Real-Time Systems

Lecture 15: Extended Timed Automata

2018-01-09

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

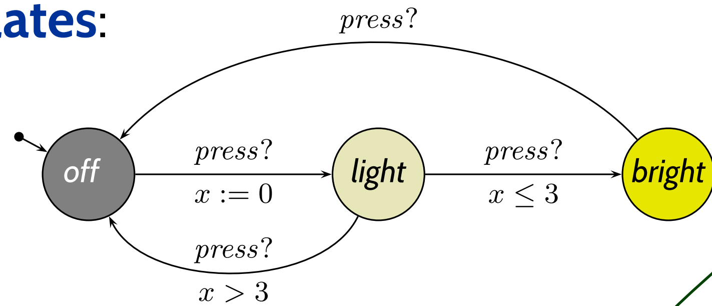
- **Extended Timed Automata** – **Syntax**
  - **Data Variables**
  - **Urgent** locations and channels
  - **Committed** locations
- **Extended Timed Automata** – **Semantics**
  - **labelled transition system**
  - **extended** valuations, timeshift, modification
  - **examples** for urgent / committed
- **Extended** vs. **Pure** Timed Automata
- The **Reachability Problem** of **Extended Timed Automata**
- **Uppaal Query Language**
  - **Transition graph** (!)
  - **By-the-way**: satisfaction relation **decidable**.

Extended Timed Automata

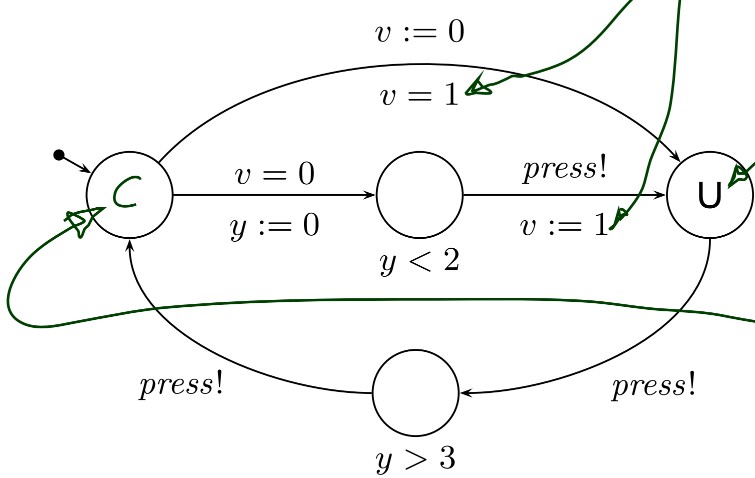
Example (Partly Already Seen in Uppaal Demo)

Templates:

- $\mathcal{L}$:



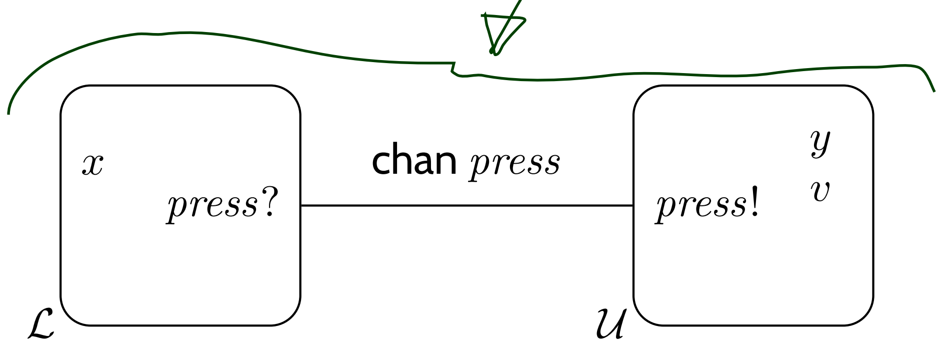
- $\mathcal{U}$:



Extensions:

- Data Variables (Expressions, Constraints, Updates)
- Structuring
- Urgent/Committed Locations, Urgent Channels

