Contents & Goals

Last Lectures:
• (Un)decidability results for fragments of DC in discrete and continuous time.

This Lecture:
• Educational Objectives:
  • What does this standard form mean? Give an satisfying interpretation.
  • What are implementables? What is a control automaton?
  • Please specify (and prove correct) a controller which satisfies this requirement.

Content:
• DC Standard Forms
• Control Automata
• DC Implementables

Requirements vs. Implementations

• Problem: In general, a DC requirement doesn’t tell how to achieve it, how to build a controller/write a program which ensures it.

• What a controller (clearly) can do is:
  • Consider inputs now, plant sensors, actuators, controller
  • Change (local) state, or
  • Wait,
  • Set outputs now. (But not, e.g., consider future inputs now.)

• So, if we have:
  • a DC requirement ‘Req’,
  • a description ‘Impl’ in DC, which “uses” just these operations,

• proving correctness amounts to proving $|\neg \square (\neg \neg P)$ Impl $\Rightarrow$ Req (in DC)

• and we (more or less) know how to program (the correct) ‘Impl’ in a PLC language, or in C on a real-time OS, or or or.

Approach: Control Automata and DC Implementables

Plan:
• Introduce DC Standard Forms
• Introduce Control Automata
• Introduce DC Implementables as subset of DC Standard Forms

DC Standard Forms: Followed-by

In the following:
• $F$ is a DC formula,
• $P_a$ a state assertion,
• $\theta$ a rigid term.

Followed-by:

$F \rightarrow \llceil P \rrceil$:

$\Leftrightarrow \Rightarrow \neg \Delta (F; \llceil \neg P \rrceil)$

$\Leftrightarrow \Rightarrow \Box \neg (F; \llceil \neg P \rrceil)$

In other symbols:

$\forall x \cdot \Box (((F \land \ell = x); \ell > 0 = \Rightarrow (F \land \ell = x); \llceil P \rrceil); \text{true})$
∀ \( x \) \( \Box \left( (F \land \ell = x) ; \ell > 0 \right) \implies \left( F \land \ell = x \right) ; \left\lceil P \right\rceil \land \text{true} \)

\( \left\lceil Q \right\rceil \leftarrow \left\lceil P \right\rceil \land \ell = 1 \)
Control Automata

Let \( X_1, \ldots, X_k \) be \( k \) state variables ranging over finite domains \( D(X_1), \ldots, D(X_k) \).

With a DC formula 'Impl' ranging over \( X_1, \ldots, X_k \) we have a system of \( k \) control automata.

'Impl' is typically a conjunction of DC implementables.

A state assertion of the form \( X_i = d_i, d_i \in D(X_i) \), which constrains the values of \( X_i \), is called basic phase of \( X_i \).

A phase of \( X_i \) is a Boolean combination of basic phases of \( X_i \).

Abbreviations:

- Write \( X_i \) instead of \( X_i = 1 \), if \( X_i \) is Boolean.
- Write \( d_i \) instead of \( X_i = d_i \), if \( D(X_i) \) is disjoint from \( D(X_j) \), \( i \neq j \).

Example: Gas Burner

\[
\begin{align*}
H & \text{ Boolean, representing heat request (input)} \\
F & \text{ Boolean, representing flame (input)} \\
C & \text{ with } D(C) = \{ \text{idle, purge, ignite, burn} \}, \text{ representing the status of the controller (local)} \\
G & \text{ Boolean, representing gas valve (output)}
\end{align*}
\]

Basic phase of \( C \):
\( C = \text{purge} \) (or only: \( \text{purge} \))

Phase of \( C \):
\( \text{purge} \lor \text{idle} \)

Specification by DC Implementables

Let \( X_1, \ldots, X_k \) be a system of \( k \) control automata.

Let 'Impl' be a conjunction of DC implementables.

Then 'Impl' specifies all interpretations \( I \) of \( X_1, \ldots, X_k \) and all valuations \( V \) such that \( I, V| = 0 \) Impl.

Example: Gas Burner

\[
\text{Example: Gas Burner controller as a system of four control automata:}
\]

\[
\begin{align*}
\text{• } & H \text{ Boolean, representing heat request (input)} \\
\text{• } & F \text{ Boolean, representing flame (input)} \\
\text{• } & C \text{ with } D(C) = \{ \text{idle, purge, ignite, burn} \}, \text{ representing the status of the controller (local)} \\
\text{• } & G \text{ Boolean, representing gas valve (output)}
\end{align*}
\]

Basic phase of \( C \):
\( C = \text{purge} \) (or only: \( \text{purge} \))

Phase of \( C \):
\( \text{purge} \lor \text{idle} \)
Lemma 3.16: \( \leq \Rightarrow \int \neg \cdot \cdot \cdot \Rightarrow \int \neg \cdot \cdot \cdot \)

Case 0:

\( \bullet \quad \epsilon \leq F \Rightarrow \int \neg \cdot \cdot \cdot \)

Case 1:

\( \bullet \quad \epsilon = | \cdot \cdot \cdot \)

Let \( \epsilon \leq F \Rightarrow \int \neg \cdot \cdot \cdot \)

Case 2:

\( \bullet \quad \epsilon = | \cdot \cdot \cdot \)

Lemma 3.15:

\[ \frac{\text{Gas Burner Controller Specification: Assumptions}}{\text{Gas Burner Controller Correctness Proof}} \]
Lemma 3.16 Cont'd

• Case 2: \( I, V, [b, e] | = \lceil \text{burn} \rceil; \text{true} \land \ell \leq 30 \rightarrow \lceil \text{burn} \lor \text{idle} \rceil \) (Seq-4)

• Case 3: \( I, V, [b, e] | = \lceil \text{ignite} \rceil; \text{true} \land \ell \leq 30 \rightarrow \lceil \text{ignite} \lor \text{burn} \rceil \) (Seq-3)

• Case 4: \( I, V, [b, e] | = \lceil \text{purge} \rceil; \text{true} \land \ell \leq 30 \rightarrow \lceil \text{purge} \lor \text{ignite} \rceil \) (Seq-2)

Discussion

We used only 'Seq-1', 'Seq-2', 'Seq-3', 'Seq-4', 'Prog-2', 'Syn-2', 'Syn-3', 'Stab-2', 'Stab-5', 'Stab-6'.

What about \( \text{Prog-1} = \lceil \text{purge} \rceil \geq 30 + \epsilon \rightarrow \lceil \neg \text{purge} \rceil \) for instance?

References
References