

# *Real-Time Systems*

## *Lecture 14: Extended Timed Automata*

*2013-06-25*

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

# Contents & Goals

## Last Lecture:

- Decidability of the location reachability problem:
  - region automaton
  - zones

## This Lecture:

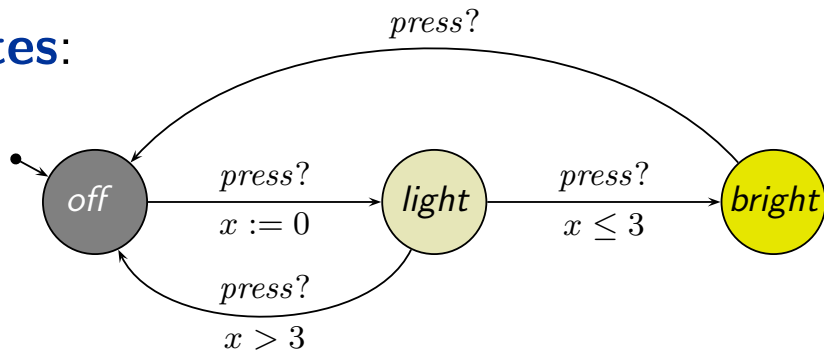
- **Educational Objectives:** Capabilities for following tasks/questions.
  - By what are TA extended? Why is that useful?
  - What's an urgent/committed location? What's the difference?
  - What's an urgent channel?
  - Where has the notion of “input action” and “output action” correspondences in the formal semantics?
- **Content:**
  - Extended TA:
    - Data-Variables
    - Structuring Facilities
    - Restriction of Non-Determinism
  - The Logic of Uppaal

# *Extended Timed Automata*

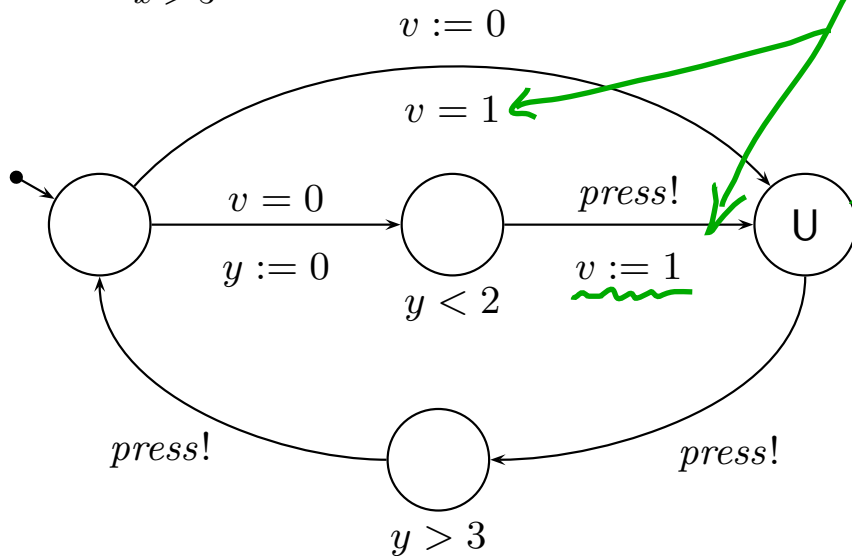
# Example (Partly Already Seen in Uppaal Demo)

## Templates:

•  $\mathcal{L}$ :



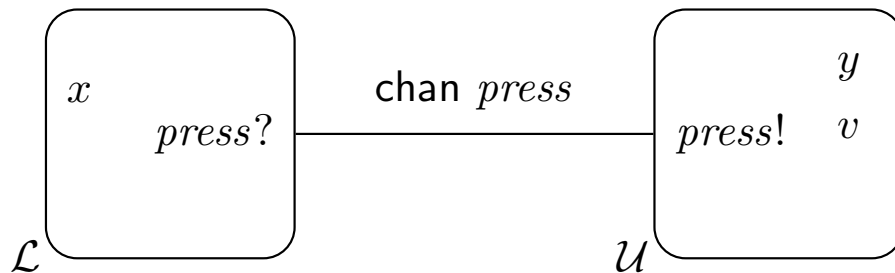
•  $\mathcal{U}$ :



**Extensions:**

- Data Variables (Expressions, Constraints, Updates)
- Structuring
- Urgent/Committed Location, Urgent Channel

