Checking Thorough Refinement on Modal Transition Systems Is EXPTIME-Complete

Nikola Beneš  Jan Křetínský  Kim G. Larsen  Jiri Srba

Masaryk University, Czech Republic
Aalborg University, Denmark

MEMICS 2009
Modal Transition Systems: Motivation

A specification formalism introduced by K.G. Larsen and B. Thomsen more than 20 years ago.

Step-wise, component-based design of a software system.

- Specifications are gradually **refined** until a concrete system (*implementation*) is produced.
- If the specification satisfies certain properties, so does the implementation.
Modal Transition Systems: Example

\[ \rightarrow \text{must} \text{ transitions} – \text{required behavior} \]

\[ \longrightarrow \text{may} \text{ transitions} – \text{allowed behavior} \]

\[ S_1 \rightarrow \text{coin} \rightarrow \bullet \]

\[ \text{tea} \]

\[ \text{coffee} \]
Modal Transition Systems: Example

\[ S_1 \xrightarrow{\text{coin}} \]

\[ I_1 \xrightarrow{\text{coin}} \]

\[ \text{must transitions} \text{ – required behavior} \]

\[ \text{may transitions} \text{ – allowed behavior} \]
Modal Transition Systems: Example

→ must transitions – required behavior
--→ may transitions – allowed behavior

\[ S_1 \bullet \xrightarrow{\text{coin}} \bullet \quad S_2 \bullet \xleftarrow{\text{coin}} \bullet \]

\[ I_1 \bullet \xrightarrow{\text{coin}} \bullet \]

Jan Křetínský

Thorough Refinement Is EXPTIME-Complete
Modal Transition Systems: Example

→ must transitions – required behavior
---→ may transitions – allowed behavior

\[ S_1 \quad \text{tea} \quad \text{coffee} \]
\[ S_2 \quad \text{tea} \quad \text{coffee} \]

\[ I_1 \quad \text{coin} \quad \text{coffee} \]
\[ I_2 \quad \text{coin} \quad \text{coffee} \]
Modal Transition Systems: Example

→ must transitions – required behavior
→→ may transitions – allowed behavior

\[ S_1 \quad S_2 \quad I_1 \quad I_2 \quad I_3 \]

\[ \begin{align*}
S_1 & \xrightarrow{\text{coin}} \quad \xleftarrow{\text{coffee}} \quad \xrightarrow{\text{tea}} \\
S_2 & \xrightarrow{\text{coin}} \quad \xleftarrow{\text{coffee}} \quad \xrightarrow{\text{tea}} \\
I_1 & \xrightarrow{\text{coin}} \quad \xleftarrow{\text{coffee}} \\
I_2 & \xrightarrow{\text{coin}} \\
I_3 &
\end{align*} \]
Definition

An *MTS* is a triple \((P, \rightarrow, \rightarrow\rightarrow)\) where \(\rightarrow\rightarrow \subseteq \rightarrow\rightarrow\).

An MTS is an *implementation* if \(\rightarrow = \rightarrow\rightarrow\).
Modal and Thorough Refinements

**Definition**

An *MTS* is a triple \((P, \rightarrow, \longrightarrow)\) where \(\longrightarrow \subseteq \rightarrow\).

An MTS is an *implementation* if \(\longrightarrow = \rightarrow\).

**Definition (Modal refinement)**

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

- if \(A \rightarrow A'\) then \(B \rightarrow B'\) and \((A', B') \in R\)
- if \(B \rightarrow B'\) then \(A \rightarrow A'\) and \((A', B') \in R\)
Modal and Thorough Refinements

Definition
An MTS is a triple \((P, \rightarrow, \to)\) where \(\to \subseteq \rightarrow\).
An MTS is an implementation if \(\to = \rightarrow\).

Definition (Modal refinement)
\(S \leq_m T\) if there is a relation \(R\) such that for every \((A, B) \in R\)
- if \(A \to A'\) then \(B \to B'\) and \((A', B') \in R\)
- if \(B \rightarrow B'\) then \(A \rightarrow A'\) and \((A', B') \in R\)

Definition (Thorough refinement)
\(S \leq_t T\) if every implementation modally refining \(S\) also modally refines \(T\).
Refinement: Example

![Diagram of a refinement example](image-url)
Modal and Thorough Refinements

\[ S \leq_t T, \text{ but } S \not\leq_m T \]
### Complexity Issues of Refinement Relations

#### Modal Refinement Checking Problem
- Given two states $S$ and $T$ in a finite MTS, does $S \leq_m T$?  
- $P$-complete

