Real-Time Systems

Lecture 15: Extended TA Cont'd, Uppaal Queries, Testable DC

2014-07-24

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- Decidability of the location reachability problem:
  - region automaton & zones
- Extended Timed Automata syntax

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's an urgent/committed location? What's the difference? Urgent channel?
  - Where has the notion of "input action" and "output action" correspondences in the formal semantics?
  - How can we relate TA and DC formulae? What's a bit tricky about that?
  - Can we use Uppaal to check whether a TA satisfies a DC formula?
- **Content:**
  - Extended TA semantics
  - The Logic of Uppaal
  - Testable DC

Extended Timed Automata

Recall: 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 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

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

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**.

Operational Semantics of Networks

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

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

where

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

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

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

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

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

Op. Sem. 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 $\ell_i \in C_i$.

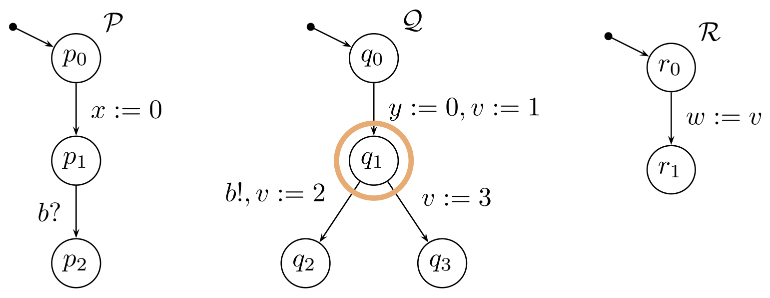
Op. Sem. of Networks: Synchronisation Transitions

- 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 $(\ell_j, b?, \varphi_j, \vec{r}_j, \ell_j') \in E_j$,
  - $\nu \models \varphi_i \wedge \varphi_j$,
  - $\vec{\ell}' = \vec{\ell}[\ell_i := \ell_i'][\ell_j := \ell_j']$,
  - $\nu' = \nu[\vec{r}_i][\vec{r}_j]$,
  - $\nu' \models I_i(\ell_i') \wedge I_j(\ell_j')$,
  - ($\clubsuit$) if $\ell_k \in C_k$ for some $k \in \{1, \dots, n\}$ then $\ell_i \in C_i$ or $\ell_j \in C_j$.

Op. Sem. of Networks: Delay Transitions

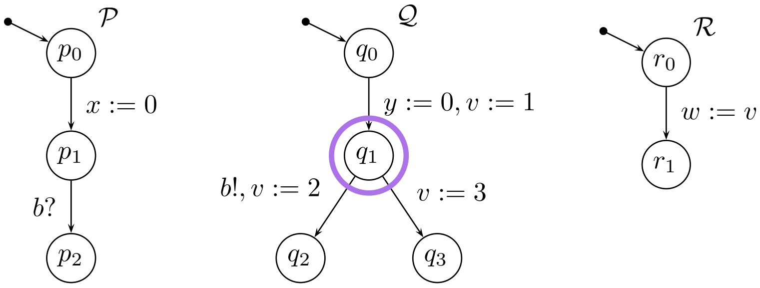
- A **delay transition** $\langle \vec{\ell}, \nu \rangle \xrightarrow{t} \langle \vec{\ell}, \nu + t \rangle$ occurs if
  - $\nu + t \models \bigwedge_{k=1}^{n} I_k(\ell_k)$,
  - (♣) there are no $i, j \in \{1, \dots, n\}$ and $b \in U$ with $(\ell_i, b!, \varphi_i, \vec{r}_i, \ell'_i) \in E_i$ and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

