Real-Time Systems

Lecture 17: Automatic Verification of DC Properties for TA II

2018-01-18

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

Introduction

- **Observables and Evolutions**
- **Duration Calculus** (DC)
- Semantical Correctness Proofs
- DC Decidability
- DC Implementables
- **PLC-Automata**

$obs : \mathsf{Time} \to \mathscr{D}(obs)$

- **Timed Automata** (TA), Uppaal
- Networks of Timed Automata
- Region/Zone-Abstraction
- TA model-checking
- Extended Timed Automata
- Undecidability Results

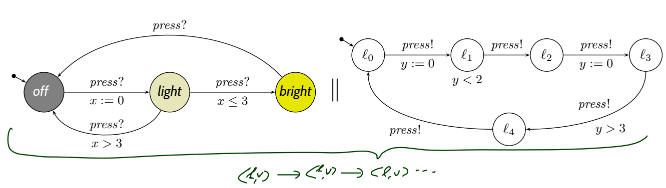
$\langle obs_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle obs_1, \nu_1 \rangle, t_1 \ldots$

- **Automatic Verification**...
  ...whether a TA satisfies a DC formula, observer-based
- **Recent Results**:
  - **Timed Sequence Diagrams**, or **Quasi-equal Clocks**, or **Automatic Code Generation**, or ...

Content

- A **satisfaction relation** between timed automata and DC formulae
  - **observables** of timed automata
  - **evolution** induced by **computation path**
- A **simple and wrong** solution.
  - **ad-hoc** fix for invariants
- **Testable DC Properties**
  - **observer construction**
  - **untestable DC properties**

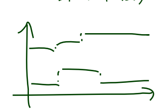
Model-Checking DC Properties with Uppaal



DC formula

$\models^? F$

$\mathcal{I}$: Obs $\to$ (Time $\to \mathcal{D}$)



- **Question 1**: what is the "$\models$"-relation here?
- **Question 2**: 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.)
  - **Maybe**: $F = \ell > 5 \implies \Diamond\lceil \text{off} \rceil^5$
  - **Not so clear**: $F = \neg\Diamond(\lceil \text{bright} \rceil ; \lceil \text{light} \rceil)$

Observing Timed Automata

Observables of a Network of Timed Automata

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

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

For simplicity: assume that all L_i and V_i are pairwise disjoint (otherwise rename).

Definition. The observables $\mathsf{Obs}(\mathcal{N})$ of $\mathcal{N}$ are

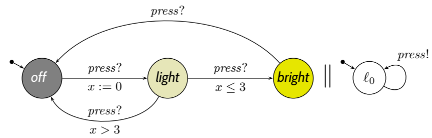
$$\{\ell_1, \dots, \ell_n\} \,\dot\cup \bigcup_{1 \leq i \leq n} V_i$$

{ ☺$_1$, ..., ☺$_n$ }

with

- $\mathcal{D}(\ell_i) = L_i$,
- $\mathcal{D}(v)$ is the domain of data-variable v in $\mathcal{A}_{e,i}$.

Example

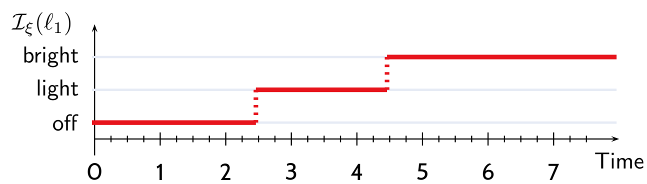


- **Observables**: $\mathsf{Obs}(\mathcal{N}) = \{\ell_1, \ell_2\}$ with
  - $\mathcal{D}(\ell_1) = \{off, light, bright\}$, $\mathcal{D}(\ell_2) = \{\ell_0\}$. (No data variables in $\mathcal{N}$!)

