Applications for variants of modal transition systems

Harald Fecher

Imperial College London, United Kingdom

Copenhagen, 20 September 2007

Main Issue

Discussion on

When (variants of) modal transition systems are appropriate.

When they are less appropriate.

Application of abstract models

Abstract models have applications in

- Top down development
  Allowing to describe a system at different level of abstraction within the same formalism.
- Property specification
  Describing property as abstract model and make refinement check.
- Verification
  Tackle the state explosion by making an abstraction (e.g., via predicate abstraction) and check the abstraction instead.
  CEGAR: Counterexample-guided abstraction refinement

Application of abstract models

Abstract models have applications in

- Top down development
  Allowing to describe a system at different level of abstraction within the same formalism.
- Property specification
  Describing property as abstract model and make refinement check.
- Verification
  Tackle the state explosion by making an abstraction (e.g., via predicate abstraction) and check the abstraction instead.
  CEGAR: Counterexample-guided abstraction refinement

Question:
Which abstract model should be chosen in which circumstance?

Non-application of MTS

Are modal transition systems and their variants suitable for classical settings?

Non-application of MTS

Are modal transition systems and their variants suitable for classical settings?

In my opinion they are not.

Non-application of MTS

Are modal transition systems and their variants suitable for classical settings?

In my opinion they are not.

Reason:
Executions of programs usually behave deterministic up to the environment.
But MTS describes sets of general (labeled) transition systems and not only deterministic transition systems (transitions leaving the same state must have different labels).

Non-application illustration

Suppose executions behave deterministic up to the environment (i.e. concrete models are deterministic transition systems):

Modal transition system specification:

May transitions become redundant as soon as leaving must transitions exists: always the must transition can be used to match the unique concrete step.

Appropriate abstract models

Abstract models suitable for deterministic systems:

- Transition systems with trace inclusion for non-reactive systems
- Labeled transition systems with ready simulation, ready, ready trace, failure, failure trace for reactive systems
- Synchronously-communicating transition systems [Fecher Grabe FSEN'07] which are labeled transition system with predicate if a label is allowed to be removed during refinement (must transition counterpart).

Modal transition system irrelevant?

Are modal transition systems and their variants irrelevant?

No they are **relevant**.

Modal transition system irrelevant?

Are modal transition systems and their variants irrelevant?

No they are **relevant**.

Applications are:

- compact specifications
- abstraction for LTL
- when *real* branching time occurs: **persistent nondeterminism** (existing in implementation)

MTS for more compact specification

MTS allows more compact specifications, especially for conjunctive behavior. For example:
After a communication via two a (first must be possible), either action a_i or action b_i for $i \leq n$ are possible.

MTS for more compact specification

MTS allows more compact specifications, especially for conjunctive behavior. For example:
After a communication via two a (first must be possible), either action a_i or action b_i for $i \leq n$ are possible.

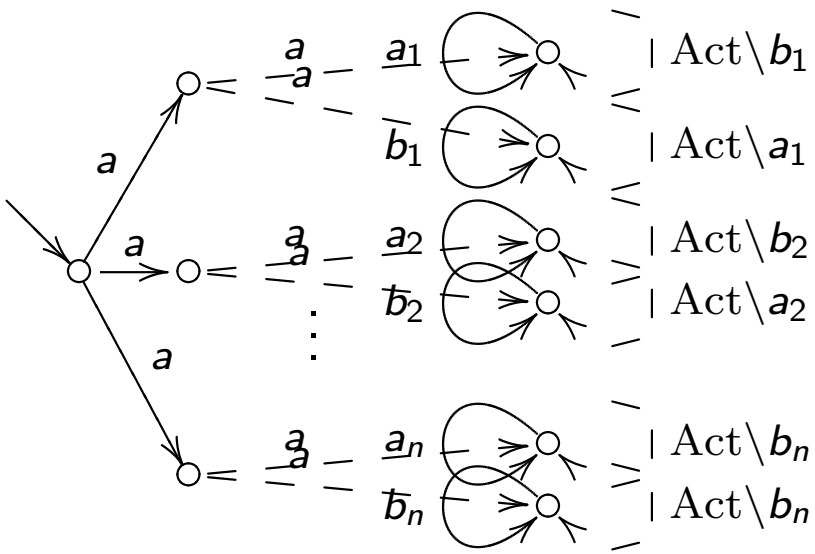
Via ready simulation: Any combination how to choose a_i and b_i necessary as aa-successor $\Rightarrow 2 + 2^n$ states

MTS for more compact specification

MTS allows more compact specifications, especially for conjunctive behavior. For example:
After a communication via two a (first must be possible), either action a_i or action b_i for $i \leq n$ are possible.

