IA159 Formal Verification Methods

Model Checking: An Overview

Jan Strejček

Department of Computer Science
Faculty of Informatics
Masaryk University

Focus and sources

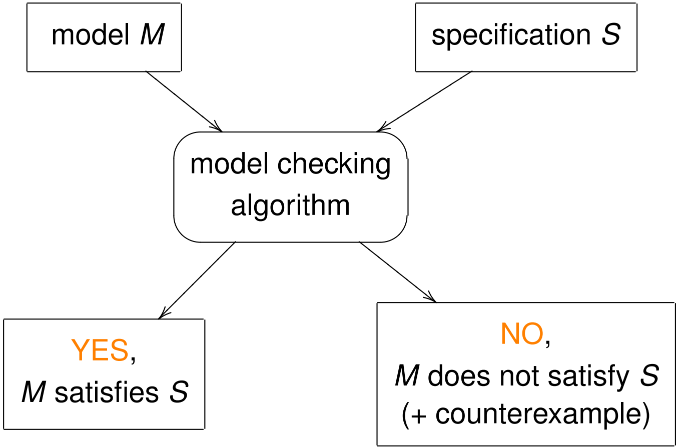
Focus

- model checking in general
- specifications, linear temporal logic (LTL), Büchi automata
- models, Kripke structure, process rewrite systems (PRS)
- model checking problems and decidability
- LTL model checking of finite systems
- state explosion problem

Sources

- Chapters 1, 2, 3 and 9 of *E. M. Clarke, O. Grumberg, and D. A. Peled: Model Checking, MIT, 1999.*
- *R. Mayr: Decidability and Complexity of Model Checking Problems for Infinite-State Systems. PhD thesis, 1998.*

Model checking schema



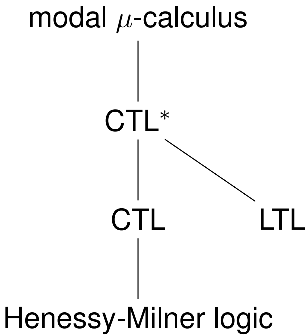
Model checking

Specification

Specification

- a finite formal description of some property that should be satisfied by all behaviours of the system
- usually does not fully specify the system
- typically given by a formula of some temporal logic
  - Linear Temporal Logic (LTL) (linear time)
  - Computational Tree Logic (CTL) (branching time)
  - CTL*, Hennessy–Milner logic, μ calculus, ...
- can be given also by a Büchi automaton, etc.

The hierarchy of basic temporal logics.



The hierarchy of selected temporal logics according to their expressive power.

State-based vs. action-based logics

state-based These logics talk about properties of states of a system. Properties of a single state are reflected by validity of atomic propositions in the state. State-based logic are interpreted over behaviours of the system represented by sequences (or trees) of sets of valid atomic propositions.

action-based Every transition of a system is labelled with an action. Action-based logic are interpreted over behaviours of the system represented only by sequences (or trees) of actions.

We provide definition of both state-based and action-based LTL.

Syntax of state-based LTL

State-based Linear Temporal Logic (LTL) is defined by

$$\varphi \; ::= \; \top \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{X}\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

where $\top$ stands for true and a ranges over a countable set AP of atomic propositions.

Abbreviations $\quad \bot \equiv \neg\top \quad \mathsf{F}\varphi \equiv \top\,\mathsf{U}\,\varphi \quad \mathsf{G}\varphi \equiv \neg\mathsf{F}\neg\varphi$

Terminology and intuitive meaning

$\mathsf{X}a$	next	$\bullet\; a\; \bullet\; \bullet\; \bullet\; \ldots$
$a\,\mathsf{U}\,b$	until	$a\; a\; \ldots\; a\; b\; \bullet\; \bullet\; \bullet\; \ldots$
$\mathsf{F}a$	eventually	$\bullet\; \bullet\; \ldots\; \bullet\; a\; \bullet\; \bullet\; \bullet\; \ldots$
$\mathsf{G}a$	always	$a\; a\; a\; a\; \ldots$

