Verification of infinite state systems

Angelo Montanari and Gabriele Puppis

Department of Mathematics and Computer Science
University of Udine, Italy
{montana,puppis}@dimi.uniud.it

Outline of the course (and tentative schedule)

Day 1	Introduction
Day 1-2	Basics results and techniques for MSO
Day 3	Context-free and prefix-recognizable graphs
Day 3-4	Contraction method
Day 4	Rational and automatic graphs
Day 5	Reachability over pushdown systems and Petri nets

Aim of the course

To present selected techniques and results about **automatic verification of properties of systems**.

Aim of the course

To present selected techniques and results about **automatic verification of properties of systems**.

$\Rightarrow$ two aspects need to be taken into account:

- what kind of systems?

  transitions systems (e.g., programs, devices, protocols, ...) that have *infinitely many possible configurations/states*.

- what kind of properties?

  reachability properties (e.g., 'does the system never reach a dangerous configuration from a given initial one?') and, more generally, properties expressed by *logical formulas* (e.g., monadic second-order logic formulas).

Definition (Transition system)

A **transition system** is described by

- a set S of possible *configurations*
- a set $\delta \subseteq S \times S$ of *transitions* (non-determinism is allowed).

Definition (Transition system)

A **transition system** is described by

- a set S of possible *configurations*
- a set $\delta \subseteq S \times S$ of *transitions* (non-determinism is allowed).

Note: transition systems can perform actions
and react to stimuli from external environments.

How to model system actions and external events?

We assign to each transition a **label** from a finite alphabet A
$\Rightarrow$ we replace δ with a tuple $(\delta_a)_{a \in A}$ of transition relations.

Definition (Transition system)

A **transition system** is described by

- a set S of possible *configurations*
- a set $\delta \subseteq S \times S$ of *transitions* (non-determinism is allowed).

Note: transition systems can perform actions
and react to stimuli from external environments.

How to model system actions and external events?

We assign to each transition a **label** from a finite alphabet A
$\Rightarrow$ we replace δ with a tuple $(\delta_a)_{a \in A}$ of transition relations.

How to distinguish good configurations from dangerous ones?

We assign to each configuration a **color** from a finite alphabet C
$\Rightarrow$ the system is expanded with a partition $(P_c)_{c \in C}$ of S.

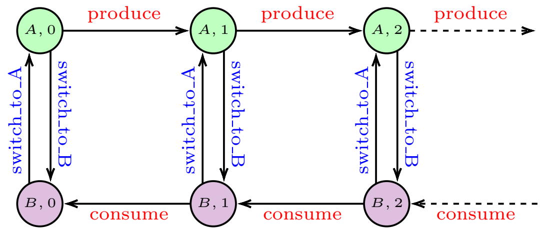
Abstract format of transition systems

Any (labeled/colored) transition system can be viewed as a (decorated) directed graph (**transition graph**), where

- vertices represent system configurations (colors can be associated with vertices),
- edges represent system transitions (labels can be associated with edges).

A : **repeat forever**
  atomic
    produce
    count := count+1

B : **repeat forever**
  atomic
    if count>0 **then**
      consume
      count := count–1



In order to effectively manipulate and reason about *infinite state systems*, we need to provide them with finite presentations.

We distinguish between two kinds of presentation:

- **internal presentations**:
  configurations and transitions are explicitly given by means of (different variants of) rewriting systems
- **external presentations**:
  transition systems are described (up to isomorphism) as the graphs resulting from applications of suitable transformations, starting from well-known structures.

Note: there exist alternative (internal and external) presentations for several classes of transition systems.

