Verification of temporal logics on infinite-state systems

Day 1, Lecture 2
Verification of the basic tense logic on rational Kripke models

Stéphane Demri and Valentin Goranko

ESSLLI 2007, Dublin, July 2007
Basic tense logic for transition systems

For labelled transition systems of type $T = \langle S, \{Ra\}_{a \in A} \rangle$, we associate a propositional language with a set of atomic propositions $PROP$ and a set of modal operators

$$\{\langle Ra\rangle, \langle Ra^{-1}\rangle\}_{a \in A}$$

with the standard Kripke semantics:

$M, u \models \langle Ra\rangle \phi$ iff $M, w \models \phi$ for some immediate $Ra$-successor $w$ of $v$, i.e., such that $vRw$;

$M, u \models \langle Ra^{-1}\rangle \phi$ iff $M, w \models \phi$ for some immediate $Ra$-predecessor $w$ of $v$, i.e. such that $wRv$.

Dual operators: $[Ra] := \neg\langle Ra\rangle \neg$ and $[Ra^{-1}] = \neg\langle Ra^{-1}\rangle \neg$.

The resulting logic is the basic tense logic $K_t$.

$K_t$ is not very expressive. In particular, it can only express local properties but no reachability properties.
Finite automata and regular languages (courtesy slide)

(Non-deterministic) finite automaton: tuple \( \langle \Sigma, S, s, F, \delta \rangle \), where:

- \( \Sigma \) is a finite set of symbols, called (input) alphabet,
- \( S \) is a finite set of states,
- \( s \in S \) is an initial state,
- \( F \subseteq S \) is a set of accepting states,
- \( \delta \subseteq S \times \Sigma \times S \) is a transition relation

A (finite) word in \( \Sigma \) is any (finite) sequence of symbols from \( \Sigma \). The set of all finite words in \( \Sigma \) is denoted by \( \Sigma^* \). A language in \( \Sigma \) is any subset of \( \Sigma^* \).

A word \( w \in \Sigma^* \) is recognized by the finite automaton \( A \) if, starting from the initial state and ‘reading’ \( w \) symbol by symbol, \( A \) ends in an accepting state.

The set of all words recognized by the finite automaton \( A \) is called the language of \( A \), denoted by \( L(A) \).

A language in \( \Sigma \) is rational or regular if it is the language of some finite automaton over \( \Sigma \).
Rational Transducers

Finite automata can recognize relations by reading all arguments synchronously, presented by their ‘convolution’.

Rational transducers are asynchronous finite automata on pairs (possibly, tuples) of words. Formally, a rational transducer is:

$$ T = \langle Q, \Sigma, \Gamma, q_i, F, \rho \rangle $$

where:
- $Q$ is a finite set of states,
- $\Sigma$ is a finite input alphabet,
- $\Gamma$ is a finite output alphabet,
- $q_i \in Q$ is an initial state,
- $F$ is a set of accepting states,
- $\rho \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ is the transition relation;

More generally (yet, equivalently): $\rho \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$. 
The relation $R \subseteq \Sigma^* \times \Gamma^*$ recognized by the transducer $\mathcal{T}$ is the set of all pairs of words for which $\mathcal{T}$ has a reading that ends in an accepting state.

Two views on a transducer:
- static, as a recognizer of a relation,
- and dynamic, as a transformer of words.

Hereafter, we assume that $\Gamma = \Sigma$. 
Example

\[ T = \langle Q, \Sigma, \Gamma, q_i, F, \rho \rangle \]

where:
\[ Q = \{q_1, q_2\}; \quad \Sigma = \Gamma = \{0, 1\}; \quad q_i = q_1; \quad F = \{q_2\}; \quad \rho = \{(q_1, 0, 0, q_1), (q_1, 1, 1, q_1), (q_1, \epsilon, 0, q_2), (q_1, \epsilon, 1, q_2)\} \]

\( T \) recognizes the pairs of words \((u, uw)\) where \( u \in \Sigma^* \), \( w \in \Sigma^+ \).
Rational relations and graphs

**Rational relation**: relation, recognizable by a rational transducer.

Equivalently, [Elgot and Mezei, 1965]: given an alphabet $\Sigma$, a (binary) rational relation over $\Sigma^*$ is rational iff it is a rational subset of $\Sigma^* \times \Sigma^*$, i.e., a relation generated by a rational expression from a finite subset of $\Sigma^* \times \Sigma^*$.

Example: $((0, 10)^* + (11, \varepsilon))^* \equiv (((0, 1)(\varepsilon, 0))^* + (1, \varepsilon)(1, \varepsilon))^*$.

The class of rational relations is closed under unions (but not intersections and complements), compositions, and inverses.

**Rational graph**: $(W, R)$, where $W \subseteq \Sigma^*$ is a rational language in a finite alphabet $\Sigma$ and $R \subseteq \Sigma^* \times \Sigma^*$ is a rational relation on $\Sigma$. 
Example: the infinite grid as a rational graph

The infinite grid:

A transducer that recognizes it:
Example: the complete binary tree as a rational graph

The complete binary tree $T_2$:

A labeled transducer recognizing $T_2$: 

![Diagram of the complete binary tree and the labeled transducer recognizing it]
Synchronous transducers
Automatic relations and graphs

Sometimes the reading heads of a transducer can be synchronized, e.g. when all transitions $u/w$ occurring in a loop are length preserving, i.e., $|u| = |w|$.

That synchronisation can possibly be achieved by appending padding symbols at the end of the shorter word.

