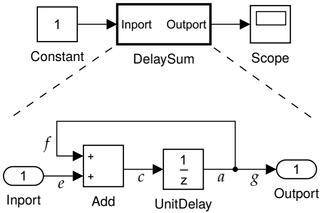
Compositional Semantics and Analysis of Hierarchical Block Diagrams

Iulia Dragomir[1]

joint work with Viorel Preoteasa[1] and Stavros Tripakis[1,2]

[1]Aalto University, Finland
[2]UC Berkeley, USA

Hierarchical block diagrams



Consist of:

- atomic components
- composed components (or subsystems)
- communication links (instantaneous)

Simulink is a HBD language for embedded control system design.

Goal: compositional semantics and analysis of HBDs

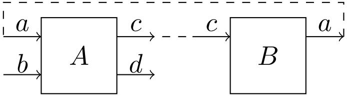
Compositional semantics and analysis of HBDs

- Compositional semantics:
  - How to translate HBDs into a formal compositional reasoning framework
- Compositional analysis:
  - Compositional verification
  - Compatibility checking

Refinement calculus for reactive systems (RCRS): a compositional reasoning framework

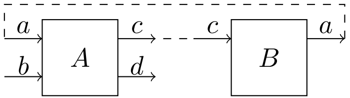
- Introduced in [Tripakis et al., TOPLAS 2011], and [Preoteasa et al., EMSOFT 2014]
- Formal model:
  - monotonic predicate transformers
  - 3 composition operators: serial ($\circ$), parallel ($\|$) and feedback (feedback)
  - refinement operator
- Allows for:
  - modeling open, non-deterministic, and non-input-receptive systems
  - modeling safety and liveness properties
  - component substitutability, reusability
  - compositional and incremental design

A non-trivial problem: translating HBDs into RCRS

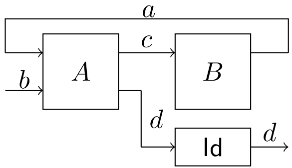


Input diagram

A non-trivial problem: translating HBDs into RCRS
Translation 1

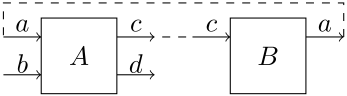


Input diagram

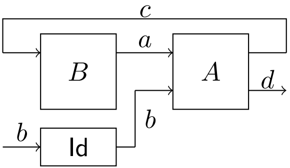


RCRS term: $\mathsf{feedback}_a(P_A \circ (P_B \parallel \mathsf{Id}))$

A non-trivial problem: translating HBDs into RCRS
Translation 2

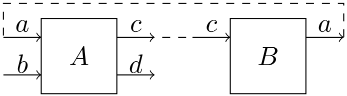


Input diagram

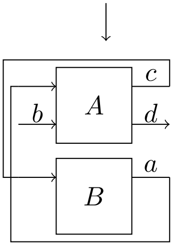


RCRS term: $\text{feedback}_c((P_B \parallel \text{Id}) \circ P_A)$

A non-trivial problem: translating HBDs into RCRS
Translation 3

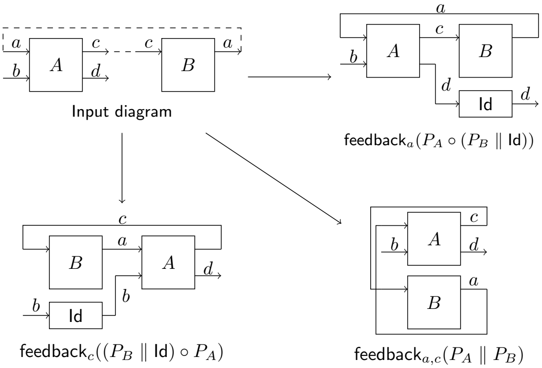


Input diagram



RCRS term: $\mathsf{feedback}_{a,c}(P_A \parallel P_B)$

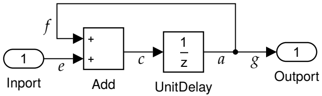
A non-trivial problem: translating HBDs into RCRS
Questions



- What are the advantages/drawbacks of these expressions?
  $\rightarrow$ How efficiently can these terms be analyzed?
- Are these expressions semantically equivalent?