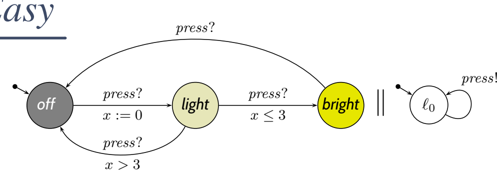
Consider **computation path**

$$\xi = \langle {}^{\text{off}}_{0} \rangle, 0 \xrightarrow{2.5} \langle {}^{\text{off}}_{2.5} \rangle, 2.5 \xrightarrow{\tau} \langle {}^{\text{light}}_{0} \rangle, 2.5 \xrightarrow{2.0} \langle {}^{\text{light}}_{2.0} \rangle, 4.5 \xrightarrow{\tau} \langle {}^{\text{bright}}_{2.0} \rangle, 4.5 \ldots$$

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



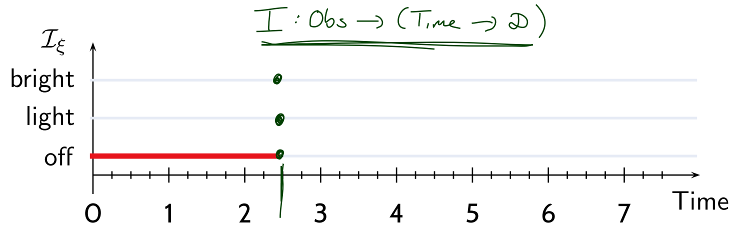
That'd be Too Easy



Consider **computation path**

$$\xi = \langle {}^{\text{off}}_{0} \rangle, 0 \xrightarrow{2.5} \underline{\langle {}^{\text{off}}_{2.5} \rangle, 2.5} \xrightarrow{\tau} \underline{\langle {}^{\text{light}}_{0} \rangle, 2.5} \xrightarrow{\tau} \underline{\langle {}^{\text{bright}}_{0} \rangle, 2.5} \xrightarrow{\tau} \langle {}^{\text{off}}_{0} \rangle, 2.5 \xrightarrow{1.0} \ldots$$

$\underline{\mathcal{I} : Obs \to (Time \to \mathcal{D})}$



Evolutions of TA Network

Our approach:

- **Consider** only those configurations assumed for more than 0 time units.
- **Extend** finite computation paths by keeping last discrete configuration.

Definition. Let

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

be a computation path of network $\mathcal{N}$ (infinite or of length n).

Then

$$\begin{aligned} \bar{\xi} : \text{Time} &\to \mathit{Conf}(\mathcal{N}) \\ t &\mapsto \Big(\langle \vec{\ell}_j, \nu_j + t - t_j \rangle \text{ where } j = \max\{i \in \mathbb{N}_0 \mid t_i \leq t\}\Big) \\ &\quad \text{and} \Big(\text{(if } \xi \text{ finite) } \langle \vec{\ell}_n, \nu_n + t - t_n \rangle \text{ for } t > t_n\Big) \end{aligned}$$

Recall: $\xi(t)$ used for the query language yielded the set of all configurations at t.

Evolutions of TA Network Cont'd

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

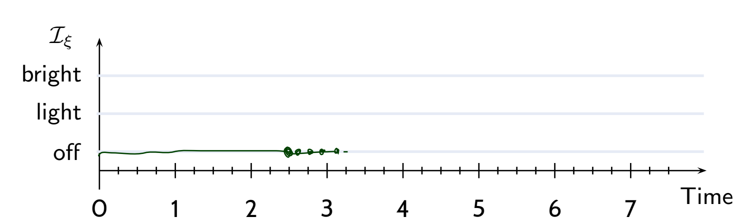
$$\mathcal{I}_\xi : \text{Obs}(\mathcal{N}) \to (\text{Time} \to \mathcal{D})$$

which is defined pointwise as follows:

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

Example:

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



Evolutions of TA Network Cont'd

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

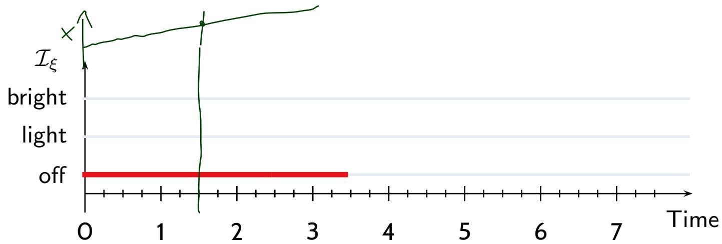
$$\mathcal{I}_\xi : \mathsf{Obs}(\mathcal{N}) \to (\mathsf{Time} \to \mathcal{D})$$

which is defined pointwise as follows:

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

Example:

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



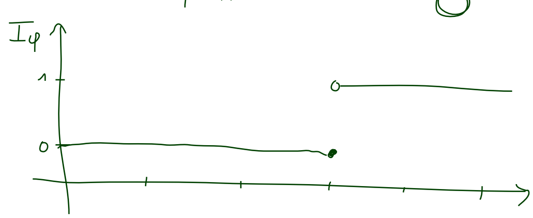
Clocks in Evolutions of TA Networks

- 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

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

- But...

