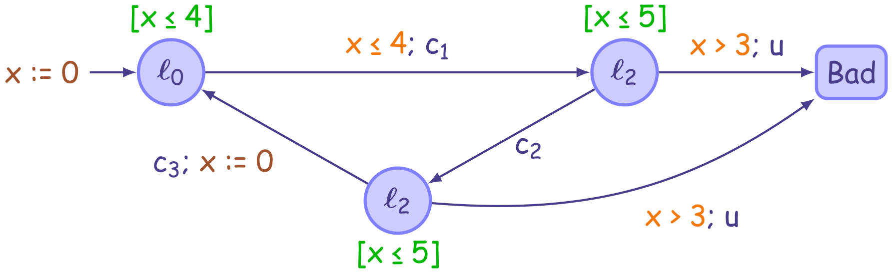
$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



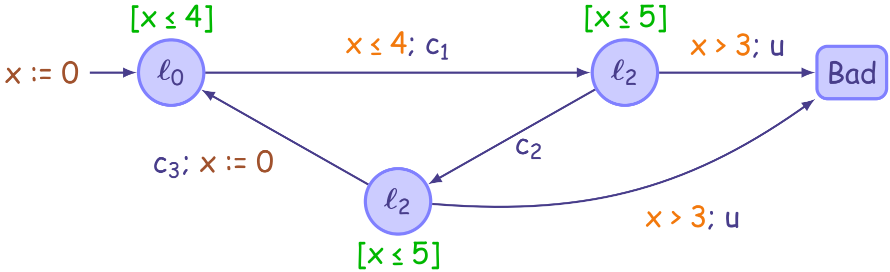
Runs = alternating sequence of discrete and time steps

$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

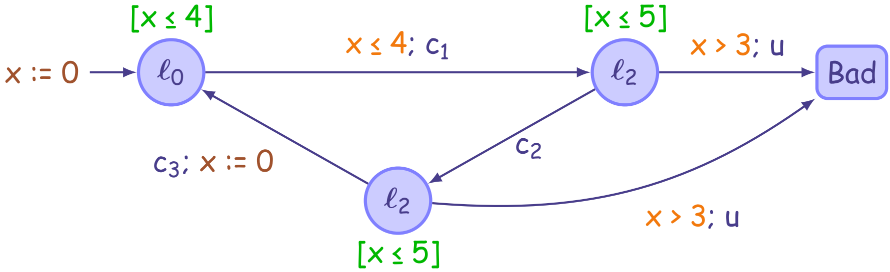
$\rho_1: \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

$\rho_2: \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3: \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

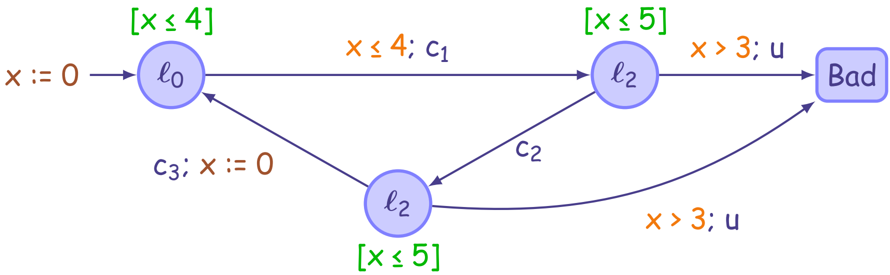
ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0)$ $\cdots\cdots\cdots$

ρ_3 : $(\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



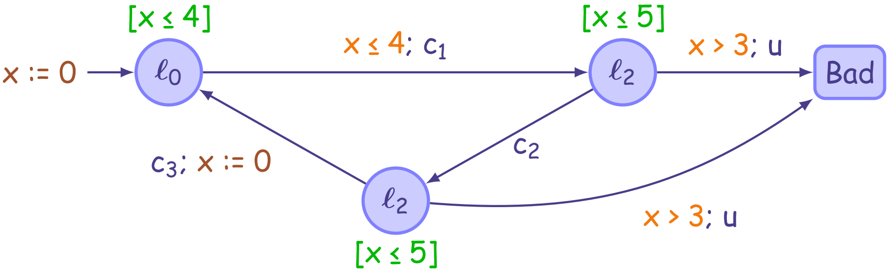
Runs = alternating sequence of discrete and time steps

