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
**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

\[ \models_0 \text{Ctrl} \implies \text{Spec}. \]
(i) Choose observables:
- two boolean observables \( G \) and \( F \)
  (i.e. \( \text{Obs} = \{G, F\}, D(G) = D(F) = \{0, 1\} \))
- \( G = 1 \): gas valve open (output)
- \( F = 1 \): have flame (input)
- define \( L := G \land \neg F \) (leakage)

(ii) Provide the requirement:

\[
\text{Req} : \iff \square (\ell \geq 60 \implies 20 \cdot \int L \leq \ell)
\]

(iii) Provide a description 'Ctrl' of the controller in form of a DC formula (over 'Obs'). Here, firstly consider a design:
- \( \text{Des-1} : \iff \square ([L] \implies \ell \leq 1) \)
- \( \text{Des-2} : \iff \square ([L] ; [\neg L] ; [L] \implies \ell > 30) \)

(iv) Prove correctness:
- We want (or do we want \( \models \...? \)):
  \[
  \models (\text{Des-1} \land \text{Des-2} \implies \text{Req})
  \]
  (Thm. 2.16)
- We do show
  \[
  \models \text{Req-1} \implies \text{Req}
  \]
  (Lem. 2.17)
  with the simplified requirement
- \( \text{Req-1} := \square (\ell \leq 30 \implies \int L \leq 1) \),
- and we show
  \[
  \models (\text{Des-1} \land \text{Des-2}) \implies \text{Req-1}
  \]
  (Lem. 2.19)
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 \( L_I \) be any interpretation of \( L \) and \([b, e]\) an interval with \( e - b \geq 60 \).
- Show "20 \cdot \int L \leq \ell", i.e.

\[ 20 \cdot \int_{L_I} L_I(t) \, dt \leq (e - b) \]

i.e.

\[ 20 \cdot \int_{[b, e]} L_I(t) \, dt \leq (e - b) \]

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

\[
\begin{array}{cccccc}
& b + 30 & b + 60 & b + 30(n - 2) & b + 30(n - 1) & b + 30n \\
\hline
b & & & & & e \\
\end{array}
\]

\[
20 \cdot \int_{[b, e]} L_I(t) \, dt = 20 \cdot \left( \int_{[b, b + 30]} L_I(t) \, dt + \int_{b + 30}^{b + 30(n - 1)} L_I(t) \, dt \right)
\]

\[ \leq 20 \cdot \sum_{i=1}^{n-1} \int_{b + 30i}^{b + 30(i+1)} L_I(t) \, dt + \int_{b + 30(n-1)}^{e} L_I(t) \, dt \]

\[ \leq 20 \cdot \left( \frac{(e - b)}{30} + 1 \right) \]

\[ \leq 20 \cdot \left( \frac{(e - b)}{30} + 1 \right) \]

\[ \leq \frac{20}{3}(e - b) + 20 \]

\[ \leq e - b \]
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) \( \models \int P \leq \ell \),

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

(iii) \( \models \lnot P \implies \int P = 0 \),

(iv) \( \models \emptyset \implies \int P = 0 \).

---

**Gas Burner Revisited: Lemma 2.18**

Claim: \( \models \emptyset \) Therefore all \( \Box \), \( \lceil L \rceil \), \( \lceil \lnot L \rceil \)\n
\[ \models \Box(\lceil L \rceil \implies \ell \leq 1) \land \Box(\lceil \lnot L \rceil ; \lceil L \rceil \implies \ell > 30) \implies \Box(\ell \leq 30 \implies \int L \leq 1) \]

**Proof:**

\[ \{\text{Def-1}\} \models \Box (\emptyset) \]

\[ \{\text{Def-2}\} \models (\emptyset) \]

\[ \{\text{Def-1}\} \models \Box \]

\[ \{\text{Def-2}\} \models \Box \text{ \ l.e.t.} \]
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)
- 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? \textbf{YES!}

(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
  \[ \text{Ctrl} \implies \text{Des} \]  \hspace{1cm} (1)

  and

  \[ \text{Des} \implies \text{Spec} \]  \hspace{1cm} (2)

  (then concluding Ctrl \implies Spec by transitivity)
- Any preference on the order?
(iii): Different Observables

- Assume, ‘Spec’ uses more abstract observables Obs\textsubscript{A} and ‘Ctrl’ more concrete ones Obs\textsubscript{C}.
- For instance:
  - in Obs\textsubscript{A}: only consider gas valve open or closed ($\mathcal{D}(G) = \{0, 1\}$)
  - in Obs\textsubscript{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\textsubscript{A} and Obs\textsubscript{C}.
- If we’re given the linking invariant as a DC formula, say ‘Link\textsubscript{C,A}’, then proving correctness of ‘Ctrl’ wrt. ‘Spec’ amounts to proving validity (from 0) of

\[
\text{Ctrl} \land \text{Link}_{\text{C}, \text{A}} \implies \text{Spec}.
\]

- For instance,

\[
\text{Link}_{\text{C}, \text{A}} = \lceil G \iff (G_1 \land G_2 \lor 0) \rceil
\]

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.
\section*{Decidability Results: Motivation}

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

\[
|\rightarrow_0 \text{Ctrl} \land \text{Asm} \implies \text{Spec} \quad (1)
\]

- If ‘Asm’ is **not satisfiable** then (1) is trivially valid, and thus each ‘Ctrl’ correct wrt. ‘Spec’.
- So: strong interest in assessing the \textbf{satisfiability} of DC formulae.

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

- More interesting for ‘Spec’: is it \textbf{realisable} (from 0)?
- Question: is it \textbf{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>decidable</td>
<td>decidable</td>
</tr>
<tr>
<td>RDC + ℓ = 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 + ℓ = (x, \forall x)</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>DC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### References