Real-Time Systems

Lecture 06: DC Properties I

2013-05-08

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

Specification and Semantics-based Correctness Proofs of Real-Time Systems with DC

Methodology: Ideal World...

(i) Choose a collection of **observables** 'Obs'.

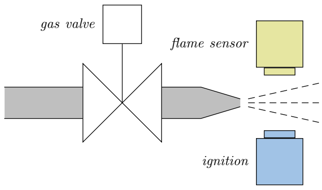
(ii) Provide the **requirement**/**specification** 'Spec' as a conjunction of DC formulae (over 'Obs').

(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

$$\models_0 \mathsf{Ctrl} \implies \mathsf{Spec}.$$

Gas Burner Revisited



(i) Choose **observables**:

- two boolean observables G and F (i.e. Obs $= \{G, F\}$, $\mathcal{D}(G) = \mathcal{D}(F) = \{0, 1\}$)
- $G = 1$: gas valve open (output)
- $F = 1$: have flame (input)
- define $L := G \wedge \neg F$ (leakage)

(ii) Provide the **requirement**:

$$\text{Req} :\iff \Box(\ell \geq 60 \implies 20 \cdot \textstyle\int L \leq \ell)$$

Gas Burner Revisited

(iii) Provide a description 'Ctrl' of the **controller** in form of a DC formula (over 'Obs'). Here, firstly consider a **design**:

- Des-1 $:\iff \Box(\lceil L \rceil \implies \ell \leq 1)$ "phases of leakage have length at most 1"
- Des-2 $:\iff \Box((\lceil L \rceil \,;\, \lceil \neg L \rceil \,;\, \lceil L \rceil) \implies \ell > 30)$ "intervals where L_I looks like (L ¬L L) should length bigger than 30"

(iv) Prove **correctness**:

- We want (or do we want $\models_0$...?):

$$\models (\text{Des-1} \wedge \text{Des-2} \implies \text{Req}) \qquad \text{(Thm. 2.16)}$$

Gas Burner Revisited

(iii) Provide a description 'Ctrl'
of the **controller** in form of a DC formula (over 'Obs').
Here, firstly consider a **design**:

- Des-1 : $\iff \Box(\lceil L \rceil \implies \ell \leq 1)$

- Des-2 : $\iff \Box(\lceil L \rceil \, ; \lceil \neg L \rceil \, ; \lceil L \rceil \implies \ell > 30)$

(iv) Prove **correctness**:

- We want (or do we want $\models_0 \ldots$?):

$$\models (\text{Des-1} \wedge \text{Des-2} \implies \text{Req}) \qquad \text{(Thm. 2.16)}$$

- We do show

$$\models \text{Req-1} \implies \text{Req} \qquad \text{(Lem. 2.17)}$$

with the simplified requirement

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

- and we show

$$\models (\text{Des-1} \wedge \text{Des-2}) \implies \text{Req-1}. \qquad \text{(Lem. 2.19)}$$

Gas Burner Revisited: Lemma 2.17

Claim: for all $\mathcal{I}, \mathcal{V}, [b,e]$

$$\models \underbrace{\Box(\ell \leq 30 \implies \textstyle\int L \leq 1)}_{\text{Req-1}} \implies \underbrace{\Box(\ell \geq 60 \implies 20 \cdot \textstyle\int L \leq \ell)}_{\text{Req}}$$

Proof:

- Assume 'Req-1'.
- Let $L_\mathcal{I}$ be any interpretation of L, and $[b,e]$ an interval with $e - b \geq 60$, let $\mathcal{V}$ a valuation.
- Show "$20 \cdot \int L \leq \ell$", i.e.

$$\mathcal{I}[\![20 \cdot \textstyle\int L \leq \ell]\!](\mathcal{V}, [b,e]) = \mathrm{tt}$$

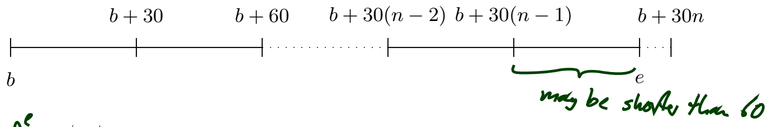
i.e.

$$20 \cdot \int_b^e L_\mathcal{I}(t)\, dt \overset{?}{\leq} (e-b)$$

Gas Burner Revisited: Lemma 2.17

$$\models \underbrace{\Box(\ell \le 30 \implies \textstyle\int L \le 1)}_{\text{Req-1}} \implies \Box(\ell \ge 60 \implies 20 \cdot \textstyle\int L \le \ell)$$

- Set $n := \lceil \frac{e-b}{30} \rceil$, i.e. $n \in \mathbb{N}$ with $n - 1 < \frac{e-b}{30} \le n$, and split the interval



$20 \cdot \int_b^e L_I(t)\,dt$

$= 20 \cdot \left(\sum_{i=0}^{n-2} \underbrace{\int_{b+30i}^{b+30(i+1)} L_I(t)\,dt}_{\le 1} + \int_{b+30(n-1)}^{e} L_I(t)\,dt \right)$