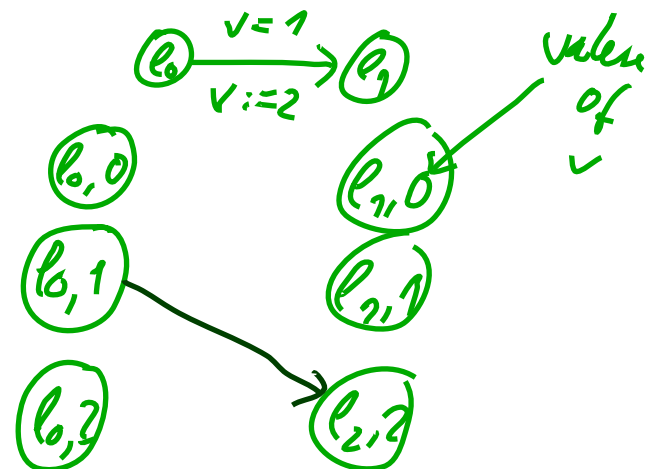
## System:



# Data-Variables

- When modelling controllers as timed automata, it is sometimes desirable to have (local and shared) 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 straightforward:
  - If we have control locations  $L_0 = \{\ell_1, \dots, \ell_n\}$ ,
  - and want to model, e.g., the valve range as a variable  $v$  with  $\mathcal{D}(v) = \{0, 1, 2\}$ ,
  - then just use  $L = L_0 \times \mathcal{D}(v)$  as control locations, i.e. encode the current value of  $v$  in the control location, and consider updates of  $v$  in the  $\xrightarrow{\lambda}$ .

$L$  is still finite, so we still have a proper TA.



# Data-Variables

- When modelling controllers as timed automata, it is sometimes desirable to have (local and shared) 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 straightforward:
  - If we have control locations  $L_0 = \{\ell_1, \dots, \ell_n\}$ ,
  - and want to model, e.g., the valve range as a variable  $v$  with  $\mathcal{D}(v) = \{0, \dots, 2\}$ ,
  - then just use  $L = L_0 \times \mathcal{D}(v)$  as control locations, i.e. encode the current value of  $v$  in the control location, and consider updates of  $v$  in the  $\xrightarrow{\lambda}$ .

$L$  is still finite, so we still have a proper TA.

- But: writing  $\xrightarrow{\lambda}$  is tedious.
- So: have variables as “first class citizens” and let compilers do the work.
- **Interestingly**, many examples in the literature live without 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

$$\psi_{int} ::= v \mid f(\psi_1, \dots, \psi_n) \quad v \in V$$

$$f \in \{+, -, \dots\}$$

- Let  $(v, w \in) V$  be a set of (integer) variables.
  - $(\psi_{int} \in) \Psi(V)$ : **integer expressions** over  $V$  using func. symb.  $+, -, \dots$
  - $(\varphi_{int} \in) \Phi(V)$ : **integer (or data) constraints** over  $V$  using **integer expressions**, predicate symbols  $=, <, \leq, \dots$ , and boolean logical connectives. (incl.  $\vee, \wedge, \Rightarrow, \Leftrightarrow, \neg, \dots$ )
- Let  $(x, y \in) X$  be a set of clocks.
  - $(\varphi \in) \Phi(X, V)$ : **(extended) guards**, defined by

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

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)  $x < y \wedge v > 2$ ,  
 $\underbrace{x < y}_{\in \Phi(X)} \quad \underbrace{v > 2}_{\in \Psi(V)}$   
 $(x - y < 0) \quad \checkmark$

(b)  $x < y \vee v > 2$ ,  
 $\underbrace{x < y}_{\in \Phi(X)} \quad \underbrace{v > 2}_{\in \Psi(V)}$   
 $\uparrow$   
 NO NO

(c)  $v < 1 \vee v > 2$ ,  
 $\underbrace{v < 1 \vee v > 2}_{\in \Psi(V)}$   
 $\checkmark$

(d)  $x < v$   
 $\notin \Phi(X)$   
 $\notin \Psi(V)$

clocks and data variables are never compared

NO

# Modification or Reset Operation

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

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

or

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

- By  $R(X, V)$  we denote the set of all resets.
- By  $\vec{r}$  we denote a finite list  $\langle r_1, \dots, r_n \rangle$ ,  $n \in \mathbb{N}_0$ , 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.