Reachability properties are evaluated over a transition system $\mathcal{T} = (S, \delta)$ and two sets $I, F \subseteq S$:

Overview of reachability problems

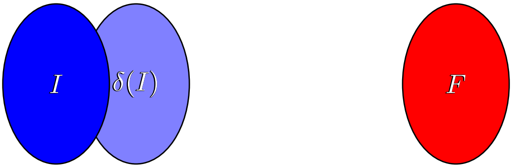
Different kinds of reachability properties can be evaluated on a transition system, for instance

- **plain reachability:**
  does $\mathcal{T}$ contain a path from I to F?
- **recurrent reachability:**
  does $\mathcal{T}$ contain a path from I that meets F infinitely often?
- **universal reachability:**
  does every path in $\mathcal{T}$ that starts in I eventually meet F?
- **...**

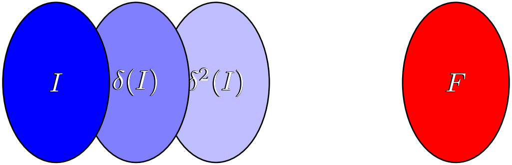
We shall focus on the plain reachability problem (experience shows that this is a crucial problem in automatic verification).

Reachability problems are usually solved via **forward analysis**...

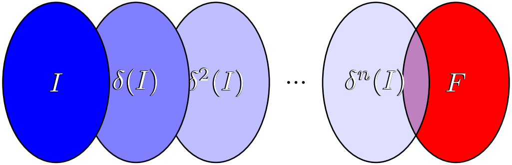
Reachability problems are usually solved via **forward analysis**...



Reachability problems are usually solved via **forward analysis**...



Reachability problems are usually solved via **forward analysis**...

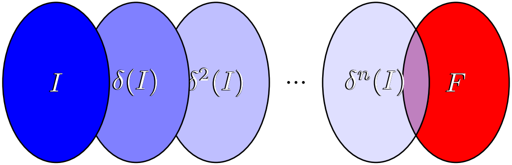


$$\exists \text{ path } \pi \text{ in } \mathcal{T}.\ I \xrightarrow{\pi} F$$

$$\Updownarrow$$

$$\delta^*(I) \cap F \neq \emptyset$$

Reachability problems are usually solved via **forward analysis**...



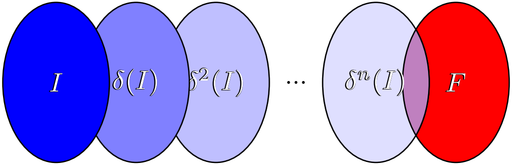
$$\exists \text{ path } \pi \text{ in } \mathcal{T}.\ I \xrightarrow{\pi} F$$

$$\Updownarrow$$

$$\delta^*(I) \cap F \neq \emptyset$$

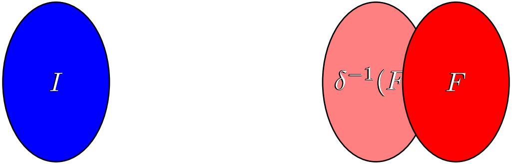
...or via **backward analysis**...

Reachability problems are usually solved via **forward analysis**...

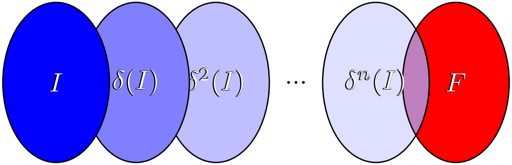


$$\exists \text{ path } \pi \text{ in } \mathcal{T}.\ I \xrightarrow{\pi} F$$
$$\Updownarrow$$
$$\delta^*(I) \cap F \neq \emptyset$$

...or via **backward analysis**...



Reachability problems are usually solved via **forward analysis**...

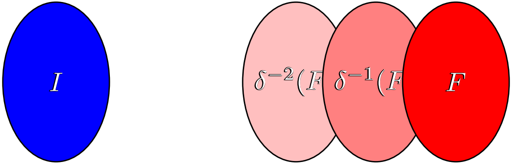


$\exists$ path π in $\mathcal{T}$. $I \xrightarrow{\pi} F$

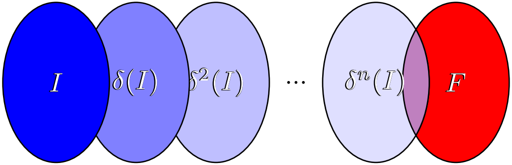
$$\Updownarrow$$

$$\delta^*(I) \cap F \neq \emptyset$$

...or via **backward analysis**...

