Protocol synthesis from requirements

Ensuring correctness via asynchronous control

Loïc Hélouët\textsuperscript{1}, Claude Jard\textsuperscript{2}, Rouwaida Abdallah\textsuperscript{2}

\textsuperscript{1}INRIA Rennes, \textsuperscript{2}ENS Cachan-Ker Lann
Motivations

Requirements =
- High-level descriptions: behavior, architecture, distribution
- global control
- abstracted Data
- ...

Implementation =
- Closer to code / code
- distributed "programs": local control only
- communications among agents
Motivations

A problem addressed since the 80’s:

Derive a correct implementation from requirements

Variations on:

- on the requirement (finite automata, logic, scenarios,...)
- on the implementation model (Petri nets, communicating automata,...)
- on the notion of correctness
Outline

- Requirement & Implementation models
- Correctness
- Synthesis by projection
- Locality & synchronization loss problem
- Asynchronous control architecture
- Correctness
basic MSC over a set of processes $\mathcal{P}$

= Partial order labeled by

$\Sigma = \{ p!q(m), p?q(m), p(act) \mid p, q \in \mathcal{P}, m \in \mathcal{M}, act \in ACT \}$

$L(M) = \{ u \in \Sigma^* \mid \exists v, u.v \text{ linearization of } M \}$

Execution of $M$ can start with $p!q(m1)$ or $r!q(m3)$

$p, r$ are called the deciding processes of $M$
Concatenation of bMSCs: \( M_1 \circ M_2 \)
Models: requirements

A **HMSC** \( H \) = an automaton with transitions labeled by bMSCs (states are called **nodes**)

\[
H \text{ defines a set of Sequences of bMSCs } Seq(H).
\]

\[
\mathcal{L}(H) = \{ w \mid \exists \rho = M_1 \circ \ldots \circ M_k \in Seq(H), w \in \mathcal{L}(\rho) \}
\]
Models: implementation

\[ CFSM \{ A_p \}_{p \in P} = \text{finite automata with FIFO queues} \]
Models: implementation

CFSM \( \{A_p\}_{p \in \mathcal{P}} \) = finite automata with FIFO queues

Execution: \( p!q(m1) \)
Models: implementation

CFSM \( \{ A_p \}_{p \in P} \) = finite automata with FIFO queues

Execution: \( p!q(m1).p!q(m2) \)
Models: implementation

CFSM \( \{ A_p \}_{p \in P} \) = finite automata with FIFO queues

Execution: \( p!q(m1).p!q(m2).q?p(m1) \)
Models: implementation

CFSM \( \{A_p\}_{p \in \mathcal{P}} \) = finite automata with FIFO queues

Execution: \[ p!q(m1).p!q(m2).q?p(m1).q?p(m2) \]

\[ \mathcal{L}(\{A_p\}_{p \in \mathcal{P}}) = \text{sequences of actions allowed by automata} \]

(prefix closed)
Correctness

Let $T(.)$ be an algorithm that:

- takes as input a HMSC $H$ over a set of processes $\mathcal{P}$
- outputs a CFSM $T(H) = \{A_p\}_{p \in \mathcal{P}}$

$T$ is correct if for every $H$, $\mathcal{L}(H) = \mathcal{L}(T(H))$
The behavior of a HMSC projected on each process is a regular language.

Projection: for every $p$ in $\mathcal{P}$,

- "erase" the behavior of processes in $\mathcal{P} \setminus \{p\}$ from $H$.
- The result is an automaton with transitions labeled by words
- transform the obtained automaton into a new automaton which transitions are labeled by a single letter.