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

\[ \text{obs} : \text{Time} \rightarrow \mathcal{D}(\text{obs}) \]

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

\[ \langle \text{obs}_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle \text{obs}_1, \nu_1 \rangle, t_1 \ldots \]

- 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

**Question 1:** what is the “|=”-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:** \( F = \Box \text{[off]} \)  or \( F = \neg \Diamond \text{[light]} \)
  (Use Uppaal’s fragment of TCTL, something like (!) \( \forall \Box \text{off} \).)

- **Maybe:** \( F = \ell > 5 \implies \Diamond \text{[off]}^5 \)

- **Not so clear:** \( F = \neg \Diamond (\text{[bright]} \mathbin{;} \text{[light]}) \)
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 $\text{Obs}(\mathcal{N})$ of $\mathcal{N}$ are

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

with

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

- **Observables**: \( \text{Obs}(\mathcal{N}) = \{ l_1, l_2 \} \) with
  - \( \mathcal{D}(l_1) = \{ \text{off, light, bright} \}, \quad \mathcal{D}(l_2) = \{ l_0 \} \).  
  (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 \( \mathcal{I}_\xi : \text{Obs}(\mathcal{N}) \rightarrow (\text{Time} \rightarrow \mathcal{D}) \):
Consider computation path

\[ \xi = \langle \text{off} \rangle, 0 \xrightarrow{2.5} \langle \text{off} \rangle, 2.5 \xrightarrow{\tau} \langle \text{light} \rangle, 2.5 \xrightarrow{\tau} \langle \text{bright} \rangle, 2.5 \xrightarrow{\tau} \langle \text{off} \rangle, 2.5 \xrightarrow{1.0} \ldots \]
Our approach:

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

**Definition.** Let

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

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

Then

$$\bar{\xi} : \text{Time} \rightarrow \text{Conf}(\mathcal{N})$$

$$t \mapsto \langle \vec{l}_j, \nu_j + t - t_j \rangle \text{ where } j = \max\{i \in \mathbb{N}_0 \mid t_i \leq t\}$$

and \(\text{(if } \xi \text{ finite) } \langle \vec{l}_n, \nu_n + t - t_n \rangle \text{ for } t > t_n\) \)

**Recall:** $\xi(t)$ used for the query language yielded the set of all configurations at $t$. 
\( \bar{\xi} \) induces the unique interpretation

\[
\mathcal{I}_\bar{\xi} : \text{Obs}(\mathcal{N}) \rightarrow (\text{Time} \rightarrow \mathcal{D})
\]

which is defined pointwise as follows:

\[
\mathcal{I}_\bar{\xi}(\ell_i)(t) = \ell^i \quad , \text{if } \bar{\xi}(t) = \langle (\ell^1, \ldots, \ell^n), \nu \rangle \\
\mathcal{I}_\bar{\xi}(w)(t) = \nu(w) \quad , \text{if } \bar{\xi}(t) = \langle \ell, \nu \rangle
\]

Example:

\( \xi = \langle \text{off}_0 \rangle, 0 \overset{2.5}{\rightarrow} \langle \text{off}_{2.5} \rangle, 2.5 \overset{\tau}{\rightarrow} \langle \text{light}_0 \rangle, 2.5 \overset{\tau}{\rightarrow} \langle \text{bright}_0 \rangle, 2.5 \overset{\tau}{\rightarrow} \langle \text{off}_0 \rangle, 2.5 \overset{1.0}{\rightarrow} \langle \text{off}_1 \rangle, 3.5 \overset{\tau}{\rightarrow} \ldots \)
\( \bar{\xi} \) induces the unique interpretation

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

which is defined pointwise as follows:

\[
\mathcal{I}_\xi(\ell_i)(t) = \ell_i, \quad \text{if } \bar{\xi}(t) = \langle (\ell_1, \ldots, \ell_n), \nu \rangle
\]

\[
\mathcal{I}_\xi(w)(t) = \nu(w), \quad \text{if } \bar{\xi}(t) = \langle \ell, \nu \rangle
\]

Example:

\[
\xi = \langle \text{off}_0 \rangle, 0 \xrightarrow{2.5} \langle \text{off}_2 \rangle, 2.5 \xrightarrow{\tau} \langle \text{light}_0 \rangle, 2.5 \xrightarrow{\tau} \langle \text{bright}_0 \rangle, 2.5 \xrightarrow{\tau} \langle \text{off}_0 \rangle, 2.5 \xrightarrow{1.0} \langle \text{off}_1 \rangle, 3.5 \xrightarrow{\tau} \ldots
\]
• But **what about clocks?** Why not $x \in \text{Obs}(\mathcal{N})$ 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...
But what about clocks? Why not \( x \in \text{Obs}(\mathcal{N}) \) 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) \( \Phi(X_1 \cup \cdots \cup X_n) \) to \( \text{Obs}(\mathcal{N}) \),
  with \( \mathcal{D}(\varphi) = \{0, 1\} \) for \( \varphi \in \Phi(X_1 \cup \cdots \cup X_n) \).

