1. Component-based design

Questions:
What are the right building blocks?
Which ones to use? How to connect them?
What is a component? How to reason about components?

4. Compositionality

Many existing frameworks, e.g., interface theories [dAH01]. But:
- focus on functional/correctness models and properties;
- not suitable for cyber-physical systems: timing, performance.

7. Actors

Formal descriptions of components:

\[ A = (P, Q, R_A) \]

- \( P \): set of input ports.
- \( Q \): set of output ports.
- \( R_A \subseteq Tr(P) \times Tr(Q) \): relation between input and output event traces.

Example: CSDF (cyclo-static dataflow [BELP96]) actor refining SDF actor:

8. Event traces

Event trace: port-wise vector of event sequences.
Event sequence: sequence of “pure” (non-valued) events in time.

10. The earlier-the-better refinement

\[ B = (P, Q, R_B) \] refines \( A = (P, Q, R_A) \), denoted \( B \sqsubseteq A \), iff
1. \( in_A \subseteq in_B \) (legal inputs of \( A \) are also legal in \( B \));
2. \( \forall x \in in_A \forall y \in B y \iff \exists y' \in y \wedge Ay' \) (outputs of \( B \) come no later than those of \( A \)).

Where \( y \sqsubseteq y' \) if events in \( y \) happen no later than those in \( y' \). E.g., \( \tau_1 \sqsubseteq \tau_2 \), \( \tau_2 \sqsubseteq \tau_1 \), but \( \tau_1 \nsubseteq \tau_2 \).

Example: CSDF (cyclo-static dataflow [BELP96]) actor refining SDF actor:

11. Results

- Refinement is compositional w.r.t. parallel, serial and feedback composition (under some conditions).
- Refinement preserves worst-case throughput and latency.
- Algorithms to check refinement and compute compositions for various finite representations (SDF, automata, ...).

- Semantical unification of existing frameworks (dataflow, automata, service curves, ...).

12. References


13. Conclusion

A compositional theory for timing and performance properties.
Deterministic abstractions for non-deterministic systems!