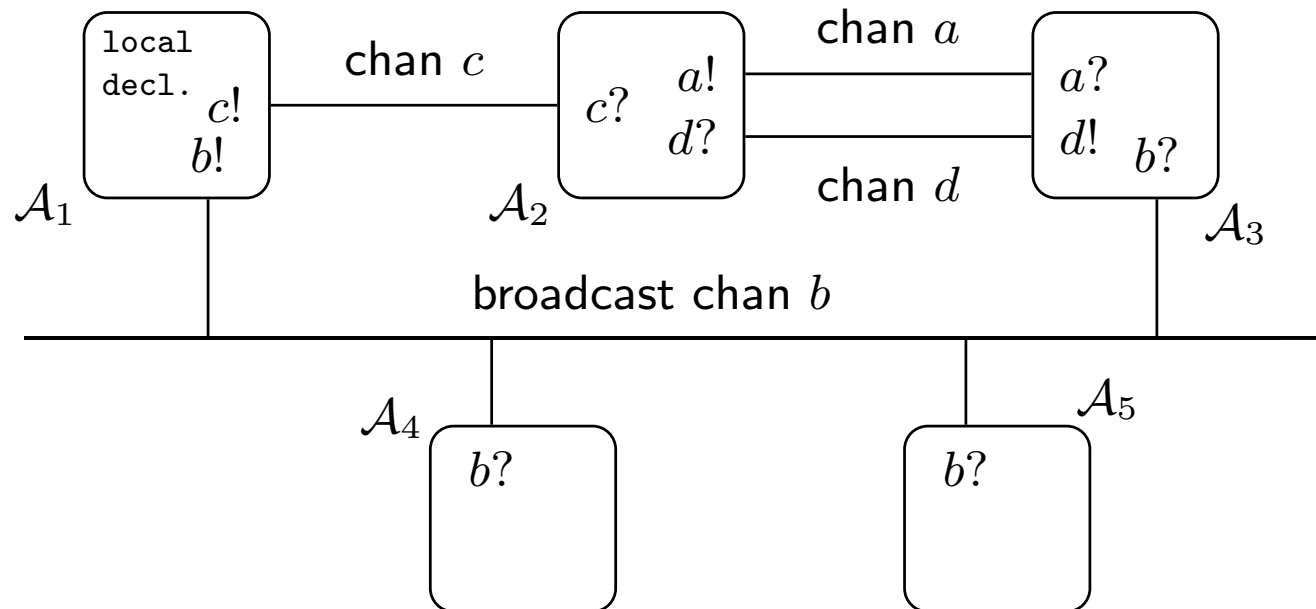
**Examples:** Modification or not? Why?

(a) $x := y,$	(b) $x := v,$	(c) $v := x,$	(d) $v := w,$	(e) $v := 0$
$\nearrow \in X$ $\nearrow \text{not } 0$	$\nearrow \in X$ $\nearrow \text{not } 0$	$\nearrow \in V$ $\nearrow \notin \Psi(V)$	$\nearrow \in V$ $\nearrow \in \Psi(V)$	$\nearrow \in V$ $\nearrow \in \Psi(V)$
NO	NO	NO	✓	✓



# Structuring Facilities

global decl.: clocks, variables, channels, constants



- 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.

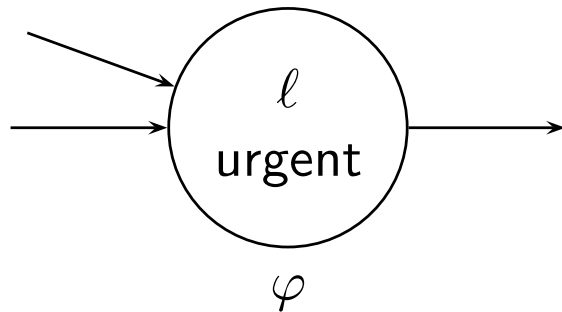
C

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

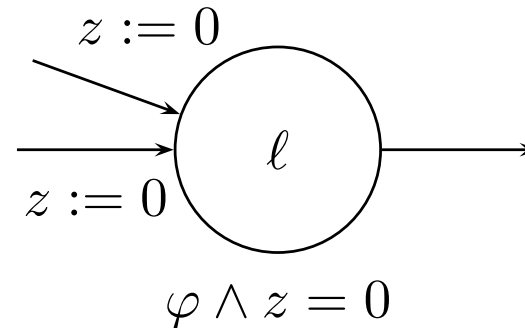
`urgent chan press;`

# Urgent Locations: Only an Abbreviation...

Replace



with



where  $z$  is a fresh clock:

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

$$N=3$$

number of urgent loc. is 20

at least one U-loc. per automaton

• 1 -  
• 3 | ∇  
•  
• 20 ||

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

# Extended Timed Automata

so still:  $I: L \rightarrow \Phi(x)$

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

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

where  $L, B, X, I, \ell_{ini}$  are as in Def. 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,
- $E \subseteq L \times B_{!?} \times \Phi(X, V) \times R(X, V)^* \times L$ : a set of **directed edges** such that

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

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

$\varphi$  is d.c.  
iff

$$\forall v: X \rightarrow \text{Time} \bullet$$

$$v \models \varphi$$

$\implies$

$$(\forall t \in \text{Time} \bullet$$

$$v.t \models \varphi)$$

$\uparrow$   
 $\circ$

•  $x < 3$  is d.c.

•  $x > 3$  is not d.c.

**Definition 4.40.** Let  $\mathcal{A}_{e,i} = (L_i, C_i, B_i, U_i, X_i, V_i, I_i, E_i, \ell_{ini,i})$ ,  $1 \leq i \leq n$ , be extended timed automata with pairwise disjoint sets of clocks  $X_i$ .

The operational semantics of  $\mathcal{C}(\mathcal{A}_{e,1}, \dots, \mathcal{A}_{e,n})$  (closed!) is the labelled transition system

$$\begin{aligned} \mathcal{T}_e(\mathcal{C}(\mathcal{A}_{e,1}, \dots, \mathcal{A}_{e,n})) \\ = (\mathit{Conf}, \mathit{Time} \cup \{\tau\}, \{\xrightarrow{\lambda} \mid \lambda \in \mathit{Time} \cup \{\tau\}\}, C_{ini}) \end{aligned}$$

where

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

and the transition relation consists of transitions of the following three types.

# Helpers: Extended Valuations and Timeshift

- **Now:**  $\nu : X \cup V \rightarrow \text{Time} \cup \mathcal{D}(V)$
- Canonically extends to  $\nu : \Psi(V) \rightarrow \mathcal{D}$  (valuation of expression).
- " $\models$ " extends canonically to expressions from  $\Phi(X, V)$ .

$$\Psi ::= \nu \mid f(\Psi_1, \dots, \Psi_n)$$

assume  $I(f) : \mathbb{Z}^n \rightarrow \mathbb{Z}$

$$I(\nu, \nu) := \nu(\nu) \in \mathcal{D}(V)$$

$$I(f(\Psi_1, \dots, \Psi_n), \nu) := I(f)(I(\Psi_1, \nu), \dots, I(\Psi_n, \nu))$$

$$I(\nu + \omega, \underbrace{\{\nu \mapsto 3, \omega \mapsto 24\}}_{\nu := \nu})$$

$$= I(+)(I(\nu, \nu), I(\omega, \nu)) = \hat{+}(\nu(\nu), \nu(\omega)) \\ = \hat{+}(3, 24) = 27$$

# Helpers: Extended Valuations and Timeshift

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

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

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

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

# Operational Semantics of Networks: Internal Transitions

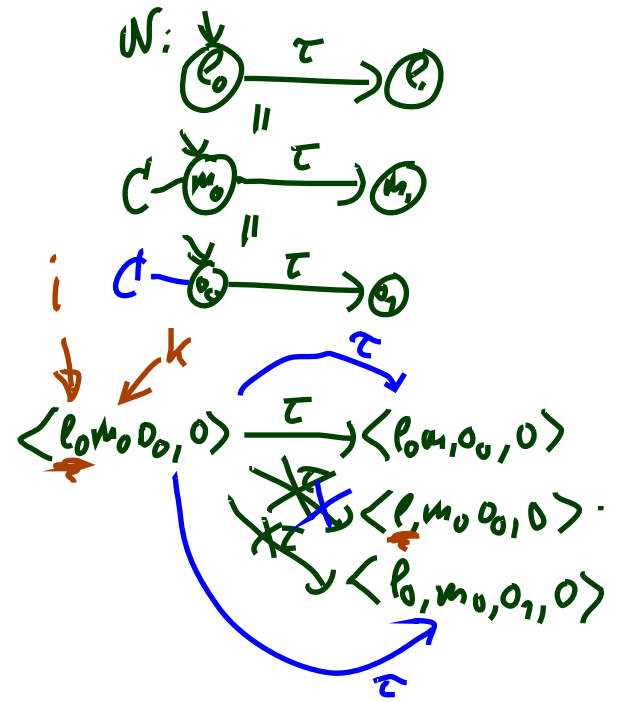
- An **internal transition**  $\langle \vec{l}, \nu \rangle \xrightarrow{\tau} \langle \vec{l}', \nu' \rangle$  occurs if there is  $i \in \{1, \dots, n\}$  such that

number of automata in the network

- there is a  $\tau$ -edge  $(l_i, \tau, \varphi, \vec{r}, l'_i) \in E_i$ ,
- $\nu \models \varphi$ ,
- $\vec{l}' = \vec{l}[l_i := l'_i]$ ,
- $\nu' = \nu[\vec{r}]$ ,
- $\nu' \models I_i(l'_i)$ ,

location of the  $i$ -th automaton in  $\vec{l}$   
 modification at the  $i$ -th position