φ: $x > 3$



- 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

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

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

Better (if needed):

- add (a finite subset of) $\Phi(X_1 \cup \dots \cup X_n)$ to $\mathsf{Obs}(\mathcal{N})$,
  with $\mathcal{D}(\varphi) = \{0, 1\}$ for $\varphi \in \Phi(X_1 \cup \dots \cup X_n)$.
- 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 may persist over non-point intervals.

Some Checkable Properties

"For every complex problem there is an answer that is clear, simple, and wrong."

Can't we **directly check** $\mathcal{N} \models F$ for   : iff for all comp. path ξ of $\mathcal{N}$, $\mathcal{I}_\xi \models_{DC} F$

- $F = \Box\lceil \mathrm{off} \rceil$ and $F = \neg\Diamond\lceil \mathrm{light} \rceil$

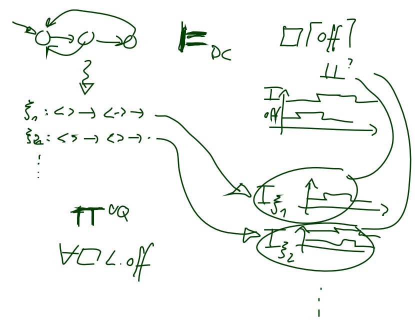
by checking **queries**

- $\forall\Box\, \mathcal{L}.\mathit{off}$ and $\exists\Diamond\, \mathcal{L}.\mathit{light}$?

$\mathcal{N} \models_{DC} \Box\lceil off \rceil$

$\Downarrow$ $\not\Leftarrow$ $\Uparrow$

$\mathcal{N} \models_{UA} \forall\Box\, \mathcal{L}.off$



Model-Checking DC Properties with Uppaal

"For every complex problem there is an answer that is clear, simple, and wrong."

Can't we **directly check** $\mathcal{N} \models F$ for

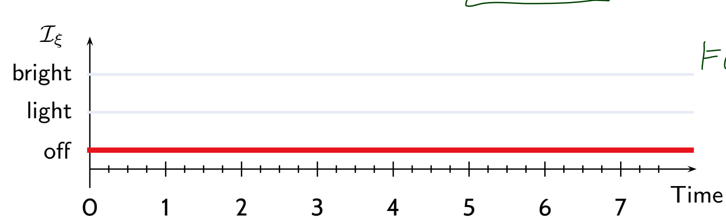
- $F = \Box\lceil \mathrm{off} \rceil$ and $F = \neg\Diamond\lceil \mathrm{light} \rceil$

by checking **queries**

- $\forall\Box\, \mathcal{L}.\mathit{off}$ and $\exists\Diamond\, \mathcal{L}.\mathit{light}$?

Well, we have $\mathcal{N} \models \forall\Box\, \mathcal{L}.\mathit{off}$ (2) **implies** $F = \Box\lceil \mathrm{off} \rceil$ (1), but **not vice versa**. $\models$ (2) ✗

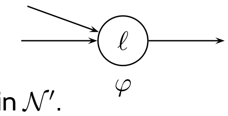
$$\xi = \langle {\mathrm{off} \atop 0} \rangle, 0 \xrightarrow{2.5} \langle {\mathrm{off} \atop 2.5} \rangle, 2.5 \xrightarrow{\tau} \langle {\mathrm{light} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\mathrm{bright} \atop 0} \rangle, 2.5 \xrightarrow{\tau} \langle {\mathrm{off} \atop 0} \rangle,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



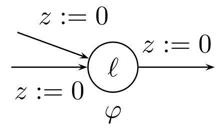
$\models$ (1) ✓

Model-Checking Invariants with Uppaal

- Ad-hoc fix: measure duration explicitly, transform $\mathcal{N}$ by



to



and obtain $\mathcal{N}'$.

Then check

$$\mathcal{N}' \models \forall\Box(z > 0 \implies P)$$

$(z=0 \vee P)$

to verify

$$\mathcal{N} \models \Box\lceil P \rceil.$$

Content

- A **satisfaction relation** between timed automata and DC formulae
  - **observables** of timed automata
  - **evolution** induced by **computation path**
- A **simple and wrong** solution.
  - **ad-hoc** fix for invariants
- **Testable DC Properties**
  - **observer construction**
  - **untestable DC properties**

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

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

for some $\mathcal{A}'_i$.

