Real-Time Systems

Lecture 06: DC Properties I

2012-05-24

Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:
- DC Syntax and Semantics: Abbreviations ("almost everywhere")
- Satisfiable/Realisable/Valid (from 0)
- Semantical Correctness Proof

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:
  - (Un-)Decidable problems of DC variants in discrete and continuous time
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.
Obstacle (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)
- We shall specify such assumptions as a DC formula ‘Asm’ on the input observables and verify correctness correctness of ‘Ctrl’ wrt. ‘Spec’ by proving validity (from 0) of
  \[ Ctrl \land Asm \implies Spec \]
- Shall we care whether ‘Asm’ is satisfiable?

Obstacle (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
  \[ Ctrl \implies Des \quad (1) \]
  and
  \[ Des \implies Spec \quad (2) \]
  (then concluding Ctrl \implies Spec by transitivity)
- Any preference on the order?
Obstacle (iii): Different Observables

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

- Example:
  - 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
    $$\mathcal{D}(G_1) = \{0, 1, 2, 3\}, \quad \mathcal{D}(G_2) = \{0, 1, 2, 3\}$$

- To prove correctness, we need information how the observables are related — an invariant which links the data values of $\text{Obs}_A$ and $\text{Obs}_C$.

- Formally: If linking invariant is given as a DC formula, say ‘Link$_{C,A}$’, then proving correctness of ‘Ctrl’ wrt. ‘Spec’ amounts to proving
  $$\models_0 \quad \text{Ctrl} \land \text{Link}_{C,A} \implies \text{Spec}.$$ 

- Example for linking invariant:
  $$\text{Link}_{C,A} = \left\lceil G \iff ( G_1 \lor G_2 > 0 ) \right\rceil$$

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.
Decidability Results: Motivation

- Recall:
  Given assumptions as a DC formula ‘Asm’ on the input observables, verifying correctness of ‘Ctrl’ wrt. ‘Spec’ amounts to proving
  \[ \models_0 \text{Ctrl} \land \text{Asm} \implies \text{Spec} \quad (1) \]
  - 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’: 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>Continuous Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDC</td>
<td>decidable</td>
<td>decidable</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>undecidable</td>
</tr>
<tr>
<td>DC</td>
<td>undecidable</td>
<td>undecidable</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 \( \mathcal{V} \).
- **cheap is this**
  - no \( \int \), no \( \mathcal{L} \) (in general)
  - no function and predicate symbols
  - \( \mathcal{G} \mathcal{T} \ ... ? 
  - \( \mathcal{T} \mathcal{G} \ ... ? 

**Discrete Time Interpretations**

- An interpretation \( \mathcal{I} \) is called **discrete time interpretation** if and only if, for each state variable \( X \),
  \[ X_\mathcal{I} : \text{Time} \to \mathcal{D}(X) \]

with
- Time = \( \mathbb{R}^+ \)
- all discontinuities are in \( \mathbb{N}_0 \).
Discrete Time Interpretations

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

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

with

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

- An interval $[b, e] \in \text{Intv}$ 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 $[b, e]$)

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

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

$$\mathcal{I}, [b, m] \models F_1 \quad \text{and} \quad \mathcal{I}, [m, e] \models F_2$$

Differences between Continuous and Discrete Time

- Let $P$ be a state assertion, e.g. $X^{<\omega}$

<table>
<thead>
<tr>
<th></th>
<th>Continuous Time</th>
<th>Discrete Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\models \exists \left([P] ; [P]\right)$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</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>$\models ? (\lfloor P \rfloor ; \lfloor P \rfloor)$</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>$\implies \lfloor P \rfloor$</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>$\models ? \lfloor P \rfloor \implies (\lfloor P \rfloor ; \lfloor P \rfloor)$</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 $\iff \ell = 0 \lor \ell (\ell = 0)$
- $\int P = 0 \iff [\neg P] \lor \ell = 0$
- $\int P = 1 \iff (\int P > 0) ; ([P] \land \ell = 0) ; (\int P = 0)$
- $\int P = k + 1 \iff (\int P = k) ; (\int P = 1)$
- $\int P \geq k \iff (\int P = k) ; +\ell$
- $\int P > k \iff \int P \geq k + 1$
- $\int P \leq k \iff \neg(\int P > k)$
- $\int P < k \iff \int P \leq k - 1$

where $k \in \mathbb{N}$. 

Decidability of Satisfiability/Realisability from 0

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 $\mathcal{L}(F)$ such that
  \[ \mathcal{I}, [0, n] \models F \text{ if and only if } w \in \mathcal{L}(F) \] (1)
  where word $w$ describes $\mathcal{I}$ on $[0, n]$
  (procedure: in a minute)
  (procedure has property (1): Lemma 3.4)

- then $F$ is satisfiable in discrete time if and only if $\mathcal{L}(F)$ is not empty (Lemma 3.5)

- Theorem 3.6 follows because
  - $\mathcal{L}(F)$ can effectively be constructed (by that procedure),
  - 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 of \( \text{Obs} \) on an interval of length 1,
  - a word of length \( n \) describes an interpretation of \( \text{Obs} \) on interval \([0, n]\).

- **Example:** Assume \( F \) contains exactly state variables \( X, Y, Z \), then
  \[
  \Sigma(F) = \{ X \land Y \land Z, X \land Y \land \neg Z, X \land \neg Y \land Z, X \land \neg Y \land \neg Z, 
  \neg X \land Y \land Z, 
  \neg X \land Y \land \neg Z, 
  \neg X \land \neg Y \land Z, 
  \neg X \land \neg Y \land \neg Z \}. 
  \]

- 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 \( \text{DNF}(P) := \{ a_1, \ldots, a_m \} \ (\subseteq \Sigma(F)) \).

- Define \( \mathcal{L}(F) \) inductively:
  \[
  \mathcal{L}([P]) = \text{DNF}(P)^\uparrow, \\
  \mathcal{L}(\neg F_1) = \Sigma(F) \setminus \mathcal{L}(F_1), \\
  \mathcal{L}(F_1 \lor F_2) = \mathcal{L}(F_1) \cup \mathcal{L}(F_2), \\
  \mathcal{L}(F_1 \land F_2) = \mathcal{L}(F_1) \cdot \mathcal{L}(F_2). 
  \]

**Construction of \( \mathcal{L}(F) \) more Formally**

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

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

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 \( \text{DNF}(P) := \{ a_1, \ldots, a_m \} \ (\subseteq \Sigma(F)) \).

- Define \( \mathcal{L}(F) \) inductively:
**Lemma 3.4**

Lemma 3.4. For all RDC formulae $F$, discrete interpretations $I$, $n \geq 0$, and all words $w \in \Sigma(F)^*$ which describe $I$ on $[0, n]$, $I, [0, n] \models F$ if and only if $w \in \mathcal{L}(F)$.

**Proof:**

Sketch: Proof of Theorem 3.9

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