- $(\clubsuit)$  if  $l_k \in C_k$  for some  $k \in \{1, \dots, n\}$  then  $l_i \in C_i$ .





# Operational Semantics of Networks: Synchronisation Transitions

- A **synchronisation transition**  $\langle \vec{l}, \nu \rangle \xrightarrow{\tau} \langle \vec{l}', \nu' \rangle$  occurs if there are  $i, j \in \{1, \dots, n\}$  with  $i \neq j$  such that

- there are edges  $(l_i, b!, \varphi_i, \vec{r}_i, l'_i) \in E_i$  and  $(l_j, b?, \varphi_j, \vec{r}_j, l'_j) \in E_j$ ,

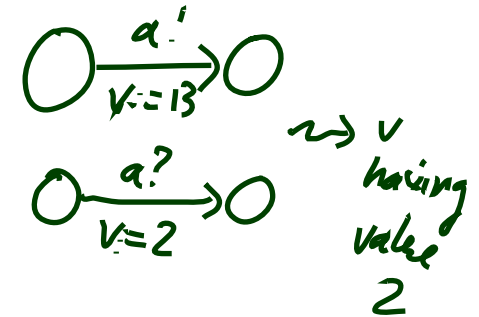
- $\nu \models \varphi_i \wedge \varphi_j$ ,

- $\vec{l}' = \vec{l}[l_i := l'_i][l_j := l'_j]$ ,

- $\nu' = \nu[\vec{r}_i][\vec{r}_j]$ ,  $\leftarrow$  "sender updates are applied first"

- $\nu' \models I_i(l'_i) \wedge I_j(l'_j)$ ,

- ( $\clubsuit$ ) if  $l_k \in C_k$  for some  $k \in \{1, \dots, n\}$  then  $l_i \in C_i$  or  $l_j \in C_j$ .

