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

• Oftentimes, 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:

\[ Ctrl \land Asm \Rightarrow Spec \]

• 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:

\[ Ctrl = \Rightarrow Des(1) \]
\[ Des = \Rightarrow Spec(2) \]

(then concluding \( Ctrl = \Rightarrow Spec \) 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
    \[ D(G) = \{0, 1\} \]
  - in \(\text{Obs}_C\): may control two valves and care for intermediate positions, for instance, to react to different heating requests
    \[ D(G_1) = \{0, 1, 2, 3\}, D(G_2) = \{0, 1, 2, 3\} \]

• 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
  \[ |\neg \text{Ctrl} \land \text{Asm} \Rightarrow \text{Spec} | = 0 \]

• Example for linking invariant:
  \[ \text{Link}_{C,A} = \left\lceil G \Leftarrow \Rightarrow (G_1 + G_2 > 0) \right\rceil \]

Obstacle (iv): 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: Motivation

• Recall: Given assumptions as a DC formula 'Asm' on the input observables, verifying correctness of 'Ctrl' wrt. 'Spec' amounts to proving
  \[ |\neg \text{Ctrl} \land \text{Asm} \Rightarrow \text{Spec} | = 0 \]

• 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?
Infinity of regular languages is decidable.

\[ L \subseteq \{ \Sigma(\in) \} \]

Each state assertion has a propositional interpretation. For all RDC formulae, there is realizability problem for RDC with discrete time is decidable.