Real-Time Systems

Lecture 06: DC Properties I

2014-05-22

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:

- DC Syntax and Semantics: Abbreviations (“almost everywhere”)
- Satisfiable/Realisable/Valid (from 0)

This Lecture:

- Educational Objectives: Capabilities for following tasks/questions.
  - What are obstacles on proving a design correct in the real-world, and how to overcome them?
  - Facts: decidability properties.
  - What’s the idea of the considered (un)decidability proofs?

- Content:
  - Semantical Correctness Proof
  - (Un-)Decidable problems of DC variants in discrete and continuous time
Specification and Semantics-based Correctness Proofs of Real-Time Systems with DC
(i) Choose a collection of observables ‘Obs’.

(ii) Provide the requirement/specification ‘Spec’ as a conjunction of DC formulae (over ‘Obs’).

(iii) Provide a description ‘Ctrl’ of the controller in form of a DC formula (over ‘Obs’).

(iv) We say ‘Ctrl’ is correct (wrt. ‘Spec’) iff

\[ \models_0 \text{Ctrl} \implies \text{Spec.} \]
(i) Choose **observables**:

- two boolean observables $G$ and $F$
  
  (i.e. $\text{Obs} = \{G, F\}, \mathcal{D}(G) = \mathcal{D}(F) = \{0, 1\}$)

- $G = 1$: gas valve open
- $F = 1$: have flame

- define $L := G \land \neg F$ (leakage)

(ii) Provide the **requirement**:

\[
\text{Req} : \leftrightarrow \quad \square (\ell \geq 60 \implies 20 \cdot \int L \leq \ell)
\]
(iii) Provide a description ‘Ctrl’ of the controller in form of a DC formula (over ‘Obs’). Here, firstly consider a design:

- Des-1: \( \iff \Box([L] \implies \ell \leq 1) \)
- Des-2: \( \iff \Box([L] \land \lnot[L] \land [L] \implies \ell > 30) \)

(iv) Prove correctness:

- We want (or do we want \( \models_{0...?} \)):
  \[ \models (\text{Des-1} \land \text{Des-2} \implies \text{Req}) \]  (Thm. 2.16)
- We do show
  \[ \models \text{Req-1} \implies \text{Req} \]  (Lem. 2.17)

with the simplified requirement

\[ \text{Req-1} := \Box(\ell \leq 30 \implies \int L \leq 1), \]

- and we show
  \[ \models (\text{Des-1} \land \text{Des-2}) \implies \text{Req-1}. \]  (Lem. 2.19)
Gas Burner Revisited: Lemma 2.17

Claim:

\[ \models \square (\ell \leq 30 \implies \int L \leq 1) \implies \square (\ell \geq 60 \implies 20 \cdot \int L \leq \ell) \]

Proof:
Claim:

\[ \models \Box (\ell \leq 30 \implies \int L \leq 1) \implies \Box (\ell \geq 60 \implies 20 \cdot \int L \leq \ell) \]

Proof:

- Assume ‘Req-1’.
Gas Burner Revisited: Lemma 2.17

Claim:

\[
\models \Box (\ell \leq 30 \implies \int L \leq 1) \implies \Box (\ell \geq 60 \implies 20 \cdot \int L \leq \ell)
\]

Proof:

- Assume ‘Req-1’.
- Let \( L_\mathcal{I} \) be any interpretation of \( L \), and \([b, e]\) an interval with \( e - b \geq 60 \).
Claim:

\[ \models \square (\ell \leq 30 \implies \int L \leq 1) \implies \square (\ell \geq 60 \implies 20 \cdot \int L \leq \ell) \]

Proof:

- Assume ‘Req-1’.
- Let \( L_I \) be any interpretation of \( L \), and \([b, e]\) an interval with \( e - b \geq 60 \).
- Show “\( 20 \cdot \int L \leq \ell \)”, i.e.

\[ I[20 \cdot \int L \leq \ell](\forall, [b, e]) = \text{tt} \]

i.e.