Restricting Non-determinism: Urgent Location



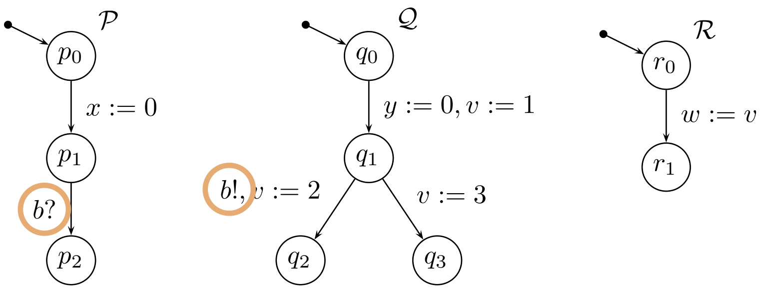
	Property 1	Property 2	Property 3
	$\exists\Diamond\, w = 1$	$\forall\Box\, \mathcal{Q}.q_1 \implies y \le 0$	$\forall\Box(\mathcal{P}.p_1 \wedge \mathcal{Q}.q_1 \implies (x \ge y \implies y \le 0))$
$\mathcal{N}$	✔	✘	✘
$\mathcal{N}$, q_1 urgent	✓	✓	✓
$\mathcal{N}$, q_1 comm.			
$\mathcal{N}$, b urgent			

Restricting Non-determinism: Committed Location



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

Restricting Non-determinism: Urgent Channel



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

Extended vs. Pure Timed Automata

Extended vs. Pure Timed Automata

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

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 $\vec{r} = \langle r_1, \ldots, r_n \rangle$, 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

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

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

Recall

> **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 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 hurt.
  - 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 (or change) the configuration space Conf.
  - Restricting non-determinism only **removes** certain transitions, so makes reachable part of the region automaton even smaller (not necessarily strictly smaller).

The Logic of Uppaal

Uppaal Fragment of Timed Computation Tree Logic

Consider $\mathcal{N} = \mathcal{C}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ 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**”)

$$e\text{-}formula ::= \exists\Diamond\, term \mid \exists\Box\, term$$

- **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**:

$$F ::= e\text{-}formula \mid a\text{-}formula$$

Configurations at Time t

- Recall: **computation path** (or path) **starting in** $\langle \vec{\ell}_0, \nu_0 \rangle, t_0$:

$$\xi = \langle \vec{\ell}_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_1} \langle \vec{\ell}_1, \nu_1 \rangle, t_1 \xrightarrow{\lambda_2} \langle \vec{\ell}_2, \nu_2 \rangle, t_2 \xrightarrow{\lambda_3} \dots$$

  which is **infinite or maximally finite**.

- Given ξ and $t \in \mathrm{Time}$, we use $\xi(t)$ to denote the set

$$\{\langle \vec{\ell}, \nu \rangle \mid \exists\, i \in \mathbb{N}_0 : t_i \leq t \leq t_{i+1} \wedge \vec{\ell} = \vec{\ell}_i \wedge \nu = \nu_i + t - t_i\}.$$

  of **configurations at time** t.

  - Why is it a set?
  - Can it be empty?

Satisfaction of Uppaal-Logic by Configurations

- We define a **satisfaction relation**

$$\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models F$$

  between **time stamped configurations**

$$\langle \vec{\ell}_0, \nu_0 \rangle, t_0$$

  of a network $\mathcal{C}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ and **formulae** F of the Uppaal logic.

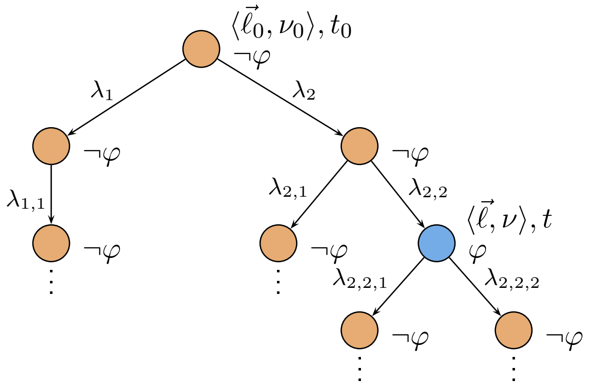
- It is defined inductively as follows:
- $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \mathcal{A}_i.\ell$ iff $\ell_{0,i} = \ell$
- $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \varphi$ iff $\nu_0 \models \varphi$
- $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \neg term$ iff $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \not\models term$
- $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models term_1 \wedge term_2$ iff $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models term_i$, $i = 1, 2$

Satisfaction of Uppaal-Logic by Configurations

Exists finally:

- $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \exists\Diamond\, term$ iff $\exists$ path ξ of $\mathcal{N}$ starting in $\langle \vec{\ell}_0, \nu_0 \rangle, t_0$
  $\exists t \in \mathsf{Time}, \langle \vec{\ell}, \nu \rangle \in Conf :$
  $t_0 \leq t \wedge \langle \vec{\ell}, \nu \rangle \in \xi(t) \wedge \langle \vec{\ell}, \nu \rangle, t \models term$

Example: $\exists\Diamond\, \varphi$

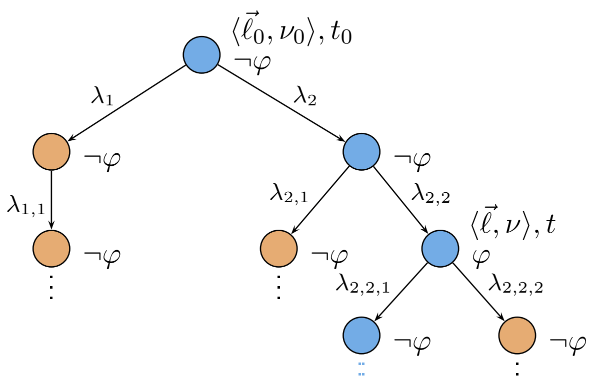


Satisfaction of Uppaal-Logic by Configurations

Exists globally:

- $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \exists\Box\, term$ iff $\exists$ path ξ of $\mathcal{N}$ starting in $\langle \vec{\ell}_0, \nu_0 \rangle, t_0$
  $\forall t \in \mathsf{Time}, \langle \vec{\ell}, \nu \rangle \in Conf :$
  $t_0 \leq t \wedge \langle \vec{\ell}, \nu \rangle \in \xi(t) \implies \langle \vec{\ell}, \nu \rangle, t \models$
  $term$

Example: $\exists\Box\, \varphi$



Satisfaction of Uppaal-Logic by Configurations

- **Always finally**:
  - $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \forall\Diamond\, term$ iff $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \not\models \exists\Box\, \neg term$
- **Always globally**:
  - $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \forall\Box\, term$ iff $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \not\models \exists\Diamond\, \neg term$

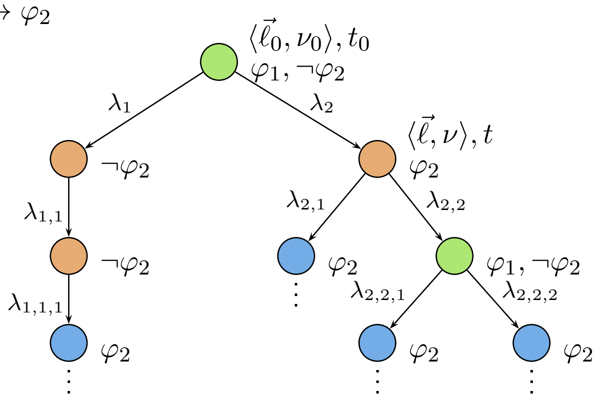
Satisfaction of Uppaal-Logic by Configurations

Leads to:

- $\langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models term_1 \longrightarrow term_2$ iff $\forall$ path ξ of $\mathcal{N}$ starting in $\langle \vec{\ell}_0, \nu_0 \rangle, t_0$
  $\forall\, t \in$ Time, $\langle \vec{\ell}, \nu \rangle \in Conf$:
  $t_0 \leq t \wedge \langle \vec{\ell}, \nu \rangle \in \xi(t)$
  $\wedge\, \langle \vec{\ell}, \nu \rangle, t \models term_1$
  implies $\langle \vec{\ell}, \nu \rangle, t \models \forall\Diamond\, term_2$

$p \longrightarrow q \rightsquigarrow$
$\forall\Box(p \Rightarrow \forall\Diamond q)$