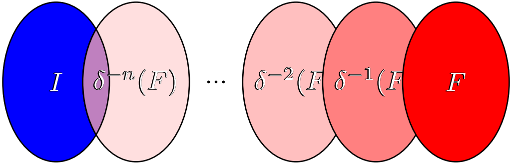


Reachability problems are usually solved via **forward analysis**...



$$\exists \text{ path } \pi \text{ in } \mathcal{T}.\ I \xrightarrow{\pi} F$$
$$\Updownarrow$$
$$\delta^*(I) \cap F \neq \emptyset$$

...or via **backward analysis**...



$$\exists \text{ path } \pi \text{ in } \mathcal{T}.\ I \xrightarrow{\pi} F$$
$$\Updownarrow$$
$$(\delta^{-1})^*(F) \cap I \neq \emptyset$$

For finite transition systems...

...reachability properties can be easily checked via forward/backward analysis.

For infinite transition systems...

...two problems arise:

- the sets $\delta^n(I)$ and $\delta^{-n}(F)$ may be infinite
  $\Rightarrow$ finite **symbolic representations** are needed
- in order to establish that no path from I to F exists an infinite number of steps may be required
  $\Rightarrow$ employment of suitable **acceleration techniques**.

Fact

The reachability problem is undecidable
for certain classes of infinite transition systems.

Fact

The reachability problem is undecidable for certain classes of infinite transition systems.

Example

Let $\mathcal{T} = (S, \delta)$ be the transition graph of a Turing machine:

- configurations (= vertices) are given by
  - a *tape inscription* $a_1, \ldots, a_n$
  - a *head position* m, with $1 \leq m \leq n$
  - a *control state* q

  $\Rightarrow$ we encode a configuration with the word $a_1 \ldots a_{m-1} q a_m \ldots a_n$

- transitions (= edges) are of the following forms

$$\begin{array}{ccc} a_1 \ldots a_{m-1} q a_m \ldots a_n & a_1 \ldots a_{m-1} q a_m \ldots a_n & a_1 \ldots a_{m-1} q a_m \ldots a_n \\ \downarrow & \downarrow & \downarrow \\ a_1 \ldots a_{m-1} q' a'_m \ldots a_m & a_1 \ldots a_{m-2} q' a_{m-1} a'_m \ldots a_m & a_1 \ldots a_{m-1} a'_m q' a_{m+1} \ldots a_m \end{array}$$

Example (continued)

There is no algorithm that decides whether a given Turing machine reaches the halting state from a given initial configuration.

$\Rightarrow$ the reachability problem is undecidable
for transition graphs of Turing machines.

Example (continued)

There is no algorithm that decides whether a given Turing machine reaches the halting state from a given initial configuration.

$\Rightarrow$ the reachability problem is undecidable for transition graphs of Turing machines.

We shall present *meaningful classes of transition systems* that enjoy *decidable reachability problems*

(e.g., the transition graphs of pushdown automata and the transition graphs of Petri nets under suitable conditions).

Logics can be used as formal languages to express interesting properties of transition systems (e.g., safety, liveness, termination, ...).

Given a logical language $\mathscr{L}$ and a class $\mathscr{C}$ of transition systems, we are interested in solving the following problem:

Definition (Model checking problem)

Input: a sentence ψ in $\mathscr{L}$ and
(a *finite presentation* of) a transition system $\mathcal{T}$ in $\mathscr{C}$

Problem: decide whether ψ holds in $\mathcal{T}$, denoted $\mathcal{T} \vDash \psi$

Definition ($\mathscr{L}$-theory of $\mathcal{T}$)

The $\mathscr{L}$-theory of a given transition system $\mathcal{T}$ is the set of all sentences $\psi \in \mathscr{L}$ such that $\mathcal{T} \vDash \psi$.