\[ \hat{20} : \int_{b}^{e} L_I(t) \, dt \leq (e - b) \]
Set $n := \lceil \frac{e-b}{30} \rceil$, i.e. $n \in \mathbb{N}$ with $n - 1 < \frac{e-b}{30} \leq n$, and split the interval

\[
\begin{align*}
&b + 30 & b + 60 & b + 30(n - 2)b + 30(n - 1) & b + 30n \\
\hline
&b & \cdots & & e \\
\end{align*}
\]
Theorem 2.18.
For all state assertions $P$ and all real numbers $r_1, r_2 \in \mathbb{R}$,

(i) $\models \int P \leq \ell$,
(ii) $\models (\int P = r_1) \land (\int P = r_2) \implies \int P = r_1 + r_2$,
(iii) $\models [\neg P] \implies \int P = 0$,
(iv) $\models [] \implies \int P = 0$. 
Claim:

\[ \models (\Box([L] \implies \ell \leq 1) \land \Box([L] ; [\neg L] ; [L] \implies \ell > 30)) \implies \Box(\ell \leq 30 \implies fL \leq 1) \]

Proof:
Claim:

\[ (i) \models \int P \leq \ell, \quad (iv) \models \square \models \int P = 0, \]

\[ (ii) \models (\int P = r_1) \; ; \; (\int P = r_2) \models \int P = r_1 + r_2, \]

\[ (iii) \models \lnot P \models \int P = 0, \]

\[ (\square (\lbrack L \rbrack \models \ell \leq 1) \wedge \square (\lbrack L \rbrack ; \lbrack \lnot L \rbrack ; \lbrack L \rbrack \models \ell > 30)) \models \square (\ell \leq 30 \models \int L \leq 1) \]

Proof:

\[ \models (\square (\lbrack L \rbrack \models \ell \leq 1) \wedge \square (\lbrack L \rbrack ; \lbrack \lnot L \rbrack ; \lbrack L \rbrack \models \ell > 30)) \models \square (\ell \leq 30 \models \int L \leq 1) \]
Gas Burner Revisited: Lemma 2.18
Obstacles in Non-Ideal World
Methodology: The World is Not Ideal...

(i) Choose a collection of **observables** ‘Obs’.
(ii) Provide **specification** ‘Spec’ (conjunction of DC formulae (over ‘Obs’)).
(iii) Provide a description ‘Ctrl’ of the **controller** (DC formula (over ‘Obs’)).
(iv) Prove ‘Ctrl’ is **correct** (wrt. ‘Spec’).

That looks **too simple to be practical**. Typical **obstacles**:

(i) It may be impossible to realise ‘Spec’ if it doesn’t consider properties of the **plant**.
(ii) There are typically intermediate **design levels** between ‘Spec’ and ‘Ctrl’.
(iii) ‘Spec’ and ‘Ctrl’ may use **different observables**.
(iv) **Proving** validity of the implication is not trivial.
(i) Assumptions As A Form of Plant Model

- Often the controller will (or can) operate correctly only under some assumptions.
- For instance, with a level crossing
  - we may assume an upper bound on the speed of approaching trains, (otherwise we’d need to close the gates arbitrarily fast)
  - we may assume that trains are not arbitrarily slow in the crossing, (otherwise we can’t make promises to the road traffic)
Assumptions As A Form of Plant Model

- Often the controller will (or can) operate correctly only under some assumptions.
- For instance, with a level crossing
  - we may assume an upper bound on the speed of approaching trains, (otherwise we’d need to close the gates arbitrarily fast)
  - we may assume that trains are not arbitrarily slow in the crossing, (otherwise we can’t make promises to the road traffic)
- We shall specify such assumptions as a DC formula ‘Asm’ on the input observables and verify correctness of ‘Ctrl’ wrt. ‘Spec’ by proving validity (from 0) of
  \[ Ctrl \land Asm \implies Spec \]
Often the controller will (or can) operate correctly only under some assumptions.

