Recurrent Reachability in Regular Model Checking

Anthony Widjaja To and Leonid Libkin

LFCS, School of Informatics, University of Edinburgh

LPAR 2008

Overview

- Verification of infinite-state systems
- Some sources of infinity:
  - unbounded stacks or FIFO queues
  - unbounded integer variable or real variable
  - unbounded number of finite processes
- Infinite systems need finite representations
- Regular model checking: use word/tree automata as finite representations

Outline

- Background and Motivation
  - The model: word/tree automatic transition systems
  - Verification questions
  - Survey of known results
- Our contributions
  - Recurrent reachability
  - Model checking for CTL-like logic
- Future work

Background and motivation

Automatic transition systems

- Two flavors:
  - word-automatic
  - tree-automatic
- Main ideas:
  - Domains are Σ^* or $\textsc{Tree}(\Sigma)$
  - Automata interpret atomic propositions
  - Regular transducers interpret transition relations

Regular transducers

- Example: $(aaabab, bab)$

Regular transducers

- Example: $(aaabab, bab)$

- Can be thought of as

$$\begin{bmatrix} a \\ b \end{bmatrix}\begin{bmatrix} a \\ a \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}\begin{bmatrix} b \\ \perp \end{bmatrix}\begin{bmatrix} a \\ \perp \end{bmatrix}\begin{bmatrix} b \\ \perp \end{bmatrix}$$

Regular transducers

- Example: $(aaabab, bab)$

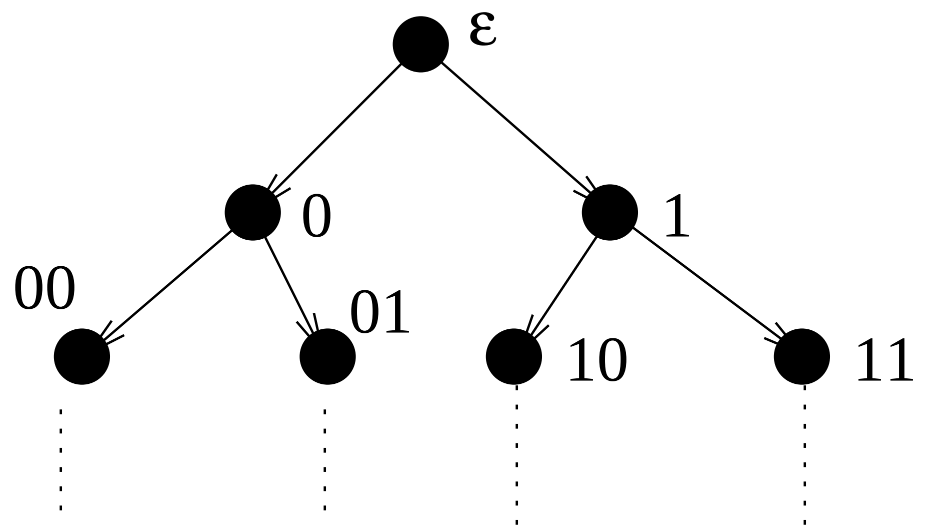
- Can be thought of as

$$\begin{bmatrix} a \\ b \end{bmatrix}\begin{bmatrix} a \\ a \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}\begin{bmatrix} b \\ \perp \end{bmatrix}\begin{bmatrix} a \\ \perp \end{bmatrix}\begin{bmatrix} b \\ \perp \end{bmatrix}$$

- word over $\Sigma_\perp \times \Sigma_\perp$, where $\Sigma_\perp := \Sigma \cup \{\perp\}$

- Automaton over $\Sigma_\perp \times \Sigma_\perp$ defines a regular binary relation over Σ^*

A concrete example: infinite binary tree



Infinite binary tree (cont)