Via ready simulation: Any combination how to choose a_i and b_i necessary as aa-successor $\Rightarrow 2 + 2^n$ states

Via MTS:

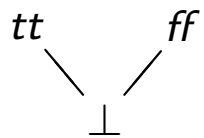


$1 + n + 2n$ states

Linear time

Linear time logics (LTL) describe property of a program (trace).

- Program execution behaves deterministic $\longrightarrow$ LTL is 2-valued.
- Unknown scheduler of parallel composition or underspecified models $\longrightarrow$ **resolvable nondeterminism**
  (transition systems with trace inclusion as refinement)
- LTL interpreted over transition systems: 3-valued



tt iff holds for all traces; ff iff neg. holds for all traces; $\bot$ otherwise

MTS for LTL model checking

If the transition system T (which is already an abstract system) is too big for checking LTL formulas, then it is often abstracted.

If the abstract system are again transition systems, then soundness is preserved: when the property is shown then it also holds for T.

MTS for LTL model checking

If the transition system T (which is already an abstract system) is too big for checking LTL formulas, then it is often abstracted.

If the abstract system are again transition systems, then soundness is preserved: when the property is shown then it also holds for T.

But when satisfaction and falsification of the abstraction fails:

- is the reason that our abstraction is too coarse and has to be refined?
- is the reason that T already yields the undefined value (has refinements that satisfy and those that falsify the property)?

Problem: end up with a chain of refinement which can never succeed.

MTS for LTL model checking

If the transition system T (which is already an abstract system) is too big for checking LTL formulas, then it is often abstracted.

If the abstract system are again transition systems, then soundness is preserved: when the property is shown then it also holds for T.

But when satisfaction and falsification of the abstraction fails:

- is the reason that our abstraction is too coarse and has to be refined?
- is the reason that T already yields the undefined value (has refinements that satisfy and those that falsify the property)?

Problem: end up with a chain of refinement which can never succeed.

If MTS are used as abstract model then this difference can be detected.

MTS for parameterized system verification

If property must hold for all parameters (e.g. mutual exclusion for n processes):

Make an initial step setting the parameter via persistent nondeterminism. Property is extended by putting a box modality in front.

Advantage: abstraction techniques can be used.

MTS for parameterized system verification

If property must hold for all parameters (e.g. mutual exclusion for n processes):

Make an initial step setting the parameter via persistent nondeterminism. Property is extended by putting a box modality in front.

Advantage: abstraction techniques can be used.

But this is oversized, since real branching only occurs as first step. Use more optimized technique for this special case.

MTS for Biological or physical systems

In biological or physical systems behavior occurs that cannot be completely determined (so far).

Persistent nondeterminism is a possibility to handle it formally.

Sometimes extended with (set of) distributions in order to obtain finer approximation of the real system.

MTS for independent environment handling

Modeling of systems where further environments invisible to the considered environment exists.
For example, computer programs dependent on a clock.

The communication with the invisible environment can be abstracted away by using persistent nondeterminism.

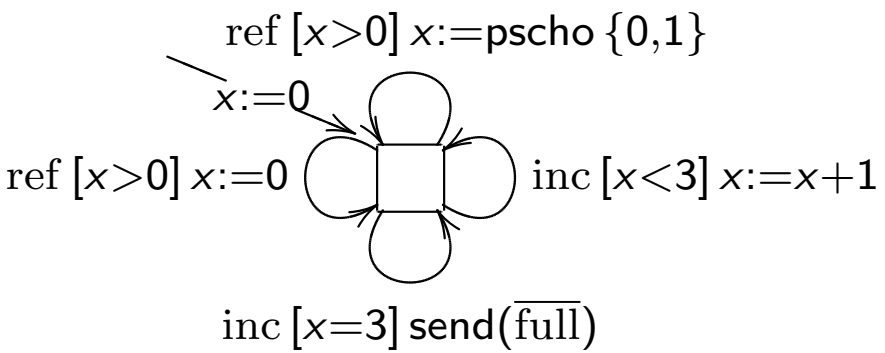
MTS for faulty system modeling

Persistent nondeterminism occurs when faulty systems, like channels or memory are modeled;
Combining with a program (makes sense: the verification property won't have to specify any fault scenarios; decreased formula complexity; model abstraction applicable)