$\{\text{Req-1}\}$ $\le 20 \cdot \sum_{i=0}^{n-2} 1 + 20 \cdot 1$

$= 20 \cdot n$

$\{n-1 < \frac{e-b}{30}\}$ $< 20 \left(\frac{e-b}{30} + 1 \right)$

$= \frac{2}{3}(e-b) + 20$

$\le e - b$

$\{ e - b \ge 60 \leadsto 20 \le \frac{1}{3}(e-b) \}$

Some Laws of the DC Integral Operator

Theorem 2.18
For all state assertions P and all real numbers $r_1, r_2 \in \mathbb{R}$,

(i) $\models \int P \le \ell$,

(ii) $\models \big((\int P = r_1)\,;(\int P = r_2)\big) \implies \big(\int P = r_1 + r_2,\big)$

(iii) $\models \lceil \neg P \rceil \implies \int P = 0$,

(iv) $\models \lceil\rceil \implies \int P = 0$.

Gas Burner Revisited: Lemma 2.18

(i) $\models \int P \leq \ell$, (iv) $\models \lceil\rceil \implies \int P = 0$.
(ii) $\models (\int P = r_1) ; (\int P = r_2) \implies \int P = r_1 + r_2$,
(iii) $\models \lceil \neg P \rceil \implies \int P = 0$,

Claim:

$$\models (\underbrace{\Box(\lceil L \rceil \implies \ell \leq 1)}_{\text{Des-1}} \wedge \underbrace{\Box(\lceil L \rceil ; \lceil \neg L \rceil ; \lceil L \rceil \implies \ell > 30)}_{\text{Des-2}}) \implies \underbrace{\Box(\ell \leq 30 \implies \int L \leq 1)}_{\text{Req-1}}$$

Proof:

$\ell \leq 30$

{finite variability} $\Rightarrow \lceil\rceil$
$\vee \lceil L \rceil ; (\lceil\rceil \vee \lceil \neg L \rceil)$
$\vee \lceil \neg L \rceil ; (\lceil\rceil \vee \lceil L \rceil)$
$\vee \lceil \neg L \rceil ; \lceil L \rceil ; \lceil \neg L \rceil$ } (*)
$\vee \ell \leq 30 \wedge \Diamond(\lceil L \rceil ; \lceil \neg L \rceil ; \lceil L \rceil)$ ↯ Des-2

{Des-2} $\Rightarrow$ (*)

{Des-1} $\Rightarrow \lceil\rceil$
$\vee (\ell \leq 1) ; (\lceil\rceil \vee \lceil \neg L \rceil)$
$\vee \lceil \neg L \rceil ; (\lceil\rceil \vee \ell \leq 1)$
$\vee \lceil \neg L \rceil ; \ell \leq 1 ; \lceil \neg L \rceil$

{(i)} $\Rightarrow \lceil\rceil$
$\vee (\int L \leq 1) ; (\lceil\rceil \vee \lceil \neg L \rceil)$
$\vee \lceil \neg L \rceil ; (\lceil\rceil \vee \int L \leq 1)$
$\vee \lceil \neg L \rceil ; \int L \leq 1 ; \lceil \neg L \rceil$

{(iv), (iii)} $\Rightarrow \int L = 0$
$\vee (\int L \leq 1) ; (\int L = 0 \vee \int L = 0)$
$\vee \int L = 0 ; (\int L = 0 \vee \int L \leq 1)$
$\vee \int L = 0 ; \int L \leq 1 ; \int L = 0$

{(ii)} $\Rightarrow \int L = 0$
$\vee \int L \leq 1 + 0$
$\vee \int L \leq 0 + 1$
$\vee \int L \leq 0 + 1 + 0$
$\Rightarrow \int L \leq 1$

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 of ‘Ctrl’ wrt. ‘Spec’ by proving validity (from 0) of

$$\mathrm{Ctrl} \wedge \mathrm{Asm} \implies \mathrm{Spec}$$

- Shall we **care** whether ‘Asm’ is satisfiable?

$\mathrm{Ctrl} \wedge \mathit{false} \Rightarrow \mathrm{Spec}$ if Asm not 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

$$\mathsf{Ctrl} \implies \mathsf{Des} \tag{1}$$

and

$$\mathsf{Des} \implies \mathsf{Spec} \tag{2}$$

(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.
- For instance:
  - 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 $(\mathcal{D}(G_1) = \{0, 1, 2, 3\}, \mathcal{D}(G_2) = \{0, 1, 2, 3\})$
- To prove correctness, we need information how the observables are related — an **invariant** which **links** the data values of Obs_A and Obs_C.
- **If** we're given the linking invariant as a DC formula, say '$\mathsf{Link}_{C,A}$', **then** proving correctness of 'Ctrl' wrt. 'Spec' amounts to proving validity (from 0) of

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

- For instance,

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

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

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

- 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?