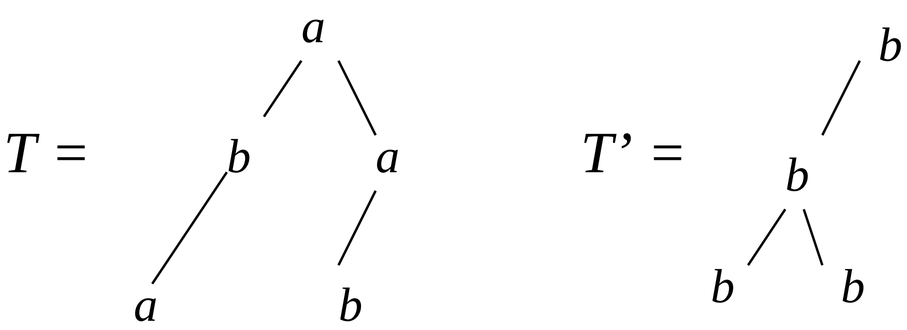
$\mathfrak{T} = \langle \{0,1\}^*; <, L_0, L_1 \rangle$:

- $<= \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^* \cdot \left(\begin{bmatrix} \perp \\ 0 \end{bmatrix} + \begin{bmatrix} \perp \\ 1 \end{bmatrix} \right)$
- $L_0 = (0+1)^*0$
- $L_1 = (0+1)^*1$

Note: $<^*$ is also a regular relation.

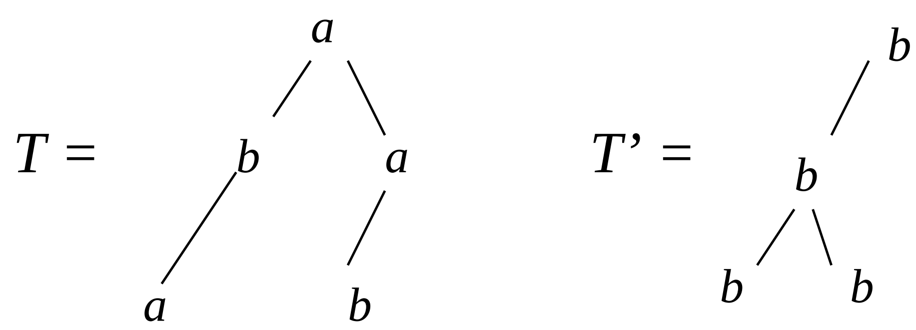
Regular tree transducers

- Example:

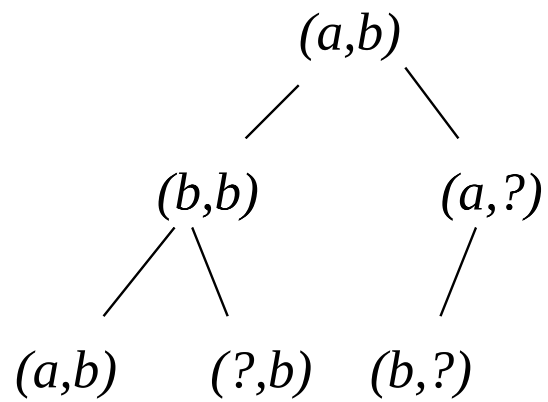


Regular tree transducers

- Example:



- (T, T') can be thought of as



Other examples

Word-automatic

- Pushdown systems
- Prefix-recognizable systems
- Lossy channel systems
- Parameterized systems

Tree-automatic

- PA-processes (minus commutativity)
- Ground tree rewrite systems

Verification questions

- Reachability (safety):

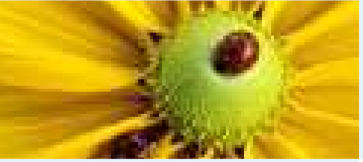
  Input: two states s_1, s_2
  Task: decide whether $s_1 \longrightarrow^* s_2$

- Recurrent reachability (liveness):

  Input: state s, and a set S of states
  Task: decide whether s can visit S infinitely often

Some known results

- **Theorem** (Folklore): The transitive closure $\rightarrow^+$ of a regular relation $\rightarrow$ is not necessarily regular.



Some known results

- **Theorem** (Folklore): The transitive closure $\rightarrow^+$ of a regular relation $\rightarrow$ is not necessarily regular.

