Real-Time Systems

Lecture 15: Automatic Verification of DC Properties for TA

2012-07-12

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- Extended Timed Automata
- Uppaal Query Language

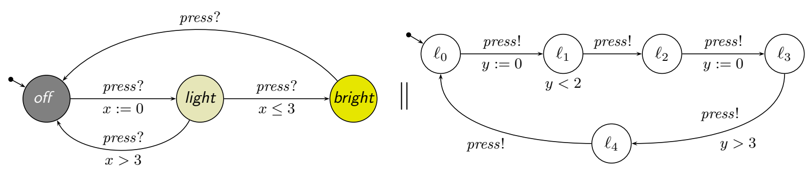
This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - 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:**
  - An evolution-of-observables semantics of TA
  - A satisfaction relation between TA and DC
  - Model-checking DC properties with Uppaal

Observer-based Automatic Verification of DC Properties for TA

Model-Checking DC Properties with Uppaal

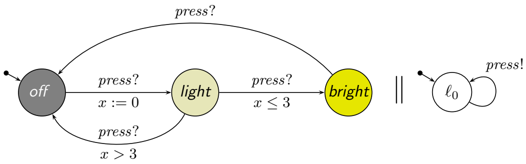
DC formula



$\models^? F$

- **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**: $F = \Box\lceil \mathsf{off} \rceil$ or $F = \neg\Diamond\lceil \mathsf{light} \rceil$ $\sim \forall\Box\, L.\mathit{off}$? $\sim \forall\Box\, \neg L.\mathit{light}$?
    (Use Uppaal's fragment of TCTL, something like $\forall\Box$ off,
    but not exactly (see later).)
  - **Maybe**: $F = \Box(\ell > 5 \implies \Diamond\lceil \mathsf{off} \rceil^5)$
  - **Not so clear**: $F = \neg\Diamond(\lceil \mathsf{bright} \rceil \,;\, \lceil \mathsf{light} \rceil)$

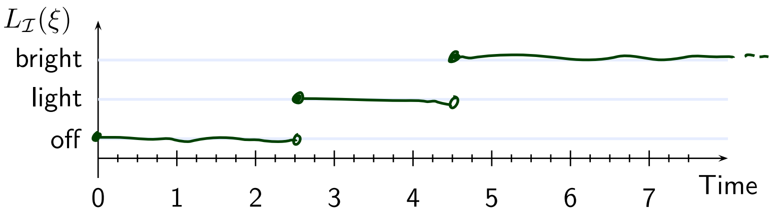
Example: Let's Start With Single Runs



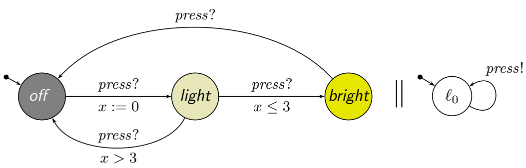
$\models^{?} \neg\Diamond\lceil \mathsf{light} \rceil$

$$\xi = \langle {\mathsf{off} \atop 0} \rangle, 0 \xrightarrow{2.5} \langle {\mathsf{off} \atop 2.5} \rangle, 2.5 \xrightarrow{\tau} \langle {\mathsf{light} \atop 0} \rangle, 2.5 \xrightarrow{2.0} \langle {\mathsf{light} \atop 2.0} \rangle, 4.5 \xrightarrow{\tau} \langle {\mathsf{bright} \atop 2.0} \rangle, 4.5 \ldots$$

Construct interpretation $L_{\mathcal{I}}(\xi)$: Time $\rightarrow$ {off, light, bright}:

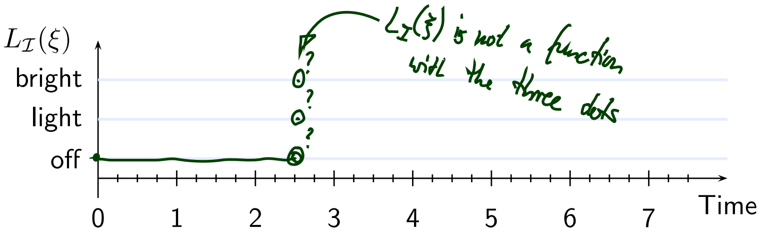


Example 2: Another Single Run



$\models^{?} \neg\Diamond\lceil \mathsf{light} \rceil$

$$\xi = \langle {\mathsf{off} \atop 0} \rangle, 0 \xrightarrow{2.5} \langle {\mathsf{off} \atop 2.5} \rangle, 2.5 \xrightarrow{\tau} \langle {\mathsf{light} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\mathsf{bright} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\mathsf{off} \atop 0} \rangle, 2.5 \xrightarrow{1.0} \ldots$$

