
Contract-based formalisms for heterogeneous and hybrid systems

Roberto Passerone

PARADES / Dipartimento di Ingegneria e Scienze dell'Informazione
University of Trento
Trento, Italy

with Alberto L. Sangiovanni-Vincentelli

PARADES / UC Berkeley

and Jerry R. Burch

Synopsys, Inc.

Motivations

- Many formalisms for component-based design, many purposes
 - Dill's asynchronous trace structures, for verification
 - The Tagged-Signal Model, for comparisons
 - Interface Automata, for incremental design
 - SPEEDS HRC, for speculative design
 - And many others (BIP, FunState, Synchronous Languages, Simulink, Reactive Modules, Process Algebras, etc.)
- But all deal with the same basic notions
 - What are components?
 - How do you compose them?
 - When are components compatible?
 - How do they refine?

Objectives

- Develop a framework that encompasses many models
 - Models of time, continuous and discrete values, communication paradigms
- Study relationship between the models
 - Show where they are similar and dissimilar
 - Approximate one model with another
 - Embed a model in another
- Prove results that apply to all models
 - Trade-off between generality and structure
 - Do more than simply classify existing models

Constructive vs. Axiomatic

- Constructive (Metropolis, Ptolemy)
 - Create a model with a **particular** structure
 - Prove properties from the particular structure
 - Encode other models in the structure to show they have the properties
- Axiomatic (Tagged-Signal Model)
 - Create an **abstract** structure made of uninterpreted operations on a yet to be defined set
 - Require certain properties (axioms) from the operations
 - Prove other properties or results from the axioms
 - Prove axioms for other models to show they have the properties

Agent Algebras

- Axiomatic framework
 - Non specific
 - Granularity at the “block” level
- A mathematical structure Q with the following elements
 - A set of agents D
 - For each agent, the set of the “signals” it uses (alphabet)
 - Operators on agents
- Typical operators (partial functions)
 - scoping: $\text{proj}(B)(p)$
 - instantiation: $\text{rename}(r)(p)$
 - composition: $p_1 \parallel p_2$
 - sequential: $p_1 \wedge p_2$

Axioms

- To comply with their intuitive interpretation, the operators must satisfy certain axioms, e.g.
 - $\alpha(\text{proj}(B)(p)) = B \cap \alpha(p)$
 - $\text{proj}(A)(p) = p$
 - $\alpha(\text{rename}(r)(p)) = r(\alpha(p))$
 - $\text{rename}(id)(p) = p$
 - $\alpha(p_1 \parallel p_2) = \alpha(p_1) \cup \alpha(p_2)$
 - **Parallel composition is associative and commutative**
 - $\text{proj}(B)(\text{proj}(B')(p)) = \text{proj}(B \cap B')(p)$
 - $\text{rename}(r)(\text{proj}(B)(p)) = \text{proj}(r'(B))(\text{rename}(r'')(p))$
 - $\text{proj}(B)(p) = \text{proj}(B)(\text{rename}(r)(p))$
 - $r(A) \cap A' \subseteq B$
 - **$\text{proj}(B)(p_1 \parallel p_2) = \text{proj}(B)(p_1) \parallel \text{proj}(B)(p_2)$**
 - **If $\alpha(p_1) \cap \alpha(p_2) \subseteq B$**

Synchronous Agent Algebra

(synchronous reactive)

- Agents are sets of behaviors
 - They communicate through events
- Let $A = \{ a, b, c \}$ be a set of signals
 - Behaviors are sequences in $\mathcal{L}_A = (2^A)^*$, e.g.
 - $x = \langle \{a\}, \{b\}, \{bc\}, \emptyset, \{ab\}, \emptyset, \{b\}, \dots \rangle$
 - $p = (A, P)$ where $P \subseteq (2^A)^*$
- Projection operator
 - Set intersection
 - $\text{proj}(\{a, c\})(x) = \langle \{a\}, \emptyset, \{c\}, \emptyset, \{a\}, \emptyset, \emptyset, \dots \rangle$
- Parallel composition operator
 - $p_1 \parallel p_2 = (A = A_1 \cup A_2, \text{proj}^{-1}(A)(P_1) \cap \text{proj}^{-1}(A)(P_2))$

Asynchronous Agent Algebra

(asynchronous trace structures)