#### Thorough Refinement Checking Problem
- Given two states $S$ and $T$ in a finite MTS, does $S \leq_t T$?  
- We show it is $\text{EXPTIME}$-complete  
- Lower-bound improves previously introduced PSPACE-hardness. [Antonik et al., FOSSACS’09]  
- Upper-bound is the first direct, goal-oriented algorithm running in $\text{EXPTIME}$.  
  (A sketch of a reduction to validity checking of vectorized modal $\mu$-calculus was given in [Antonik et al., FOSSACS’09])
Reduction from the EXPTIME-complete problem of acceptance for alternating LBA:

1. Encoding of LBA computation trees into implementations.

2. Construct an MTS $L$ which implements almost all encodings of computation trees of LBA (also incorrect ones).

3. Construct an MTS $R$ which implements encodings of all incorrect or non-accepting computation trees of LBA.

4. Show that LBA accepts a string $w$ iff $L \not\leq_t R$. 
Alternating Linear Bounded Automaton (LBA)

- Alternating Turing machine where the input word $w$ is given on the tape as $\overleftarrow{w}\overrightarrow{\cdot}$ such that $\overleftarrow{\cdot}$ and $\overrightarrow{\cdot}$ cannot be overwritten.
- Control states divided into existential and universal states. We assume that every universal branching has exactly two successor configurations.

Computation Tree of LBA

- A computation tree is rooted with the initial configuration.
- Every existential configuration has exactly one successor configuration according to the LBA transition function.
- Every universal configuration has exactly two successors according to the LBA transition function.
- A tree is accepting iff every leaf configuration contains the control state $q_{\text{acc}}$. 
Configurations

Configurations are written in the form:

\[ \vdash w_1 XY q w_2 \]

where the control state \( q \) is preceded by \( XY \in \{ \forall 1, \forall 2, \exists^* \} \).

- \( \forall 1 \) \ldots the previous configuration was universal and the first successor was chosen
- \( \forall 2 \) \ldots the previous configuration was universal and the second successor was chosen
- \( \exists^* \) \ldots the previous configuration was existential
For example a configuration

$$\frac{a \forall 1 q b}{\vdash}$$

of an LBA is encoded as
For example a configuration
\[ \vdash a \forall 1 q b \]
of an LBA is encoded as

Encoding of a part of a computation tree with universal branching
Specification $L$

code of initial configuration of the LBA

$\vdash \ldots \vdash \{\forall\}$

$M$

1 2

$\forall$

Jan Křetínský

Thorough Refinement Is EXPTIME-Complete
Encoding of an accepting computation tree is an implementation of \( L \).

In every implementation of \( L \) (the encoding of) a universal configuration has always both successors.
Specification $R$

**Intuition**

Implementations of $R$ should be all encodings of *incorrect* or *non-accepting* computation trees of the given LBA.

Assume a given LBA $M$ with an input string $w = w_1 \ldots w_n$.

- Configurations of $M$ are of length $n + 5$ ($\vdash, \dashv, \forall/\exists, 1/2/\ast, q$).
- **Bad sequence** is a sequence $c_1c_2c_3c_4c_5 \ldots d_1d_2d_3d_4d_5$ such that $c_1 \ldots c_5$ and $d_1 \ldots d_5$ do not form a legal computation window in $M$. 

$n$ symbols
Specification $R$

Intuition
Implementations of $R$ should be all encodings of incorrect or non-accepting computation trees of the given LBA.

Goal
Let $I \leq_m L$.
- If $I$ contains a path with a bad sequence, or
- if $I$ has a branch that does not contain the state $q_{acc}$,
then $I \leq_m R$.  

For every illegal window $c_1 c_2 c_3 c_4 c_5 \ldots d_1 d_2 d_3 d_4 d_5$
For every illegal window \( c_1 c_2 c_3 c_4 c_5 \ldots d_1 d_2 d_3 d_4 d_5 \)
For every illegal window \( c_1 c_2 c_3 c_4 c_5 \ldots d_1 d_2 d_3 d_4 d_5 \).
For every illegal window $c_1 c_2 c_3 c_4 c_5 \ldots d_1 d_2 d_3 d_4 d_5$
Specification $R$ — Preliminary Version

For every illegal window $c_1 c_2 c_3 c_4 c_5 \ldots d_1 d_2 d_3 d_4 d_5$.

