Supervisory control synthesis

Bert van Beek

Systems Engineering Group
Dept. of Mechanical Engineering

5 November 2009
System view

A system can be divided in (uncontrolled) plant $P$ and supervisor (controller) $S$:

$S$ ensures that $P$ satisfies its control requirements $R_S$. 
Traditional engineering

\[ \text{define} \quad R_{S/P} \rightarrow \text{design} \quad D_{S/P} \rightarrow \text{realize} \quad Z_S \rightarrow \text{integrate} \quad \text{Infrastructure} \rightarrow \]

\[ \text{define} \quad R_P \rightarrow \text{design} \quad D_P \rightarrow \text{realize} \quad Z_P \rightarrow \text{integrate} \]
Model based engineering
Model based engineering
Model based engineering

hardware-in-the-loop simulation and testing
Model based engineering

- Define $R_S/P$
- Design $D_{S/P}$
- Model $M_S$
- Realize $Z_S$

- Define $R_P$
- Design $D_P$
- Model $M_P$
- Realize $Z_P$

- Infrastructure $I$
- Integrate

Final implementation testing
Synthesis based engineering
Supervisory control synthesis

The resulting supervisor is:

- by construction mathematically correct w.r.t. $M_{RS}$
- controllable (allows uncontrollable events)
- non-blocking (deadlock free and livelock free)
- maximally permissive allowing selection of 'optimal' sequence of events
Supervisory synthesis flavors

Event-based:

- Plant and control requirements modeled by means of automata
- Monolithic SCS: synthesis of one (big) monolithic supervisor in the form of an automaton
- Modular SCS: control requirements partitioned into groups; synthesis of a supervisor for each group
- Can deal with unobservable events

State-based (state tree structures):

- Plant modeled by means of automata
- Control requirements modeled by means of automata, by means of predicates over states (state exclusion), and by means of state transition exclusion
- Monolithic SCS: synthesis of a BDD (Binary Decision Diagram) for each event (very efficient algorithm)
- No unobservable events
Motor and brake

Plant model:
Motor and brake

Equivalent plant model:
Motor and brake event-based

The brake may not be on when the motor is on.

Green area represents safe behavior in plant model:
Motor and brake event-based

Event-based control requirement:
Motor and brake event-based

In this case, supervisor equals control requirement:
Motor and brake state-based

Plant model:

State-based control requirement:

$$\neg (\text{BrakeOn} \land \text{MotorOn})$$
Motor and brake state-based

Plant model:

State-based control requirement:

\[ \neg (\text{BrakeOn} \land \text{MotorOn}) \]
Motor and brake state-based

Synthesis of supervisor, forbidden state:

\[ \neg (\text{BrakeOn} \land \text{MotorOn}) \]

State-based control requirement:
Motor and brake state-based

Synthesis of supervisor, disable events:

State-based control requirement:
\[
\neg (\text{BrakeOn} \land \text{MotorOn})
\]
Tool chain
Real-time control of a patient support system

Supervisory control synthesis using:
- modular supervisory control
- state-based supervisory control

Uncontrolled system: 6.3 billion states
Plant model

Patient support table modeled by
- 19 automata

User interface modeled by
- 8 automata

Total uncontrolled systems
- 6.3 billion states
Control requirements

Event-based:
- 57 automata

State-based:
- 4 automata
- 50 logical expressions

Examples of requirements:
- Do not move beyond end sensors
- Only motorized movement if clutch is active
- Only move vertically if horizontally in maximal out position
- Tumble switch moves table up and down, or in and out
Synthesized supervisor

Supervisor event-based

- The model of the supervisor consists of 14 modular supervisors
- The size of each supervisor is in the range of 20 – 2,000 states

Supervisor state-based

- One BDD for each action

Validation:

- The synthesized supervisor has been simulated in parallel with a (hybrid) model of the plant
- A real-time supervisory controller execution environment has been implemented
- The synthesized supervisor has been executed in this environment to control the actual patient support system
Conclusions

• Supervisory control synthesis focusses on designing control requirements instead of designing controllers
• Control requirements can be read and understood by domain experts and software experts
• Debugging and improving control requirements instead of debugging and improving control code
• Prove of concepts in several industrial cases
Acknowledgements

- Dennis Hendriks, programmer SE
- Albert T. Hofkamp, scientific programmer SE
- Jason Markovski, postdoc SE
- Asia van de Mortel-Fronczak, UD SE
- Damian Nadalus, PhD student SE
- Jacobus (Koos) E. Rooda, group leader SE
- Ramon R.H. Schiffelers, postdoc SE
- Rong Su, postdoc SE
- Rolf Teunissen, PhD student SE
- Peter Thijs, programmer SE