Otherwise F is called **untestable**.

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

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

for some $\mathcal{A}'_i$.

Otherwise F is called **untestable**.

Theorem 6.4. DC implementables are testable.

Proposition 6.3. There exist untestable DC formulae.

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.

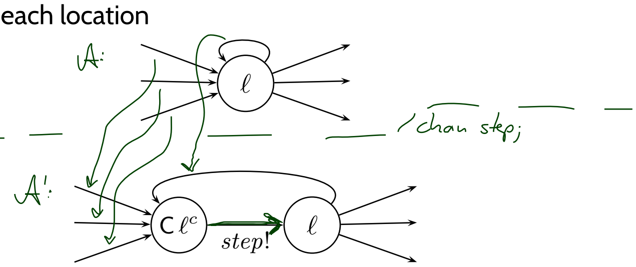
  Example: $\Diamond(\lceil v = 0 \rceil \,;\, \lceil v = 1 \rceil)$

- **Recall**: transitions of TA are only triggered by syncronisation, not by changes of data-variables.

- **Approach**: have **auxiliary** *step* action. $\mathcal{A} \rightsquigarrow \mathcal{A}'$

  Technically, replace each location

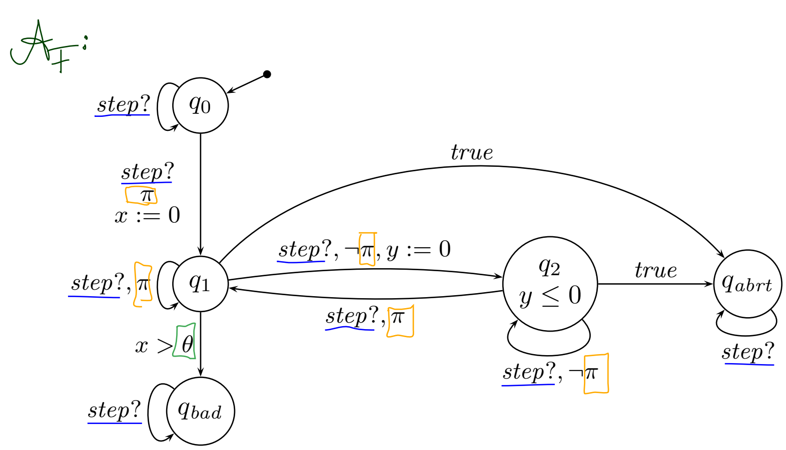
  with



  Note: the observer will consider data variables **after** all updates.

Proof of Theorem 6.4: Sketch

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



Counterexample Formulae

Definition 6.5.

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

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

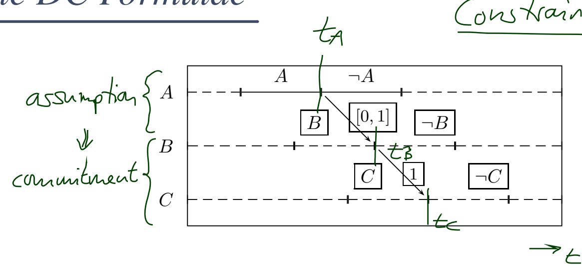
  where for $1 \leq i \leq k$,
  - π_i are state assertions,
  - I_i are non-empty, and open, half-open, or closed time intervals of the form
    - (b,e) or $[b,e)$ with $b \in \mathbb{Q}_0^+$ and $e \in \mathbb{Q}_0^+ \,\dot\cup\, \{\infty\}$,
    - $(b,e]$ or $[b,e]$ with $b,e \in \mathbb{Q}_0^+$.

    (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.

Theorem 6.7. CE formulae are testable.

Untestable DC Formulae

Constraint Diagram



"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

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

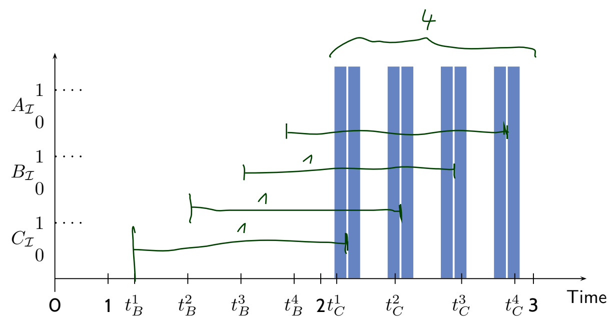
Assume the number of clocks in $\mathcal{A}_F$ is $n \in \mathbb{N}_0$.