$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



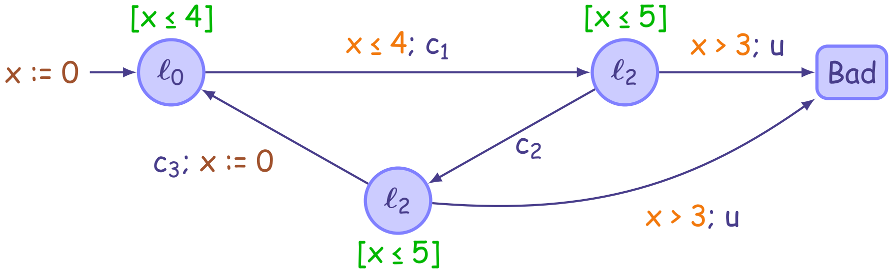
Runs = alternating sequence of discrete and time steps

$\rho_1: \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

$\rho_2: \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3: \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

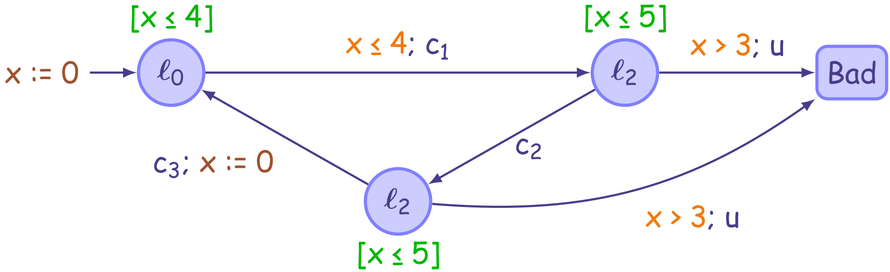
$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



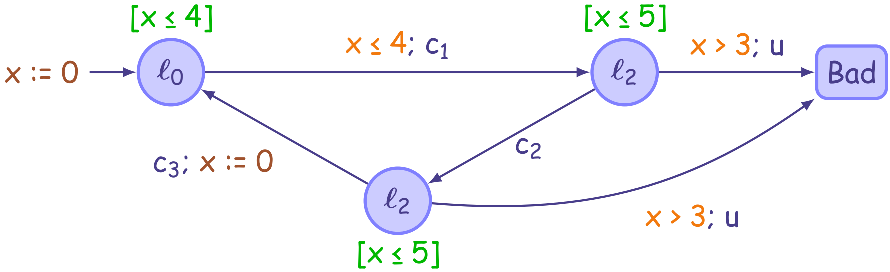
Runs = alternating sequence of discrete and time steps

$$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$$

$$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2) \xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$$

$$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

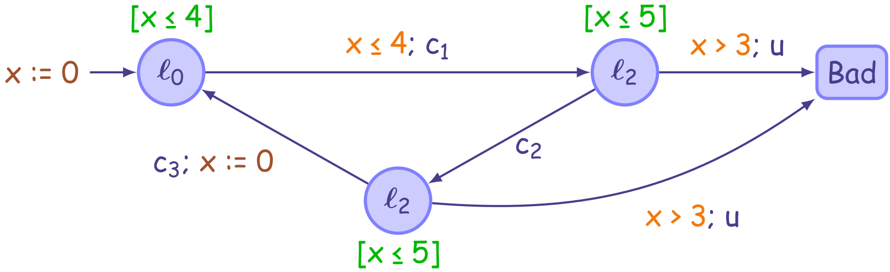
ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0)$ ....

ρ_3 : $(\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



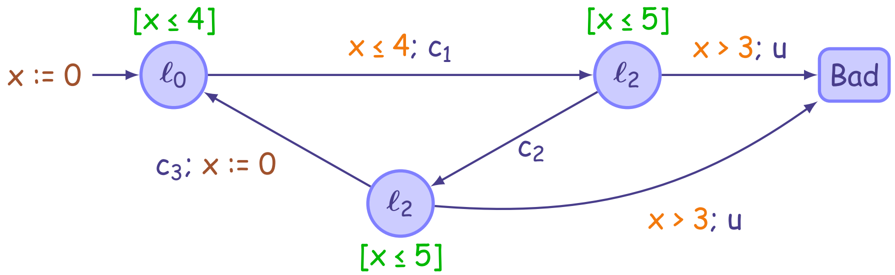
Runs = alternating sequence of discrete and time steps