Example: $\varphi_1 \longrightarrow \varphi_2$



Satisfaction of Uppaal-Logic by Networks

- We write $\mathcal{N} \models e\text{-}formula$ if and only if

$$\textbf{for some } \langle \vec{\ell}_0, \nu_0 \rangle \in C_{ini}, \langle \vec{\ell}_0, \nu_0 \rangle, 0 \models e\text{-}formula, \tag{1}$$

  and $\mathcal{N} \models a\text{-}formula$ if and only if

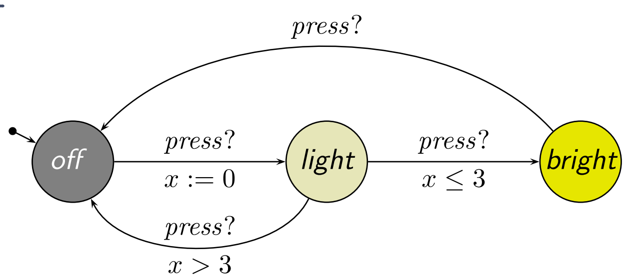
$$\textbf{for all } \langle \vec{\ell}_0, \nu_0 \rangle \in C_{ini}, \langle \vec{\ell}_0, \nu_0 \rangle, 0 \models a\text{-}formula, \tag{2}$$

  where C_{ini} are the initial configurations of $\mathcal{T}_e(\mathcal{N})$.

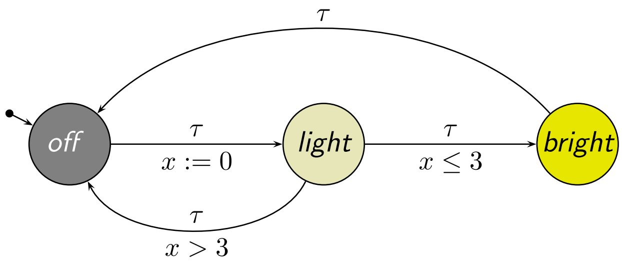
- If $C_{ini} = \emptyset$, (1) is a contradiction and (2) is a tautology.
- If $C_{ini} \neq \emptyset$, then

$$\mathcal{N} \models F \text{ if and only if } \langle \vec{\ell}_{ini}, \nu_{ini} \rangle, 0 \models F.$$

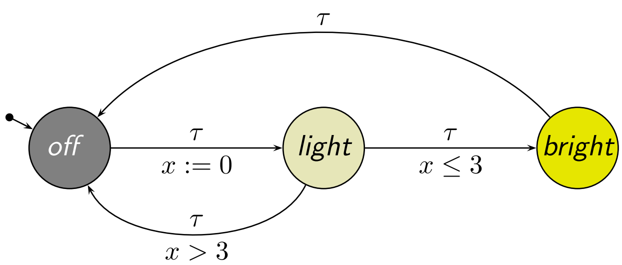
Example



Example



Example

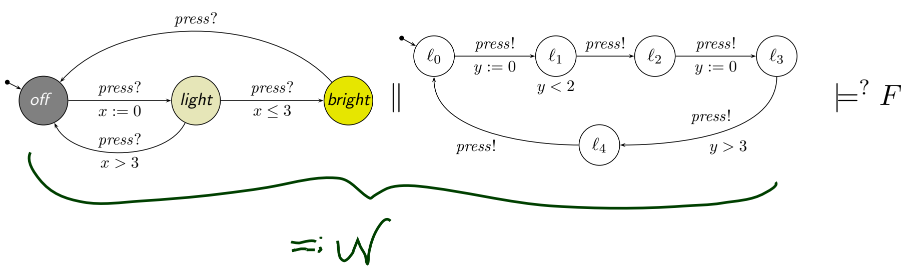


- $\mathcal{N} \models \exists\Diamond\, \mathcal{L}.bright$?
- $\mathcal{N} \models \exists\Box\, \mathcal{L}.bright$?
- $\mathcal{N} \models \exists\Box\, \mathcal{L}.off$?
- $\mathcal{N} \models \forall\Diamond\, \mathcal{L}.light$?
- $\mathcal{N} \models \forall\Box\, \mathcal{L}.bright \implies x \geq 3$?
- $\mathcal{N} \models \mathcal{L}.bright \longrightarrow \mathcal{L}.off$?