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: $n = 3$



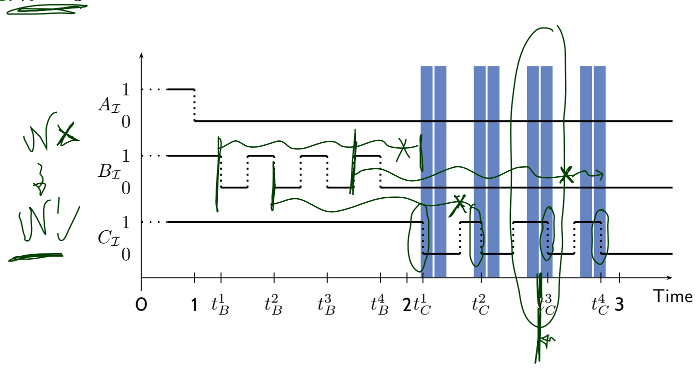
– 17 – 2018-01-18 – Sdctest –

Untestable DC Formulae Cont'd

Consider the following time points:

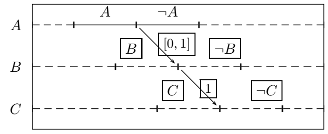
- $t_A := 1$
- $t_B^i := t_A + \frac{2i-1}{2(n+1)}$ for $i = 1, \dots, n+1$
- $t_C^i \in \left]t_B^i + 1 - \frac{1}{4(n+1)}, t_B^i + 1 + \frac{1}{4(n+1)}\right[$ for $i = 1, \dots, n+1$
  with $t_C^i - t_B^i \neq 1$ for $1 \leq i \leq n+1$.

Example: $n = 3$

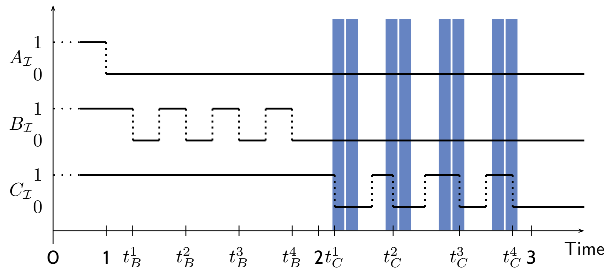


– 17 – 2018-01-18 – Sdctest –

Untestable DC Formulae Cont'd

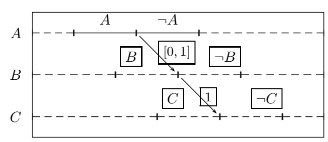


Example: $n = 3$

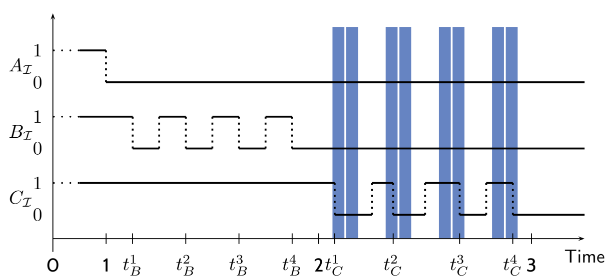


- The shown interpretation $\mathcal{I}$ satisfies the **assumption** of the property.
- It has $n+1$ candidates to satisfy the **commitment**.
- By choice of t_C^i, the commitment is not satisfied; so F is not satisfied.
- Because $\mathcal{A}_F$ is a test automaton for F, is has a computation path 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**.

Content

- A **satisfaction relation** between timed automata and DC formulae
  - **observables** of timed automata
  - **evolution** induced by **computation path**
- A **simple and wrong** solution.
  - **ad-hoc** fix for invariants
- **Testable DC Properties**
  - **observer construction**
  - **untestable DC properties**

Tell Them What You've Told Them...

- For **testable** DC formulae F, we can automatically verify whether a network $\mathcal{N}$ satisfies F.
  - by constructing an **observer automaton**
  - and **transforming** $\mathcal{N}$ appropriately.
- There are **untestable** DC formulae.

  (Everything else would be surprising.)

References

References

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