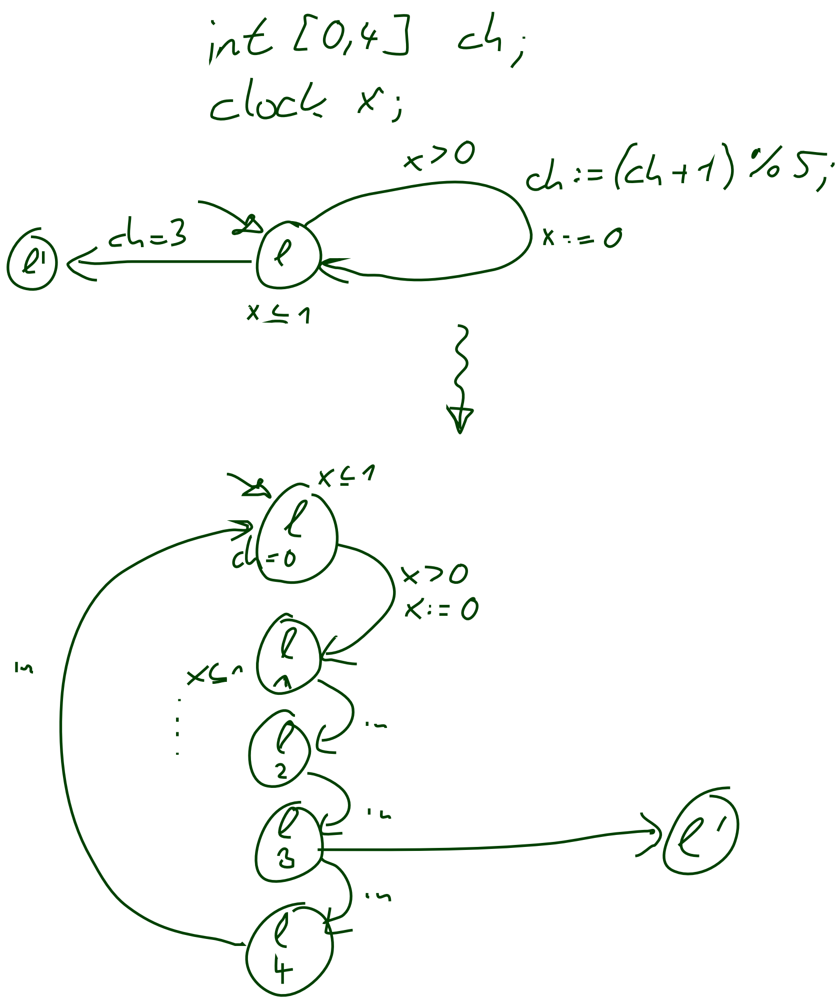
System:



Data-Variables

- When modelling controllers as timed automata,
  it is sometimes desirable to have (local and shared) **non-clock variables**.

  E.g. count number of open doors, or intermediate positions of gas valve.



Data-Variables

- When modelling controllers as timed automata,
  it is sometimes desirable to have (local and shared) **non-clock variables**.

  E.g. count number of open doors, or intermediate positions of gas valve.

- Adding variables with **finite** range (possibly grouped into **finite** arrays)
  to **any** finite-state automata concept is **straighforward**:
  - If we have control locations $L_0 = \{\ell_1, \dots, \ell_n\}$,
  - and want to model, e.g., the **valve position** as a variable v with domain $\mathcal{D}(v) = \{0, 1, 2\}$,
  - then just use $L = L_0 \times \mathcal{D}(v)$ as control locations,
    i.e. **encode** the current value of v in **locations**, and **consider updates** of v in the edges.

  L is still **finite**, so we still have a **proper timed automaton**.

- **But**: writing **edges** is tedious then.
- So: have variables as "first class citizens" and let compilers do the work.

- **Interestingly**, many case-studies in the literature live without non-clock variables:

  The more abstract the model is, i.e., the fewer information it keeps track of (e.g. in data variables), the easier the verification task.

Data Variables and Expressions

- Let $(v, w \in)\ V$ be a set of **(integer) variables**.