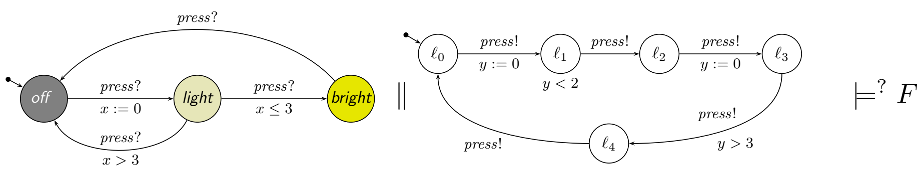


$L_{\mathcal{I}}(\xi)$ is not a function with the three dots

We know this problem from the exercises...

Observing Timed Automata

DC Properties of Timed Automata



Wanted: A satisfaction relation between networks of timed automata and DC formulae, a notion of $\mathcal{N}$ **satisfies** F, denoted by $\mathcal{N} \models F$.

Plan:

- Consider network $\mathcal{N}$ consisting of TA

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

- Define observables Obs($\mathcal{N}$) of $\mathcal{N}$.
- Define evolution $\mathcal{I}_\xi$ of Obs($\mathcal{N}$) induced by computation path $\xi \in CompPaths(\mathcal{N})$ of $\mathcal{N}$,
  $CompPaths(\mathcal{N}) = \{\xi \mid \xi$ is a computation path of $\mathcal{N}\}$
- Say $\mathcal{N} \models F$ if and only if $\forall\, \xi \in CompPaths(\mathcal{N}) : \mathcal{I}_\xi \models_0 F$.

Observables of TA Network

Let $\mathcal{N}$ be a network of n extended timed automata

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

For simplicity: assume that the L_i and X_i are pairwise disjoint and that each V_i is pairwise disjoint to every L_i and X_i (otherwise rename).

- **Definition**: The observables $\mathsf{Obs}(\mathcal{N})$ of $\mathcal{N}$ are

$$\{\ell_1, \dots, \ell_n\} \cup \bigcup_{1 \leq i \leq n} V_i$$

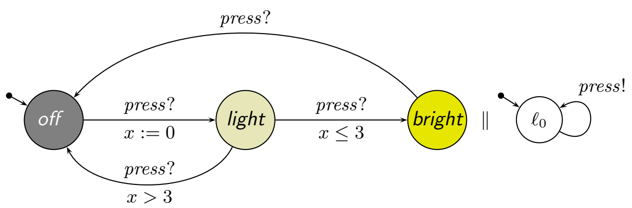
current location of $\mathcal{A}_{e,1}$
(would be less confusing if
we used {☺$_1$, …, ☺$_n$}

with
- $\mathcal{D}(\ell_i) = L_i$,
- $\mathcal{D}(v)$ as given, $v \in V_i$.

Observables of TA Network: Example

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

The observables $\mathsf{Obs}(\mathcal{N})$ of $\mathcal{N}$ are $\{\ell_1, \dots, \ell_n\} \cup \bigcup_{1 \leq i \leq n} V_i$ with
- $\mathcal{D}(\ell_i) = L_i$,
- $\mathcal{D}(v)$ as given, $v \in V_i$.



- ℓ_1: off, light, bright
- ℓ_2: ℓ_0

Evolutions of TA Network

Recall: computation path

$$\xi = \langle \vec{\ell}_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_1} \langle \vec{\ell}_1, \nu_1 \rangle, t_1 \xrightarrow{\lambda_2} \langle \vec{\ell}_2, \nu_2 \rangle, t_2 \xrightarrow{\lambda_3} \dots \quad j=2$$

of $\mathcal{N}$, $\vec{\ell}_k$ denotes a tuple $\langle \ell^1_k, \dots, \ell^n_k \rangle \in L_1 \times \dots \times L_n$.

Recall: Given ξ and $t \in$ 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.

New: $\bar{\xi}(t)$ denotes $\langle \vec{\ell}_j, \nu_j + t - t_j \rangle$ where $j = \max\{i \in \mathbb{N}_0 \mid t_i \leq t \wedge \vec{\ell} = \vec{\ell}_i\}$.