- Agents are sets of behaviors
 - They communicate through events
- Let $A = \{ a, b, c \}$ be a set of signals
 - Behaviors are sequences in $\mathcal{L}_A = (2^A - \emptyset)^*$, e.g.
 - $x = \langle \{a\}, \{b\}, \{bc\}, \{ab\}, \{b\}, \{ac\}, \dots \rangle$
 - $p = (A, P)$ where $P \subseteq (2^A - \emptyset)^*$
- Projection operator
 - Set intersection + shrink of sequence
 - $\text{proj}(\{a, c\})(x) = \langle \{a\}, \{c\}, \{a\}, \{ac\}, \dots \rangle$
- Parallel composition operator
 - $p_1 \parallel p_2 = (A = A_1 \cup A_2, \text{proj}^{-1}(A)(P_1) \cap \text{proj}^{-1}(A)(P_2))$

Two-set models

(Dill's trace structures, Interface Automata, SPEEDS HRC)

- Each agent includes a set of *successes* and a set of *failures*
 - $p = (S, F); P = S \cup F$
 - The failures represent possible behaviors that are illegal uses of the component
 - Provides assumptions relative to the environment
- Composition
 - $p \parallel p' = (S \cap S', (F \cap P') \cup (F' \cap P))$
- HRC has explicit assumptions and guarantees
 - $p = (A, G);$
 - $p \parallel p' = ((A \cap A') \cup \neg G \cup \neg G', G \cap G')$
 - $F = \neg A, S = A \cap G$
- Interface Automata
 - Non-progressive states give assumptions
 - Autofailure manifestation during parallel composition (“hopeless” states)

Compatibility

- Two agents p and q may be incompatible
 - Roughly speaking, p and q are compatible when their parallel composition is defined
- No matter what compatibility is, the following must be true
 - If p' is compatible in a certain context, and if $p \leq p'$, then p must be compatible in the same context
 - In fact, this must be true for all contexts in which p' is compatible
- **Decision.** Compatibility becomes a parameter of our model
 - We express compatibility in terms of a set G of agents, called conformance set
 - p' is compatible in E iff $E[p'] \in G$
 - Similarly, p and q are compatible iff $p \parallel q \in G$

Order and Compatibility

- G induces an order on the agents called G-conformance order
 - $p \leq_G p'$ if and only if for all contexts E , if $E[p'] \in G$ then $E[p] \in G$
- If the order \leq in Q is the same as the order \leq_G induced by G , we say that Q has a G-conformance order
 - Note: different G 's induce different conformance orders
- We are particularly interested in composition contexts
 - Call it conformance relative to composition
 - $p \leq_G p'$ if and only if for all q , if $p' \parallel q \in G$ then $p \parallel q \in G$
- The maximally compatible agent is called a mirror
 - $p \leq p'$ if and only if $p \parallel \text{mirror}(p') \in G$
 - If Q has a mirror function, the mirror of p' completely characterizes its refinements
- Mirror properties (similar to complementation)
 - $p \leq q$ if and only if $\text{mirror}(q) \leq \text{mirror}(p)$
 - $p = \text{mirror}^2(p)$