$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$

$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1c_2c_3 \text{ in } \frac{1}{8}} \cdots$

Example 1: Timed Automaton



Runs = alternating sequence of discrete and time steps

$$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$$

$$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$$
$$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \quad \cdots\cdots\cdots$$

$$\rho_3 : \quad (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{2}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{4}} (\ell_0, 0) \xrightarrow{c_1 c_2 c_3 \text{ in } \frac{1}{8}} \cdots$$

Languages Generated by Timed Automata

A Timed Automaton A is a tuple $(L, \ell_0, X, \Sigma_\tau, E, F)$

$\Sigma_\tau = \Sigma \cup \{\tau\}$, τ = invisible/silent
F = subset of L, accepting locations

A run ϱ of A is a sequence of the form:

$$\begin{aligned}\varrho \quad = \quad & (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_0} (\ell_1, v_1) \cdots \\ & \cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n)\end{aligned}$$

$tr(\varrho)$ is the trace of ϱ which is the timed word

$$\pi_\Sigma((a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)) \; with \; t_i = \sum_{k=0}^{i} \delta_k$$

$Tr(A)$ = set of traces of words generated by A
w is accepted by A if $w = tr(\varrho)$ with $last(\varrho) \in F$
$\mathcal{L}(A) \subseteq Tr(A)$ is the set timed words accepted by A.

Languages Generated by Timed Automata

A Timed Automaton A is a tuple $(L, \ell_0, X, \Sigma_\tau, E, F)$

$\Sigma_\tau = \Sigma \cup \{\tau\}$, τ = invisible/silent
F = subset of L, accepting locations

A run ϱ of A is a sequence of the form:

$$\varrho = (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_0} (\ell_1, v_1) \cdots$$
$$\cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n)$$

$tr(\varrho)$ is the trace of ϱ which is the timed word

$$\pi_\Sigma\big((a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)\big) \; \textit{with } t_i = \sum_{k=0}^{i} \delta_k$$

$Tr(A)$ = set of traces of words generated by A
w is accepted by A if $w = tr(\varrho)$ with $last(\varrho) \in F$
$\mathcal{L}(A) \subseteq Tr(A)$ is the set timed words accepted by A.

Languages Generated by Timed Automata

A Timed Automaton A is a tuple $(L, \ell_0, X, \Sigma_\tau, E, F)$

$\Sigma_\tau = \Sigma \cup \{\tau\}$, τ = invisible/silent

F = subset of L, accepting locations

A run ϱ of A is a sequence of the form:

$$\begin{aligned}\varrho \quad = \quad & (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_0} (\ell_1, v_1) \cdots \\ & \cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n)\end{aligned}$$

$tr(\varrho)$ is the trace of ϱ which is the timed word

$$\pi_\Sigma\big((a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)\big) \textit{ with } t_i = \sum_{k=0}^{i} \delta_k$$

$Tr(A)$ = set of traces of words generated by A

w is accepted by A if $w = tr(\varrho)$ with $last(\varrho) \in F$

$\mathcal{L}(A) \subseteq Tr(A)$ is the set timed words accepted by A.

Languages Generated by Timed Automata

A Timed Automaton A is a tuple $(L, \ell_0, X, \Sigma_\tau, E, F)$

- $\Sigma_\tau = \Sigma \cup \{\tau\}$, τ = invisible/silent
- F = subset of L, accepting locations

A run ϱ of A is a sequence of the form:

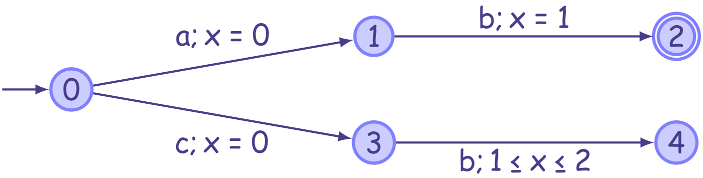
$$\begin{aligned} \varrho \quad = \quad & (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_0} (\ell_1, v_1) \cdots \\ & \cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n) \end{aligned}$$

$tr(\varrho)$ is the trace of ϱ which is the timed word

$$\pi_\Sigma\big((a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)\big) \textit{ with } t_i = \sum_{k=0}^{i} \delta_k$$

$Tr(A)$ = set of traces of words generated by A
w is accepted by A if $w = tr(\varrho)$ with $last(\varrho) \in F$
$\mathcal{L}(A) \subseteq Tr(A)$ is the set timed words accepted by A.