  $(\psi_{int} \in)\ \Psi(V)$: **integer expressions** over V using function symbols $+, -, \ldots$

  $(\varphi_{int} \in)\ \Phi(V)$: **integer** (or **data**) **constraints** over V,

  using **integer expressions**, predicate symbols $=, <, \leq, \ldots$, and $\underbrace{\text{logical connectives}}_{(\wedge, \neg, \vee, \ldots)}$.

- Let $(x, y \in)\ X$ be a set of clocks.

  $(\varphi \in)\ \Phi(X, V)$: The set of (**extended**) **guards** is defined by the following grammar:

$$\varphi ::= \varphi_{clk} \mid \varphi_{int} \mid \varphi_1 \wedge \varphi_2$$

  where $\varphi_{clk} \in \Phi(X)$ is a **simple clock constraint** (as defined before)
  and $\varphi_{int} \in \Phi(V)$ an **integer (or data) constraint**.

Examples: Extended guard or not extended guard? Why?

(a) $\underbrace{x < y}_{\in \Phi(X)} \wedge \underbrace{v > 2}_{\in \Phi(V)}$, ✓ (b) $x < y \vee v > 2$, ✗ (c) $\underbrace{\overbrace{v < 1}^{\in \Phi(V)} \vee \overbrace{v > 2}^{\in \Phi(V)}}_{\in \Phi(V)}$, ✓ (d) $x < v$ ✗

V : |||
X : |

Modification or Reset Operation

- **New**: a **modification** or **reset (operation)** is

$$x := 0, \qquad x \in X,$$

or

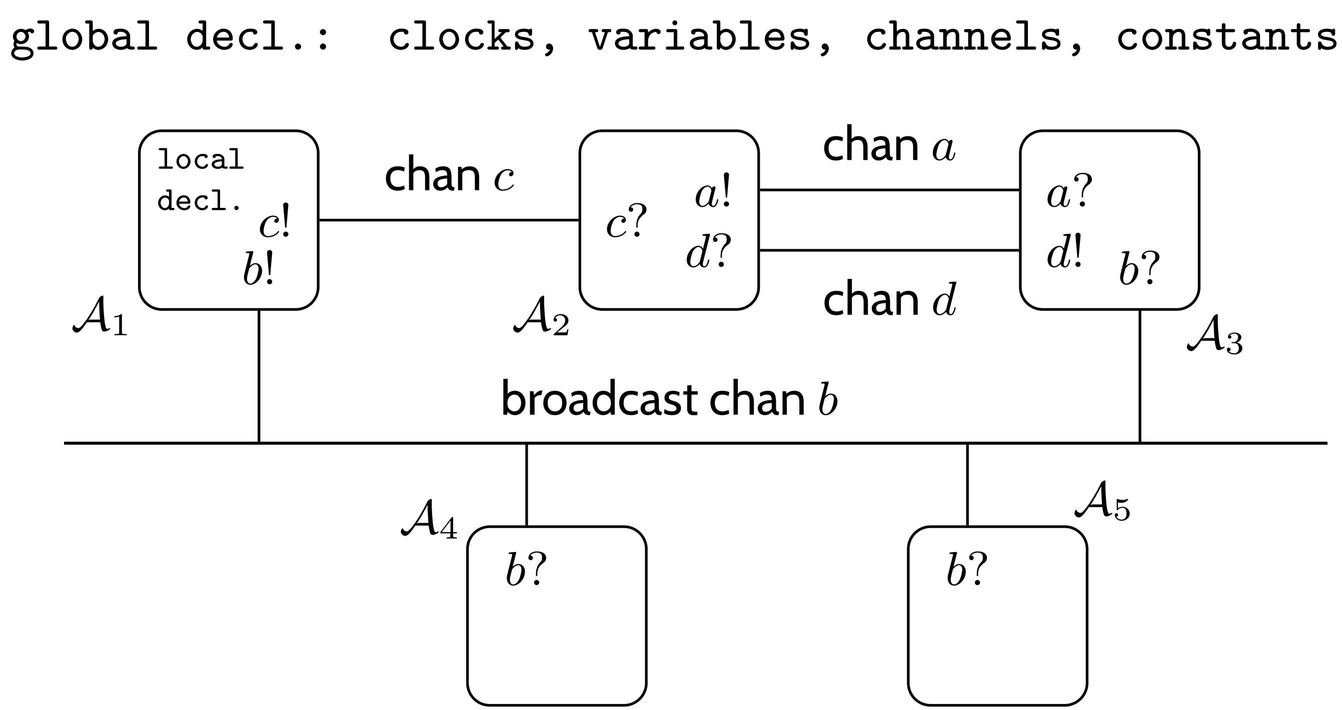
$$v := \psi_{int}, \qquad v \in V, \quad \psi_{int} \in \Psi(V).$$