– 06 – 2013-05-08 – Smotiv –

Decidability Results for Realisability: Overview

restricted DC

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

– 06 – 2013-05-08 – Smotiv –

RDC in Discrete Time

Restricted DC (RDC)

$0 \mid 1 \mid X=0 \mid X=1 \mid \neg P \mid P_1 \vee P_2$

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

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

Note:

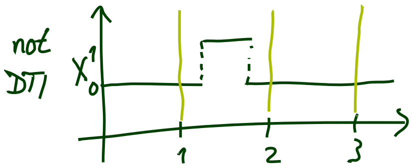
- No global variables, thus don't need $\mathcal{V}$.
- chop is there
- no $\int$, no ℓ (in general)
- no predicate, no function symbols
- $\Diamond F$ ...?
- $\lceil\rceil$ ...?

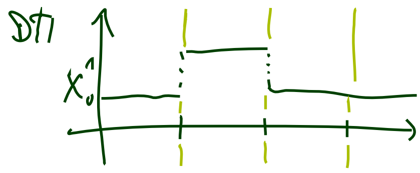
Discrete Time Interpretations

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

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

  with
  - $\mathsf{Time} = \mathbb{R}_0^+$,
  - 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,

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

  with
  - $\mathsf{Time} = \mathbb{R}_0^+$,
  - 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 $[b, e] \subset \mathsf{Intv}$ 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 $[b, e]$)

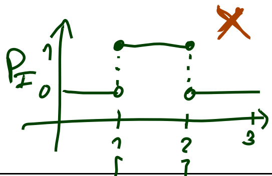
$$\mathcal{I}, [b, e] \models F_1 \,;\, F_2$$

  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.

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

[ ] $\lceil P \rceil$ holds

[ ] $\lceil P \rceil$ holds

b/m e

$\lceil P \rceil$ does not hold, because $m-b=0 \not> 0$

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

- $\ell = 1 \quad :\Longleftrightarrow \quad \lceil 1 \rceil \wedge \neg(\lceil 1 \rceil \,;\, \lceil 1 \rceil)$
- $\ell = 0 \quad :\Longleftrightarrow \quad \neg\lceil 1 \rceil$
- $true \quad :\Longleftrightarrow \quad \ell = 0 \vee \neg(\ell = 0)$
- $\int P = 0 \quad :\Longleftrightarrow \quad \lceil \neg P \rceil \vee \ell = 0$
- $\int P = 1 \quad \Longleftrightarrow \quad (\int P = 0) \,;\, (\lceil P \rceil \wedge \ell = 1) \,;\, \int P = 0$
- $\int P = k+1 \quad \Longleftrightarrow \quad (\int P = k) \,;\, (\int P = 1)$
- $\int P \geq k \quad \Longleftrightarrow \quad (\int P = k) \,;\, true$
- $\int P > k \quad \Longleftrightarrow \quad \int P \geq k+1$
- $\int P \leq k \quad \Longleftrightarrow \quad \neg(\int P > k)$
- $\int P < k \quad \Longleftrightarrow \quad \int P \leq k-1$

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

still
$\Diamond F := true \,;\, F \,;\, true$
in RDC

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.

RDC formula F, DT/ $\mathcal{I}$.

$$\begin{array}{ccc} F & \dashv & \mathcal{I} \\ \Big\downarrow \text{construct} & \Updownarrow \text{prove} & \Big\downarrow \text{construct} \\ \mathcal{L}(F) & \ni & w \in \Sigma^* \text{ word} \end{array}$$

regular language

Sketch: Proof of Theorem 3.6

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

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

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

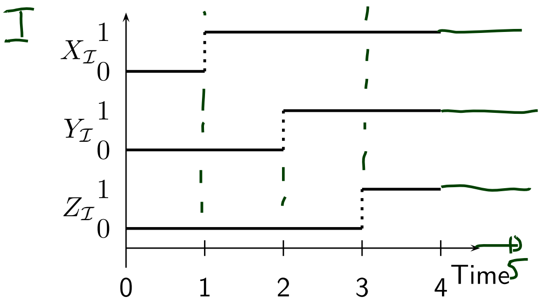
- then F is satisfiable in discrete time if and only if $\mathcal{L}(F)$ is not empty (**Lemma 3.5**)

- Theorem 3.6 follows because
  - $\mathcal{L}(F)$ can **effectively** be constructed,
  - 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 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

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



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

$\bullet(X \wedge Y \wedge Z)$
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 $DNF(P) := \{a_1, \dots, a_m\}$ $(\subseteq \Sigma(F))$. finite words of length at least one
- Define $\mathcal{L}(F)$ inductively:

$$\mathcal{L}(\lceil P \rceil) = DNF(P)^+,$$
$$\mathcal{L}(\neg F_1) = \Sigma(F) \setminus \mathcal{L}(F_1),$$
$$\mathcal{L}(F_1 \vee F_2) = \mathcal{L}(F_1) \cup \mathcal{L}(F_2),$$
$$\mathcal{L}(F_1 \,;\, F_2) = \mathcal{L}(F_1) \cdot \mathcal{L}(F_2)$$

References

References

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