Towards Compositional Feedback in Non-Deterministic and Non-Input-Receptive Systems

Stavros Tripakis[1,2]

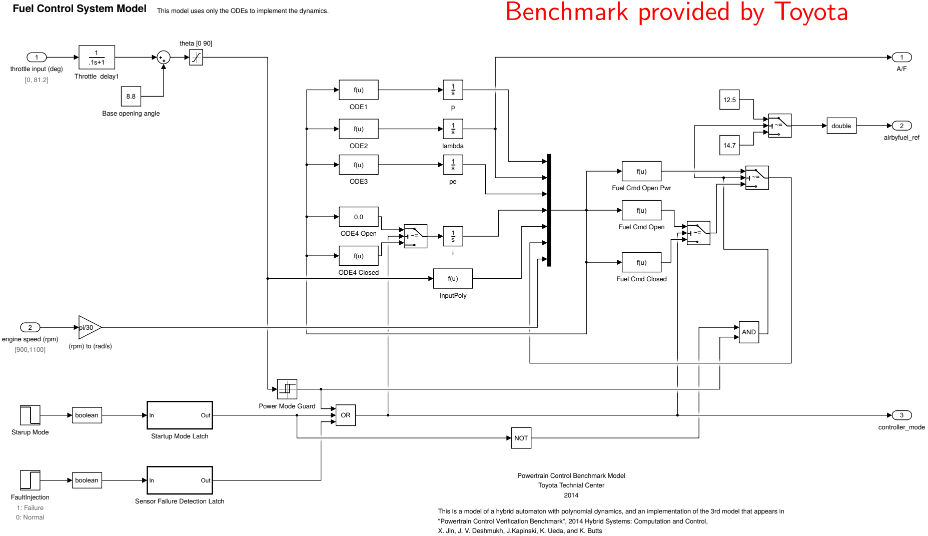
Joint LICS'16 paper with Viorel Preoteasa[2]
Contributions to the RCRS framework also by Iulia Dragomir[2]

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

ExCAPE PI Meeting 2016

Motivation: compositional reasoning for Simulink

Benchmark provided by Toyota



What does "compositional reasoning for Simulink" mean?

1. Be able to model basic Simulink blocks: constants, adders, integrators, ...
2. Be able to model arbitrary Simulink models: **hierarchical block diagrams**.

What does "compositional reasoning for Simulink" mean?

1. Be able to model basic Simulink blocks: constants, adders, integrators, ...
2. Be able to model arbitrary Simulink models: **hierarchical block diagrams**.
3. Be able to **check compatibility**: "lightweight verification", akin to type-checking.
4. Be able to **synthesize system from subsystems**: compute component compositions bottom-up.
5. Be able to **check substitutability**: component **refinement**.

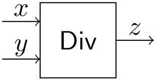
What does "compositional reasoning for Simulink" mean?

1. Be able to model basic Simulink blocks: constants, adders, integrators, ...
2. Be able to model arbitrary Simulink models: **hierarchical block diagrams**.
3. Be able to **check compatibility**: "lightweight verification", akin to type-checking.
4. Be able to **synthesize system from subsystems**: compute component compositions bottom-up.
5. Be able to **check substitutability**: component **refinement**.
6. Be able to express and verify **safety and liveness** properties.

Refinement Calculus of Reactive Systems (RCRS)

[EMSOFT 2009, TOPLAS 2011, SPIN 2013, EMSOFT 2014, FPS 2014, SPIN 2016, LICS 2016]

- Relational interfaces: symbolic, synchronous version of Alfaro-Henzinger's *interface automata*
- Example: relational interface for division component



contract: $y \neq 0 \wedge z = \frac{x}{y}$

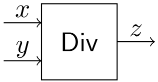
- Can model non-input-receptive systems:

$y \neq 0 \wedge z = \frac{x}{y}$ instead of $y \neq 0 \rightarrow z = \frac{x}{y}$

Refinement Calculus of Reactive Systems (RCRS)

[EMSOFT 2009, TOPLAS 2011, SPIN 2013, EMSOFT 2014, FPS 2014, SPIN 2016, LICS 2016]

- Relational interfaces: symbolic, synchronous version of Alfaro-Henzinger's *interface automata*
- Example: relational interface for division component



contract: $y \neq 0 \wedge z = \frac{x}{y}$

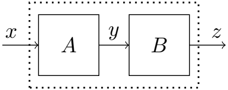
- Can model non-input-receptive systems:

$y \neq 0 \wedge z = \frac{x}{y}$ instead of $y \neq 0 \rightarrow z = \frac{x}{y}$

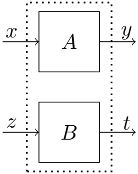
- RCRS:
  - Extends relational interfaces to handle liveness properties
  - Semantics based on monotonic property transformers
  - Can handle stateful systems, and **continuous** blocks by Euler discretization, e.g., Integrator: $s' = s + x \cdot dt$

Composition operators

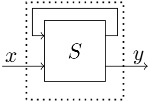
- Serial composition (∀-∃ synthesis inside!)



- Parallel composition (conjunction)

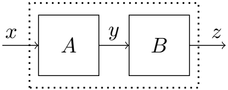


- Feedback composition

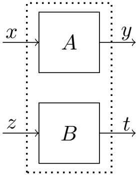


Composition operators

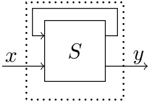
- Serial composition (∀-∃ synthesis inside!)



- Parallel composition (conjunction)

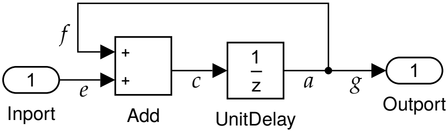


- Feedback composition



How to define feedback composition?

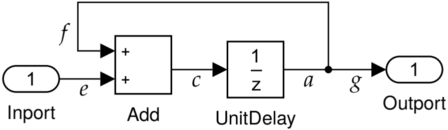
“Easy” feedback: “broken” by unit delays



$$a(k) = c(k-1)$$

No instantaneous cyclic dependency.

"Easy" feedback: "broken" by unit delays

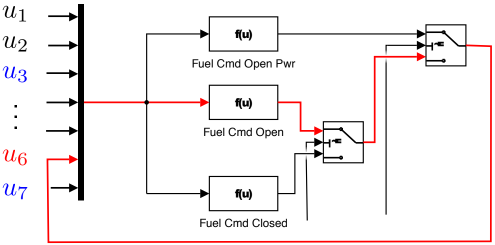


$$a(k) = c(k-1)$$

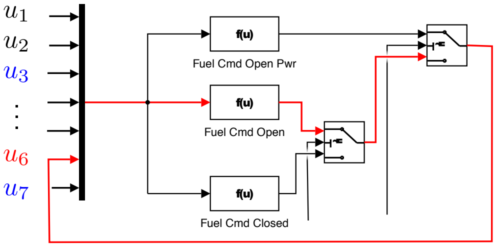
No instantaneous cyclic dependency.

Handled in our earlier work on relational interfaces:
Tripakis, Lickly, Henzinger, Lee. *A Theory of Synchronous Relational Interfaces*, TOPLAS 2011.

"Not too hard" feedback



“Not too hard” feedback

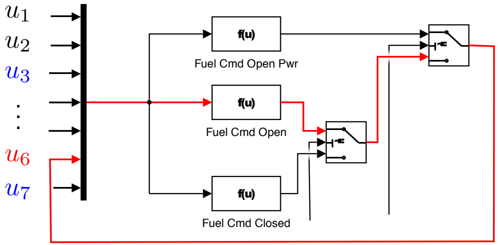


Output of FuelCmdOpen does not depend on u_6:

$$\text{FuelCmdOpen}(u_1, u_2, \ldots, u_7) = \frac{1}{14.7}(-0.366 + 0.08979 u_7 u_3 - 0.0337 u_7 {u_3}^2 + 0.0001 {u_7}^2 u_3)$$

No instantaneous cyclic dependency

“Not too hard” feedback



Output of FuelCmdOpen does not depend on u_6:

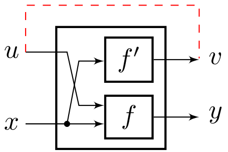
$$\text{FuelCmdOpen}(u_1, u_2, \ldots, u_7) = \frac{1}{14.7}(-0.366 + 0.08979 u_7 u_3 - 0.0337 u_7 {u_3}^2 + 0.0001 {u_7}^2 u_3)$$

No instantaneous cyclic dependency

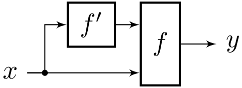
Handled in recent work:
Dragomir, Preoteasa, Tripakis. *Compositional Semantics and Analysis of Hierarchical Block Diagrams*, SPIN 2016.