MTS for faulty system modeling

Persistent nondeterminism occurs when faulty systems, like channels or memory are modeled;
Combining with a program (makes sense: the verification property won't have to specify any fault scenarios; decreased formula complexity; model abstraction applicable)

Example:



MTS for specification with abstract actions

When actions are grouped into an abstract one, then persistent nondeterminism guarantees that in a refinement both alternatives have to be present by using different concrete actions.

MTS for specification with abstract actions

When actions are grouped into an abstract one, then persistent nondeterminism guarantees that in a refinement both alternatives have to be present by using different concrete actions.

For example:

$$request.sendPhoto +_{per} request.sendPhotoLo$$

is refined by

$$requestHi.sendPhoto + requestLo.sendPhotoLo$$

where *request* abstracts *requestHi* and *requestLo*

MTS for specification abstracting time

When time is abstracted, then persistent nondeterminism guarantees that in a refinement both alternatives have to be present by using different time points.

MTS for specification abstracting time

When time is abstracted, then persistent nondeterminism guarantees that in a refinement both alternatives have to be present by using different time points.

For example:

$$second.0 +_{per} second.1 +_{per} ... +_{per} second.59$$

is refined by

$$second.0 + \tau.second.1 + \tau.\tau.second.2... + \tau...\tau.second.59$$

Take the chance to list some of my works on MTS variants.

Some of my work on MTS

- [Fecher Huth ATVA'06] hypermixed Kripke structure: conjunctive may-hypertransition (used for improved abstraction wrt ranking functions)
- [Fecher Huth VMCAI'07] model independent definition of abstraction-precision via games; abstraction into μ-automata
- [Fecher H.Schmidt SOS'07] process algebra (having persistent choice operator) semantics in terms of μ-automata
- [Fecher Shoham SPIN'07] Lazy-abstraction for the μ-calculus based on generalized Kripke modal transition systems
- [Fecher Huth H.Schmidt Schönborn AVoCS'07] ν-automata: used as semantic model for state machines having persistent choice operator

Models for abstracting transition systems.

Models for abstracting transition systems

- Modal transition systems [Larsen Thomsen LICS'88]
- Disjunctive modal transition systems [Larsen Xinxin LICS'90]
- μ-automata [Janin Walukiewicz *Mathematical Foundations of Computer Science*'95]
- Mixed transition systems [Dams Gerth Grumberg *Program. Lang. Syst. 19* (1997)]
- Partial Kripke structures [Bruns Godefroid CAV'99]
- Kripke modal transition systems [Huth Jagadeesan D.Schmidt ESOP'01]

MTS variants (2)

- Generalized Kripke modal transition systems [Shoham Grumberg TACAS'04]
- Focused transition systems [Dams Namjoshi LICS'04]
- Modal automata [Dams Namjoshi VMCAI'05]
- Ternary modal transition systems [Ball Kupferman Yorsh CAV'05]
- Hyper Kripke modal transition systems [Shoham Grumberg LICS'06]
- Hypermixed Kripke structures [Fecher Huth ATVA'06]
- ν-automata [Fecher Huth H.Schmidt Schönborn AVoCS'07]

Conclusion

- MTS are less appropriate when the concrete system behaves deterministic
- MTS are suitable for
  - compact specifications
  - (parameterized system verification)
  - abstraction in LTL model checking
  - biological or physical systems modeling
  - independent environments abstraction
  - faulty systems modeling
  - abstract actions specifications
  - time abstraction

Conclusion

- MTS are less appropriate when the concrete system behaves deterministic
- MTS are suitable for
  - compact specifications
  - (parameterized system verification)
  - abstraction in LTL model checking
  - biological or physical systems modeling
  - independent environments abstraction
  - faulty systems modeling
  - abstract actions specifications
  - time abstraction

Did I miss an application?

Did I miss an abstract model?