Our choice:

- **Ignore** configurations assumed for 0-time only.
- **Extend** finite computation paths to infinite length, staying in last configuration.
  Yet clocks advance – see later.

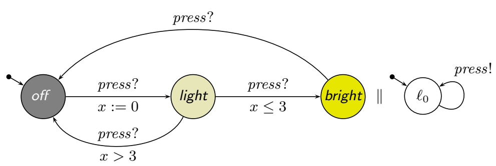
Evolutions of TA Network: Example

$\boxed{\bar{\xi}(t) \text{ denotes } \langle \vec{\ell}_j, \nu_j + t - t_j \rangle \text{ where } j = \max\{i \in \mathbb{N}_0 \mid t_i \leq t \wedge \vec{\ell} = \vec{\ell}_i\}.}$

Example:

$$\xi = \langle {\text{off} \atop 0} \rangle, 0 \xrightarrow{2.5} \langle {\text{off} \atop 2.5} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{light} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{bright} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{off} \atop 0} \rangle, 2.5 \xrightarrow{1.0} \langle {\text{off} \atop 1} \rangle, 3.5 \xrightarrow{\tau} \dots$$

- $\bar{\xi}(0) =$
- $\bar{\xi}(1.0) = \langle {\text{off} \atop x=1} \rangle$
- $\bar{\xi}(2.5) =$



Evolutions of TA Network Cont'd

$\bar{\xi}$ **induces** the unique interpretation

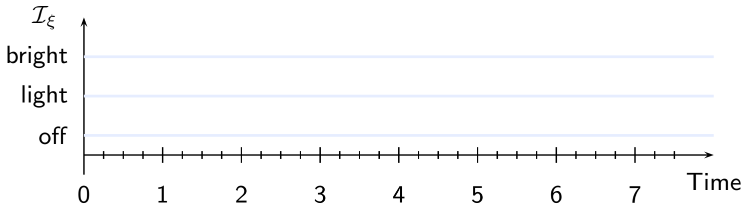
$$\mathcal{I}_\xi : \mathsf{Obs}(\mathcal{N}) \to (\mathsf{Time} \to \mathcal{D})$$

of $\mathsf{Obs}(\mathcal{N})$ defined pointwise as follows:

$$\mathcal{I}_\xi(a)(t) = \begin{cases} \ell^i & \text{, if } a = \ell_i,\ \bar{\xi}(t) = \langle\langle \ell^1, \dots, \ell^n \rangle, \nu\rangle \\ \nu(a) & \text{, if } a \in V_i,\ \bar{\xi}(t) = \langle \vec{\ell}, \nu \rangle \end{cases}$$

Example: $\mathcal{D}(\ell_1) = \{\text{off}, \text{light}, \text{bright}\}$

$$\xi = \langle {\text{off} \atop 0} \rangle, 0 \xrightarrow{2.5} \langle {\text{off} \atop 2.5} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{light} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{bright} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{off} \atop 0} \rangle, 2.5 \xrightarrow{1.0} \langle {\text{off} \atop 1} \rangle, 3.5 \xrightarrow{\tau} \dots$$



Evolutions of TA Network Cont'd

$$\xi = \langle {\text{off} \atop 0} \rangle, 0 \xrightarrow{2.5} \langle {\text{off} \atop 2.5} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{light} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{bright} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\text{off} \atop 0} \rangle, 2.5 \xrightarrow{1.0} \langle {\text{off} \atop 1} \rangle, 3.5 \xrightarrow{\tau} \dots$$

current loc. of 1st automaton

Abbreviations as usual:

- $\mathcal{I}_\xi(\ell_1)(0) =$ off (by Def.)
- $\mathcal{I}(\ell_1 = \text{off})(0) = \mathcal{I}_\xi(\ell_1)(0) = \mathcal{I}(\text{off})$
- $\mathcal{I}(\text{off})(1.0) = \mathcal{I}(\ell_1 = \text{off})(1.0)$
  if L_i pairwise disjoint.

Evolutions of TA Network Cont'd

- But **what about clocks**? Why not $x \in \mathsf{Obs}(\mathcal{N})$ for $x \in X_i$?
- We would know how to define $\mathcal{I}_\xi(x)(t)$, namely

$$\mathcal{I}_\xi(x)(t) = \nu_{\xi(t)}(x) + (t - t_{\xi(t)}).$$

a boolean observable

- But... $\mathcal{I}_\xi(x)(t)$ changes too often.