Semantics of state-based LTL

Let $\Sigma = 2^{AP'}$, where $AP' \subseteq AP$ is a finite subset. We interpret LTL on infinite words $w = w(0)w(1)\ldots \in \Sigma^\omega$. By w_i we denote the suffix of w of the form $w(i)w(i+1)w(i+2)\ldots$.
The validity of an LTL formula φ for $w \in \Sigma^\omega$, written $w \models \varphi$, is defined as

$$
\begin{array}{lll}
w \models \top & & \\
w \models a & \text{iff} & a \in w(0) \\
w \models \neg\varphi & \text{iff} & w \not\models \varphi \\
w \models \varphi_1 \wedge \varphi_2 & \text{iff} & w \models \varphi_1 \wedge w \models \varphi_2 \\
w \models \mathrm{X}\varphi & \text{iff} & w_1 \models \varphi \\
w \models \varphi_1 \,\mathrm{U}\, \varphi_2 & \text{iff} & \exists i \in \mathbb{N}_0 : w_i \models \varphi_2 \wedge \forall\, 0 \le j < i : w_j \models \varphi_1
\end{array}
$$

Given an alphabet Σ, an LTL formula φ defines the language

$$L^\Sigma(\varphi) = \{w \in \Sigma^\omega \mid w \models \varphi\}.$$

Action-based LTL

Differences between action-based and state-based LTL

- In the syntax, a ranges over countable set of actions Act.
- Formulae of action-based LTL are then interpreted over infinite sequences w of actions from a finite subset $Act' \subseteq Act$.
- Semantics of formula a is defined as follows:

$$w \models a \qquad \text{iff} \quad a = w(0)$$

Examples of LTL formulae

- $G\neg \mathit{error}$ - safety property
- $G(p \implies Fq)$ - response property
- GFp - liveness property

Büchi automata

A Büchi automaton (BA) is a tuple $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$, where

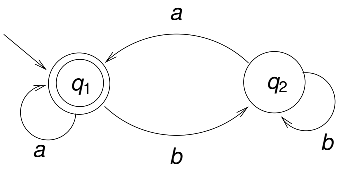
- Σ is a finite alphabet,
- Q is a finite set of states,
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function,
- $q_0 \in Q$ is an initial states,
- $F \subseteq Q$ is a set of accepting states.

A run of $\mathcal{A}$ on inifnite word $w = w(0)w(1)... \in \Sigma^\omega$ is an infinite sequence of states $\sigma = \sigma(0)\sigma(1)...$, where $\sigma(0) = q_0$ and $\sigma(i+1) \in \delta(\sigma(i), w(i))$ holds for all i.

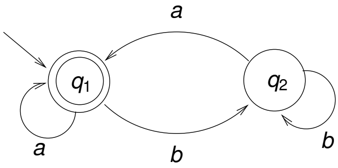
A run σ is accepting if $Inf(\sigma) \cap F \neq \emptyset$, where $Inf(\sigma)$ is the set of the states appearing in σ infinitely often. An automaton $\mathcal{A}$ accepts a word w if there is an accepting run of $\mathcal{A}$ on w. We set

$$L(\mathcal{A}) = \{w \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } w\}.$$

Example of a Büchi automaton



Example of a Büchi automaton



Accepts the words with infinitely many occurences of *a*.

Model checking

Model

Model

- a finite formal description of all possible behaviours of the system to be verified
- behaviour is a sequence (or a tree) of states/actions
- state is an image of the system in a certain moment (current values of variables, program counter, etc.)
- a state is characterized by validity of atomic propositions (e.g. $PC == start$, $x > 5$)
- many possible formalisms
  - standard languages C, Java, VHDL, ...
  - dedicated languages, e.g. ProMeLa (Process or Protocol Meta Language)
  - process algebras (infinite-state systems) BPA, BPP, PA, pushdown processes, Petri nets, ...
  - low-level formalisms: Kripke structure (for state-based approach) and labelled transition systems (for action-based approach)