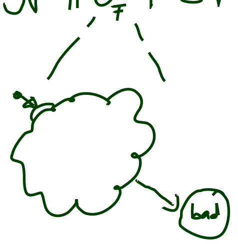
– 15 – 2014-07-24 – Sutl –
29/43

Observer-based Automatic Verification of DC Properties for TA

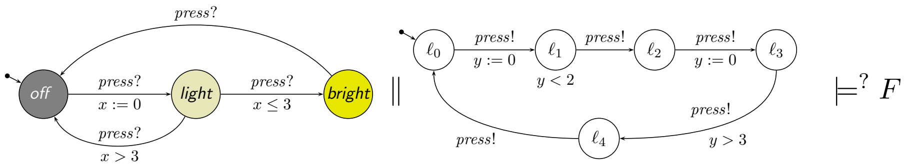
Model-Checking DC Properties with Uppaal



$\mathcal{N} \parallel \mathcal{O}_F \models \exists \Diamond\, \mathcal{O}_F.\mathit{bad}$



Model-Checking DC Properties with Uppaal



- **First Question**: what is the "$\models$" here?
- **Second Question**: what kinds of DC formulae can we check with Uppaal?
  - **Clear**: Not every DC formula.
    (Otherwise contradicting undecidability results.)
  - **Quite clear**: $F = \Box\lceil \text{off} \rceil$ or $F = \neg\Diamond\lceil \text{light} \rceil$
    (Use Uppaal's fragment of TCTL, something like $\forall\Box$ off,
    but not exactly (see later).)
  - **Maybe**: $F = \ell > 5 \implies \Diamond\lceil \text{off} \rceil^5$
  - **Not so clear**: $F = \neg\Diamond(\lceil \text{bright} \rceil \,;\, \lceil \text{light} \rceil)$

Testable DC Properties

Testability

Definition 6.1. A DC formula F is called **testable** if an observer (or test automaton (or monitor)) $\mathcal{A}_F$ exists such that for all networks $\mathcal{N} = \mathcal{C}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ it holds that

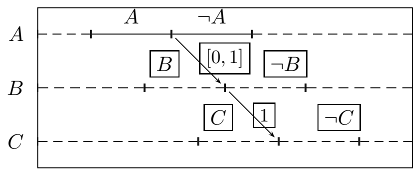
$$\mathcal{N} \models F \quad \text{iff} \quad \mathcal{C}(\mathcal{A}'_1, \dots, \mathcal{A}'_n, \mathcal{A}_F) \models \forall\Box\,\neg(\mathcal{A}_F.q_{bad})$$

Otherwise it's called **untestable**.

Proposition 6.3. There exist untestable DC formulae.

Theorem 6.4. DC implementables are testable.

Untestable DC Formulae



"Whenever we observe a change from A to $\neg A$ at time t_A,
the system has to produce a change from B to $\neg B$ at some time $t_B \in [t_A, t_A + 1]$
and a change from C to $\neg C$ at time $t_B + 1$.

Sketch of Proof: Assume there is $\mathcal{A}_F$ such that, for all networks $\mathcal{N}$, we have

$$\mathcal{N} \models F \quad \text{iff} \quad \mathcal{C}(\mathcal{A}'_1, \dots, \mathcal{A}'_n, \mathcal{A}_F) \models \forall\Box\,\neg(\mathcal{A}_F.q_{bad})$$

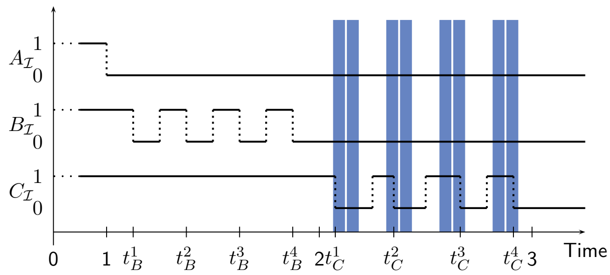
Assume the number of clocks in $\mathcal{A}_F$ is $n \in \mathbb{N}_0$.

Untestable DC Formulae Cont'd

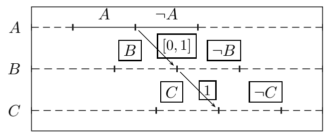
Consider the following time points:

- $t_A := 1$
- $t_B^i := t_A + \frac{2i-1}{2(n+1)}$ for $i = 1, \dots, n+1$
- $t_C^i \in \left]t_B^i + 1 - \frac{1}{4(n+1)}, t_B^i + 1 + \frac{1}{4(n+1)}\right[$ for $i = 1, \dots, n+1$
  with $t_C^i - t_B^i \neq 1$ for $1 \leq i \leq n+1$.