$\Box(\lceil off \rceil \Rightarrow \boxed{x \leq 5})$

Better (if wanted):

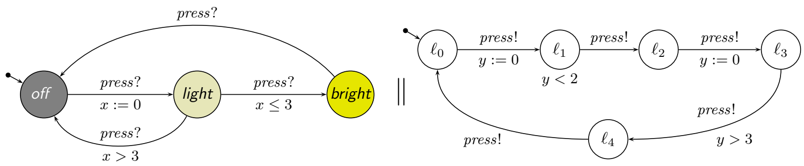
- add $\Phi(X_1 \cup \cdots \cup X_i)$ to $\mathsf{Obs}(\mathcal{N})$,
  with $\mathcal{D}(\varphi) = \{0, 1\}$ for $\varphi \in \Phi(X_1 \cup \cdots \cup X_i)$.
- set

$$\mathcal{I}_\xi(\varphi)(t) = \begin{cases} 1, \text{ if } \nu(x) \models \varphi, \bar{\xi}(t) = \langle \vec{\ell}, \nu \rangle \\ 0, \text{ otherwise} \end{cases}$$

The truth value of constraint φ can endure over non-point intervals.

Some Checkable Properties

Model-Checking DC Properties with Uppaal

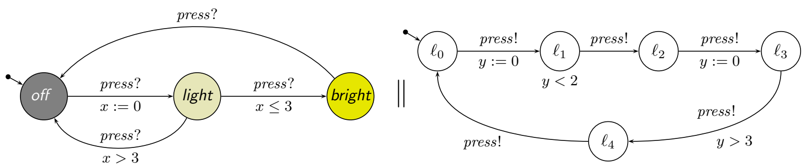


$\models^{?} F$

- **First Answer**: $\mathcal{N} \models F$ if and only if $\forall\, \xi \in \mathit{CompPaths}(\mathcal{N}) : \mathcal{I}_\xi \models_0 F$.
- **Second Question**: what kinds of DC formulae can we check with Uppaal?
  - **Clear**: Not every DC formula. (Otherwise contradicting undecidability results.)
  - **Quite clear**: $F = \Box\lceil \mathsf{off} \rceil$ or $F = \neg\Diamond\lceil \mathsf{light} \rceil$

    (Use Uppaal's fragment of TCTL, something like $\forall\Box$ off, but not exactly (see later).)
  - **Maybe**: $F = \Box(\ell > 5 \implies \Diamond\lceil \mathsf{off} \rceil^5)$
  - **Not so clear**: $F = \neg\Diamond(\lceil \mathsf{bright} \rceil \,;\, \lceil \mathsf{light} \rceil)$

Model-Checking DC Properties with Uppaal



$\models^{?} F$

- **Second Question**: what kinds of DC formulae can we check with Uppaal?

  Wanted:
  - a function f mapping DC formulae to Uppaal queries and
  - a transformation $\tilde{\cdot}$ of networks of TA

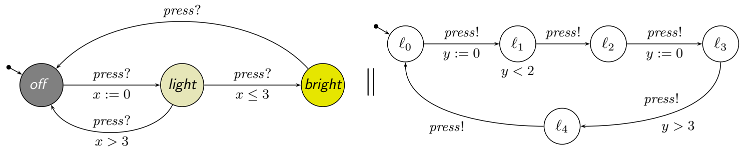
  such that

$$\widetilde{\mathcal{N}} \models_{\mathsf{Uppaal}} f(F) \iff \mathcal{N} \models F$$

One step more general: an additional **observer** construction $\mathcal{O}(\,\cdot\,)$ such that

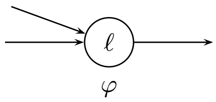
$$\widetilde{\mathcal{N}} \parallel \mathcal{O}(F) \models_{\mathsf{Uppaal}} f_{\mathcal{O}}(F) \iff \mathcal{N} \models F$$

Model-Checking Invariants with Uppaal

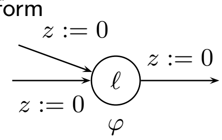


$\models^? F$

- **Quite clear**: $F = \Box\lceil P \rceil$.
  - Unfortunately, we have
  $$\mathcal{N} \models \forall\Box\, P \implies \mathcal{N} \models \Box\lceil P \rceil$$
  but in general not
  $$\mathcal{N} \models \Box\lceil P \rceil \implies \mathcal{N} \models \forall\Box\, P,$$
  because Uppaal also considers violations of P without duration, at points.
  - Possible fix: measure duration explicitly, transform

