Process Algebra for Modal Transition Systems

Nikola Beneš  Jan Křetínský

Masaryk University, Brno, Czech Republic
Technische Universität München, Germany

MEMICS
24th October 2010
Motivation – MTS

Design and verification of parallel systems

- composition
- loose specifications
- stepwise refinement
Motivation – MTS

Design and verification of parallel systems
- composition
- loose specifications
- stepwise refinement

Modal transition systems
- specification formalism
- Larsen & Thomsen, 1988
- extension of LTS
- may and must transitions
- combination of allowed and required behaviour
- bisimulation-based refinement
Motivation – MTS

Design and verification of parallel systems
- composition
- loose specifications
- stepwise refinement

Modal transition systems
- specification formalism
- Larsen & Thomsen, 1988
- extension of LTS
- may and must transitions
- combination of allowed and required behaviour
- bisimulation-based refinement
Motivation – MTS

Design and verification of parallel systems
- composition
- loose specifications
- stepwise refinement

Modal transition systems
- specification formalism
- Larsen & Thomsen, 1988
- extension of LTS
- may and must transitions
- combination of allowed and required behaviour
- bisimulation-based refinement
Motivation – MTS

Design and verification of parallel systems
- composition
- loose specifications
- stepwise refinement

Modal transition systems
- specification formalism
- Larsen & Thomsen, 1988
- extension of LTS
- may and must transitions
- combination of allowed and required behaviour
- bisimulation-based refinement
Motivation – MTS

Design and verification of parallel systems

- composition
- loose specifications
- stepwise refinement

Modal transition systems

- specification formalism
- Larsen & Thomsen, 1988
- extension of LTS
- may and must transitions
- combination of allowed and required behaviour
- bisimulation-based refinement
Motivation – Extending MTS

Modal transition systems

- **may** transitions
  - guarantee safety
- **must** transitions
  - guarantee liveness (?)
Motivation – Extending MTS

Modal transition systems

- **may** transitions
  - guarantee safety
- **must** transitions
  - guarantee liveness (?)

![Diagram of Modal Transition Systems](image-url)
Motivation – Extending MTS

Modal transition systems

- **may** transitions
  - guarantee **safety**
- **must** transitions
  - guarantee **liveness** (?)

Extensions to MTS

- Disjunctive MTS
  (several flavours)
- Mixed transition systems
Motivation – Extending MTS

Modal transition systems

- **may** transitions
  - guarantee **safety**
- **must** transitions
  - guarantee **liveness (?)**

Extensions to MTS

- Disjunctive MTS
  (several flavours)
- Mixed transition systems
Motivation – TS with Obligations

Our contribution: Transition systems with obligations

- encompasses all mentioned formalisms
- more succinct description
- equipped with process algebra
- allow us to compare the expressiveness of the MTS variants
- advantages for composition
Definition

A positive boolean formula over set $X$ of atomic propositions is given by:

$$\varphi ::= x \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid tt \mid ff$$

The set of all positive boolean formulae over $X$ is denoted as $B^+(X)$.

Definition

A transition system with obligations (OTS) over an action alphabet $\Sigma$ is a triple $(P, \rightarrow, \Omega)$, where:

- $P$ is a set of processes,
- $\rightarrow \subseteq P \times \Sigma \times P$ is the may transition relation, and
- $\Omega: P \rightarrow B^+(\Sigma \times P)$ is the set of obligations.
The Coffee Machine
Drink vending machine requirement:
“The machine may sell coffee, tea and hot chocolate. Everybody drinks coffee. Only some people drink tea. Those who don’t drink tea, drink hot chocolate. Everybody has to have something to drink.”
Drink vending machine requirement:

“The machine may sell coffee, tea and hot chocolate. Everybody drinks coffee. Only some people drink tea. Those who don’t drink tea, drink hot chocolate. Everybody has to have something to drink.”
Drink vending machine requirement:
“The machine may sell coffee, tea and hot chocolate. Everybody drinks coffee. Only some people drink tea. Those who don’t drink tea, drink hot chocolate. Everybody has to have something to drink.”

\[ \Omega(\text{idle}) = (\text{coin, select}) \]
\[ \Omega(\text{select}) = (\text{coffee, working}) \lor ((\text{tea, working}) \land (\text{hot chocolate, working})) \]
\[ \Omega(\text{working}) = (\text{output, idle}) \]
Drink vending machine requirement:

“The machine may sell coffee, tea and hot chocolate. Everybody drinks coffee. Only some people drink tea. Those who don’t drink tea, drink hot chocolate. Everybody has to have something to drink.”

\[
\Omega(\text{idle}) = (\text{coin}, \text{select}) \\
\Omega(\text{select}) = (\text{coffee}, \text{working}) \lor ((\text{tea}, \text{working}) \land (\text{hot chocolate}, \text{working})) \\
\Omega(\text{working}) = (\text{output}, \text{idle})
\]
Subclasses of OTS

Labelled transition systems (LTS, implementations)
- one behaviour only

Mixed transition systems (MixTS)
- allowed and required behaviour

Modal transition systems (MTS)
- what is required is also allowed (syntactic consistency)

Disjunctive modal transition systems (DMTS)
- must hypertransitions
- solutions to process equations

Consistent DMTS (cDMTS)
- DMTS + syntactic consistency
### Process Algebra for OTS

#### Definition (syntax)

\[
P := \text{nil} \mid \text{co-nil} \mid a.P \mid X \mid P \land P \mid P \lor P \mid \not^a P
\]

#### Definition (semantics)

\[
\begin{align*}
\Omega(\text{nil}) & = \text{tt} \\
\Omega(\text{co-nil}) & = \text{ff} \\
\Omega(a.P) & = (a, P) \\
\Omega(X) & = \Omega(P) \quad \text{for } X := P
\end{align*}
\]

\[
\begin{align*}
\Omega(P \land Q) & = \Omega(P) \land \Omega(Q) \\
\Omega(P \lor Q) & = \Omega(P) \lor \Omega(Q) \\
\Omega(\not^a P) & = \Omega(P)
\end{align*}
\]

Shortcut: \(?P = (\text{nil} \lor P)\)
\[ \Omega(\text{idle}) = (\text{coin, select}) \]
\[ \Omega(\text{select}) = (\text{coffee, working}) \lor ((\text{tea, working}) \land (\text{hot chocolate, working})) \]
\[ \Omega(\text{working}) = (\text{output, idle}) \]
**OTS Example Revisited**

\[ \Omega(\text{idle}) = (\text{coin}, \text{select}) \]
\[ \Omega(\text{select}) = (\text{coffee}, \text{working}) \lor ((\text{tea}, \text{working}) \land (\text{hot chocolate}, \text{working})) \]
\[ \Omega(\text{working}) = (\text{output}, \text{idle}) \]

\[
\begin{align*}
\text{idle} & := \text{coin}.\text{select} \\
\text{select} & := \text{coffee.}\text{working} \lor (\text{tea.}\text{working} \land \text{hot chocolate.}\text{working}) \\
\text{working} & := \text{output.}\text{idle}
\end{align*}
\]
OTS Example Revisited

\[ \Omega(\text{idle}) = (\text{coin}, \text{select}) \]
\[ \Omega(\text{select}) = (\text{coffee}, \text{working}) \lor ((\text{tea}, \text{working}) \land (\text{hot chocolate}, \text{working})) \]
\[ \Omega(\text{working}) = (\text{output}, \text{idle}) \]

\[
\begin{align*}
\text{idle} &:= \text{coin}.\text{select} \land (\text{banknote}.\text{select} \lor \text{nil}) \\
\text{select} &:= \text{coffee}.\text{working} \lor (\text{tea}.\text{working} \land \text{hot chocolate}.\text{working}) \\
\text{working} &:= \text{output}.\text{idle}
\end{align*}
\]
OTS Example Revisited

\[ \Omega(\text{idle}) = (\text{coin}, \text{select}) \]
\[ \Omega(\text{select}) = (\text{coffee}, \text{working}) \lor ((\text{tea}, \text{working}) \land (\text{hot chocolate}, \text{working})) \]
\[ \Omega(\text{working}) = (\text{output}, \text{idle}) \]

\begin{align*}
\text{idle} & := \text{coin} \cdot \text{select} \land ?\text{banknote} \cdot \text{select} \\
\text{select} & := \text{coffee} \cdot \text{working} \lor (\text{tea} \cdot \text{working} \land \text{hot chocolate} \cdot \text{working}) \\
\text{working} & := \text{output} \cdot \text{idle}
\end{align*}
OTS Example Revisited