For instance, with a level crossing

- we may assume an upper bound on the speed of approaching trains, (otherwise we’d need to close the gates arbitrarily fast)
- we may assume that trains are not arbitrarily slow in the crossing, (otherwise we can’t make promises to the road traffic)

We shall specify such assumptions as a DC formula ‘Asm’ on the input observables and verify correctness of ‘Ctrl’ wrt. ‘Spec’ by proving validity (from 0) of

\[ \text{Ctrl} \land \text{Asm} \implies \text{Spec} \]

Shall we care whether ‘Asm’ is satisfiable?
(ii) Intermediate Design Levels

- A top-down development approach may involve
  - Spec — specification/requirements
  - Des — design
  - Ctrl — implementation

- Then correctness is established by proving validity of

\[ \text{Ctrl} \rightarrow \text{Des} \quad (1) \]

and

\[ \text{Des} \rightarrow \text{Spec} \quad (2) \]

(then concluding \( \text{Ctrl} \rightarrow \text{Spec} \) by transitivity)

- Any preference on the order?
(iii): Different Observables

- Assume, ‘Spec’ uses more abstract observables $\text{Obs}_A$ and ‘Ctrl’ more concrete ones $\text{Obs}_C$.

- For instance:
  - in $\text{Obs}_A$: only consider gas valve open or closed ($\mathcal{D}(G) = \{0, 1\}$)
  - in $\text{Obs}_C$: may control two valves and care for intermediate positions, for instance, to react to different heating requests ($\mathcal{D}(G_1) = \{0, 1, 2, 3\}, \mathcal{D}(G_2) = \{0, 1, 2, 3\}$)
(iii): Different Observables

- Assume, ‘Spec’ uses more abstract observables $\text{Obs}_A$ and ‘Ctrl’ more concrete ones $\text{Obs}_C$.
- For instance:
  - in $\text{Obs}_A$: only consider gas valve open or closed ($\mathcal{D}(G) = \{0, 1\}$)
  - in $\text{Obs}_C$: may control two valves and care for intermediate positions, for instance, to react to different heating requests ($\mathcal{D}(G_1) = \{0, 1, 2, 3\}, \mathcal{D}(G_2) = \{0, 1, 2, 3\}$)
- To prove correctness, we need information how the observables are related — an **invariant** which **links** the data values of $\text{Obs}_A$ and $\text{Obs}_C$.
- If we’re given the linking invariant as a DC formula, say ‘Link$_{C,A}$’, **then** proving correctness of ‘Ctrl’ wrt. ‘Spec’ amounts to proving validity (from 0) of
  $$\text{Ctrl} \land \text{Link}_{C,A} \implies \text{Spec}.$$  
- For instance,
  $$\text{Link}_{C,A} = [G \leftrightarrow (G_1 + G_2 > 0)]$$
Obstacle (iv): How to Prove Correctness?

- by hand on the basis of DC semantics,
- maybe supported by proof rules,
- sometimes a general theorem may fit (e.g. cycle times of PLC automata),
- algorithms as in Uppaal.
DC Properties
Recall:

Given assumptions as a DC formula ‘Asm’ on the input observables, verifying correctness of ‘Ctrl’ wrt. ‘Spec’ amounts to proving

$$\models_0 \text{Ctrl} \land \text{Asm} \implies \text{Spec}$$  \hspace{1cm} (1)

If ‘Asm’ is not satisfiable...
Decidability Results: Motivation

- Recall:
  Given **assumptions** as a DC formula ‘Asm’ on the input observables, verifying **correctness** of ‘Ctrl’ wrt. ‘Spec’ amounts to proving

\[ \models_0 \text{Ctrl} \land \text{Asm} \implies \text{Spec} \] (1)

- If ‘Asm’ is **not satisfiable** then (1) is trivially valid, and thus each ‘Ctrl’ correct wrt. ‘Spec’.

- So: strong interest in assessing the **satisfiability** of DC formulae.

- Question: is there an automatic procedure to help us out?
  (a.k.a.: is it **decidable** whether a given DC formula is satisfiable?)

- More interesting for ‘Spec’:
Decidability Results: Motivation

- Recall:
  Given assumptions as a DC formula ‘Asm’ on the input observables, verifying correctness of ‘Ctrl’ wrt. ‘Spec’ amounts to proving

\[ \models_0 \text{Ctrl} \land \text{Asm} \implies \text{Spec} \]  

- If ‘Asm’ is not satisfiable then (1) is trivially valid, and thus each ‘Ctrl’ correct wrt. ‘Spec’.
- So: strong interest in assessing the satisfiability of DC formulae.
- Question: is there an automatic procedure to help us out? (a.k.a.: is it decidable whether a given DC formula is satisfiable?)
- More interesting for ‘Spec’: is it realisable (from 0)?
- Question: is it decidable whether a given DC formula is realisable?
### Decidability Results for Realisability: Overview