In these cases, the transducer is equivalent to a finite automaton reading tuples of letters.

Relations recognizable by synchronous transducers are called automatic relations and the respective rational graphs are automatic graphs.

For example, the infinite grid and the complete binary tree are automatic graphs.
Rational graphs: some important cases

- The configuration graph of every Turing machine. In particular, all pushdown graphs.
- The configuration graph of every Petri net.
- All counter systems.

In fact, all these are automatic graphs.
Rational graphs: some facts

► Reachability in a rational graph is generally undecidable.

► [Johnson’1986, Morvan’2000]: testing an input rational relation for (ir)reflexivity, transitivity, symmetry, etc., is undecidable (by reduction of Post Correspondence Problem).

► [Thomas, 2002]: there is a rational graph with undecidable first-order theory.

► Yet, inclusion and equality of rational relations are undecidable.

So, after all, what are rational graphs good for?

For model checking of basic tense logic!
Basic Tense Logic and Symbolic Model Checking

In a TCS, 2001 paper ‘Symbolic model checking with rich assertional languages’, Kesten, Maler, Marcus, Pnueli, and Shahar formulate the following minimal requirements for an assertional language $\mathcal{L}$ to be adequate for symbolic model checking:

1. The property to be verified and the initial conditions (i.e., the set of initial states) should be expressible in $\mathcal{L}$.

2. $\mathcal{L}$ should be effectively closed under the boolean operations of negation and disjunction, and possess an algorithm for deciding equivalence of two assertions.

3. There should exist an algorithm for constructing the predicate transformer $\text{pred}$, where $\text{pred}(\phi)$ is an assertion characterizing the set of states that have a successor state satisfying $\phi$.

The minimal natural logical language satisfying these requirements is basic modal logic. So, its tense extension is just good enough for model checking pre-conditions and post-conditions. For more, reachability must be definable.
A Kripke model \((W, \{R_a\}_{a \in A}, V)\) is rational if:

- \((W, \{R_a\}_{a \in A})\) is a rational graph;
- \(V\) is a rational valuation, i.e. every \(V(p)\) is a rational subset of \(W\).

For instance, the grid \(\mathbb{N} \times \mathbb{N}\) with atomic propositions for every row and column is a rational Kripke model.
Example

\[ G = (S, R, V), \]

where \( S = 0^*1^* \), \( R = \{((010)^n, u) | n \geq 0, u \in 0^* (10^*)^n\} \), 
\( V(p) = 0^*10010^* \), and \( V(q) = 1^* \), is a rational Kripke model. Indeed, \( R = (((\epsilon, 0)^* (010, 1) (\epsilon, 0)^*)^* \).

Here are the respective machines recognising \( S, V(p), V(q), R \):

\[ A_S: \]

\[ A_p: \]

\[ A_q: \]

\[ \mathcal{T}_R: \]
**Regularity Preservation Lemma:** Let $\Sigma$ be a finite non-empty alphabet, $X \subseteq \Sigma^*$ a regular set, and $R \subseteq \Sigma^* \times \Sigma^*$ a rational relation. Then the sets

$$\langle R \rangle X = \{ u \in \Sigma^* | \exists v \in X (uRv) \}$$

and

$$\langle R^{-1} \rangle X = \{ u \in \Sigma^* | \exists v \in X (vRu) \}$$

are regular subsets of $\Sigma^*$, effectively computable from the automaton for $X$ and the transducer for $R$. 
Example

Transducer $\mathcal{T}$ recognizing the relation $R = (0, \varepsilon)^+ (\varepsilon, 1)^* \{(0, 0), (\varepsilon, 1)\}$:

Automaton $\mathcal{A}$ recognizing the regular language $X = \{0\}$:

The automaton $\langle \mathcal{T} \rangle \mathcal{A}$ recognizing the regular language $\langle R \rangle X = 0^2 0^*$:
**Regularity Preservation Theorem:**
For every rational Kripke model $\mathcal{M}$ and every formula $\varphi \in K_t$, the set $\llbracket \varphi \rrbracket_{\mathcal{M}}$ is an effectively computable regular language.

Proof: structural induction on $\varphi$, using the closure of regular languages under Booleans and the Regularity Preservation Lemma.

Consequently, local model checking, global model checking, and model satisfiability checking of formulas of $K_t$-formulae in rational Kripke models are algorithmically decidable.

This result easily extends to hybrid tense logic, obtained from $K_t$ by adding nominals and universal modality.
Some questions and open problems

- Is there any logical (e.g. in terms of interpretations) or model-theoretic characterization of rational structures?
- What is the complexity of the outlined model-checking procedure? Is there a practically efficient automata-based model checking technique for $K_t$ on rational Kripke models?
- What is the strongest modal language for which model checking is decidable on rational Kripke models? I.e., what other modal operators, preserving rationality but not definable in the hybrid extension of $K_t$, can be added to the language?
- Identify natural restrictions of the class of RKM where model checking of modal logic with reachability is decidable.
- Generalize to omega-rational and tree-rational Kripke models.
The menu for tomorrow

Lecture 2/1: Reachability problems and queries. Symbolic computation of reachability.

1. Reachability problems and queries.
2. Pushdown systems.
3. Symbolic computation of reachability in pushdown systems.

Lecture 2/2: Temporal logics for specification and verification of infinite-state transition systems.

1. Linear and branching time temporal logics: LTL, CTL, CTL*.
   Expressing properties of transition systems in temporal logic.
2. Automata-based techniques for model checking in finite state systems.