Exponentially many paths!
Encoding of LBA Configurations

For example a configuration

\[ \Gamma \vdash a \forall 1 q b \]

of an LBA is encoded as
Specification $R$ — Bad Paths

\[
\begin{align*}
\text{All} & \setminus \{q_{\text{acc}}\} \\
R & \xrightarrow{\pi} \bullet \xrightarrow{\sigma} \bullet \xrightarrow{\pi} \bullet \xrightarrow{\sigma} \cdots \xrightarrow{\pi} \sigma \xrightarrow{\pi} \sigma \xrightarrow{\pi} \cdots \xrightarrow{\pi} \sigma \xrightarrow{\pi} d_1 \xrightarrow{\sigma} \cdots \xrightarrow{\sigma} d_5 \\
\end{align*}
\]
Specification $R$ — Universal Choice

$\forall \sigma \cdot \pi_1 \ldots$
code of initial configuration of the LBA

for all \( a \in \text{All} \setminus \{\forall\} \)
Summary:

- Given an alternating LBA $M$ with input $w$ we constructed MTSs $L$ and $R$, both of polynomial size in $M$ and $w$.
- If $M$ does not accept $w$ then $I \leq_m L$ implies that $I \leq_m R$, hence $L \leq_t R$.
- If $M$ accepts $w$ then let $I$ be the encoding of an accepting computation tree. Clearly $I \leq_m L$ but $I \nleq_m R$, hence $L \nleq_t R$. 
EXPTIME-Hardness

Summary:

- Given an alternating LBA $M$ with input $w$ we constructed MTSs $L$ and $R$, both of polynomial size in $M$ and $w$.
- If $M$ does not accept $w$ then $I \leq_m L$ implies that $I \leq_m R$, hence $L \leq_t R$.
- If $M$ accepts $w$ then let $I$ be the encoding of an accepting computation tree. Clearly $I \leq_m L$ but $I \not\leq_m R$, hence $L \not\leq_t R$.

Theorem

Checking thorough refinement on finite MTSs is EXPTIME-hard.

Theorem

Checking thorough refinement on finite MTSs is EXPTIME-hard even if the left-hand side process is fixed.
Containment in EXPTIME — Tableau Method

\[ A \not\leq_t B \quad \text{iff} \quad \exists I : I \leq_m A \text{ but } I \not\leq_m B \]

We call such a pair \((A, B)\) consistent.

Show consistency of a goal \((A, B)\)

by showing consistency for a number of subgoals.
Containment in EXPTIME — Tableau Method

\[ A \nsubseteq_t B \quad \text{iff} \quad \exists I : I \leq_m A \text{ but } I \nsubseteq_m B \]

We call such a pair \((A, B)\) consistent.

Show consistency of a goal \((A, B)\) by showing consistency for a number of subgoals.

- If \( B \xrightarrow{a} B' \) for some \( B' \) such that
  - for all \( A \xrightarrow{a} A_i \), the pairs \((A_1, B'), \ldots, (A_k, B')\) are consistent
  - then \((A, B)\) is consistent too.

- If \( A \xrightarrow{a} A' \) for some \( A' \) such that
  - for all \( B \xrightarrow{a} B_i \), the tuple \((A', B_1, \ldots, B_k)\) is consistent
  - then \((A, B)\) is consistent too.
Containment in EXPTIME — Tableau Method

\[ A \not\leq_t B \iff \exists I : I \leq_m A \text{ but } I \not\leq_m B \]

We call such a pair \((A, \overline{B})\) consistent.

**Show consistency of a goal \((A, \overline{B})\)**
by showing consistency for a number of subgoals.

- If \(B \xrightarrow{a} B'\) for some \(B'\) such that
  - for all \(A \xrightarrow{a} A_i\), the pairs \((A_1, \overline{B'}), \ldots, (A_k, \overline{B'})\) are consistent
  - then \((A, \overline{B})\) is consistent too.

- If \(A \xrightarrow{a} A'\) for some \(A'\) such that
  - for all \(B \xrightarrow{a} B_i\), the tuple \((A', \overline{B_1}, \ldots, \overline{B_k})\) is consistent
  - then \((A, \overline{B})\) is consistent too.

Problem: for this rule we need **tuples**, not only pairs!
By generalizing the tableau rules to tuples of the form \((A, B_1, \ldots, B_k)\) we can prove the following theorem.

**Theorem**

Thorough refinement checking on finite MTS is decidable in EXPTIME.

Proof: least fixed-point computation of all consistent tuples (there are exponentially many of them).

**Corollary**

If the right-hand side MTS is deterministic or of a fixed size, the problem is decidable in P.
Conclusion

- We proved that thorough refinement checking is EXPTIME-complete.

- This was the last open problem in the area (common implementation problem on MTS, consistency checking and thorough refinement on mixed transition systems were already known to be EXPTIME-complete).

- Possible solutions to deal with the high complexity:
  - Use the modal refinement relation instead of thorough refinement (over-approximation).
  - Consider only deterministic specifications (right-hand side processes).