- By $R(X, V)$ we denote the set of all **resets**.
- By $\vec{r}$ we denote a **finite list** $\langle r_1, \ldots, r_n \rangle$, $n \in \mathbb{N}_0$, of **reset operations** $r_i \in R(X, V)$; $\langle\rangle$ is the empty list.
- By $R(X, V)^*$ we denote the set of all such lists of reset operations (also called **reset vector**).

Examples: Modification or not? Why? (x, y clocks; v, w variables)

(a) $x := y$, ✗ (b) $x := v$, ✗ (c) $v := x$, ✗ (d) $v := w$, ✓ (e) $v := 0$ ✓

Structuring Facilities



- Global declarations of of clocks, data variables, channels, and constants.
- Binary and broadcast channels: chan c and broadcast chan b.
- Templates of timed automata.
- Instantiation of templates (instances are called **process**).
- System definition: list of processes.

Restricting Non-determinism

- **Urgent locations** – enforce local immediate progress.

  U

- **Committed locations** – enforce **atomic** immediate progress.

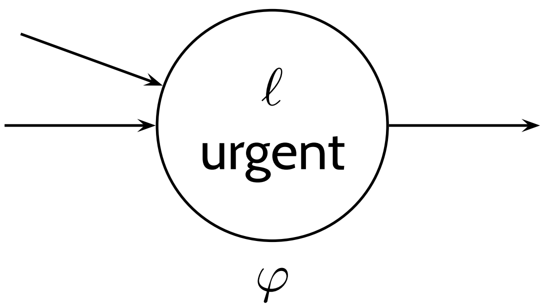
  C

- **Urgent channels** – enforce cooperative immediate progress.

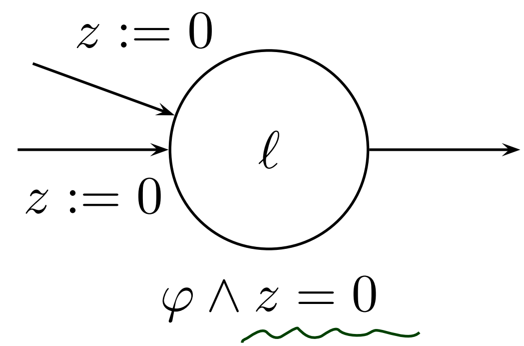
```
urgent chan press;
```

Urgent Locations: Only an Abbreviation...

Replace



with



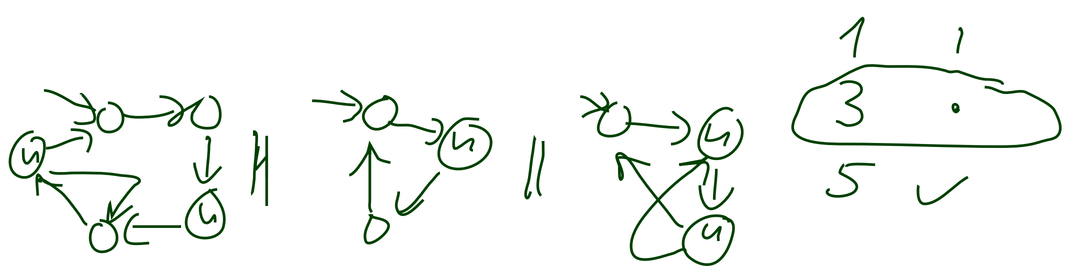
where z is a fresh clock:

- reset z on all in-going egdes,
- add $z = 0$ to invariant.

pairwise

because in the course we only consider disjoint sets of clocks

Question: How many fresh clocks do we need in the worst case for a network of N extended timed automata?