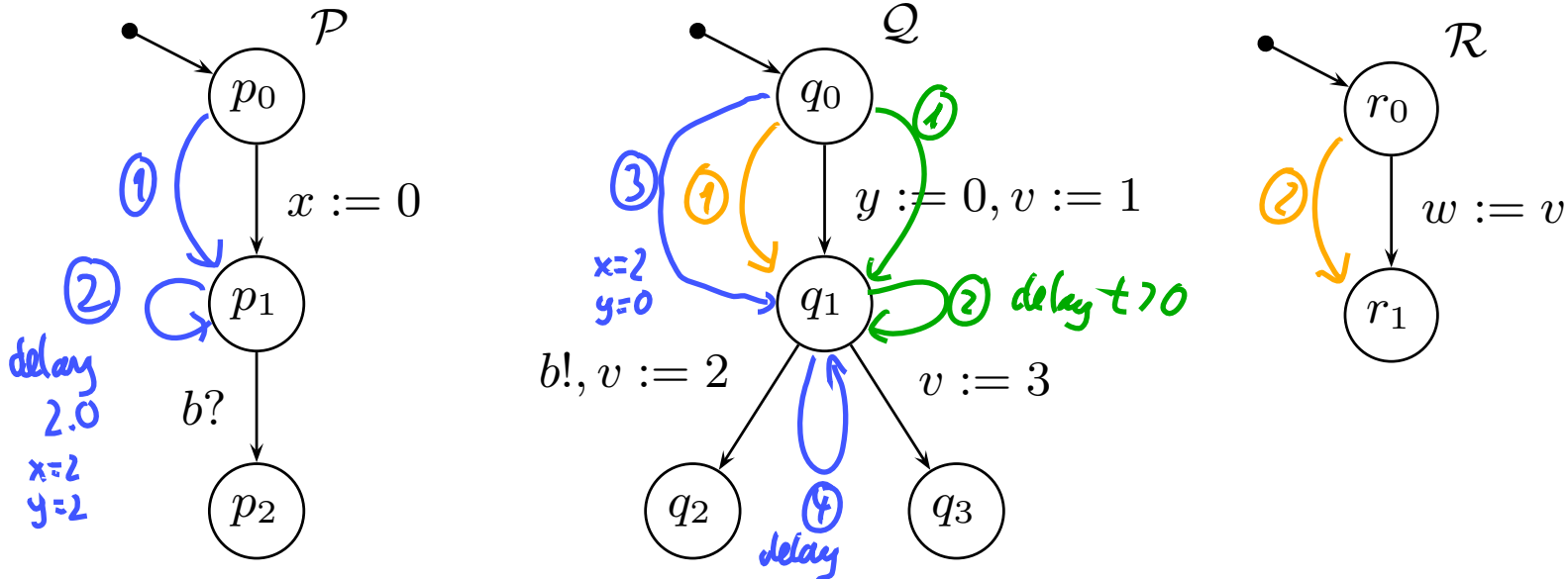


# Operational Semantics of Networks: Delay Transitions

---

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

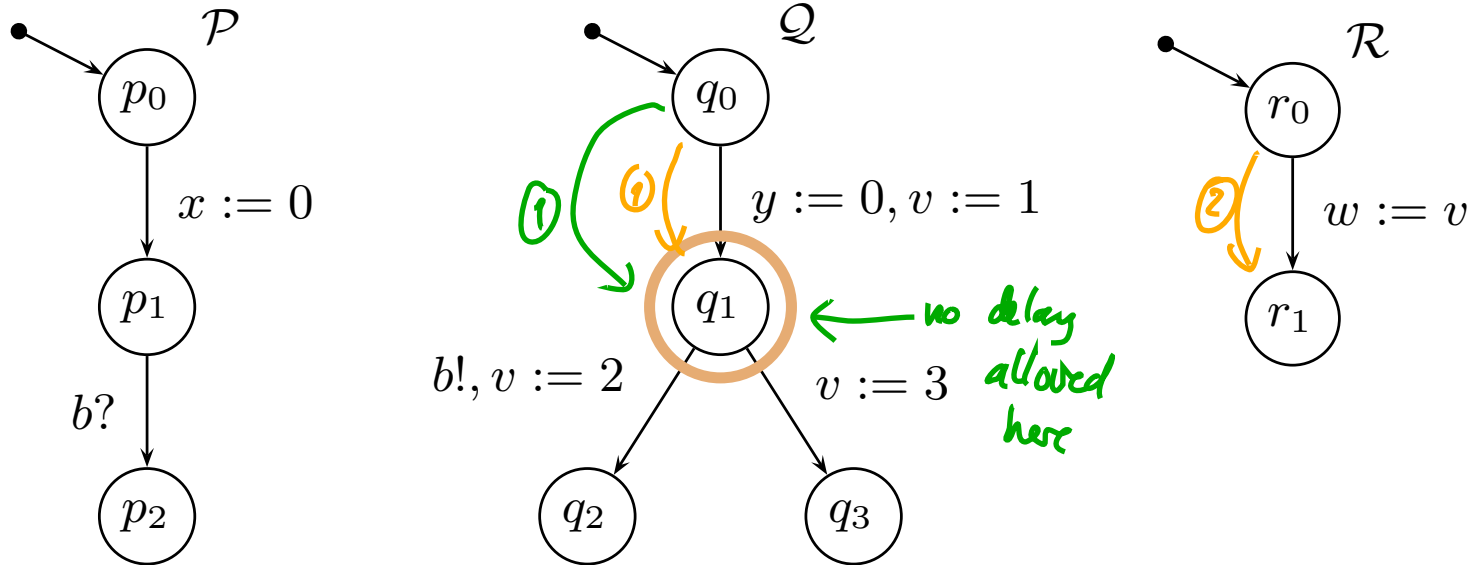
# Restricting Non-determinism: Example



∥ a config  $\langle \vec{e}, v \rangle$  with "for all reachable configs.  $\langle \vec{e}, v \rangle, v \models (Q.q_1 \Rightarrow y \leq 0)$ "  
 $v \models w=1$  is reachable  
 $Q$  in  $q_1$

