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?

Specification and Semantics-based Correctness Proofs of Real-Time Systems with DC
GasBurnerRevisited: Lemma 2.17

Claim:

\[
\begin{align*}
| & = \square (\ell \leq 30 \Rightarrow \int L \leq 1) \, \text{Req-1} \\
\end{align*}
\]

Proof:

• Assume 'Req-1'.

• Let \( L \) be any interpretation of \( L \), and \([b,e]\) an interval with \( e - b \geq 60 \).

• Show "\( 20 \cdot \int L \leq \ell \)" i.e. \( I/llbracket 20 \cdot \int L \leq \ell \rrbracket (V, [b,e]) = t \), i.e. \( \hat{20} \cdot \int_{b}^{e} L (t) \mathrm{d}t \leq (e - b) / 30 \).

• Set \( n = \lceil \frac{e - b}{30} \rceil \), i.e. \( n \in \mathbb{N} \) with \( n - 1 < \frac{e - b}{30} \leq n \).

Theorem 2.18.

For all state assertions \( P \) and all real numbers \( r_1, r_2 \in \mathbb{R} \),

\[
\begin{align*}
(i) & \ | = \int P \leq \ell, \\
(ii) & \ | = (\int P = r_1) \land (\int P = r_2) = \Rightarrow \int P = r_1 + r_2, \\
(iii) & \ | = \lceil \neg P \rceil = \Rightarrow \int P = 0, \\
(iv) & \ | = \lceil \rceil = \Rightarrow \int P = 0.
\end{align*}
\]
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 other 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 $\text{Ctrl} \land \text{Asm} \Rightarrow \text{Spec}$.

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 $\text{Ctrl} \Rightarrow \text{Des}(1)$ and $\text{Des} \Rightarrow \text{Spec}(2)$ (then concluding $\text{Ctrl} = \text{Spec}$).

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 ($\text{D}(\text{G}) = \{0, 1\}$)
  - in $\text{Obs}_C$: may control two valves and care for intermediate positions, for instance, to react to different heating requests ($\text{D}(\text{G}_1) = \{0, 1, 2, 3\}, \text{D}(\text{G}_2) = \{0, 1, 2, 3\}$).
- 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 $\text{Ctrl} \land \text{Link}_{C,A} = \text{Spec}$.

How to Prove Correctness?

- by hand on the basis of DC semantics,
- may be supported by proof rules,
- sometimes a general theorem may fit (e.g. cycle times of PLC automata),
- algorithms as in Uppaal.
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>+\ell = r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDC + \int P_1 = \int P_2</td>
<td>undecidable</td>
<td></td>
</tr>
<tr>
<td>+\ell = x, \forall x</td>
<td></td>
<td>undecidable</td>
</tr>
</tbody>
</table>

References