Extended Timed Automata

Definition 4.39. An **extended timed automaton** is a structure

$$\mathcal{A}_e = (L, C, B, U, X, V, I, E, \ell_{ini})$$

where L, B, X, I, ℓ_{ini} are as in Definition 4.3,
except that location invariants in I are **downward closed**, and where

- $C \subseteq L$: **committed locations**,
- $U \subseteq B$: **urgent channels**,
- V: a set of **data variables** (with **finite** domain),
- $E \subseteq L \times B_{!?} \times \Phi(X, V) \times R(X, V)^* \times L$
  is a set of **directed edges** such that

$$(\ell, \alpha, \varphi, \vec{r}, \ell') \in E \wedge \mathsf{chan}(\alpha) \in U \implies \varphi = true.$$

  Edges $(\ell, \alpha, \varphi, \vec{r}, \ell')$ from location ℓ to ℓ' are
  labelled with an **action** α, a **guard** φ, and a list $\vec{r}$ of **reset operations**.

Content

- **Extended Timed Automata** – **Syntax**
  - **Data Variables**
  - **Urgent** locations and channels
  - **Committed** locations
- **Extended Timed Automata** – **Semantics**
  - **labelled transition system**
  - **extended** valuations, timeshift, modification
  - **examples** for urgent / committed
- **Extended** vs. **Pure** Timed Automata
- The **Reachability Problem** of **Extended Timed Automata**
- **Uppaal Query Language**
  - **Transition graph** (!)
  - **By-the-way**: satisfaction relation **decidable**.

Definition 4.40. Let

$$\mathcal{A}_{e,i} = (L_i, C_i, B_i, U_i, X_i, V_i, I_i, E_i, \ell_{ini,i}), \quad 1 \leq i \leq n,$$

be extended timed automata with pairwise disjoint sets of clocks X_i.
The **operational semantics** of $\mathcal{N} = \mathcal{C}(\mathcal{A}_{e,1}, \dots, \mathcal{A}_{e,n})$ (closed!)
is **the labelled transition system**

$$\mathcal{T}_e(\mathcal{C}(\mathcal{A}_{e,1}, \dots, \mathcal{A}_{e,n})) = \mathcal{T}(\mathcal{N}) =$$

$$(\mathit{Conf}, \mathsf{Time} \cup \{\tau\}, \{\xrightarrow{\lambda} \mid \lambda \in \mathsf{Time} \cup \{\tau\}\}, C_{ini})$$

where

- $X = \bigcup_{i=1}^{n} X_i$ and $V = \bigcup_{i=1}^{n} V_i$,
- $\mathit{Conf} = \{\langle \vec{\ell}, \nu \rangle \mid \ell_i \in L_i, \nu : X \cup V \to \mathsf{Time}, \nu \models \bigwedge_{k=1}^{n} I_k(\ell_k)\}$,
  $\cup \mathcal{D}(V)$
- $C_{ini} = \{\langle \vec{\ell}_{ini}, \nu_{ini} \rangle\} \cap \mathit{Conf}$,

The **transition relations** consists of transitions of the following **three types**.

Helpers: Extended Valuations and Timeshift

- **Now**: $\nu : X \cup V \to \mathsf{Time} \cup \mathcal{D}(V)$
- Canonically extends to $\nu : \Psi(V) \to \mathcal{D}^{(V)}$ (valuation of expression).
- "$\models$" extends canonically to expressions from $\Phi(X, V)$.
- Extended **timeshift** $\nu + t$, $t \in \mathsf{Time}$, applies to clocks only:
  - $\underbrace{(\nu + t)}(x) := \nu(x) + t$, $x \in X$,
  - $\underbrace{(\nu + t)}(v) := \nu(v)$, $v \in V$.
- **Effect of modification** $r \in R(X, V)$ on ν, denoted by $\nu[r]$:

$$\underbrace{\nu[x := 0]}(a) := \begin{cases} 0, \text{ if } a = x, \\ \nu(a), \text{ otherwise} \end{cases}$$

$$\nu[v := \psi_{int}](a) := \begin{cases} \nu(\psi_{int}), \text{ if } a = v, \\ \nu(a), \text{ otherwise} \end{cases}$$

- We set $\nu[\langle r_1, \dots, r_n \rangle] := ((\nu[r_1])\dots[r_n]) = (\;(\;(\;\nu[r_1]\;)[r_2]\;)[r_3]\;\dots\;)\;[r_n]$.