Example: mutual exclusion in ProMeLa

```
byte cnt = 0;  // number of processes in critical sections
byte turn = 0; // token for entering a critical section

init {
   run(P0); run(P1);  // parallel execution of P0 a P1
}
```

```
proctype P0()
{
  // s0
  do
  // NC0 (noncritical section)
  :: do
     :: (turn == 0) -> break;
     :: else;
     od;
     // CS0 (critical section)
     cnt = cnt + 1;
     cnt = cnt - 1;
     turn = 1;
  od;
}
```

```
proctype P1()
{
  //s1
  do
  // NC1 (noncritical section)
  :: do
     :: (turn == 1) -> break;
     :: else;
     od;
     // CS1 (critical section)
     cnt = cnt + 1;
     cnt = cnt - 1;
     turn = 0;
  od;
}
```

Kripke structure

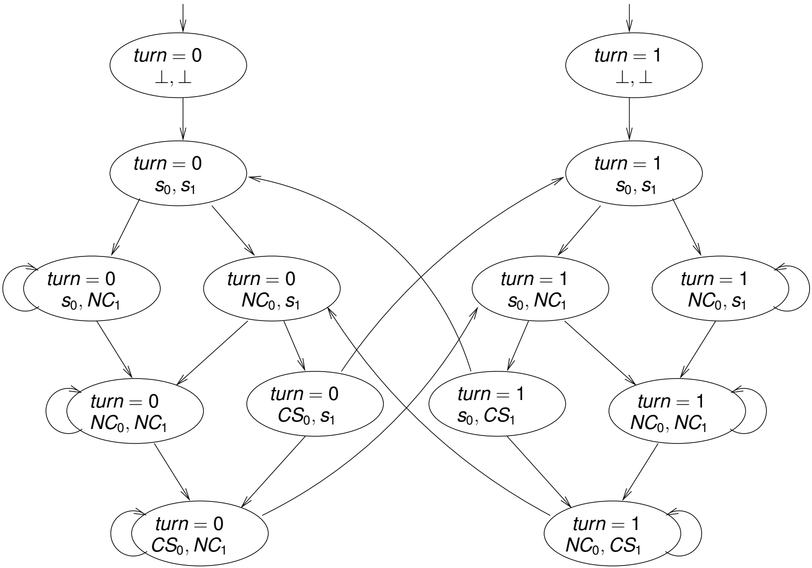
Let AP be a countable set of atomic propositions.

A Kripke structure is a tuple $M = (S, R, S_0, L)$, where

- S is a set of states
- $R \subseteq S \times S$ is transitions relation
- $S_0 \subseteq S$ is a set of initial states
- $L : S \rightarrow 2^{AP}$ is a labelling function associating to each state $s \in S$ the set of atomic propositions that are true in s.

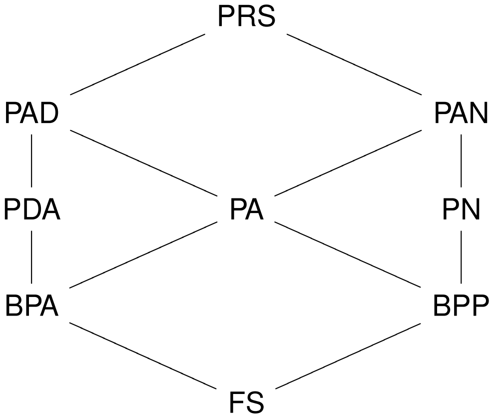
A path in M starting in a state s is an infinite sequence $\pi = s_0 s_1 s_2 \ldots$ of states such that $s_0 = s$ and $(s_i, s_{i+1}) \in R$ holds for every i.

Example: mutual exclusion as a Kripke structure



Process rewrite systems hierarchy (PRS-hierarchy)

The hierarchy compares expressive power of many classes of infinite-state systems including BPA, BPP, PA, Petri nets (PN), and pushdown processes (PDA). FS stands for finite systems.