<table>
<thead>
<tr>
<th>Fragment</th>
<th>Discrete Time</th>
<th>Continous Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDC</td>
<td><strong>decidable</strong></td>
<td><strong>decidable</strong></td>
</tr>
<tr>
<td>RDC $+ \ell = r$</td>
<td>decidable for $r \in \mathbb{N}$</td>
<td>undecidable for $r \in \mathbb{R}^+$</td>
</tr>
<tr>
<td>RDC $+ \int P_1 = \int P_2$</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>RDC $+ \ell = x, \forall x$</td>
<td>undecidable</td>
<td><strong>undecidable</strong></td>
</tr>
<tr>
<td>DC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RDC in Discrete Time
Restricted DC (RDC)

\[
F ::= [P] | \neg F_1 | F_1 \lor F_2 | F_1 ; F_2
\]

where \( P \) is a state assertion, but with **boolean** observables **only**.

Note:
- No global variables, thus don’t need \( \forall \).
An interpretation $\mathcal{I}$ is called **discrete time interpretation** if and only if, for each state variable $X$,

$$X_\mathcal{I} : \text{Time} \rightarrow \mathcal{D}(X)$$

with

- $\text{Time} = \mathbb{R}^+_0$,
- all discontinuities are in $\mathbb{N}_0$. 

**Discrete Time Interpretations**
Discrete Time Interpretations

• An interpretation $\mathcal{I}$ is called **discrete time interpretation** if and only if, for each state variable $X$,

\[ X_{\mathcal{I}} : \text{Time} \rightarrow \mathcal{D}(X) \]

with

• $\text{Time} = \mathbb{R}_0^+$,
• all discontinuities are in $\mathbb{N}_0$.

• An interval $[b, e] \subset \text{Intv}$ is called **discrete** if and only if $b, e \in \mathbb{N}_0$. 
Discrete Time Interpretations

- An interpretation $\mathcal{I}$ is called **discrete time interpretation** if and only if, for each state variable $X$,

$$X_\mathcal{I} : \text{Time} \rightarrow \mathcal{D}(X)$$

with

- Time $= \mathbb{R}_0^+$,
- all discontinuities are in $\mathbb{N}_0$.

- An interval $[b, e] \subset \text{Intv}$ is called **discrete** if and only if $b, e \in \mathbb{N}_0$.

- We say (for a discrete time interpretation $\mathcal{I}$ and a discrete interval $[b, e]$)

$$\mathcal{I}, [b, e] \models F_1 ; F_2$$

if and only if there exists $m \in [b, e] \cap \mathbb{N}_0$ such that

$$\mathcal{I}, [b, m] \models F_1 \quad \text{and} \quad \mathcal{I}, [m, e] \models F_2$$
• Let $P$ be a state assertion.

<table>
<thead>
<tr>
<th></th>
<th>Continuous Time</th>
<th>Discrete Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\models ? ([P] ; [P])$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Rightarrow [P]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\models ? [P] \Rightarrow ([P] ; [P])$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Differences between Continuous and Discrete Time

- Let $P$ be a state assertion.

<table>
<thead>
<tr>
<th></th>
<th>Continuous Time</th>
<th>Discrete Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\models^? ([P] ; [P])$</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>$\Rightarrow [P]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\models^? [P] \Rightarrow ([P] ; [P])$</td>
<td>✔</td>
<td>✘</td>
</tr>
</tbody>
</table>

- In particular: $\ell = 1 \iff ([1] \land \neg([1] ; [1]))$ (in discrete time).
Expressiveness of RDC

- $\ell = 1 \iff [1] \land \neg([1] ; [1])$
- $\ell = 0 \iff \neg[1]$
- $true \iff \ell = 0 \lor \neg(\ell = 0)$
- $\int P = 0 \iff [\neg P] \lor \ell = 0$
- $\int P = 1 \iff (\int P = 0) ; ([P] \land \ell = 1) ; (\int P = 0)$
- $\int P = k + 1 \iff (\int P = k) ; (\int P = 1)$
- $\int P \geq k \iff (\int P = k) ; true$
- $\int P > k \iff \int P \geq k + 1$
- $\int P \leq k \iff \neg(\int P > k)$
- $\int P < k \iff \int P \leq k - 1$

where $k \in \mathbb{N}$. 
Theorem 3.6.
The satisfiability problem for RDC with discrete time is decidable.