"Not too hard" feedback: solution

If we know the block's **internals and input-output dependencies**:

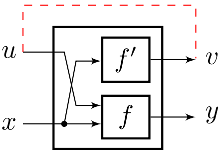


then this feedback reduces to serial composition:

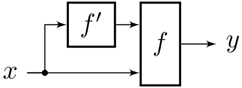


“Not too hard” feedback: solution

If we know the block’s **internals and input-output dependencies**:

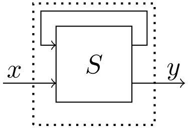


then this feedback reduces to serial composition:



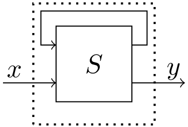
What if we don’t know the internals of the block?

How to define general feedback?



How to define feedback for **any system** S?

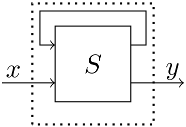
How to define general feedback?



How to define feedback for **any system** S?

A non-trivial problem.

How to define general feedback?



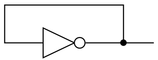
How to define feedback for **any system** S?

A non-trivial problem.

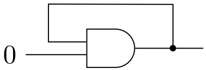
Even for deterministic and input-receptive systems.

Feedback for **deterministic and input-receptive** systems

How to distinguish invalid feedbacks:



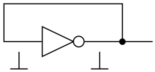
from valid ones:



Solution [Malik '94]: **fixpoint** semantics, starting with **unknown** values ("bottom" $\perp$).

Fixpoint semantics (also called *constructive* semantics)

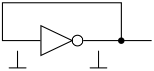
$$\neg\bot = \bot$$



Fixpoint stabilizes to $\bot$: feedback is invalid.

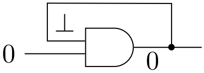
Fixpoint semantics (also called *constructive* semantics)

$$\neg\bot = \bot$$



Fixpoint stabilizes to $\bot$: feedback is invalid.

$$\bot \wedge 0 = 0$$



Fixpoint stabilizes to 0: feedback is valid.

Limitations of constructive semantics

- It only applies to **deterministic and input-receptive** systems (i.e., total functions)
  We also want to handle non-deterministic and non-input-receptive systems (partial relations).

Limitations of constructive semantics

- It only applies to **deterministic and input-receptive** systems (i.e., total functions)
  We also want to handle non-deterministic and non-input-receptive systems (partial relations).
- There is **no refinement** in existing constructive semantics frameworks.
  We not only have refinement, we also want refinement to be ***preserved*** *by feedback.*

Preservation of refinement by composition

We want feedback to be **compositional**, i.e., for a composition operator $\circ$:

If $A \sqsubseteq A'$ *(*A' *refines* A*) and* $B \sqsubseteq B'$ *then* $A \circ B \sqsubseteq A' \circ B'$.

This property is necessary for substitutability.

Preservation of refinement by composition

We want feedback to be **compositional**, i.e., for a composition operator $\circ$:

If $A \sqsubseteq A'$ *(*A' *refines* A*) and* $B \sqsubseteq B'$ *then* $A \circ B \sqsubseteq A' \circ B'$.

This property is necessary for substitutability.

In particular, for feedback, we want:
If $A \sqsubseteq A'$ *then* $\mathsf{feedback}(A) \sqsubseteq \mathsf{feedback}(A')$.

Preservation of refinement by composition

We want feedback to be **compositional**, i.e., for a composition operator $\circ$:

If $A \sqsubseteq A'$ *(*A' *refines* A*) and* $B \sqsubseteq B'$ *then* $A \circ B \sqsubseteq A' \circ B'$*.*

This property is necessary for substitutability.

In particular, for feedback, we want:
If $A \sqsubseteq A'$ *then* feedback$(A) \sqsubseteq$ feedback(A').

This is not easy to achieve – usual definitions don't work [TOPLAS 2011, FPS 2014, LICS 2016].