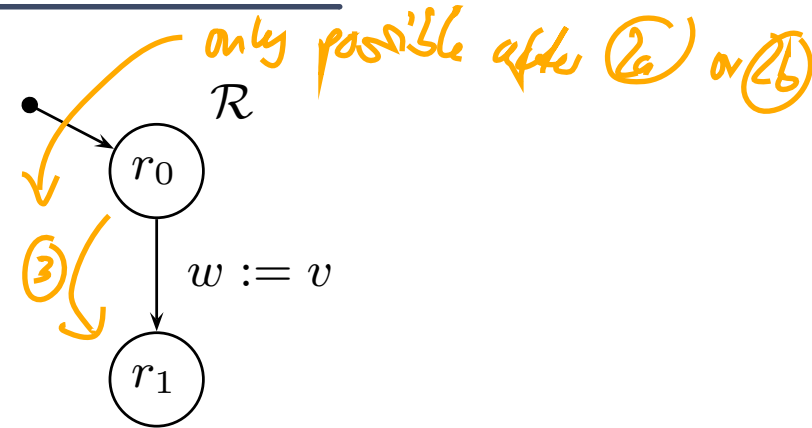
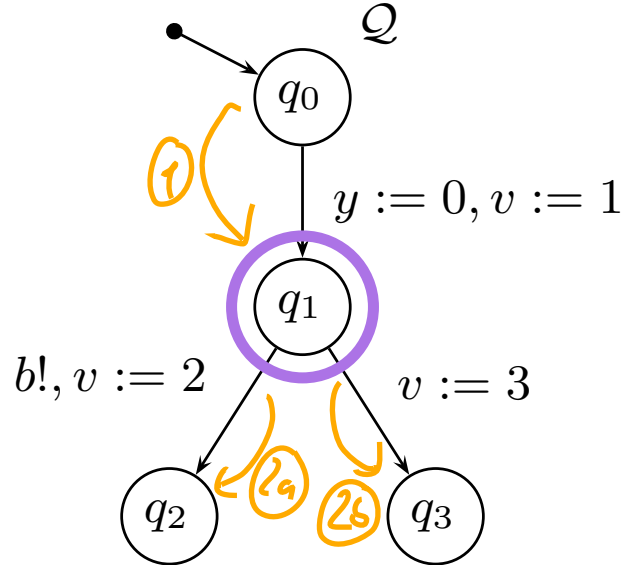
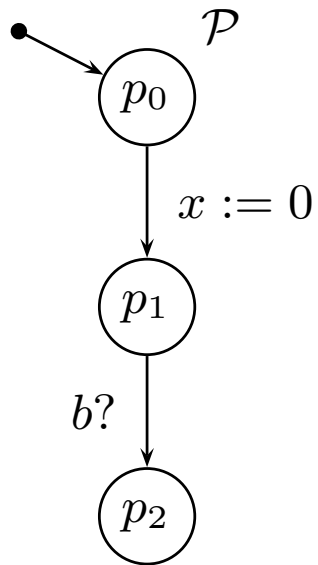
	Property 1	Property 2	Property 3
	$\exists \diamond w = 1$	$\forall \square (Q.q_1 \Rightarrow y \leq 0)$	$\forall \square (\mathcal{P}.p_1 \wedge Q.q_1 \Rightarrow (x \geq y \Rightarrow y \leq 0))$
$\mathcal{N} := \mathcal{P} \parallel \mathcal{Q} \parallel \mathcal{R}$	✓	✗	✗
$\mathcal{N}, q_1$ urgent			
$\mathcal{N}, q_1$ comm.			
$\mathcal{N}, b$ urgent			