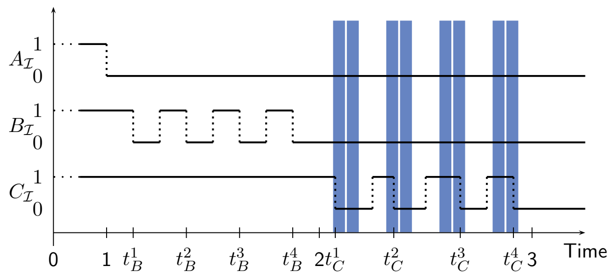
Example: $n = 3$



Untestable DC Formulae Cont'd

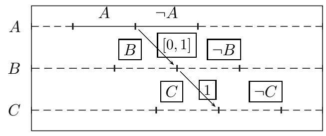


Example: $n = 3$

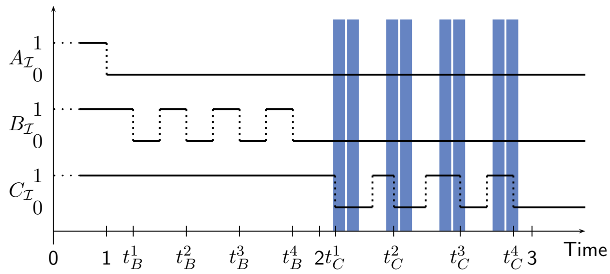


- The shown interpretation $\mathcal{I}$ satisfies **assumption** of property.
- It has $n+1$ candidates to satisfy **commitment**.
- By choice of t_C^i, the commitment is not satisfied; so F not satisfied.
- Because $\mathcal{A}_F$ is a test automaton for F, is has a computation path to q_{bad}.
- Because $n = 3$, $\mathcal{A}_F$ can not save all $n+1$ time points t_B^i.
- Thus there is $1 \leq i_0 \leq n$ such that all clocks of $\mathcal{A}_F$ have a valuation which is not in $2 - t_B^{i_0} + (-\frac{1}{4(n+1)}, \frac{1}{4(n+1)})$

Untestable DC Formulae Cont'd



Example: $n = 3$



- Because $\mathcal{A}_F$ is a test automaton for F, is has a computation path to q_{bad}.
- Thus there is $1 \leq i_0 \leq n$ such that all clocks of $\mathcal{A}_F$ have a valuation which is not in $2 - t_B^{i_0} + (-\frac{1}{4(n+1)}, \frac{1}{4(n+1)})$
- Modify the computation to $\mathcal{I}'$ such that $t_C^{i_0} := t_B^{i_0} + 1$.
- Then $\mathcal{I}' \models F$, but $\mathcal{A}_F$ reaches q_{bad} via the same path.
- That is: $\mathcal{A}_F$ claims $\mathcal{I}' \not\models F$.
- Thus $\mathcal{A}_F$ is not a test automaton. **Contradiction**.

Testable DC Formulae

Theorem 6.4. DC implementables are testable.

- **Initialisation**: $\lceil\,\rceil \lor \lceil \pi \rceil \,;\, true$
- **Sequencing**: $\lceil \pi \rceil \longrightarrow \lceil \pi \lor \pi_1 \lor \dots \lor \pi_n \rceil$
- **Progress**: $\lceil \pi \rceil \xrightarrow{\theta} \lceil \neg \pi \rceil$
- **Synchronisation**: $\lceil \pi \land \varphi \rceil \xrightarrow{\theta} \lceil \neg \pi \rceil$
- **Bounded Stability**: $\lceil \neg \pi \rceil \,;\, \lceil \pi \land \varphi \rceil \xrightarrow{\leq \theta} \lceil \pi \lor \pi_1 \lor \dots \lor \pi_n \rceil$
- **Unbounded Stability**: $\lceil \neg \pi \rceil \,;\, \lceil \pi \land \varphi \rceil \longrightarrow \lceil \pi \lor \pi_1 \lor \dots \lor \pi_n \rceil$
- **Bounded initial stability**: $\lceil \pi \land \varphi \rceil \xrightarrow{\leq \theta}_0 \lceil \pi \lor \pi_1 \lor \dots \lor \pi_n \rceil$
- **Unbounded initial stability**: $\lceil \pi \land \varphi \rceil \longrightarrow_0 \lceil \pi \lor \pi_1 \lor \dots \lor \pi_n \rceil$

Proof Sketch:

- For each implementable F, construct $\mathcal{A}_F$.
- Prove that $\mathcal{A}_F$ is a test automaton.