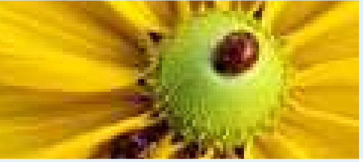
- In practice, $\rightarrow^+$ for automatic systems are often regular. Some good semi-algorithms for computing $\rightarrow^+$ have been developed.

Some known results

- Theorem (Folklore): The transitive closure $\rightarrow^+$ of a regular relation $\rightarrow$ is not necessarily regular.

- In practice, $\rightarrow^+$ for automatic systems are often regular. Some good semi-algorithms for computing $\rightarrow^+$ have been developed.

- Definition: If $\rightarrow^+$ is regular, a transducer for $\rightarrow^+$ is called an iterating transducer.



Some known results (cont)

- **Theorem**: $\longrightarrow^+$ is regular and PTIME-computable for:
  - Pushdown systems (Caucal)
  - GTRSs (Dauchet et al.)
  - PA-processes (Lugiez & Schnoebelen)

- **Theorem**: $\longrightarrow^+$ is regular and EXPTIME-computable for prefix-rec. systems

Some known results (cont.)

What about recurrent reachability?

Some known results (cont.)

What about recurrent reachability?

Partial answers:

- PTIME-computable for *pushdown systems* (Esparza et al.) and *GTRSs* (Löding);
- EXPTIME-computable for *prefix-rec. systems* (follows from Löding's).
- Undecidable for lossy channel systems (Abdulla & Jonsson)

Our contribution

Recurrent reachability

Restriction: In the following, we ONLY consider automatic transition systems:

- whose transitive closures are *regular*
- iterating transducers available as input

Recurrent reachability (cont)

Theorem: Over automatic systems:

- recurrent reachability is decidable in PTIME in the size of systems + iterating transducers;
- Buchi word/tree automata that recognize infinite witnessing paths are PTIME-computable.

Recurrent reachability (cont)

Theorem: Over automatic systems:

- recurrent reachability is decidable in PTIME in the size of systems + iterating transducers;
- Buchi word/tree automata that recognize infinite witnessing paths are PTIME-computable.

Corollary: Recurrent reachability is decidable in PTIME for pushdown systems, GTRSs, and PA-processes and is decidable in EXPTIME for prefix-rec. systems.

Recurrent reachability (proof)

Proof Idea for word case:
Inputs are:

- an NFA $\mathcal{A}$
- transducers $\rightarrow$ and $\rightarrow^{+}$
- and a word w

Recurrent reachability (proof)

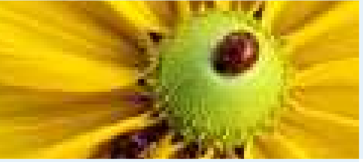
Proof Idea for word case:

Inputs are:

- an NFA $\mathcal{A}$
- transducers $\rightarrow$ and $\rightarrow^+$
- and a word w

Notation: $Rec(\mathcal{A})$ denotes the set of all words s_0 with an infinite path $s_0 \rightarrow s_1 \rightarrow \ldots$ visiting $L(\mathcal{A})$ infinitely often.

Approach: show that $Rec(\mathcal{A})$ is regular for which an automaton is constructible in PTIME



Recurrent reachability (proof)

- $s_0 \in Rec(\mathcal{A})$ iff there is an infinite witnessing sequence

$$\begin{array}{ccc} s_0 \longrightarrow^+ & s_1 \longrightarrow^+ & s_2 \longrightarrow^+ \ldots \\ & \pitchfork & \pitchfork \quad\quad \ldots \\ & L(\mathcal{A}) & L(\mathcal{A}) \quad\quad \ldots \end{array}$$

- Divide $Rec(\mathcal{A})$ into two sets Rec_1 and Rec_2:
  - $w \in Rec_1$ has a looping witnessing infinite sequence, i.e., $s_i = s_j$ for some distinct i, j.
  - $w \in Rec_2$ has a non-looping witnessing infinite sequence, i.e., $s_i \neq s_j$ for all distinct i, j.

Recurrent reachability (proof)

- An NFA for Rec_1 can easily be constructed in PTIME (Hint: simple product construction and projection)

- How do we construct an NFA for Rec_2?