Another non-trivial problem: expansion and simplification of RCRS terms

"DelaySum" block diagram:



translation $\downarrow$

$$\mathsf{DelaySum} = \mathsf{feedback}((\mathsf{Add} \parallel \mathsf{Id}) \circ \mathsf{UnitDelay} \circ (\mathsf{Split} \parallel \mathsf{Id}))$$

expansion and simplification $\downarrow$

$$\mathsf{DelaySum} = [e, s \rightsquigarrow s, s + e]$$

Contributions

1. Implementation of RCRS in the Isabelle theorem prover
2. Translation of HBDs into RCRS
3. Expansion and simplification of RCRS terms in Isabelle
4. Case study: realistic Simulink model from Toyota

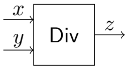
Outline

Outline

Monotonic predicate transformers

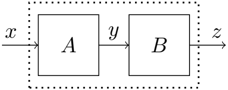
- Classic mechanism to represent programs
- Weakest precondition semantics [Dijkstra et al.]
- Atomic Simulink components can be represented by monotonic predicate transformers (MPTs)
- Example:

  $\mathsf{Div} = \{x, y : y \neq 0\} \circ [x, y \leadsto \frac{x}{y}]$



Composition operators

- Serial composition

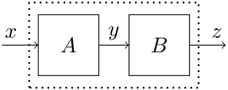


- Parallel composition

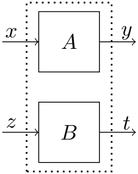
- Feedback composition

Composition operators

- Serial composition



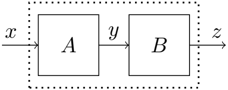
- Parallel composition



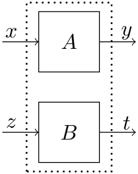
- Feedback composition

Composition operators

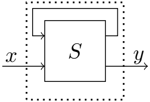
- Serial composition



- Parallel composition



- Feedback composition

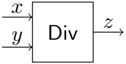


Outline

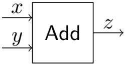
Outline

Translating (standard) atomic components

- An atomic component becomes an atomic monotonic predicate transformer.
- Examples:
  - a Div component
    $\text{Div} = \{x, y : y \neq 0\} \circ [x, y \rightsquigarrow \frac{x}{y}]$

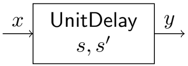


  - an Add component
    $\text{Add} = [x, y \rightsquigarrow x + y]$

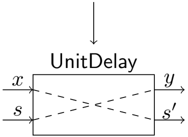


Translating stateful atomic components

- Stateful atomic components define current- and next-state variables
- Example:
  - a UnitDelay component
    UnitDelay $= [x, s \rightsquigarrow s, x]$



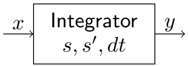
Simulink representation



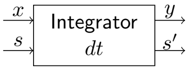
Atomic MPT representation

Translating continuous-time atomic components

- Continuous-time atomic components are discretized and parameterized by *dt*
- Example:
  - an Integrator component
    Integrator$(dt) = [x, s \rightsquigarrow s, s + x \cdot dt]$



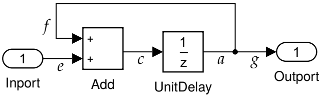
Simulink representation



Atomic MPT representation

Outline

Composite monotonic predicate transformers



Simulink diagram

translation $\big\downarrow$ **?**

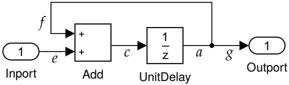
$$\text{DelaySum} = \text{feedback}((\text{Add} \parallel \text{Id}) \circ \text{UnitDelay} \circ (\text{Split} \parallel \text{Id}))$$

Composite MPT

Translation strategies

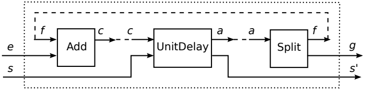
3 translation strategies:

- feedback-parallel
- incremental
- feedbackless



Simulink diagram

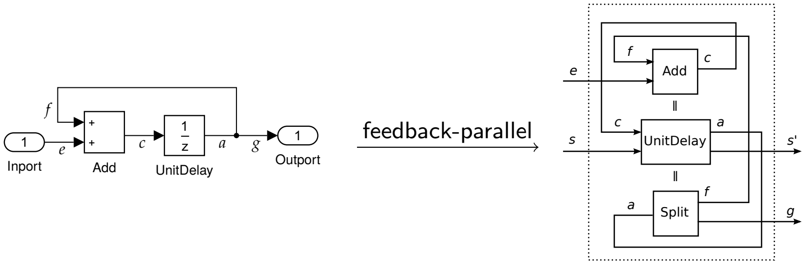
↓



Atomic MPTs representation

Feedback-parallel translation

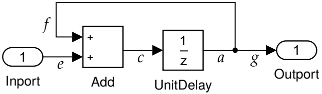
- Key idea: compose all components in parallel and then connect outputs to inputs by applying feedback operations



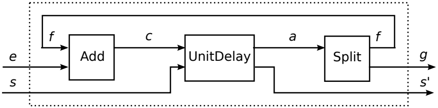
$$\text{DelaySum} = \text{feedback}_{f,c,a}(\text{Add} \parallel \text{UnitDelay} \parallel \text{Split})$$

Incremental translation

- Key idea:
  - sort components topologically according to dependencies in the diagram
  - compose components 1-by-1
  - for each pair of components determine which composition operator(s) to use



$\downarrow$ incremental

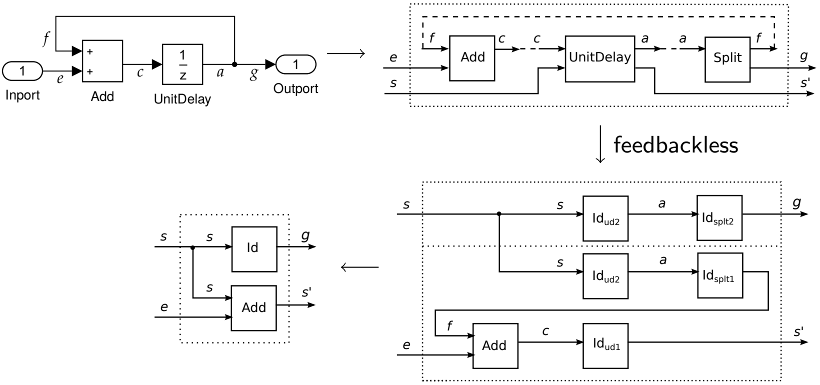


$$\text{Aux} = (\text{Add} \parallel \text{Id}) \circ \text{UnitDelay}$$

$$\text{DelaySum} = \text{feedback}_f(\text{Aux} \circ (\text{Split} \parallel \text{Id}))$$

Feedbackless translation

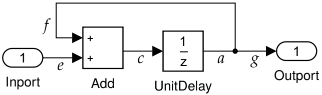
- Key idea: eliminate feedback by replacing it with direct operations on current- and next-state variables (like for stateful atomic components)



$$\text{DelaySum} = [s, e \rightsquigarrow s, s, e] \circ (\text{Id} \parallel \text{Add})$$

Outline

From composite MPTs to atomic MPTs



Simulink diagram

translation $\downarrow$

$$\mathsf{DelaySum} = \mathsf{feedback}((\mathsf{Add} \parallel \mathsf{Id}) \circ \mathsf{UnitDelay} \circ (\mathsf{Split} \parallel \mathsf{Id}))$$

Composite MPT

expansion and simplification $\downarrow$ **?**

$$\mathsf{DelaySum} = [e, s \rightsquigarrow s, s + e]$$

Simplified (atomic) MPT

Obtaining simplified MPTs

- Expand definitions of MPTs, $\circ$, $\|$ and feedback
  $\rightarrow$ an MPT of the form $\{p\} \circ [f]$ is obtained
  $\rightarrow$ but formulas p and f can grow very large ...