Operational Semantics of Networks: Internal Transitions

- An **internal transition** $\langle \vec{\ell}, \nu \rangle \xrightarrow{\tau} \langle \vec{\ell}', \nu' \rangle$ occurs if there is $i \in \{1, \dots, n\}$ such that
  - there is a τ-edge $(\ell_i, \tau, \varphi, \vec{r}, \ell_i') \in E_i$,
  - $\nu \models \varphi$,
  - $\vec{\ell}' = \vec{\ell}[\ell_i := \ell_i']$,
  - $\nu' = \nu[\vec{r}]$,
  - $\nu' \models I_i(\ell_i')$,
  - ($\clubsuit$) if $\ell_k \in C_k$ for some $k \in \{1, \dots, n\}$ then $\ell_i \in C_i$.

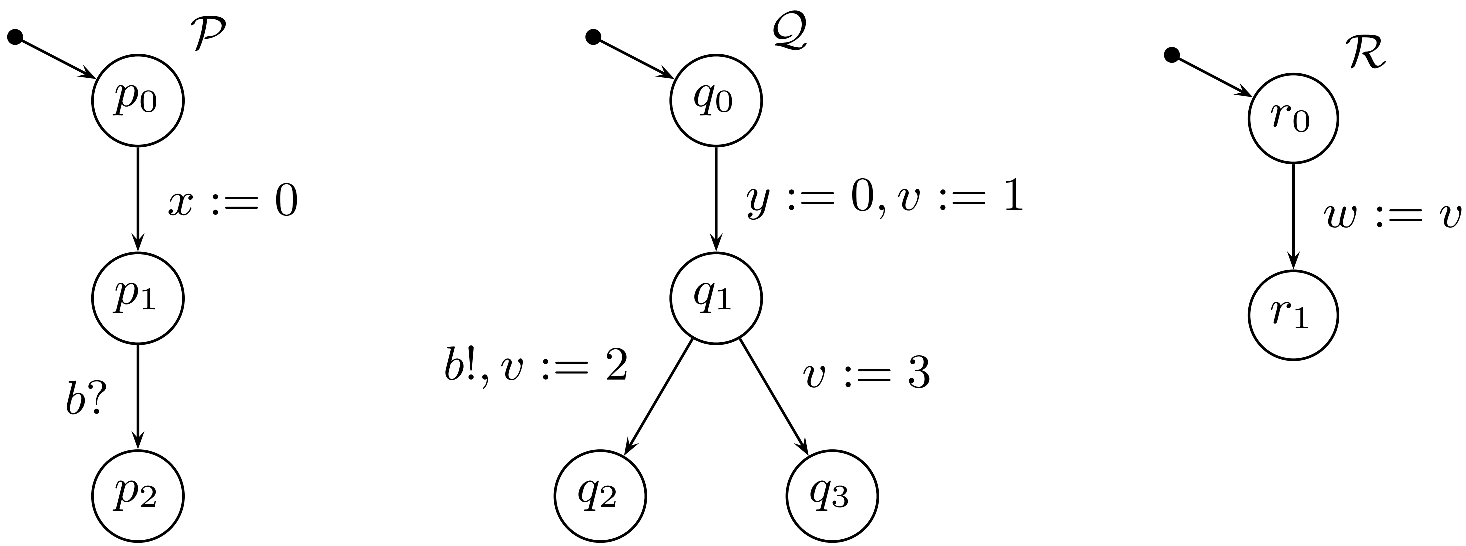
Operational Semantics of Networks: Synchronisation