Timed Language Accepted by a TA (Example 2)



$\mathcal{B}$ can generate the following runs: for $\delta_1 \geq 0$ and $1 \leq \delta_2 \leq 2$

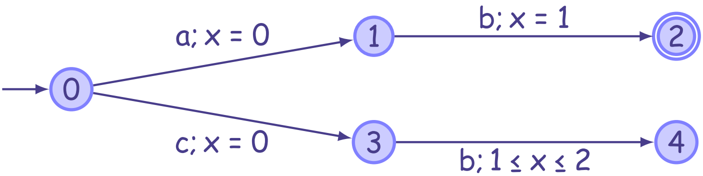
$$(0, x = 0) \xrightarrow{a} (1, x = 0) \xrightarrow{1} (1, x = 1) \xrightarrow{b} (2, x = 1) \xrightarrow{\delta_1} (2, x = 1 + \delta_1)$$

and

$$(0, x = 0) \xrightarrow{c} (3, x = 0) \xrightarrow{\delta_2} (3, x = \delta_2 2) \xrightarrow{b} (4, x = \delta_2) \xrightarrow{\delta_1} (4, x = \delta_2 + \delta_1)$$

$Tr(\mathcal{B}) = \{(a, 0)(b, 1), (c, 0)(b, t), 1 \leq t \leq 2\}$ $\mathcal{L}(\mathcal{B}) = \{(a, 0)(b, 1)\}$

Timed Language Accepted by a TA (Example 2)



$\mathcal{B}$ can generate the following runs: for $\delta_1 \geq 0$ and $1 \leq \delta_2 \leq 2$

$$(0, x = 0) \xrightarrow{a} (1, x = 0) \xrightarrow{1} (1, x = 1) \xrightarrow{b} (2, x = 1) \xrightarrow{\delta_1} (2, x = 1 + \delta_1)$$

and

$$(0, x = 0) \xrightarrow{c} (3, x = 0) \xrightarrow{\delta_2} (3, x = \delta_2 2) \xrightarrow{b} (4, x = \delta_2) \xrightarrow{\delta_1} (4, x = \delta_2 + \delta_1)$$

$Tr(\mathcal{B}) = \{(a, 0)(b, 1), (c, 0)(b, t), 1 \leq t \leq 2\}$ $\mathcal{L}(\mathcal{B}) = \{(a, 0)(b, 1)\}$

Verification of Timed Automata [Alur and Dill (TCS 94)]

- Timed Automata generate Timed Languages
  a timed word: $(a, 1.2)(b, 4.567)(a, 6)\cdots$
- Emptiness Problem: Is the language accepted by a TA empty ?
  reachability properties, Büchi-like properties
- Universal Problem: Does a TA accept all timed words ?

Decidability Result [Alur and Dill (TCS 94)]

Emptiness Problem for TA is PSPACE-Complete.
Build a finite time-bisimilar abstraction: region automaton

Undecidability/Non Closure Results [Alur and Dill (TCS 94)]

- Universal Problem for TA is undecidable
  implies that Inclusion Problem is undecidable
- TA are not determinizable nor complementable

Verification of Timed Automata [Alur and Dill (TCS 94)]

- Timed Automata generate Timed Languages
  a timed word: $(a, 1.2)(b, 4.567)(a, 6)\cdots$
- Emptiness Problem: Is the language accepted by a TA empty ?
  reachability properties, Büchi-like properties
- Universal Problem: Does a TA accept all timed words ?

Decidability Result [Alur and Dill (TCS 94)]

Emptiness Problem for TA is PSPACE-Complete.
Build a finite time-bisimilar abstraction: region automaton

Undecidability/Non Closure Results [Alur and Dill (TCS 94)]

- Universal Problem for TA is undecidable
  implies that Inclusion Problem is undecidable
- TA are not determinizable nor complementable

Verification of Timed Automata [Alur and Dill (TCS 94)]

- Timed Automata generate Timed Languages
  a timed word: $(a, 1.2)(b, 4.567)(a, 6) \cdots$
- Emptiness Problem: Is the language accepted by a TA empty ?
  reachability properties, Büchi-like properties
- Universal Problem: Does a TA accept all timed words ?

