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’).

$\models_0 \text{Ctrl} \implies \text{Spec}$

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.

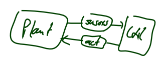
Obstacle (i): Assumptions As A Form of Plant Model

- 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

$$\text{Ctrl} \wedge \underbrace{\text{Asm}}_{\Updownarrow\ \text{false}} \implies \text{Spec}$$

false } if not sat.

- 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

$$\text{Ctrl} \implies \text{Des} \quad (1)$$

and

$$\text{Des} \implies \text{Spec} \quad (2)$$

(then concluding Ctrl $\implies$ Spec by transitivity)

- Any preference on the order?

Obstacle (iii): Different Observables

- Assume, 'Spec' uses more abstract observables Obs_A and 'Ctrl' more concrete ones Obs_C.
- Example:
  - in Obs_A: only consider gas valve open or closed

$$\mathcal{D}(G) = \{0, 1\}$$

  - in 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 Obs_A and Obs_C.
- Formally: **If** linking invariant is given as a DC formula, say '$\mathsf{Link}_{C,A}$', **then** proving correctness of 'Ctrl' wrt. 'Spec' amounts to proving

$$\models_0 \mathsf{Ctrl} \wedge \mathsf{Link}_{C,A} \implies \mathsf{Spec}.$$

- Example for linking invariant:

$$\mathsf{Link}_{C,A} = \Box\lceil G \iff (G_1 + G_2 > 0)\rceil$$

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.

DC Properties

Decidability Results: Motivation

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

$$\models_0 \mathsf{Ctrl} \wedge \mathsf{Asm} \implies \mathsf{Spec} \tag{1}$$

- 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

restricted

Fragment	Discrete Time	Continous Time
RDC	decidable	decidable
RDC $+ \ell = r$	decidable for $r \in \mathbb{N}$	undecidable for $r \in \mathbb{R}^+$
RDC $+ \int P_1 = \int P_2$	undecidable	undecidable
RDC $+ \ell = x, \forall x$	undecidable	undecidable
DC	undecidable	undecidable

RDC in Discrete Time

Restricted DC (RDC)

$1 \mid 0 \mid x=1 \mid x=0 \mid \neg P \mid P_1 \wedge P_2$

$$F ::= \lceil P \rceil \mid \neg F_1 \mid F_1 \vee F_2 \mid F_1 \,;\, F_2$$

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

Note:

- No global variables, thus don't need $\mathcal{V}$.
- chop is there
- no $\int$, no ℓ (in general)
- no function and predicate symbols
- $\Diamond F$ ... ?
- $\lceil\rceil$ ... ?

Discrete Time Interpretations

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

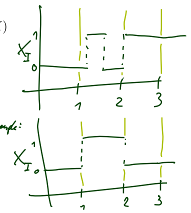
$$X_\mathcal{I} : \text{Time} \to \mathcal{D}(X)$$

with

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

not a DTI:

DTI example:



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

- We say $\mathcal{I}, [b,e] \models \lceil P \rceil$ iff $\int_b^e P_\mathcal{I}(t)\,dt = (e-b) \wedge (e-b) > 0$

- An interval $[b,e] \in$ 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

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

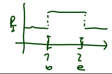
Differences between Continuous and Discrete Time

- Let P be a state assertion, e.g. $X=1$

	Continuous Time	Discrete Time
$\models^? (\lceil P \rceil \,;\, \lceil P \rceil)$	✓ 	✓ 

Differences between Continuous and Discrete Time

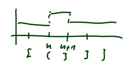
- Let P be a state assertion.

	Continuous Time	Discrete Time
$\models^? (\lceil P \rceil \,;\, \lceil P \rceil) \implies \lceil P \rceil$	✓	✓
$\models^? \lceil P \rceil \implies (\lceil P \rceil \,;\, \lceil P \rceil)$	✓	✗ 

- In particular: $\ell = 1 \iff (\lceil 1 \rceil \wedge \neg(\lceil 1 \rceil \,;\, \lceil 1 \rceil))$ (in discrete time).

Expressiveness of RDC

$\Diamond F :=$ true ; F ; true in RDC...