Recurrent reachability (proof)

- An NFA for Rec_1 can easily be constructed in PTIME (Hint: simple product construction and projection)
- How do we construct an NFA for Rec_2?
- Claim: $w \in Rec_2$ iff there exists a "nice" witnessing sequence.
- From this characterization, it will be easy to construct an NFA for Rec_2.

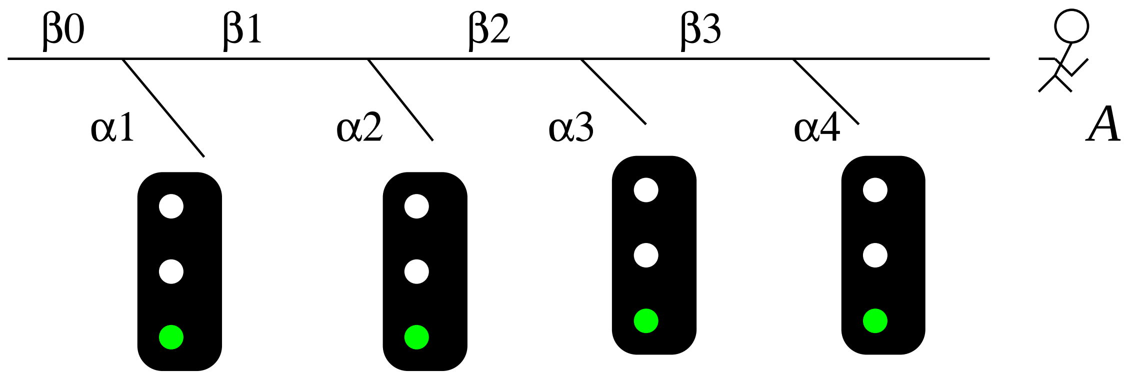
Recurrent reachability (proof)

- Lengths of words in a non-looping witnessing infinite sequence grow indefinitely.

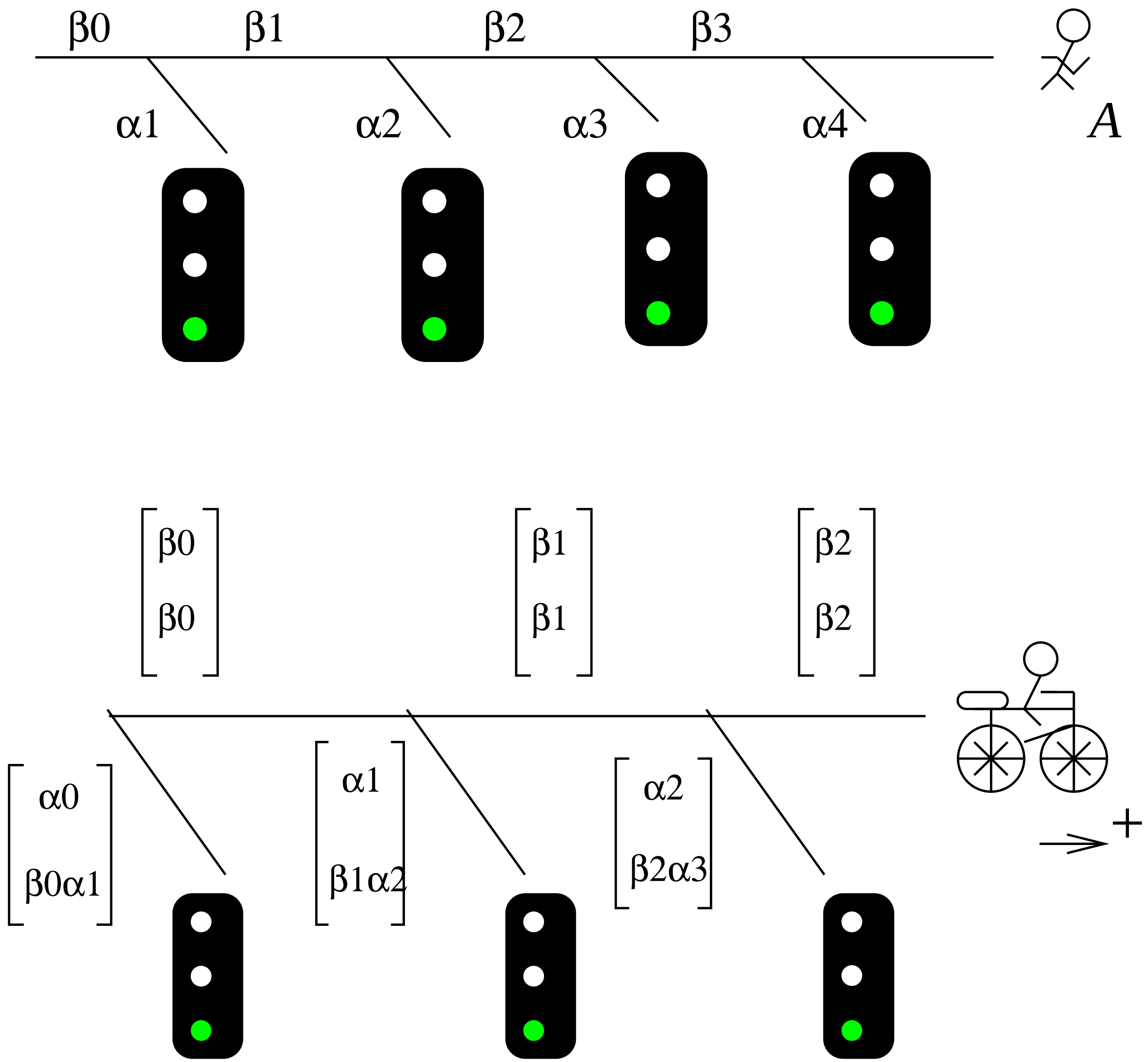
- moreover, we can extract subsequence of the form

