Modal Transition Systems

Nikola Beneš

ParaDiSe laboratory
Faculty of Informatics
Masaryk University, Brno

2011-01-11
D3S Seminar
Outline of the Talk

Modal Transition Systems
- basic definitions
- refinement
- composition
- some other problems

Our Contribution
- joint work with J. Křetínský, J. Srba, K. G. Larsen
- TCS’09: On Determinism in Modal Transition Systems
- ICTAC’09: Checking Thorough Refinement on Modal Transition Systems is EXPTIME-complete

Ongoing and Future Work
Introduction

Motivation

- specification formalism
- partial specifications
- stepwise refinement
- composition
Introduction

Motivation

- specification formalism
- partial specifications
- stepwise refinement
- composition

Modal Transition Systems

- invented by K. G. Larsen, B. Thomsen in 1988
- extension of labelled transition systems
- may and must transitions
- allowed/required behaviour
- several modifications: Mixed TS, Disjunctive MTS, ...
**Implementations:** All states are either red or blue. Red state may have an $a$-transition to a blue state. Blue state (may and) must have a $b$-transition to a red state.
**Implementations:** All states are either red or blue.

Red state may have an $a$-transition to a blue state.

Blue state (may and) must have a $b$-transition to a red state.
MTS: Simple Example

Implementations: All states are either red or blue. 
Red state may have an $a$-transition to a blue state. 
Blue state (may and) must have a $b$-transition to a red state.
**Implementations:** All states are either red or blue.

Red state may have an $a$-transition to a blue state.

Blue state (may and) must have a $b$-transition to a red state.
Implementations: All states are either red or blue. Red state may have an $a$-transition to a blue state. Blue state (may and) must have a $b$-transition to a red state.
Definition (MTS)

An MTS over an action alphabet $\Sigma$ is a triple $(P, \rightarrow, \longrightarrow)$ where

$$\longrightarrow \subseteq \rightarrow \subseteq P \times \Sigma \times P.$$
Basic Definitions

**Definition (MTS)**

An MTS over an action alphabet \( \Sigma \) is a triple \((P, \rightarrow, \rightarrow)\) where

\[ \rightarrow \subseteq \rightarrow \subseteq P \times \Sigma \times P. \]

**Definition (Modal Refinement)**

\( S \leq_m T \) if there is a relation \( R \) such that for every \((A, B) \in R\)

- if \( A \xrightarrow{a} A' \) then \( B \xrightarrow{a} B' \) and \((A', B') \in R\)
- if \( B \xrightarrow{a} B' \) then \( A \xrightarrow{a} A' \) and \((A', B') \in R\)
Definition (MTS)

An $MTS$ over an action alphabet $\Sigma$ is a triple $(P, \rightarrow, \Rightarrow)$ where

$\Rightarrow \subseteq \rightarrow \subseteq P \times \Sigma \times P.$

Definition (Modal Refinement)

$S \leq_m T$ if there is a relation $R$ such that for every $(A, B) \in R$

- if $A \xrightarrow{a} A'$ then $B \xrightarrow{a} B'$ and $(A', B') \in R$
- if $B \xrightarrow{a} B'$ then $A \xrightarrow{a} A'$ and $(A', B') \in R$

Bisimulation-like two-player game

- Attacker: may transitions on the left, must transition on the right.
- Defender: must transitions on the left, may transition on the right.
Implementations

- MTS with $\rightarrow = \rightarrow\rightarrow$.
- $I$ implements $S$ if $I$ is an implementation and $I \leq_m S$.
- **semantics** of $S$:
  $$\llbracket S \rrbracket = \{ I \mid I \text{ implements } S \}$$
Implementations

- MTS with $\longrightarrow = \longrightarrow$.
- $I$ implements $S$ if $I$ is an implementation and $I \leq_m S$.
- Semantics of $S$:

$$\llbracket S \rrbracket = \{ I \mid I \text{ implements } S \}$$

Definition (Thorough Refinement)

$$S \leq_t T \text{ if } \llbracket S \rrbracket \subseteq \llbracket T \rrbracket.$$ 

Relation between the refinements:

- $S \leq_m T$ implies $S \leq_t T$.
- Not vice versa!
Modal and Thorough Refinements

$S \leq_t T$, but $S \nleq_m T$
Example – full synchronization:

\[
\begin{align*}
  I \xrightarrow{a} I' & \quad J \xrightarrow{a} J' \\
  I \parallel J \xrightarrow{a} I' \parallel J' & \quad \forall a \in \Sigma
\end{align*}
\]
Composition

Example – full synchronization:
\[
\begin{align*}
I & \xrightarrow{a} I' \quad J \xrightarrow{a} J' \\
\begin{array}{c}
I \\
I' \\
\hline
\end{array}
& \begin{array}{c}
J \\
\hline
J' \\
\end{array} \quad \forall a \in \Sigma
\end{align*}
\]