- 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 \( \varphi \) 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

- \( F = \Box \text{off} \) and \( F = \neg \Diamond \text{light} \)

by checking **queries**

- \( \forall \Box \mathcal{L}. \text{off} \) and \( \exists \Diamond \mathcal{L}. \text{light} \)?

\[
\begin{align*}
\mathcal{N} \models_\text{dc} \Box \text{off} & \quad ? \\
\uparrow & \quad \uparrow \\
\mathcal{N} \not\models_\text{uq} \forall \Box \mathcal{L}. \text{off} & \quad \text{iff for all comp. path } \xi \text{ of } \mathcal{N}, \\
\overline{\mathcal{I}_\xi} \not\models_\text{dc} F & \quad \text{for all comp. path } \xi \text{ of } \mathcal{N},
\end{align*}
\]
“For every complex problem there is an answer that is clear, simple, and wrong.”

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

- $F = \square \lceil \text{off} \rceil$ and $F = \neg \diamond \lceil \text{light} \rceil$

by checking **queries**

- $\forall \square \mathcal{L}.\text{off}$ and $\exists \diamond \mathcal{L}.\text{light}$?

Well, we have $\mathcal{N} \models \forall \square \mathcal{L}.\text{off}$ **implies** $F = \square \lceil \text{off} \rceil$, but **not vice versa**.

\[ \xi = \langle \text{off} \rangle, 0 \overset{0.5}{\rightarrow} \langle \text{off} \rangle, 2.5 \overset{2.5}{\rightarrow} \langle \text{light} \rangle, 2.5 \overset{2.5}{\rightarrow} \langle \text{bright} \rangle, 2.5 \overset{2.5}{\rightarrow} \langle \text{off} \rangle, 2.5 \overset{1.0}{\rightarrow} \langle \text{off} \rangle, 3.5 \overset{1.0}{\rightarrow} \ldots \]

**Diagram:**

- $\mathcal{L}: \text{light, bright, off}$
- $I_\xi$:
  - bright
  - light
  - off
- $\mathcal{N}: \text{Time} = 0, 1, 2, 3, 4, 5, 6, 7$
• Ad-hoc fix: measure duration explicitly, transform $\mathcal{N}$ by

\[
\phi \quad z := 0
\]

and obtain $\mathcal{N}'$.

Then check

\[
\mathcal{N}' \models \forall \Box (z > 0 \implies P) \\
(z=0 \lor P)
\]

to verify

\[
\mathcal{N} \models \Box [P].
\]
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
Definition 6.1. A DC formula $F$ is called **testable** if an observer (or test automaton (or monitor)) $A_F$ exists such that for all networks $\mathcal{N} = C(A_1, \ldots, A_n)$ it holds that

\[
\mathcal{N} \models F \text{ iff } (C(A'_1, \ldots, A'_n, \overbrace{A_F}^{\text{here}})) \models \forall q \Box \neg (A_F.q_{bad})
\]

for some $A'_i$.

Otherwise $F$ is called **untestable**.
Testability

Definition 6.1. A DC formula $F$ is called **testable** if an observer (or test automaton (or monitor)) $A_F$ exists such that for all networks $\mathcal{N} = C(A_1, \ldots, A_n)$ it holds that

\[
\mathcal{N} \models F \text{ iff } C(A'_1, \ldots, A'_n, A_F) \models \forall \square \neg (A_F.q_{bad})
\]

for some $A'_i$.

Otherwise $F$ is called **untestable**.

Theorem 6.4. DC implementables are testable.

Proposition 6.3. There exist untestable DC formulae.
Theorem 6.4. DC implementables are testable.