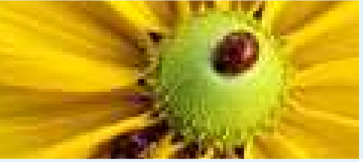
s_0	ε	ε	ε	$\ldots$
β_0	α_1	ε	ε	$\ldots$
β_0	β_1	α_2	ε	$\ldots$
β_0	β_1	β_2	α_3	ε
$\vdots$	$\vdots$	$\vdots$	β_3 $\vdots$	$\ddots$

Recurrent reachability (proof)



Recurrent reachability (proof)





Recurrent reachability (proof)

- Construct Büchi automaton $\mathcal{B}$ that recognizes ω-words of the form

$$\begin{bmatrix} \alpha_0 \\ \beta_0 \end{bmatrix} \# \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \# \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix} \# \ldots$$

  satisfying aforementioned conditions

- Construct $\mathcal{A}_2$ by taking projection and do reachability analysis in $\mathcal{B}$

Model Checking CTL-like logic

- Consider **EF**-logic with syntax:

$$\varphi, \varphi' := \top \mid P_i,\ i \leq n \mid \varphi \vee \varphi' \mid \neg\varphi \mid \mathbf{EX}\varphi \mid \mathbf{EF}\varphi$$

- Simplest meaningful branching-time logic

- Extend with formulas $\mathbf{EGF}\varphi$ interpreted as $[\![\mathbf{EGF}\varphi]\!] := Rec([\![\varphi]\!])$.

Model Checking CTL-like logic

- Consider **EF**-logic with syntax:

$$\varphi, \varphi' := \top \mid P_i,\ i \leq n \mid \varphi \vee \varphi' \mid \neg\varphi \mid \mathbf{EX}\varphi \mid \mathbf{EF}\varphi$$

- Simplest meaningful branching-time logic

- Extend with formulas $\mathbf{EGF}\varphi$ interpreted as $[\![\mathbf{EGF}\varphi]\!] := Rec([\![\varphi]\!])$.

- Corollary: Model checking ($\mathbf{EF} + \mathbf{EGF}$)-logic over automatic transition systems is decidable.

Future work

Plan for third year

- More examples that fit our restriction
- Some implementations
- Complexity of model-checking $(\mathbf{EF} + \mathbf{EGF})$-logic over automatic transition systems