# Restricting Non-determinism: Urgent Location



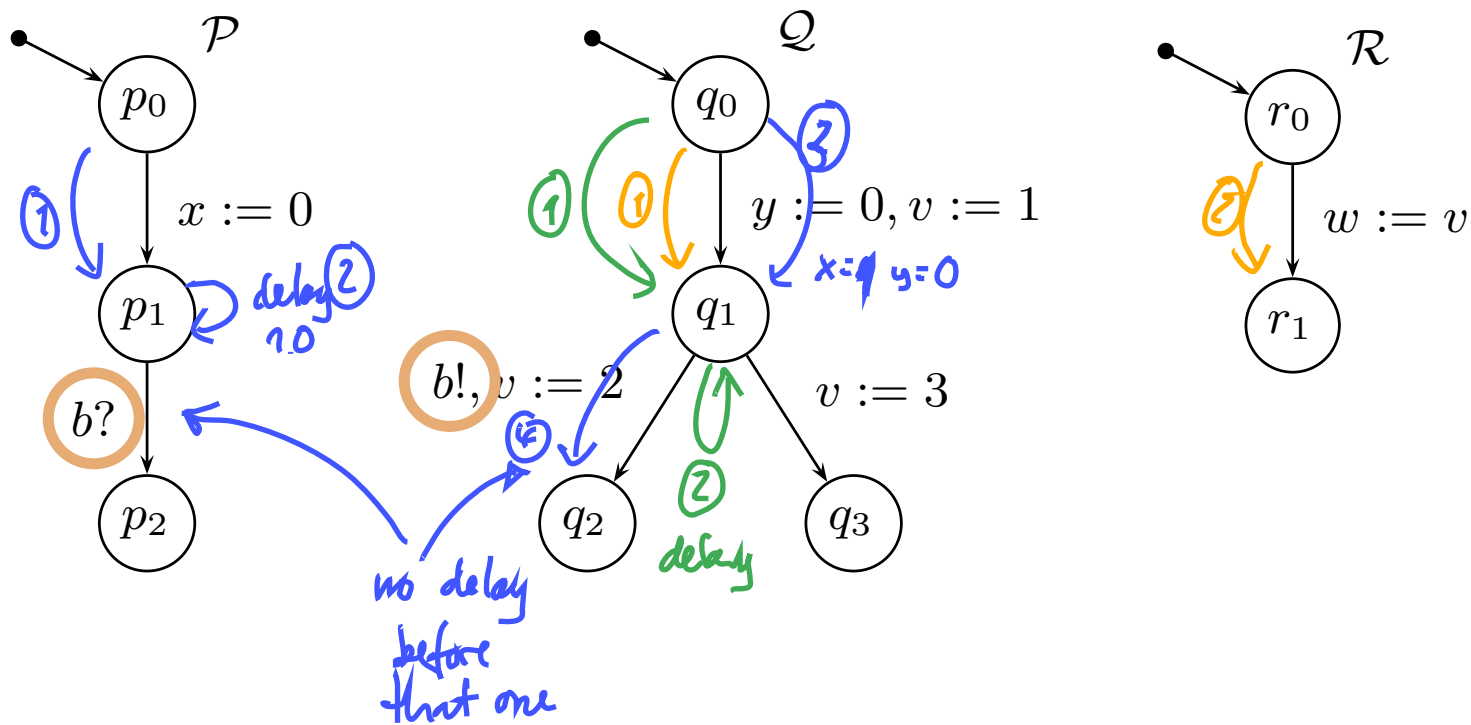
	Property 1	Property 2	Property 3
	$\exists \diamond w = 1$	$\forall \square Q.q_1 \implies y \leq 0$	$\forall \square (\mathcal{P}.p_1 \wedge Q.q_1 \implies (x \geq y \implies y \leq 0))$
$\mathcal{N}$	✓	✗	✗
$\mathcal{N}, q_1$ urgent	✓	✓	✓
$\mathcal{N}, q_1$ comm.			
$\mathcal{N}, b$ urgent			

# Restricting Non-determinism: Committed Location



	Property 1	Property 2	Property 3
	$\exists \diamond w = 1$	$\forall \square \mathcal{Q}.q_1 \implies y \leq 0$	$\forall \square (\mathcal{P}.p_1 \wedge \mathcal{Q}.q_1 \implies (x \geq y \implies y \leq 0))$
$\mathcal{N}$	✓	✗	✗
$\mathcal{N}, q_1$ urgent	✓	✓	✓
$\mathcal{N}, q_1$ comm.	✗	✓	✓
$\mathcal{N}, b$ urgent			

# Restricting Non-determinism: Urgent Channel



	Property 1	Property 2	Property 3
	$\exists \diamond w = 1$	$\forall \square \mathcal{Q}.q_1 \implies y \leq 0$	$\forall \square (\mathcal{P}.p_1 \wedge \mathcal{Q}.q_1 \implies (x \geq y \implies y \leq 0))$
$\mathcal{N}$	✓	✗	✗
$\mathcal{N}, q_1$ urgent	✓	✓	✓
$\mathcal{N}, q_1$ comm.	✗	✓	✓
$\mathcal{N}, b$ urgent	✓	✗	✓

## *Extended vs. Pure Timed Automata*

# Extended vs. Pure Timed Automata

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

$$(\ell, \alpha, \varphi, \vec{r}, \ell') \in L \times B_{!} \times \Phi(X, V) \times R(X, V)^* \times L$$

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  $V = \emptyset$ .



**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

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

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

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

# *Reachability Problems for Extended Timed Automata*

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

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

- And what about  $\text{tea}^{\sim W}$  **extended** timed automata?

# *References*

## References

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