Theorem 3.9.
The realisability problem for RDC with discrete time is decidable.
Sketch: Proof of Theorem 3.6

• give a procedure to construct, given a formula $F$, a regular language $L(F)$ such that

$$I, [0, n] \models F \text{ if and only if } w \in L(F)$$

where word $w$ describes $I$ on $[0, n]$
(suitability of the procedure: Lemma 3.4)

• then $F$ is satisfiable in discrete time if and only if $L(F)$ is not empty
(Lemma 3.5)

• Theorem 3.6 follows because
  • $L(F)$ can effectively be constructed,
  • the emptiness problem is decidable for regular languages.
Construction of $\mathcal{L}(F)$

- **Idea:**
  - alphabet $\Sigma(F)$ consists of basic conjuncts of the state variables in $F$,
  - a letter corresponds to an interpretation on an interval of length 1,
  - a word of length $n$ describes an interpretation on interval $[0, n]$. 
Construction of $\mathcal{L}(F)$

- **Idea:**
  - alphabet $\Sigma(F)$ consists of basic conjuncts of the state variables in $F$,
  - a letter corresponds to an interpretation on an interval of length 1,
  - a word of length $n$ describes an interpretation on interval $[0, n]$.

- **Example:** Assume $F$ contains exactly state variables $X, Y, Z$, then
  \[
  \Sigma(F) = \{ X \land Y \land Z, X \land Y \land \neg Z, X \land \neg Y \land Z, X \land \neg Y \land \neg Z, \\
  \neg X \land Y \land Z, \neg X \land Y \land \neg Z, \neg X \land \neg Y \land Z, \neg X \land \neg Y \land \neg Z \}. 
  \]
Construction of $\mathcal{L}(F)$ more Formally

**Definition 3.2.** A word $w = a_1 \ldots a_n \in \Sigma(F)^*$ with $n \geq 0$ describes a discrete interpretation $\mathcal{I}$ on $[0, n]$ if and only if

$$\forall j \in \{1, \ldots, n\} \ \forall t \in ]j - 1, j[ : \mathcal{I}[a_j](t) = 1.$$  

For $n = 0$ we put $w = \varepsilon$.

- Each state assertion $P$ can be transformed into an equivalent disjunctive normal form $\bigvee_{i=1}^{m} a_i$ with $a_i \in \Sigma(F)$.
- Set $DNF(P) := \{a_1, \ldots, a_m\} (\subseteq \Sigma(F))$.
- Define $\mathcal{L}(F)$ inductively:

  $$\mathcal{L}([P]) = DNF(P)^+, \quad \mathcal{L}(\neg F_1) = \Sigma(F)^* \setminus \mathcal{L}(F_1), \quad \mathcal{L}(F_1 \lor F_2) = \mathcal{L}(F_1) \cup \mathcal{L}(F_2), \quad \mathcal{L}(F_1 ; F_2) = \mathcal{L}(F_1) \cdot \mathcal{L}(F_2).$$
Lemma 3.4. For all RDC formulae $F$, discrete interpretations $\mathcal{I}$, $n \geq 0$, and all words $w \in \Sigma(F)^*$ which describe $\mathcal{I}$ on $[0, n]$, 

$$\mathcal{I}, [0, n] \models F \text{ if and only if } w \in \mathcal{L}(F).$$
Theorem 3.9. The realisability problem for RDC with discrete time is decidable.

- \( \text{kern}(L) \) contains all words of \( L \) whose prefixes are again in \( L \).
- If \( L \) is regular, then \( \text{kern}(L) \) is also regular.
- \( \text{kern}(\mathcal{L}(F)) \) can effectively be constructed.
- We have

Lemma 3.8. For all RDC formulae \( F \), \( F \) is realisable from 0 in discrete time if and only if \( \text{kern}(\mathcal{L}(F)) \) is infinite.

- Infinity of regular languages is decidable.
References