Decidability of model checking

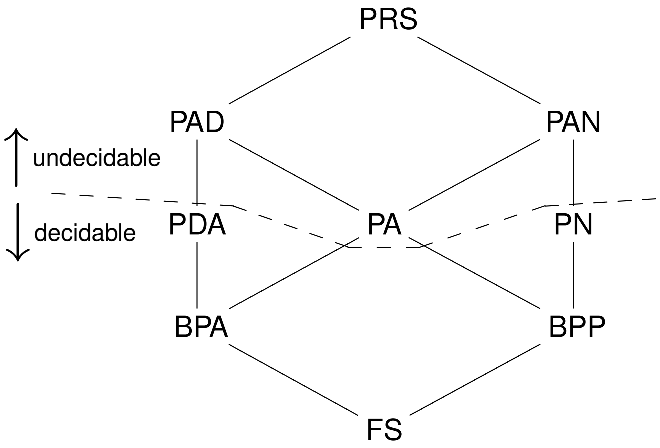
Model checking

Model checking problem is to decide whether all behaviours of a given system satisfy a given specification.

- specific problems for specific input
  - state-based LTL model checking of finite systems
  - action-based CTL model checking of finite systems
  - state-based LTL model checking of pushdown processes
  - action-based LTL model checking of pushdown processes
  - ...
- model checking problem is not decidable for some kinds of input (e.g. action-based LTL model checking of PA processes)
- all model checking problems are decidable for finite systems

The decidability boundary

The decidability boundary of the action-based LTL model checking in the PRS-hierarchy.



Model checking

Automata-based LTL model checking of finite systems

Automata-based LTL model checking of finite systems

Kripke structure M
with finitely many states

↓

Büchi automaton $\mathcal{A}_M$
accepts paths of M starting in initial states and projected by L to $2^{AP(\varphi)}$

LTL formula φ

↓

Büchi automaton $\mathcal{A}_{\neg\varphi}$
words over $2^{AP(\varphi)}$ violating φ

↘ ↙

product Büchi automaton $\mathcal{B}$
$L(\mathcal{B}) = L(\mathcal{A}_M) \cap L(\mathcal{A}_{\neg\varphi})$

↓

$L(\mathcal{B}) \stackrel{?}{=} \emptyset$
e.g. nested DFS algorithm

↙ ↘

YES

NO
+ counterexample

Complexity notes

Complexity

Time and space complexity of the LTL model checking algorithm is $\mathcal{O}(|M| \cdot 2^{\mathcal{O}(|\varphi|)})$, where $|M|$ is the number of states and transitions in the Kripke structure M.

- LTL model checking problem is PSPACE-complete.
- state explosion problem - $|M|$ is often exponential in the size of implicit description of the system due to
  - parallelism
  - large data domains
  - dynamically allocated memory
  - ...

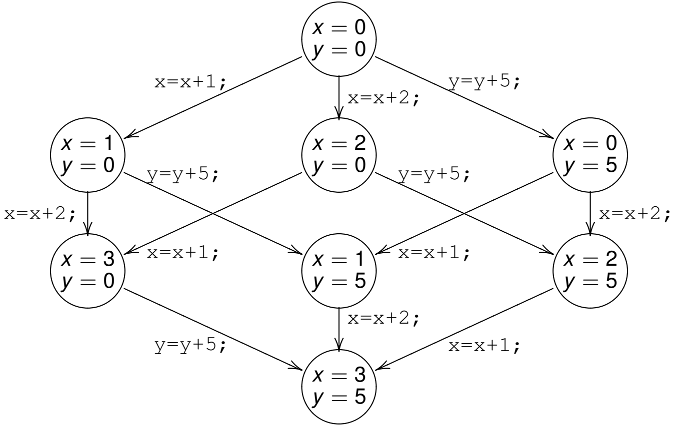
State explosion problem - an example

```
byte x = 0;
byte y = 0;

proctype A() {
   x = x + 1;
}

proctype B() {
   x = x + 2;
}

proctype C() {
   y = y + 5;
}
```



Partial solutions of the state explosion problem

- abstraction
- partial order reduction
- symmetry reduction
- on-the-fly algorithms
- symbolic model checking
- distributed algorithms
- ...

Our topics

- translation LTL→BA (via alternating 1-weak BA)
- partial order reduction
- state-based LTL model checking of pushdown processes
- abstraction
- counterexample guided abstraction refinement (CEGAR)

Coming next week

LTL→BA via alternating 1-weak BA

- What is an alternating 1-weak Büchi automaton?
- Can we see it?