Lifting to MTS:
\[
\begin{align*}
I & \xrightarrow{a} I' \quad J \xrightarrow{a} J' \\
\begin{array}{c}
I \\
I' \\
\hline
\end{array}
& \begin{array}{c}
J \\
\hline
J' \\
\end{array} \quad \forall a \in \Sigma
\end{align*}
\]

\[
\begin{align*}
I & \begin{array}{c}
\xrightarrow{a} \\
\hline
\end{array} I' \quad J \begin{array}{c}
\xrightarrow{a} \\
\hline
\end{array} J' \\
\begin{array}{c}
I \\
I' \\
\hline
\end{array}
& \begin{array}{c}
J \\
\hline
J' \\
\end{array} \quad \forall a \in \Sigma
\end{align*}
\]

\[
[S] \parallel [T] \subseteq [S \parallel T]
\]

Liftable Composition Operators

- all that satisfy certain monotonicity property
Common Implementation

- given $n$ specifications $S_1, \ldots, S_n$
- decide whether $[S_1] \cap \cdots \cap [S_n]$ is nonempty
- (maybe also synthesise a common implementation)
- more general problem: **logical conjunction**

Generalized Model Checking

- given a specification $S$ and a logic formula $\varphi$
- decide whether all or some implementations of $S$ satisfy $\varphi$
Our Contribution – Determinism

(Beneš, Křetínský, Srba, Larsen: *On Determinism in Modal Transition Systems*)

Determinism

- in specifications – high-level description
- in implementations – deterministic hardware
- deterministic approximations

Does determinism help?
Previously Known

- \( S \leq_m T \) implies \( S \leq_t T \) but not vice versa.
- Checking \( \leq_m \) is \textsc{PTIME}-complete.

Refinements in Deterministic Case

- If \( D \) is deterministic, \( S \leq_t D \) implies \( S \leq_m D \).
- Checking \( \leq_m \) is \textsc{NLOGSPACE}-complete if the right-hand side process is deterministic.

\textit{Note}: difference between \textsc{PTIME} and \textsc{NLOGSPACE}: parallelization.
Our Contribution – Deterministic Hull

**Deterministic Hull** $\mathcal{D}(S)$
- smallest deterministic MTS refined by $S$
- $S \leq_m T$ is an under-approximation of $S \leq_t T$
- $S \leq_m \mathcal{D}(T)$ is an over-approximation of $S \leq_t T$

**Application**
- choosing a fitting component from a large set

**Computation**
- extension of the subset construction
Our Contribution – Composition

Previously Known
- MTS composition is an over-approximation
  (sound, not complete)

Determinism in Specifications
- does not help (next slide)

Determinism in Implementations
- certain conditions for $\llbracket S_1 \parallel S_2 \rrbracket = \llbracket S_1 \parallel S_2 \rrbracket$
Incomplete Composition

\[ S_1 \quad S_2 \quad S_1 \parallel S_2 \]

\[ \begin{align*}
S_1 &\quad a \quad b \quad c \\
S_2 &\quad a \quad b \quad c \\
S_1 \parallel S_2 &\quad b \quad c \\
\end{align*} \]
Our Contribution – Common Implementation

Previously Known

- common implementation is \textbf{EXPTIME}-complete in general
- common implementation of fixed number of specifications is \textbf{PTIME}-complete

Deterministic Case

- \textbf{PSPACE}-complete
- \textbf{NLOGSPACE}-complete if the number of specifications is fixed
Our Contribution – Thorough Refinement

(Beneš, Křetínský, Srba, Larsen: 
*Checking Thorough Refinement on Modal Transition Systems is EXPTIME-complete*)

Previously Known

- checking $\leq_t$ is PSPACE-hard
- checking $\leq_t$ is in EXPTIME
  (argument relies on folklore knowledge)

Our Contribution

- checking $\leq_t$ is EXPTIME-hard
  - reduction from acceptance problem for alternating LBA
  - proof quite involved
- checking $\leq_t$ is in EXPTIME
  - new proof
  - tableau method
Ongoing Work

Logical Conjunction

- given $S_1, \ldots, S_n$
- find $S$ such that $[S] = [S_1] \cap \cdots \cap [S_n]$
- MTS: may not exist
- disjunctive MTS: conjunction always exists
Logical Conjunction

- given $S_1, \ldots, S_n$
- find $S$ such that $[S] = [S_1] \cap \cdots \cap [S_n]$
- MTS: may not exist
- disjunctive MTS: conjunction always exists
Logical Conjunction

- given $S_1, \ldots, S_n$
- find $S$ such that $[S] = [S_1] \cap \cdots \cap [S_n]$
- MTS: may not exist
- disjunctive MTS: conjunction always exists
Ongoing and Future Work

**Generalized Model Checking**
- previously known results on CTL
- our focus: LTL
- extending to disjunctive MTS

**Extensions to MTS**
- Disjunctive MTS, Mixed TS, ...
- compare their expressiveness
- look at all previous problems
Factorization Problem

- dual to composition
- given $A$ and $B$
- find (most general) $X$ such that $A \parallel X \approx B$
- various forms, depending on the choice of $\parallel$ and $\approx$
Conclusion

Modal Transition Systems
- based on LTS, *may* and *must* relations
- refinement (modal, thorough); composition

Our Contribution
- determinism helps (refinements coincide; lower complexities)
- thorough refinement is in general EXPTIME-complete

Ongoing and Future Work
- conjunction
- generalized LTL model checking
- expressiveness of various extensions
- factorization