- Simplify p and f using rewriting rules

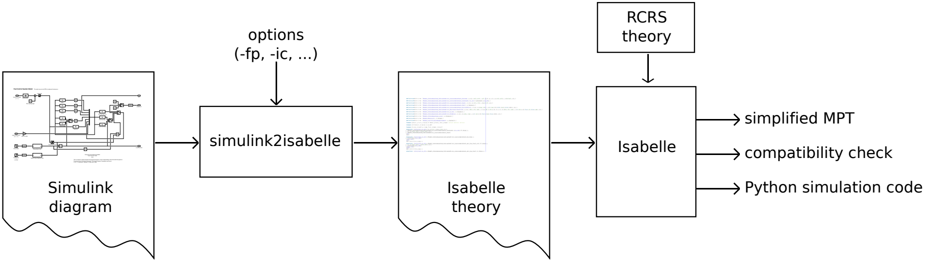
- 2050 lines of Isabelle code

Compatibility checking

- Simplify the CPT to an MPT $\{p\} \circ [f]$
- Verify that p is not false
- A satisfiability problem

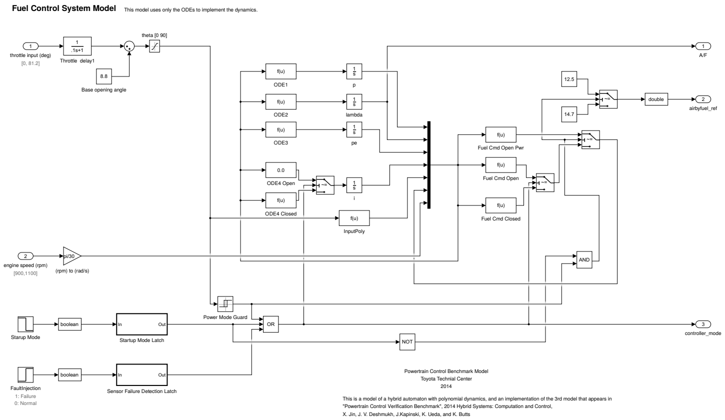
Outline

Toolset



Publicly available at: `users.ics.aalto.fi/iulia/sim2isa.shtml`

- Publicly available benchmark: `http://cps-vo.org/group/ARCH/benchmarks`
- Simulink model:
  - 3-level hierarchy
  - 104 blocks: 97 atomic blocks and 7 subsystems
  - 101 links of which 7 feedbacks



Evaluation results I

- Negligible translation time (< 1sec) for all 3 strategies
- Expansion/simplification time:
  - feedback-parallel strategy: 15min to 1h (depending on translation options)
  - incremental strategy: 10min to 14min (depending on translation options)
  - feedbackless strategy: < 1min

Evaluation results I

- Negligible translation time (< 1sec) for all 3 strategies
- Expansion/simplification time:
  - feedback-parallel strategy: 15min to 1h (depending on translation options)
  - incremental strategy: 10min to 14min (depending on translation options)
  - feedbackless strategy: $<$ 1min

Evaluation results II

- Length of the final, top-level, simplified MPT: 122k characters

```
NFB_Model_type_reals ?dt =
{.(si_vh1, si_vh2, si_rh, si_cf, si_qe, si_xb, si_qc1, si_pd, si_f, si_x, si_qc2, si_xd, si_ga, si_ab, si_ba, si_ge).0 ≤ si_qe.} ∘
[-λ(si_vh1, si_vh2, si_rh, si_cf, si_qe, si_xb, si_qc1, si_pd, si_f, si_x, si_qc2, si_xd, si_ga, si_ab, si_ba, si_ge).
     (si_vh1 + 1,
      if si_vh1 * ?dt < 3 then 0
      else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then si_vh2 + 1 else if 3 ≤ si_vh1 * ?dt ∧ 205223 ≤ si_vh2 * ?dt * 20000 ∧ si_vh2 * ?dt < 205223 / 10000 - ?dt then si_vh2 + 1 else 0,
      si_rh + 1, si_cf + 1, 0, si_xb + 1, si_qc1 + 1, si_pd + 1, if 60 ≤ si_xb * ?dt ∨ si_f ≠ 0 then 1 else 0, if 10 ≤ si_pd * ?dt ∨ si_x ≠ 0 then 1 else 0,
      (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) / exp (10 * si_qc1 * ?dt),
      if 0 ≤ (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
              exp (10 * si_qc1 * ?dt) +
              44 / 5 ∧
         (90 < (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
               exp (10 * si_qc1 * ?dt) +
               44 / 5 ∨
          70 ≤ (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                exp (10 * si_qc1 * ?dt) +
                44 / 5)
      then 1 else if (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                     exp (10 * si_qc1 * ?dt) +
                     44 / 5
                     < 0 ∨
                     (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                     exp (10 * si_qc1 * ?dt) +
                     44 / 5
                     ≤ 90 ∧
                     (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                     exp (10 * si_qc1 * ?dt) +
                     44 / 5
                     ≤ 50
                  then 0 else si_xd,
      si_ab + (if 0 ≤ (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                       exp (10 * si_qc1 * ?dt) +
                       44 / 5 ∧
                  ((si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                   exp (10 * si_qc1 * ?dt) +
                   44 / 5
                   ≤ 90 ⟶
                   70 ≤ (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                         exp (10 * si_qc1 * ?dt) +
                         44 / 5) ∨
                  0 ≤ (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                       exp (10 * si_qc1 * ?dt) +
                       44 / 5 ∧
                  ((si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
                   exp (10 * si_qc1 * ?dt) +
                   44 / 5
                   ≤ 90 ⟶
                   ¬ (si_qc2 * exp ((10 * si_qc1 - 10) * ?dt) + 10 * (if si_vh1 * ?dt < 3 then 0 else if 3 ≤ si_vh1 * ?dt ∧ si_vh2 * ?dt * 20000 < 205223 then 13367 / 250 else 0) * exp (10 * si_qc1 * ?dt) * ?dt) /
```