Conformance and mirrors

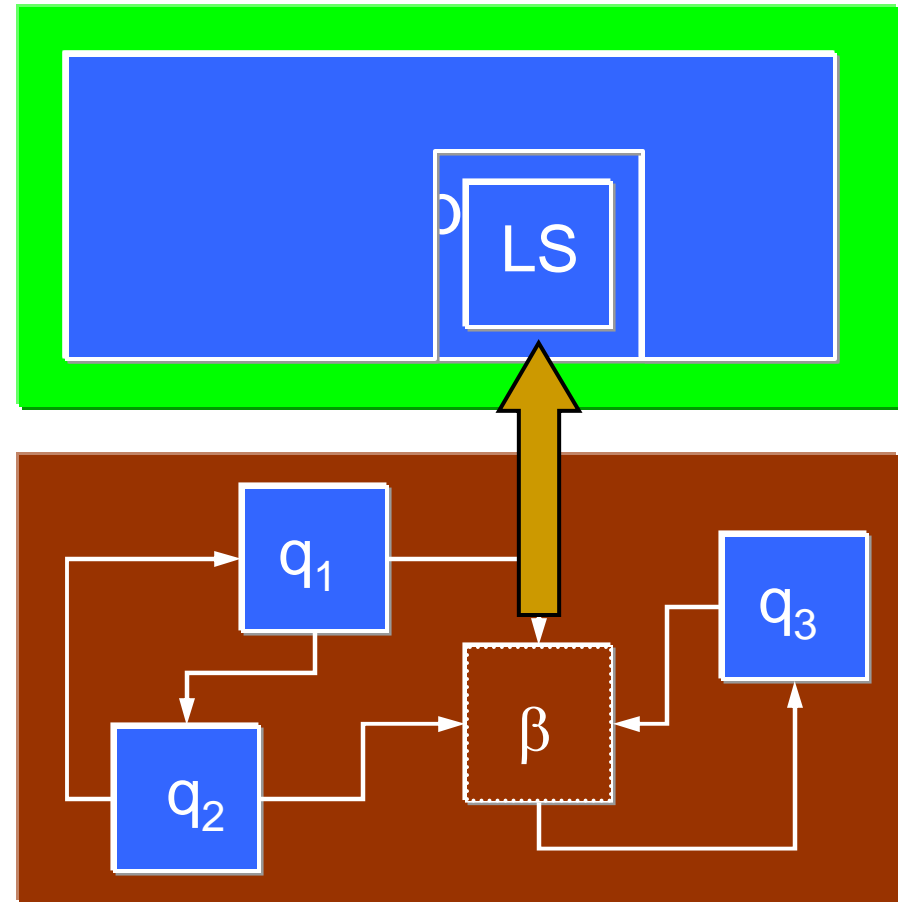
- Synchronous/asynchronous model
 - $G = \{ p \mid p \text{ has no behaviors } \}$
 - Compatibility means maximal guarantees
 - $p_1 \leq p_2$ if and only if $P_1 \subseteq P_2$
 - $\text{mirror}(p) = (A, \mathcal{L}_A - P)$
- Two-set models
 - $G = \{ p \mid p \text{ has no failures } (F = \emptyset) \}$
 - Compatibility means satisfying the assumptions
 - $p \leq p'$ if and only if $F \subseteq F'$ and $P \subseteq P'$
 - $\text{mirror}(p) = (S - F, \mathcal{L}_A - P)$
- HRC model
 - $p \leq p'$ if and only if $G \subseteq G'$ and $A' \subseteq A$
- Interface Automata
 - Alternating simulation (research direction)

Relating Models

- Common underlying algebraic structure
 - Define homomorphisms (operator preserving) functions between sets of behaviors
 - Derive corresponding abstractions between domains
- Conservative approximations give upper and lower bound
 - They preserve refinement from abstract to concrete
 - Similar to pairs of Galois connections (abstract interpretations) going in opposite directions
- Induce embeddings from abstract to concrete
 - Determine when agents can be expressed in both models
 - Help understand how one model fits in another

Local Specification Synthesis

- Global specification p'
- Implementation
 - ▣ $p = \beta \parallel q_1 \parallel q_2 \parallel q_3$
- Find the local specification of a component β such that p refines p'
- The components around β form its context



Application areas

- Supervisory control
 - Context: a plant to be controlled
 - Global specification: what the plant is supposed to do
 - Local specification: controller for the plant
 - Engineering Change (ECO), Rectification
 - Global specification: corrected functionality
 - Context: untouchable part of the system
 - Local specification: replacement part
 - Optimization
 - Global specification: old functionality
 - Context: rest of the system
 - Local specification: optimized part
 - Protocol conversion
-

Local Specification Synthesis

$$\text{proj}(B)(\beta \parallel q) \leq p'$$

if and only if

$$\beta \leq \text{mirror}(q \parallel \text{proj}(B)(\text{mirror}(p')))$$

- Result similar to many specific results
 - But the formulation here is generic
 - In particular we don't say what mirror is
 - Complexity depends on the model and its implementation

Local Specification Synthesis

- Result based on the relation between
 - Refinement order
 - Compatibility
 - Mirror
- Simplified proof
 - $p \parallel q \leq p'$
 - $(p \parallel q) \parallel \text{mirror}(p')$ $\in G$
 - $p \parallel (q \parallel \text{mirror}(p')) \in G$
 - $p \parallel \text{mirror}(\text{mirror}(q \parallel \text{mirror}(p')))) \in G$
 - $p \leq \text{mirror}(q \parallel \text{mirror}(p'))$

Research Directions

- Effective computation
 - Approximate complementation
 - See also residuation
- Operational models
 - Sequential composition is key
 - Explicitly account for causality
 - Define how a model evolves at every step
 - Parameterized composition (via communication media)
- Understand choice
 - Refinement in two-set models more distinguishing than behavior containment (vending machine example)
 - What is the relation with observation equivalence?
- Model adaptors (with A. Pinto)
 - Based on operational models
 - Correctness criterion based on common semantic domain

Thank You