Definition (First-Order (FO) Logic)

Given a (labeled and colored) transition system $\mathcal{T} = (S, (\delta_a)_{a \in A}, (P_c)_{c \in C})$, FO-formulas over $\mathcal{T}$ are defined as follows:

- variables $x, y, z, \ldots$ denote single elements in S

Definition (**First-Order (FO) Logic**)

Given a (labeled and colored) transition system $\mathcal{T} = (S, (\delta_a)_{a \in A}, (P_c)_{c \in C})$, FO-formulas over $\mathcal{T}$ are defined as follows:

- variables $x, y, z, \ldots$ denote single elements in S
- atomic formulas have one of the following forms:
  - $x = y$, meaning 'x *denotes the same vertex as* y'
  - $\delta_a(x, y)$, meaning '(x, y) *denotes an a-labeled edge*'
  - $P_c(x)$, meaning 'x *denotes a c-colored vertex*'

Definition (First-Order (FO) Logic)

Given a (labeled and colored) transition system $\mathcal{T} = (S, (\delta_a)_{a \in A}, (P_c)_{c \in C})$, FO-formulas over $\mathcal{T}$ are defined as follows:

- variables $x, y, z, \ldots$ denote single elements in S
- atomic formulas have one of the following forms:
  - $x = y$, meaning 'x *denotes the same vertex as* y'
  - $\delta_a(x, y)$, meaning '(x, y) *denotes an a-labeled edge*'
  - $P_c(x)$, meaning 'x *denotes a c-colored vertex*'
- more complex formulas are build up via
  - the Boolean connectives $\wedge$, $\vee$, $\neg$
  - and the existential and universal quantifications $\exists x, \forall x$.

Definition (First-Order (FO) Logic)

Given a (labeled and colored) transition system $\mathcal{T} = (S, (\delta_a)_{a \in A}, (P_c)_{c \in C})$, FO-formulas over $\mathcal{T}$ are defined as follows:

- variables $x, y, z, \ldots$ denote single elements in S
- atomic formulas have one of the following forms:
  - $x = y$, meaning 'x *denotes the same vertex as* y'
  - $\delta_a(x, y)$, meaning '(x, y) *denotes an a-labeled edge*'
  - $P_c(x)$, meaning 'x *denotes a c-colored vertex*'
- more complex formulas are build up via
  - the Boolean connectives $\wedge$, $\vee$, $\neg$
  - and the existential and universal quantifications $\exists x, \forall x$.

Example

'*The system can always switch from a good state to a bad one*' is translated into $\forall\, x.\ \big(P_{good}(x) \ \rightarrow\ \exists\, y.\ \delta(x, y)\ \wedge\ P_{bad}(y)\big)$.

First-order logic cannot express reachability properties.

First-order logic cannot express reachability properties.
$\Rightarrow$ we extend it with **set-variables**.

Definition (Monadic Second-Order (MSO) Logic)

Given a transition system $\mathcal{T} = (S, (\delta_a)_{a \in A}, (P_c)_{c \in C})$, MSO-formulas over $\mathcal{T}$ are defined as follows:

- FO-variables $x, y, z, \ldots$ denote single elements in S
- MSO-variables $X, Y, Z, \ldots$ denote subsets of S

First-order logic cannot express reachability properties.
$\Rightarrow$ we extend it with **set-variables**.

Definition (Monadic Second-Order (MSO) Logic)

Given a transition system $\mathcal{T} = (S, (\delta_a)_{a \in A}, (P_c)_{c \in C})$, MSO-formulas over $\mathcal{T}$ are defined as follows:

- FO-variables $x, y, z, \ldots$ denote single elements in S
- MSO-variables $X, Y, Z, \ldots$ denote subsets of S
- atomic formulas have one of the following forms:
  - $\delta_a(x, y)$, meaning '(x, y) *denotes an a-labeled edge*'
  - $P_c(x)$, meaning 'x *denotes a c-colored vertex*'
  - $X(y)$, meaning 'y *denotes a vertex in the set* X'

First-order logic cannot express reachability properties.
$\Rightarrow$ we extend it with **set-variables**.