Semantical equivalence of the translation strategies

- For all studied examples, including FCS, the simplified MPTs are semantically equivalent

  $\rightarrow$ proved in Isabelle

- Proving this in general: ongoing work

Compatibility checking

- The FCS Simulink model is proven compatible $\forall dt > 0$
- i.e., the model's simplified assert condition is satisfiable $\forall dt > 0$

$\rightarrow$ proved in Isabelle

All Isabelle proofs available at `users.ics.aalto.fi/iulia/sim2isa.shtml`

Compatibility checking

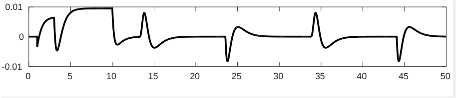
- The FCS Simulink model is proven compatible $\forall dt > 0$
- i.e., the model's simplified assert condition is satisfiable $\forall dt > 0$

$\rightarrow$ proved in Isabelle

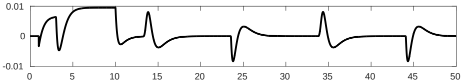
All Isabelle proofs available at `users.ics.aalto.fi/iulia/sim2isa.shtml`

Validation by simulation

- From Isabelle we can automatically generate simulation code (in Python)
- Simulation plots obtained from the FCS model using Simulink vs. our tool are nearly identical
  - $|\text{error}| \leq 6.1487 \cdot 10^{-5}$



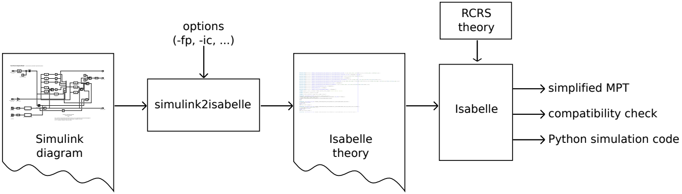
Simulink simulation



Simulation of the simplified MPT

Outline

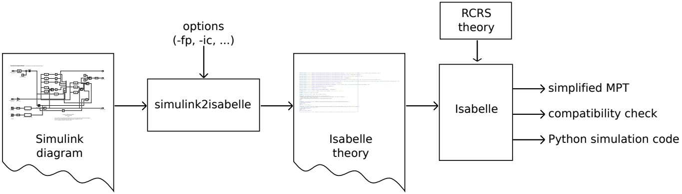
Conclusion



- Compositional semantics of HBDs
- 3 translation strategies of HBDs to RCRS
- Implementation of the RCRS framework in Isabelle
- Evaluation on real-life automotive case study

Thank you!
Questions?

Conclusion



- Compositional semantics of HBDs
- 3 translation strategies of HBDs to RCRS
- Implementation of the RCRS framework in Isabelle
- Evaluation on real-life automotive case study

Thank you!
Questions?