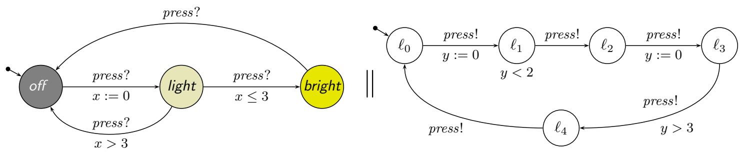


to



Then check for $\mathcal{N} \models \forall\Box(z > 0 \implies P)$.

Model-Checking Certain Durations with Uppaal

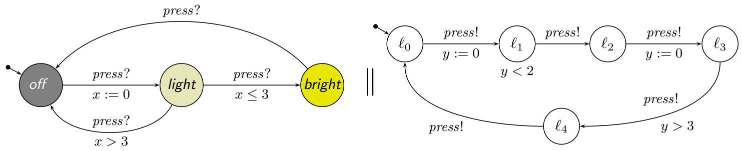


$\models^? F$

- **Maybe**: $F = \Box(\ell > t_0 \implies \Diamond\lceil P \rceil^t_1)$, $t > 0$

Check for
$$\mathcal{N} \models \forall\Diamond(P \wedge z_0 > t_0 \wedge z_1 = t_1).$$

Model-Checking Certain Chops with Uppaal



$\models^? F$

- **Not so clear**: $F = \neg\Diamond(\lceil \mathsf{bright} \rceil \,;\, \lceil \mathsf{light} \rceil)$ (Expectation? Holds or not?)

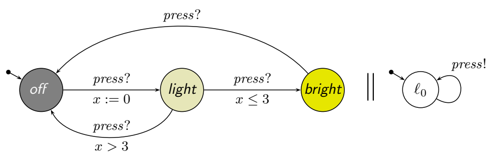
  Off-hand approach:

  - Add two auxiliary duration clocks z_{light} and z_{bright}.
  - Add auxiliary variable $\mathtt{prev}$ with $\mathcal{D}(\mathtt{prev}) = \{\mathsf{off}, \mathsf{light}, \mathsf{bright}\}$ keeping track of where we came from.
  - Observe: $\lceil \mathsf{bright} \rceil \,;\, \lceil \mathsf{light} \rceil$ means "get from bright directly to light".
  - Check for

$$\mathcal{N} \models \forall\Diamond(\mathcal{L}.\mathsf{light} \wedge z_{\mathsf{light}} > 0 \wedge z_{\mathsf{bright}} > 0 \wedge \mathtt{prev} = \mathsf{bright})$$

  - **Exercise**: Prove $\mathcal{N} \models F \iff \widetilde{\mathcal{N}} \models f(F)$.

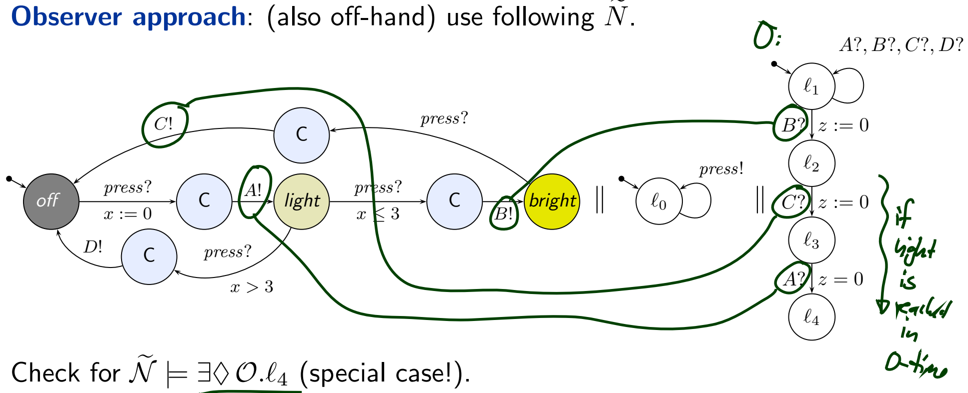
Model-Checking Certain Chops with Uppaal



$\models^? F$

- **Not so clear**: $F = \neg\Diamond(\lceil \mathsf{bright} \rceil \,;\, \lceil \mathsf{light} \rceil)$ (Expectation? Holds or not?)