- A **synchronisation transition** $\langle \vec{\ell}, \nu \rangle \xrightarrow{\tau} \langle \vec{\ell'}, \nu' \rangle$ occurs if there are $i, j \in \{1, \dots, n\}$ with $i \neq j$ such that
  - there are edges $(\ell_i, b!, \varphi_i, \vec{r}_i, \ell_i') \in E_i$ and $(\ell_j, b?, \varphi_j, \vec{r}_j, \ell_j') \in E_j$,
  - $\nu \models \varphi_i \wedge \varphi_j$,
  - $\vec{\ell'} = \vec{\ell}[\ell_i := \ell_i'][\ell_j := \ell_j']$,
  - $\nu' = (\nu[\vec{r}_i])[\vec{r}_j]$,
  - $\nu' \models I_i(\ell_i') \wedge I_j(\ell_j')$,
  - (♣) if $\ell_k \in C_k$ for some $k \in \{1, \dots, n\}$ then $\ell_i \in C_i$ or $\ell_j \in C_j$.

Operational Semantics of Networks: Delay Transitions

- A **delay transition** $\langle \vec{\ell}, \nu \rangle \xrightarrow{t} \langle \vec{\ell}, \nu + t \rangle$ occurs if
  - $\nu + t \models \bigwedge_{k=1}^{n} I_k(\ell_k)$,
  - (♣) there are no $i \neq j \in \{1, \dots, n\}$ and $b \in U$ with $(\ell_i, b!, \varphi_i, \vec{r}_i, \ell'_i) \in E_i$ and $(\ell_j, b?, \varphi_j, \vec{r}_j, \ell'_j) \in E_j$,
  - (♣) there is no $i \in \{1, \dots, n\}$ such that $\ell_i \in C_i$.

Restricting Non-determinism: Example



	Property 1	Property 2	Property 3
	w can become 1	$y \leq 0$ holds when $\mathcal{Q}$ is in q_1	$(x \geq y \implies y \leq 0)$ holds when in p_1 and q_1
$\mathcal{N} := \mathcal{P} \Vert \mathcal{Q} \Vert \mathcal{R}$	✔	✘	✘
$\mathcal{N}$, q_1 urgent	✓	✓	✓
$\mathcal{N}$, q_1 committed	✗	✓	✓
$\mathcal{N}$, b urgent	✓	✗	✓

Content

- **Extended Timed Automata** – **Syntax**
  - **Data Variables**
  - **Urgent** locations and channels
  - **Committed** locations
- **Extended Timed Automata** – **Semantics**
  - **labelled transition system**
  - **extended** valuations, timeshift, modification
  - **examples** for urgent / committed
- **Extended** vs. **Pure** Timed Automata
- The **Reachability Problem** of **Extended Timed Automata**
- **Uppaal Query Language**
  - **Transition graph** (!)
  - **By-the-way**: satisfaction relation **decidable**.

Extended vs. Pure Timed Automata

Extended vs. Pure Timed Automata

$$\mathcal{A}_e = (L, C, B, U, X, V, I, E, \ell_{ini})$$
$$(\ell, \alpha, \varphi, \vec{r}, \ell') \in L \times B_{!?} \times \Phi(X, V) \times R(X, V)^* \times L$$

vs.

$$\mathcal{A} = (L, B, X, I, E, \ell_{ini})$$
$$(\ell, \alpha, \varphi, Y, \ell') \in E \subseteq L \times B_{?!} \times \Phi(X) \times 2^X \times L$$

- $\mathcal{A}_e$ is in fact (or specialises to) a **pure** timed automaton if
  - $C = \emptyset$,
  - $U = \emptyset$,
  - $V = \emptyset$,
  - for each $\vec{r} = \langle r_1, \dots, r_n \rangle$, every r_i is of the form $x := 0$ with $x \in X$.

  - $I(\ell), \varphi \in \Phi(X)$ is then a consequence of $V = \emptyset$.