Definition (Monadic Second-Order (MSO) Logic)

Given a transition system $\mathcal{T} = (S, (\delta_a)_{a \in A}, (P_c)_{c \in C})$, MSO-formulas over $\mathcal{T}$ are defined as follows:

- FO-variables $x, y, z, \ldots$ denote single elements in S
- MSO-variables $X, Y, Z, \ldots$ denote subsets of S
- atomic formulas have one of the following forms:
  - $\delta_a(x, y)$, meaning '(x, y) *denotes an a-labeled edge*'
  - $P_c(x)$, meaning 'x *denotes a c-colored vertex*'
  - $X(y)$, meaning 'y *denotes a vertex in the set* X'
- more complex formulas are build up via
  - the Boolean connectives $\wedge$, $\vee$, $\neg$
  - quantifications $\exists x, \forall x$ over FO-variables
  - quantifications $\exists X, \forall X$ over MSO-variables.

Definability of reflexive and transitive closures

The **reflexive and transitive closure** δ^*
of any relation δ is definable in MSO logic:

$$\delta^*(x,y) \;:=\; \forall X.\ \big(X(x) \wedge \forall z,w.\ (X(z) \wedge \delta(z,w) \rightarrow X(w))\big) \;\rightarrow\; X(y)$$

Definability of reflexive and transitive closures

The **reflexive and transitive closure** δ^* of any relation δ is definable in MSO logic:

$$\delta^*(x,y) \ := \ \forall X. \ (X(x) \wedge \forall z, w. \ (X(z) \wedge \delta(z,w) \rightarrow X(w))) \ \rightarrow \ X(y)$$

$\Rightarrow$ MSO logic is powerful enough to express non-trivial properties of systems like reachability, planarity...

Example (1)

'*The system can eventually reach a bad state from a good one*' is translated into $\exists\, x, y. \ \big(P_{good}(x) \ \wedge \ P_{bad}(y) \ \wedge \ \delta^*(x,y)\big)$.

Example (2)

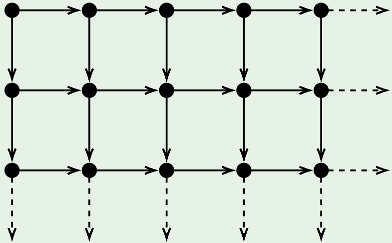
'*The system satisfies the Church-Rosser property*' is translated into $\forall\, x, y, z. \ \big((\delta^*(x,y) \ \wedge \ \delta^*(x,z)) \ \rightarrow \ \exists\, w. \ (\delta^*(y,w) \ \wedge \ \delta^*(z,w))\big)$.

Unfortunately, model checking problems for MSO logic (and even MSO-theories of single graphs) are often undecidable.

Example (Undecidability of the MSO-theory of the grid)

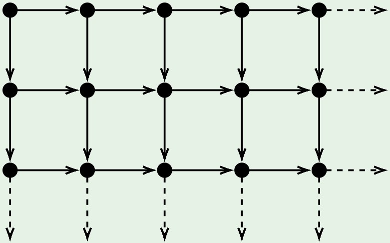
Consider the infinite grid $\mathcal{G}$ and a generic Turing machine M.

We build an MSO-sentence ψ_M such that $\mathcal{G} \vDash \psi_M$ iff M reaches the halting state (starting from the empty tape).



Unfortunately, model checking problems for MSO logic (and even MSO-theories of single graphs) are often undecidable.

Example (Undecidability of the MSO-theory of the grid)



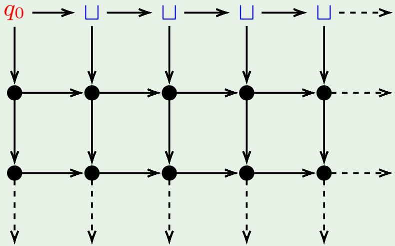
We mark the vertices of $\mathcal{G}$ with *tape symbols* and *control states* to encode, row by row, each configuration of a halting run.