Proof of Theorem 6.4: Preliminaries

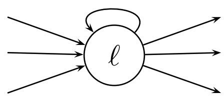
- **Note**: DC does not refer to communication between TA in the network, but only to data variables and locations.

  Example:

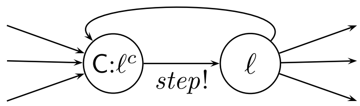
  $$\Diamond(\lceil v = 0 \rceil \,;\, \lceil v = 1 \rceil)$$

- **Recall**: transitions of TA are only triggered by syncronisation, not by changes of data-variables.
- **Approach**: have auxiliary *step* action.

  Technically, replace each

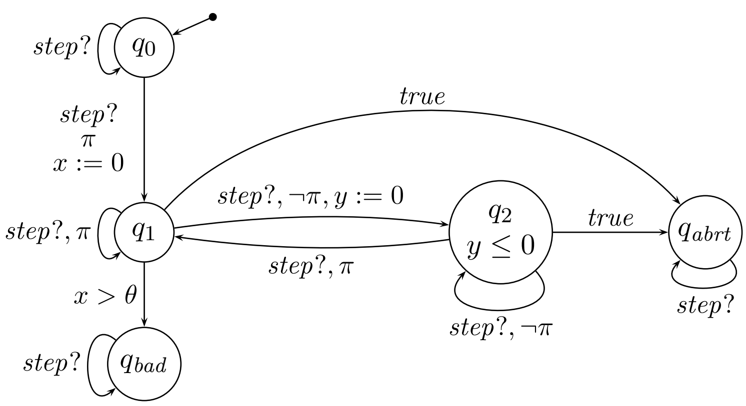


by



Proof of Theorem 6.4: Sketch

- Example: $\lceil \pi \rceil \xrightarrow{\theta} \lceil \neg\pi \rceil$



Counterexample Formulae

Definition 6.5.

- A **counterexample formula** (CE for short) is a DC formula of the form:

  $$true \,;\, (\lceil \pi_1 \rceil \wedge \ell \in I_1) \,;\, \ldots \,;\, (\lceil \pi_k \rceil \wedge \ell \in I_k) \,;\, true$$

  where for $1 \leq i \leq k$,

  - π_i are state assertions,
  - I_i are non-empty, and open, half-open, or closed time intervals of the form
    - (b, e) or $[b, e)$ with $b \in \mathbb{Q}_0^+$ and $e \in \mathbb{Q}_0^+ \dot\cup \{\infty\}$,
    - $(b, e]$ or $[b, e]$ with $b, e \in \mathbb{Q}_0^+$.

    (b, ∞) and $[b, \infty)$ denote unbounded sets.

- Let F be a DC formula. A DC formula F_{CE} is called **counterexample formula for** F if $\models F \iff \neg(F_{CE})$ holds.

Counterexample Formulae

Definition 6.5.

- A **counterexample formula** (CE for short) is a DC formula of the form:

  $$true \,;\, (\lceil \pi_1 \rceil \wedge \ell \in I_1) \,;\, \ldots \,;\, (\lceil \pi_k \rceil \wedge \ell \in I_k) \,;\, true$$

  where for $1 \leq i \leq k$,

  - π_i are state assertions,
  - I_i are non-empty, and open, half-open, or closed time intervals of the form
    - (b, e) or $[b, e)$ with $b \in \mathbb{Q}_0^+$ and $e \in \mathbb{Q}_0^+ \dot\cup \{\infty\}$,
    - $(b, e]$ or $[b, e]$ with $b, e \in \mathbb{Q}_0^+$.

    (b, ∞) and $[b, \infty)$ denote unbounded sets.

- Let F be a DC formula. A DC formula F_{CE} is called **counterexample formula for** F if $\models F \iff \neg(F_{CE})$ holds.

– 15 – 2014-07-24 – Sdctest –

41/43

References

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