Operational Semantics of Extended TA

Theorem 4.41. If $\mathcal{A}_1, \dots, \mathcal{A}_n$ **specialise to pure** timed automata, then the operational semantics of

$$\mathcal{C}(\mathcal{A}_1, \dots, \mathcal{A}_n)$$

and

$$\textsf{chan}\, b_1, \dots, b_m \bullet (\mathcal{A}_1 \parallel \dots \parallel \mathcal{A}_n),$$

where $\{b_1, \dots, b_m\} = \bigcup_{i=1}^{n} B_i$, **coincide**, i.e.

$$\mathcal{T}_e(\mathcal{C}(\mathcal{A}_1, \dots, \mathcal{A}_n)) = \mathcal{T}(\textsf{chan}\, b_1, \dots, b_m \bullet (\mathcal{A}_1 \parallel \dots \parallel \mathcal{A}_n)).$$

Content

- **Extended Timed Automata** – **Syntax**
  - **Data Variables**
  - **Urgent** locations and channels
  - **Committed** locations
- **Extended Timed Automata** – **Semantics**
  - **labelled transition system**
  - **extended** valuations, timeshift, modification
  - **examples** for urgent / committed
- **Extended** vs. **Pure** Timed Automata
- The **Reachability Problem**
  of **Extended Timed Automata**
- **Uppaal Query Language**
  - **Transition graph** (!)
  - **By-the-way**: satisfaction relation **decidable**.

Reachability Problems for Extended Timed Automata

Recall

Theorem 4.33. [*Location Reachability*]
The location reachability problem for **pure** timed automata is **decidable**.

Theorem 4.34. [*Constraint Reachability*]
Constraint reachability is **decidable** for **pure** timed automata.

- And what about tea`^W` **extended** timed automata?

What About Extended Timed Automata?

Extended Timed Automata add the following features:

- **Data-Variables**
  - As long as the domains of all variables in V are **finite**, adding data variables doesn't harm decidability.
  - If they're **infinite**, we've got a problem (encode two-counter machine!).
- **Structuring Facilities**
  - Don't hurt – they're merely abbreviations.
- **Restricting Non-determinism**
  - Restricting non-determinism doesn't affect the configuration space $Conf$.
  - Restricting non-determinism only **removes** certain transitions, so it makes the **reachable part** of the region automaton **even smaller** (not necessarily strictly smaller).

The Logic of Uppaal

$$\mathcal{N} \models^{?} \underbrace{\overset{EF}{E\Diamond}(\overbrace{w=1}^{\varphi})}$$

Uppaal Fragment of Timed Computation Tree Logic

aka, Uppaal query language

Consider $\mathcal{N} = \mathcal{C}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ over data variables V.

- **basic formula**:
$$atom ::= \mathcal{A}_i.\ell \mid \varphi$$
where $\ell \in L_i$ is a location and φ a constraint over X_i and V.
- **configuration formulae**:
$$term ::= atom \mid \neg term \mid term_1 \wedge term_2$$
- **existential path formulae**: ("**exists finally**", "**exists globally**")
$$e\text{-}formula ::= \underbrace{\exists\Diamond}_{EF}\, term \mid \underbrace{\exists\Box}_{EG}\, term$$
- **universal path formulae**: ("**always finally**", "**always globally**", "**leads to**")
$$a\text{-}formula ::= \forall\Diamond\, term \mid \forall\Box\, term \mid term_1 \longrightarrow term_2$$
- **formulae**:
$$F ::= e\text{-}formula \mid a\text{-}formula$$

Tell Them What You've Told Them...

- **For convenience**, time automata can be **extended** by
  - **data variables**, and
  - **urgent / committed** locations.
- **None** of these **extensions** harm decidability,
  as long as the data variables have a **finite** domain.
- Properties **to be checked** for a timed automata model
  can be specified using the **Uppaal Query Language**,
  - which is a **tiny little fragment** of Timed CTL (TCTL),
  - and as such **by far** not as expressive as Duration Calculus.
- **TCTL** is another **means** to **formalise requirements**.

References

References

Olderog, E.-R. and Dierks, H. (2008). *Real-Time Systems - Formal Specification and Automatic Verification*. Cambridge University Press.