Results of LICS 2016 paper

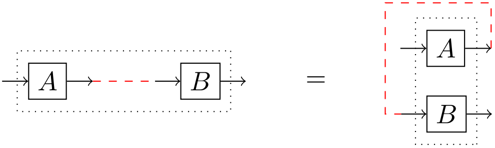
1. Two feedback operators:
   - **Instantaneous feedback**: applies to stateless ("memoryless") systems.
   - **Feedback with unit-delay**: applies to stateful systems; instantaneous dependencies "broken" by a unit delay.
2. Both operators can handle non-deterministic and non-input-receptive systems.
3. Both operators proven to be compositional: they preserve refinement.

Results of LICS 2016 paper

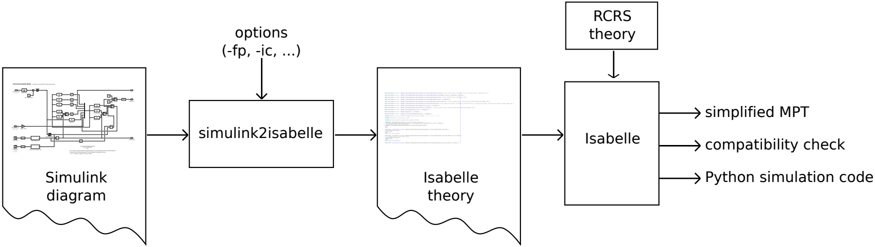
1. Two feedback operators:
   - **Instantaneous feedback**: applies to stateless ("memoryless") systems.
   - **Feedback with unit-delay**: applies to stateful systems; instantaneous dependencies "broken" by a unit delay.
2. Both operators can handle non-deterministic and non-input-receptive systems.
3. Both operators proven to be compositional: they preserve refinement.
4. In the special case of deterministic and input-receptive systems, both operators specialize to the standard solutions (instantaneous feedback specializes to constructive semantics).

Results of LICS 2016 paper

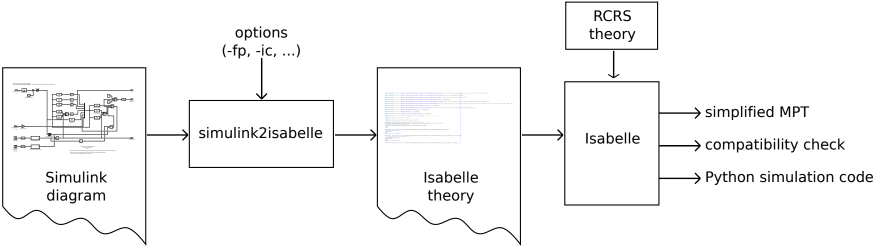
1. Two feedback operators:
   - **Instantaneous feedback**: applies to stateless ("memoryless") systems.
   - **Feedback with unit-delay**: applies to stateful systems; instantaneous dependencies "broken" by a unit delay.
2. Both operators can handle non-deterministic and non-input-receptive systems.
3. Both operators proven to be compositional: they preserve refinement.
4. In the special case of deterministic and input-receptive systems, both operators specialize to the standard solutions (instantaneous feedback specializes to constructive semantics).
5. Serial composition = parallel composition followed by feedback.



Toolset – downloadable from: `rcrs.cs.aalto.fi`



Toolset – downloadable from: `rcrs.cs.aalto.fi`



Another non-trivial problem: *Translation of arbitrary block diagrams to terms using the three primitive composition operators (serial, parallel, feedback).*

See: Dragomir, Preoteasa, Tripakis. *Compositional Semantics and Analysis of Hierarchical Block Diagrams*, SPIN 2016.

Thank you – questions?

Bibliography:

S. Tripakis, B. Lickly, T. A. Henzinger, and E. A. Lee.
A Theory of Synchronous Relational Interfaces.
ACM Transactions on Programming Languages and Systems (TOPLAS), 33(4), July 2011.

V. Preoteasa and S. Tripakis.
Refinement Calculus of Reactive Systems.
In *Proceedings of the 14th ACM & IEEE International Conference on Embedded Software (EMSOFT'14)*, 2014.

I. Dragomir, V. Preoteasa, and S. Tripakis.
Compositional Semantics and Analysis of Hierarchical Block Diagrams.
In *23rd International SPIN symposium on Model Checking of Software (SPIN 2016)*, LNCS. Springer, 2016.

V. Preoteasa and S. Tripakis.
Towards Compositional Feedback in Non-Deterministic and Non-Input-Receptive Systems.
In *31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2016.