Decidability Result [Alur and Dill (TCS 94)]

Emptiness Problem for TA is PSPACE-Complete.
Build a finite time-bisimilar abstraction: region automaton

Undecidability/Non Closure Results [Alur and Dill (TCS 94)]

- Universal Problem for TA is undecidable
  implies that Inclusion Problem is undecidable
- TA are not determinizable nor complementable

Timed Opacity Problem

Given: a timed automaton $A = (L, \ell_0, X, \Sigma_T, E, F)$
F = set of secret locations
$\Sigma_o \subseteq \Sigma$, the set of observable actions

- $\pi(Tr(A))$ = set of projections on Σ_o of words generated by A
- $w \in \pi(Tr(A))$
  - $[w] = \pi^{-1}(w) \cap Tr(A)$
  - $last([w])$ set of locations A can be in after observing w

Definition (Opacity)

The secret F is opaque with respect to A and $\Sigma_o \subseteq \Sigma$ iff for each $w \in \pi(Tr(A))$, $last([w]) \not\subseteq F$.

Opacity Verification Problem for timed automata:

Check wether F is opaque w.r.t. (A, Σ_o).

Timed Opacity Problem

Given: a timed automaton $A = (L, \ell_0, X, \Sigma_\tau, E, F)$
F = set of secret locations
$\Sigma_o \subseteq \Sigma$, the set of observable actions

- $\pi(Tr(A))$ = set of projections on Σ_o of words generated by A
- $w \in \pi(Tr(A))$
  - $[w] = \pi^{-1}(w) \cap Tr(A)$
  - *$last([w])$ set of locations A can be in after observing w*

Definition (Opacity)

The secret F is opaque with respect to A and $\Sigma_o \subseteq \Sigma$ iff for each $w \in \pi(Tr(A))$, $last([w]) \not\subseteq F$.

Opacity Verification Problem for timed automata:

Check wether F is opaque w.r.t. (A, Σ_o).

Timed Opacity Problem

Given: a timed automaton $A = (L, \ell_0, X, \Sigma_T, E, F)$
F = set of secret locations
$\Sigma_o \subseteq \Sigma$, the set of observable actions

- $\pi(Tr(A))$ = set of projections on Σ_o of words generated by A
- $w \in \pi(Tr(A))$
  - $[w] = \pi^{-1}(w) \cap Tr(A)$
  - $last([w])$ *set of locations A can be in after observing w*

Definition (Opacity)

The secret F is opaque with respect to A and $\Sigma_o \subseteq \Sigma$ iff for each $w \in \pi(Tr(A))$, $last([w]) \not\subseteq F$.

Opacity Verification Problem for timed automata:

Check wether F is opaque w.r.t. (A, Σ_o).

Timed Opacity Problem

Given: a timed automaton $A = (L, \ell_0, X, \Sigma_\tau, E, F)$
F = set of secret locations
$\Sigma_o \subseteq \Sigma$, the set of observable actions

- $\pi(Tr(A))$ = set of projections on Σ_o of words generated by A
- $w \in \pi(Tr(A))$
  - $[w] = \pi^{-1}(w) \cap Tr(A)$
  - $last([w])$ *set of locations A can be in after observing w*

Definition (Opacity)

The secret F is opaque with respect to A and $\Sigma_o \subseteq \Sigma$ iff for each $w \in \pi(Tr(A))$, $last([w]) \not\subseteq F$.

Opacity Verification Problem for timed automata:

Check wether F is opaque w.r.t. (A, Σ_o).

Results: Undecidability of Timed Opacity

Theorem

The opacity problem is undecidable for TA.

The proof is by reduction of the universality problem to the opacity problem.

Simpler Classes of Timed Automata

- Deterministic: no silent action and next state determined by (time,action)
- Event-Recording: deterministic, clocks are associated with actions [Alur et al. (CAV'94)]

Theorem

The opacity problem is undecidable for Event-Recording TA.

Results: Undecidability of Timed Opacity

Theorem

The opacity problem is undecidable for TA.

The proof is by reduction of the universality problem to the opacity problem.

Simpler Classes of Timed Automata

- Deterministic: no silent action and next state determined by (time,action)
- Event-Recording: deterministic, clocks are associated with actions [Alur et al. (CAV'94)]

Theorem

The opacity problem is undecidable for Event-Recording TA.