- $\ell = 1 \iff \lceil 1 \rceil \wedge \neg(\lceil 1 \rceil \,;\, \lceil 1 \rceil)$
- $\ell = 0$, $\lceil\rceil$ $\iff \neg\lceil 1 \rceil$
- *true* $\iff \ell = 0 \vee \neg(\ell = 0)$
- $\int P = 0 \iff \lceil \neg P \rceil \vee \ell = 0$
- $\int P = 1 \iff (\int P = 0) \,;\, (\lceil P \rceil \wedge \ell = 1) \,;\, (\int P = 0)$
- $\int P = k+1 \iff (\int P = k) \,;\, (\int P = 1)$
- $\int P \geq k \iff (\int P = k) \,;\,$ 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 $k \in \mathbb{N}$.

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

F
$\Big\}$ construct
$\Downarrow$
$\mathcal{L}(F)$

- give a procedure to construct, given a formula F, a **regular** language $\mathcal{L}(F)$ such that

$$\mathcal{I}, [0, n] \models F \text{ if and only if } w \in \mathcal{L}(F) \qquad (1)$$

  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 emptyness problem is **decidable** for regular languages.

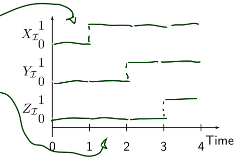
Construction of $\mathcal{L}(F)$

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

$$\Sigma(F) = \{X \wedge Y \wedge Z, \quad X \wedge Y \wedge \neg Z, \quad X \wedge \neg Y \wedge Z, \quad X \wedge \neg Y \wedge \neg Z,$$
$|\Sigma(F)| = 8$
$$\neg X \wedge Y \wedge Z, \quad \neg X \wedge Y \wedge \neg Z, \quad \neg X \wedge \neg Y \wedge Z, \quad \neg X \wedge \neg Y \wedge \neg Z\}.$$

$$w = (\neg X \wedge \neg Y \wedge \neg Z) \cdot (X \wedge \neg Y \wedge \neg Z) \cdot (X \wedge Y \wedge \neg Z) \cdot (X \wedge Y \wedge Z) \in \Sigma(F)^*$$

Concatenation



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

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

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

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 $DNF(P) := \{a_1, \dots, a_m\}$ $(\subseteq \Sigma(F))$.
- Define $\mathcal{L}(F)$ inductively:

word of length at least 1

$$\mathcal{L}(\lceil P \rceil) = DNF(P)^+, \qquad \text{(regular)}$$
$$\mathcal{L}(\neg F_1) = \Sigma(F)^* \setminus \mathcal{L}(F_1), \qquad \text{(again regular)}$$
$$\mathcal{L}(F_1 \vee F_2) = \mathcal{L}(F_1) \cup \mathcal{L}(F_2), \qquad \text{– " –}$$
$$\mathcal{L}(F_1 \,;\, F_2) = \mathcal{L}(F_1) \cdot \mathcal{L}(F_2). \qquad \text{– " –}$$

Lemma 3.4

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 $[0, n]$,

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

Proof: Structural induction

Base $F = \lceil P \rceil$: assume $w = a_1 \dots a_n$ describes $\mathcal{I}$ on $[0,n]$

$\mathcal{I}, [0,n] \models \lceil P \rceil \iff \mathcal{I}, [0,n] \models \lceil P \rceil$ and $n \geq 1$
$\iff n \geq 1$ and $\forall\, 1 \leq j \leq n \bullet \mathcal{I}, [j-1, j] \models \lceil P \rceil$
$\iff n \geq 1$ and $\forall\, 1 \leq j \leq n \bullet \mathcal{I}, [j-1, j] \models (\lceil P \rceil \wedge \lceil a_j \rceil)$ and $a_j \in DNF(F)$

"describes" $\Uparrow$ $\qquad$ $\Downarrow$ clear

$\iff n \geq 1$ and $\forall\, 1 \leq j \leq n \bullet a_j \in DNF(F)$
$\iff w \in DNF(F)^+$
$\iff w \in \mathcal{L}(F)$

Steps: • $F = F_1 ; F_2$
$F = \neg F_1$
$F = F_1 \vee F_2$

Sketch: Proof of Theorem 3.9

Theorem 3.9.
The realisability problem for RDC with discrete time is decidable.

- $kern(L)$ contains all words of L whose prefixes are again in L.
- If L is regular, then $kern(L)$ is also regular.
- $kern(\mathcal{L}(F))$ 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 $kern(\mathcal{L}(F))$ is infinite.

- Infinity of regular languages is decidable.

References

References

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