ψ_M expresses the existence of such a marking as follows:

Unfortunately, model checking problems for MSO logic (and even MSO-theories of single graphs) are often undecidable.

Example (Undecidability of the MSO-theory of the grid)

We mark the vertices of $\mathcal{G}$ with *tape symbols* and *control states* to encode, row by row, each configuration of a halting run.



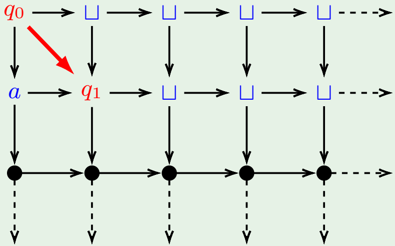
ψ_M expresses the existence of such a marking as follows:

- '*the first row is the initial configuration of M*'

Unfortunately, model checking problems for MSO logic (and even MSO-theories of single graphs) are often undecidable.

Example (Undecidability of the MSO-theory of the grid)

We mark the vertices of $\mathcal{G}$ with *tape symbols* and *control states* to encode, row by row, each configuration of a halting run.



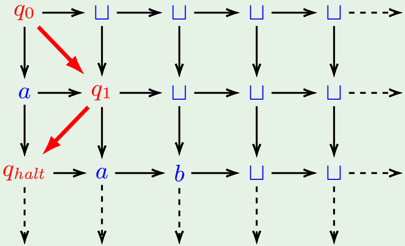
ψ_M expresses the existence of such a marking as follows:

- '*the first row is the initial configuration of M*'
- '*the next row is the next configuration of M*'

Unfortunately, model checking problems for MSO logic (and even MSO-theories of single graphs) are often undecidable.

Example (Undecidability of the MSO-theory of the grid)

We mark the vertices of $\mathcal{G}$ with *tape symbols* and *control states* to encode, row by row, each configuration of a halting run.



ψ_M expresses the existence of such a marking as follows:

- '*the first row is the initial configuration of M*'
- '*the next row is the next configuration of M*'
- '*there is a row containing the halting state of M*'

Example (Some details)

- we use one MSO-variable X_q for each control state q to represent the set of grid positions where q occurs
- we use one MSO-variable Y_a for each tape symbol a
- $\forall$ position x, $\exists!$ state q or symbol a such that $X_q(x)$ or $Y_a(x)$
- each row contains exactly one position x such that $X_q(x)$ for some q

Example (Some details)

- we use one MSO-variable X_q for each control state q to represent the set of grid positions where q occurs
- we use one MSO-variable Y_a for each tape symbol a
- $\forall$ position x, $\exists!$ state q or symbol a such that $X_q(x)$ or $Y_a(x)$
- each row contains exactly one position x such that $X_q(x)$ for some q
- '*the first row is the initial configuration of M*':
  $\exists\, x.\ (\nexists\, y.\ (\delta_h(y,x)\ \vee\ \delta_v(y,x))\ \wedge\ X_{q_0}(x)\ \wedge\ \forall\, y.\ (\delta_h^*(x,y)\ \rightarrow\ Y_{\sqcup}(y))$
- '*there is one row containing the halting state of M*':
  $\exists\, x.\ X_{q_{halt}}(x)$
- '*the next row is the next configuration of M*':
  consider a generic window of 4×2 cells
  the marking on the second cell in the lower row is uniquely determined by M-transitions and the marking on the upper row
  $\Rightarrow$ we can write a formula constraining 8-tuples of FO-variables to be compatible with M-transitions.

- We shall describe basic techniques for establishing the decidability of FO/MSO-theories of infinite transition systems

  (e.g., interpretations, unfoldings, reductions).

- We shall present *meaningful classes of transition systems* that *enjoy decidable model checking problems for FO/MSO logics*

  (e.g., the transition graphs of pushdown automata, prefix-recognizable graphs, automatic graphs).

Note: one could also consider logics in between FO and MSO (e.g., reachability logics, μ-calculus, ...)

Go

Go