\[ \Omega(\text{idle}) = (\text{coin}, \text{select}) \]
\[ \Omega(\text{select}) = (\text{coffee}, \text{working}) \lor ((\text{tea}, \text{working}) \land (\text{hot chocolate}, \text{working})) \]
\[ \Omega(\text{working}) = (\text{output}, \text{idle}) \]

idle := coin.select \land \text{?} banknote.select

select := coffee.working \lor (tea.working \land \not\text{hot chocolate}.working)

working := output.idle
Process Algebra w.r.t. Subclasses of OTS

DMTS \[ P ::= \text{nil} \mid a.P \mid X \mid P \land P \mid P \lor P \mid \nmid P \mid \text{co-nil} \]

cDMTS \[ P ::= \text{nil} \mid a.P \mid X \mid P \land P \mid P \lor P \]

MixTS \[ P ::= \text{nil} \mid a.P \mid X \mid P \land P \mid P \lor \text{nil} \mid \nmid P \mid \text{co-nil} \]

MTS \[ P ::= \text{nil} \mid a.P \mid X \mid P \land P \mid P \lor \text{nil} \]

LTS \[ P ::= \text{nil} \mid a.P \mid X \mid P \land P \]
Composition

Composition of OTS

- based on synchronous message passing
- synchronizing alphabet $\Gamma \subseteq \Sigma$

Definition

\[
\begin{align*}
S_1 & \xrightarrow{a} S_1' & S_2 & \xrightarrow{a} S_2' \\
S_1 \parallel S_2 & \xrightarrow{a} S_1' \parallel S_2' \\
\end{align*}
\]

\[
\begin{align*}
S_1 & \xrightarrow{a} S_1' \\
S_1 \parallel S_2 & \xrightarrow{a} S_1' \parallel S_2 \\
\end{align*}
\]

- obligations – a bit more complicated
- disjunctive normal form
- advantage over DMTS
Semantics and expressiveness

- $[S]$ – set of all implementations of $S$

$\forall C \in \mathcal{C}$ with $[C] \neq \emptyset$ there is $D \in \mathcal{D}$ such that $[C] = [D]$ 

(strict expressiveness $\prec$, equivalence $\equiv$)
Hierarchy Results – Strictness

\[
\begin{align*}
\text{LTS} & \prec \text{MTS} \\
& \quad \bullet \ ?a.\text{nil} \\
\text{MTS} & \prec \text{MixTS} \\
& \quad \bullet \ ?a.b.\text{nil} \land ?a.c.\text{nil} \land \not\exists a.(?b.\text{nil} \land ?c.\text{nil}) \\
\text{MTS} & \prec \text{cDMTS} \\
& \quad \bullet \ a.\text{nil} \lor b.\text{nil} \\
\text{MixTS} & \prec \text{DMTS} \\
& \quad \bullet \ a.\text{nil} \lor b.\text{nil}
\end{align*}
\]
Hierarchy Results – Equivalence

\[ \text{DMTS} \equiv \text{OTS} \]
- every positive boolean formula may be converted into CNF

\[ \text{cDMTS} \equiv \text{DMTS} \]
- powerset construction
Hierarchy Results – Syntactic Characterizations

Syntactic characterization of consistent OTS

- $P ::= \text{nil} \mid a.P \mid X \mid P \land P \mid P \lor P$

Syntactic characterization of consistent MixTS?

- previous construction
  - CNF formulae, all literals in one clause have same action
- idea: $P ::= \text{nil} \mid X \mid a.P \mid P \land P \mid \bigvee_{i} a.P_i$
Syntactic characterization of consistent OTS

- $P ::= \text{nil} \mid a.P \mid X \mid P \land P \mid P \lor P$

Syntactic characterization of consistent MixTS?

- previous construction
  - CNF formulae, all literals in one clause have same action
- idea: $P ::= \text{nil} \mid X \mid a.P \mid P \land P \mid \bigvee_{i} a.P_{i}$
- unfortunately, does not work
- $(a.(a.\text{nil} \land b.\text{nil}) \lor a.\text{nil}) \land ?a.a.\text{nil}$
- the question remains open
### Conclusion

**Transition systems with obligations**
- equipped with process algebra
- generalise MTS, MixTS, DMTS
- more succinct than DMTS
- optimization in composition

**Comparison of various kinds of MTSs**
- $LTS$ (implementations) $\prec MTS \prec MixTS \prec cDMTS \equiv DMTS$ (OTS)
- main result $DMTS \equiv cDMTS$

**Syntactic characterization**
- semantic consistency
- future work: MixTS