Conclusion & Further Results

Opacity + Dense-Time

- Checking Opacity is undecidable for TA
- Undecidability holds for simple timed systems like ERA
- Undecidability holds for time Petri Nets
  Timed automata and time Petri nets are equally expressive
  [Cassez and Roux (JSS 2006)]

Opacity + Discrete time

- Decidable but expensive

Opacity + Digital Clocks [Cassez and Tripakis (FI 2008)]

- A clock is a timed automaton (dense-time)
- Clock issues tick events
- External observer can only see $\Sigma_o \cup \{tick\}$
- Opacity with digital clocks is decidable in EXPTIME

Thanks !

Conclusion & Further Results

Opacity + Dense-Time

- Checking Opacity is undecidable for TA
- Undecidability holds for simple timed systems like ERA
- Undecidability holds for time Petri Nets

  Timed automata and time Petri nets are equally expressive [Cassez and Roux (JSS 2006)]

Opacity + Discrete time

- Decidable but expensive

Opacity + Digital Clocks [Cassez and Tripakis (FI 2008)]

- A clock is a timed automaton (dense-time)
- Clock issues tick events
- External observer can only see $\Sigma_o \cup \{tick\}$
- Opacity with digital clocks is decidable in EXPTIME

Thanks !

Conclusion & Further Results

Opacity + Dense-Time

- Checking Opacity is undecidable for TA
- Undecidability holds for simple timed systems like ERA
- Undecidability holds for time Petri Nets
  Timed automata and time Petri nets are equally expressive
  [Cassez and Roux (JSS 2006)]

Opacity + Discrete time

- Decidable but expensive

Opacity + Digital Clocks [Cassez and Tripakis (FI 2008)]

- A clock is a timed automaton (dense-time)
- Clock issues tick events
- External observer can only see $\Sigma_o \cup \{tick\}$
- Opacity with digital clocks is decidable in EXPTIME

Thanks !

Conclusion & Further Results

Opacity + Dense-Time

- Checking Opacity is undecidable for TA
- Undecidability holds for simple timed systems like ERA
- Undecidability holds for time Petri Nets

  Timed automata and time Petri nets are equally expressive [Cassez and Roux (JSS 2006)]

Opacity + Discrete time

- Decidable but expensive

Opacity + Digital Clocks [Cassez and Tripakis (FI 2008)]

- A clock is a timed automaton (dense-time)
- Clock issues tick events
- External observer can only see $\Sigma_o \cup \{tick\}$
- Opacity with digital clocks is decidable in EXPTIME

Thanks !

References

[Mazaré (WITS'2004)] Mazaré, L.:
Using unification for opacity properties.
In: Proceedings of the 4th IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'04), Barcelona (Spain) (2004) 165–176

[Bryans et al. (FAST'2005)] Bryans, J., Koutny, M., Mazaré, L., Ryan, P.Y.A.:
Opacity generalised to transition systems.
In Dimitrakos, T., Martinelli, F., Ryan, P.Y.A., Schneider, S.A., eds.: Formal Aspects in Security and Trust. Volume 3866 of Lecture Notes in Computer Science., Springer (2005) 81–95

[Alur and Dill (TCS 94)] Alur, R., Dill, D.:
A theory of timed automata.
Theoretical Computer Science (TCS) 126(2) (1994) 183–235

[Alur et al. (CAV'94)] Alur, R., Fix, L., Henzinger, T.A.:
Event clock automata: A determinizable class of timed automata.
In: Proc. 6th International Conference on Computer Aided Verification (CAV'94). Volume 818 of Lecture Notes in Computer Science., Springer (1994) 1–13

[Cassez and Roux (JSS 2006)] Cassez, F., Roux, O.H.:
Structural translation from time petri nets to timed automata.
Journal of Software and Systems 79(10) (2006) 1456–1468

[Cassez and Tripakis (FI 2008)] Cassez, F., Tripakis, S.:
Fault diagnosis with static or dynamic diagnosers.
Fundamenta Informatica 88(4) (November 2008) 497–540.

[Cassez et al. (ATVA'09)] Cassez, F., Dubreil, J. and Marchand, H.:
Dynamic Observers for the Synthesis of Opaque Systems.
In: Proc. 7th International Symposium on Automated Technology for Verification and Analysis (ATVA'09). LNCS, Forthcoming.