- Initialisation:
  \( \left[ \emptyset \right] \lor \left[ \pi \right] ; \text{true} \)
  \( \left[ \pi \right] \rightarrow \left[ \pi \lor \pi_1 \lor \cdots \lor \pi_n \right] \)

- Sequencing:
  \( \left[ \pi \right] \xrightarrow{\theta} \left[ \neg \pi \right] \)
  \( \left[ \pi \land \varphi \right] \xrightarrow{\theta} \left[ \neg \pi \right] \)

- Progress:
  \( \left[ \pi \right] \theta \rightarrow \left[ \neg \pi \right] \)

- Synchronisation:
  \( \left[ \pi \land \varphi \right] \theta \rightarrow \left[ \neg \pi \right] \)

- Bounded Stability:
  \( \left[ \neg \pi \right] ; \left[ \pi \land \varphi \right] \xleq{\theta} \left[ \pi \lor \pi_1 \lor \cdots \lor \pi_n \right] \)

- Unbounded Stability:
  \( \left[ \neg \pi \right] ; \left[ \pi \land \varphi \right] \xrightarrow{\theta} \left[ \pi \lor \pi_1 \lor \cdots \lor \pi_n \right] \)

- Bounded initial stability:
  \( \left[ \pi \land \varphi \right] \xleq{\theta} \left[ \pi \lor \pi_1 \lor \cdots \lor \pi_n \right] \)

- Unbounded initial stability:
  \( \left[ \pi \land \varphi \right] \xrightarrow{\theta} \left[ \pi \lor \pi_1 \lor \cdots \lor \pi_n \right] \)

Proof Sketch:

- For each implementable \( F \), construct \( A_F \).
- Prove that \( A_F \) 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:** \(\diamond ([v = 0] ; [v = 1])\)

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

- **Approach:** have auxiliary step action.

Technically, replace each location

![Diagram showing the step action](image)

with

Note: the observer will consider data variables after all updates.
Proof of Theorem 6.4: Sketch

- Example: $[\pi] \xrightarrow{\theta} [\neg \pi] = \top$

\[
\begin{align*}
\text{step?} & \quad q_0 \\
\text{step?} & \quad x := 0 \\
\text{step?}, \pi & \quad q_1 \\
\text{step?}, \pi, x > \theta & \quad q_{bad} \\
\text{step?}, \neg \pi & \quad q_2 \quad y \leq 0 \\
\text{true} & \quad q_{abrt} \\
\end{align*}
\]
Definition 6.5.

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

\[
true \land (\lceil \pi_1 \rceil \land \ell \in I_1) \land \ldots \land (\lceil \pi_k \rceil \land \ell \in I_k) \land true
\]

where for \(1 \leq i \leq k\),
- \(\pi_i\) 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 \(b \in Q_0^+\) and \(e \in Q_0^+ \cup \{\infty\}\),
  - \((b, e]\) or \([b, e]\) with \(b, e \in Q_0^+\).
- \((b, \infty)\) 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.
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 $A_F$ such that, for all networks $\mathcal{N}$, we have

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

Assume the number of clocks in $A_F$ is $n \in \mathbb{N}_0$. 
Consider the following time points:

- \( t_A := 1 \)
- \( t_i^B := t_A + \frac{2i-1}{2(n+1)} \) for \( i = 1, \ldots, n + 1 \)
- \( t_i^C \in \left[ t_i^B + 1 - \frac{1}{4(n+1)}, t_i^B + 1 + \frac{1}{4(n+1)} \right] \) for \( i = 1, \ldots, n + 1 \)

with \( t_i^C - t_i^B \neq 1 \) for \( 1 \leq i \leq n + 1 \).

**Example: \( n = 3 \)**
Consider the following time points:

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

**Example:** $n = 3$
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 $t^i_C$, the commitment is not satisfied; so $F$ is not satisfied.
- Because $\mathcal{A}_F$ is a test automaton for $F$, it has a computation path to $q_{bad}$.

- Because $n = 3$, $\mathcal{A}_F$ can not save all $n + 1$ time points $t^i_B$.
- Thus there is $1 \leq i_0 \leq n$ such that all clocks of $\mathcal{A}_F$ have a valuation which is not in $2 - t^{i_0}_B + \left( -\frac{1}{4(n+1)}, \frac{1}{4(n+1)} \right)$.
**Example:** $n = 3$

- Because $\mathcal{A}_F$ is a test automaton for $F$, it 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 $\mathcal{T}'$ such that $t_C^{i_0} := t_B^{i_0} + 1$.
- Then $\mathcal{T}' \models F$, but $\mathcal{A}_F$ reaches $q_{bad}$ via the same path.
- That is: $\mathcal{A}_F$ claims $\mathcal{T}' \not\models F$.
- 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
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