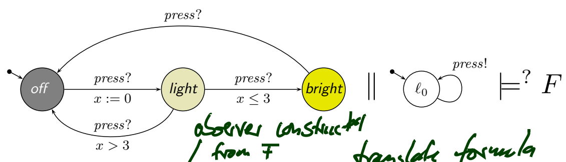
  Observer approach: (also off-hand) use following $\widetilde{N}$.



Check for $\widetilde{\mathcal{N}} \models \exists\Diamond\, \mathcal{O}.\ell_4$ (special case!).

Testable DC Properties

A More Systematic Approach



observer constructed from F

translate formula into query

- We have seen $f_{\mathcal{O}}$, $\widetilde{\cdot}$, and $\mathcal{O}(\,\cdot\,)$ with

modification of $\mathcal{N}$

$$\widetilde{\mathcal{N}} \parallel \mathcal{O}(F) \models_{\mathsf{Uppaal}} f_{\mathcal{O}}(F) \iff \mathcal{N} \models F \qquad (*)$$

  for **some particular** F. **Tedious**: always have to prove $(*)$.

- **Better**:
  - characterise a subset (or fragment) of DC,
  - give procedures to construct $f_{\mathcal{O}}(\,\cdot\,)$, $\widetilde{\cdot}$, and $\mathcal{O}(\,\cdot\,)$
  - prove once and for all that, if F is in this fragment, then

$$\widetilde{\mathcal{N}} \parallel \mathcal{O}(F) \models_{\mathsf{Uppaal}} f_{\mathcal{O}}(F) \iff \mathcal{N} \models F$$

- **Even better**: exact (syntactic) characterisation of the DC fragment that is testable (not in the lecture).

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

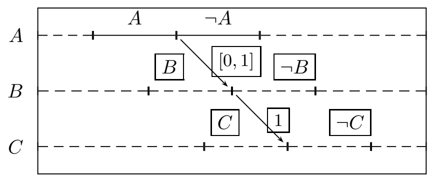
$$\mathcal{N} \models F \quad \text{iff} \quad \mathcal{C}(\mathcal{A}'_1, \dots, \mathcal{A}'_n, \mathcal{A}_F) \models \forall \Box \neg(\mathcal{A}_F.q_{bad})$$

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

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

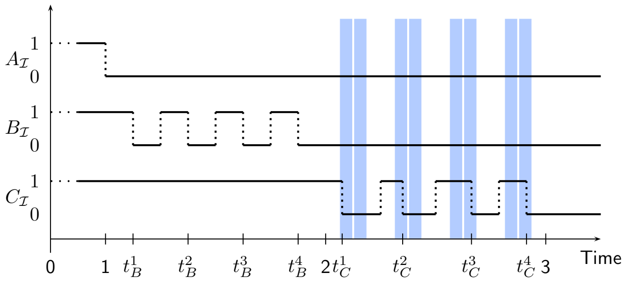
Assume the number of clocks in $\mathcal{A}_F$ is $n \in \mathbb{N}_0$.

Untestable DC Formulae Cont'd

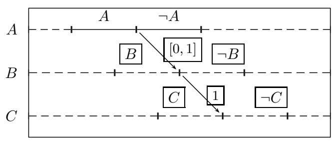
Consider the following time points:

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

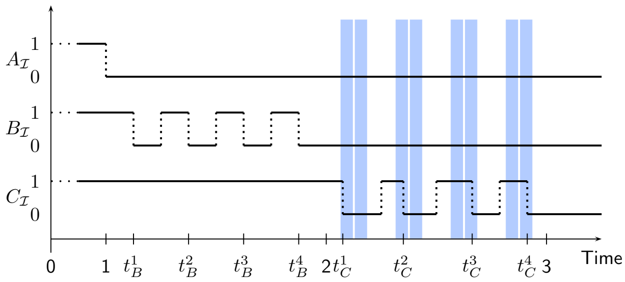
Example: $n = 3$



Untestable DC Formulae Cont'd

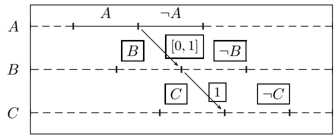


Example: $n = 3$

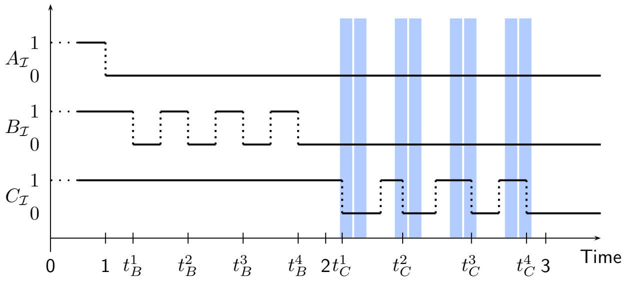


- The shown interpretation $\mathcal{I}$ satisfies **assumption** of property.
- It has $n + 1$ candidates to satisfy **commitment**.
- By choice of t_C^i, the commitment is not satisfied; so F not satisfied.
- Because $\mathcal{A}_F$ is a test automaton for F, is has a computation path to q_{bad}.
- 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**: $\lceil\rceil \vee \lceil \pi \rceil \,;\, true$
- **Sequencing**: $\lceil \pi \rceil \longrightarrow \lceil \pi \vee \pi_1 \vee \dots \vee \pi_n \rceil$
- **Progress**: $\lceil \pi \rceil \xrightarrow{\theta} \lceil \neg\pi \rceil$
- **Synchronisation**: $\lceil \pi \wedge \varphi \rceil \xrightarrow{\theta} \lceil \neg\pi \rceil$
- **Bounded Stability**: $\lceil \neg\pi \rceil \,;\, \lceil \pi \wedge \varphi \rceil \xrightarrow{\leq\theta} \lceil \pi \vee \pi_1 \vee \dots \vee \pi_n \rceil$
- **Unbounded Stability**: $\lceil \neg\pi \rceil \,;\, \lceil \pi \wedge \varphi \rceil \longrightarrow \lceil \pi \vee \pi_1 \vee \dots \vee \pi_n \rceil$
- **Bounded initial stability**: $\lceil \pi \wedge \varphi \rceil \xrightarrow{\leq\theta}_0 \lceil \pi \vee \pi_1 \vee \dots \vee \pi_n \rceil$
- **Unbounded initial stability**: $\lceil \pi \wedge \varphi \rceil \longrightarrow_0 \lceil \pi \vee \pi_1 \vee \dots \vee \pi_n \rceil$

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

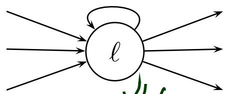
- **Note**: DC does not refer to communication between TA in the network, but only to data variables and locations (and clock constraints, if added).

  Example:

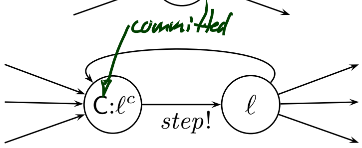
$$\Diamond(\lceil v = 0 \rceil \,;\, \lceil v = 1 \rceil)$$

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

  Technically, replace each



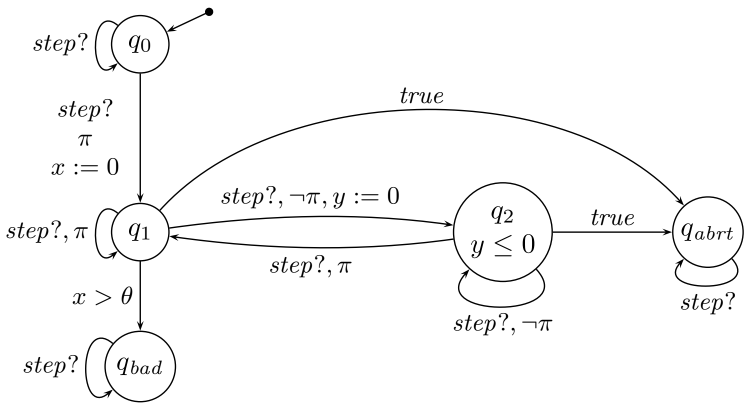
by



Note: the observer sees the data variables **after** the update.

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:

  $$\mathit{true} \,;\, (\lceil \pi_1 \rceil \wedge \ell \in I_1) \,;\, \dots \,;\, (\lceil \pi_k \rceil \wedge \ell \in I_k) \,;\, \mathit{true}$$

  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.

Theorem 6.7. CE formulae are testable.

– 15 – 2012-07-12 – Sdctest –

References

– 15 – 2012-07-12 – main –

References

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