Model Checking

Lecture #6: Verifying Omega-Regular Properties

[Baier & Katoen, Chapter 4.4]

Joost-Pieter Katoen

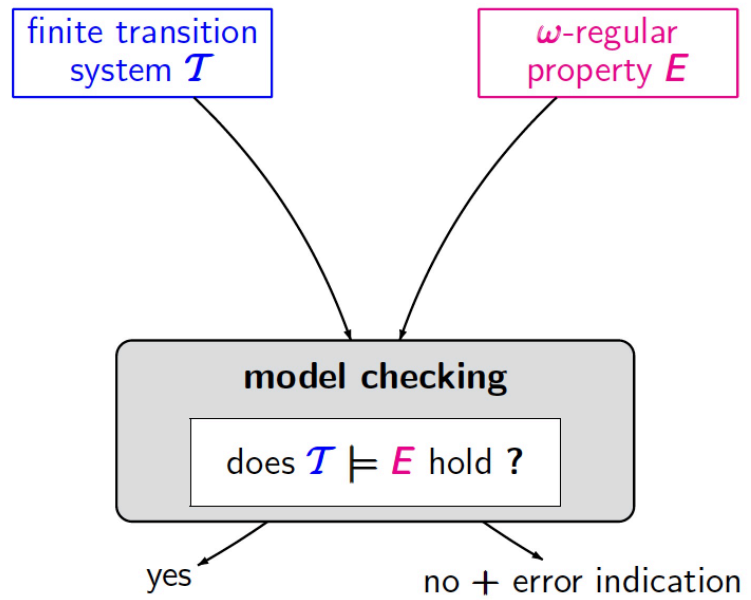
Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, WiSe 2019/2020

Overview

Overview

Topic



ω-Regular Properties

Definition: ω-regular language

The set $\mathfrak{L}$ of infinite works over the alphabet Σ is ω-regular if $\mathfrak{L} = \mathfrak{L}_\omega(G)$ for some ω-regular expression G over Σ.

Definition: ω-regular properties

LT property E over AP is ω-regular if E is an ω-regular language over 2^{AP}.

This is equivalent to:

LT property E over AP is ω-regular if E is accepted by a non-deterministic Büchi automaton (over the alphabet 2^{AP}).

Example ω-Regular Properties

- Any invariant E is an ω-regular property
  - Φ^ω describes E with invariant condition Φ
- Any regular safety property E is an ω-regular property
  - $\overline{E} = BadPref(E).\left(2^{AP}\right)^\omega$ is ω-regular
  - and ω-regular languages are closed under complement
- Let $\Sigma = \{ a, b \}$ Then:
  - Infinitely often a:

$$\left((\varnothing + \{ b \})^*.(\{ a \} + \{ a, b \})\right)^\omega$$

  - Eventually a:

$$\left(2^{AP}\right)^* .(\{ a \} + \{ a, b \}).\left(2^{AP}\right)^\omega$$

Overview

Nondeterministic Büchi automata

Definition: Nondeterministic Büchi automaton

A nondeterministic Büchi automaton (NBA) $\mathfrak{A} = (Q, \Sigma, \delta, Q_0, F)$ with:

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function
- $Q_0 \subseteq Q$ a set of initial states
- $F \subseteq Q$ is a set of accept (or: final) states.

This definition is the same as for NFA.

The acceptance condition of NBA is different though.

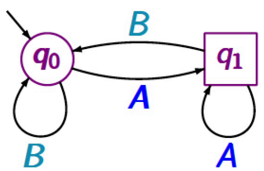
Language of a Büchi Automaton

- NBA $\mathfrak{A} = (Q, \Sigma, \delta, Q_0, F)$ and infinite word $w = A_1 A_2 \ldots \in \Sigma^\omega$
- A run for w in $\mathfrak{A}$ is an infinite sequence $q_0 \, q_1 \, \ldots \in Q^\omega$ such that:
  - $q_0 \in Q_0$ and $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $0 \leq i$
- Run $q_0 \, q_1 \, \ldots$ is accepting if $q_i \in F$ for infinitely many i
- The accepted language of $\mathfrak{A}$:

$$\mathfrak{L}_\omega(\mathfrak{A}) = \left\{ w \in \Sigma^\omega \mid \mathfrak{A} \text{ has an accepting run for } w \right\}$$

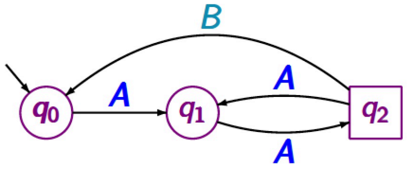
- NBA $\mathfrak{A}$ and $\mathfrak{A}'$ are equivalent if $\mathfrak{L}_\omega(\mathfrak{A}) = \mathfrak{L}_\omega(\mathfrak{A}')$

Examples



accepted language:
set of all infinite words that contain infinitely many A's

$(B^*.A)^\omega$



accepted language:
"every B is preceded by a positive even number of A's"

$$((A.A)^+.B)^\omega + ((A.A)^+.B)^*.A^\omega$$

NBA and ω-Regular Languages

Theorem

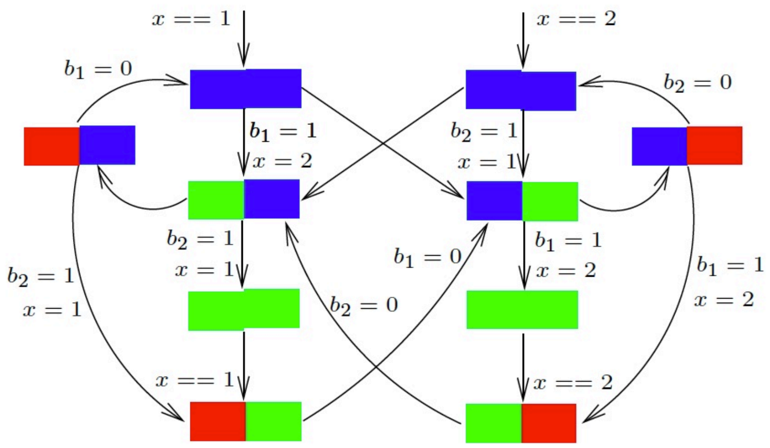
1. For every NBA $\mathfrak{A}$, the language $\mathfrak{L}_\omega(\mathfrak{A})$ is ω-regular.
2. For every ω-regular language L, there is an NBA $\mathfrak{A}$ with $L = \mathfrak{L}_\omega(\mathfrak{A})$.

Proof.

Previous lecture. □

Overview

Peterson's Transition System



If a thread wants to update the account, does it ever get the opportunity to do so?

"always (req_L $\Rightarrow$ eventually $@account_L$) $\wedge$ always (req_R $\Rightarrow$ eventually $@account_R$)"

Verifying Starvation Freedom

- Starvation freedom = when a thread wants access to account, it eventually gets it
- "Infinite bad prefix" automaton: once a thread wants access to the account, it never gets it
- Checking starvation freedom:

$$\underbrace{Traces(TS_{Pet})}_{\text{infinite traces}} \cap \mathfrak{L}_\omega(\overline{E_{live}}) = \varnothing?$$

- Intersection, complementation and emptiness of $\underbrace{\text{Büchi automata}}_{\text{accept infinite words}}$

Basic Idea

$TS \not\models E$ if and only if $Traces(TS) \not\subseteq E$

if and only if $Traces(TS) \cap \left(2^{AP}\right)^\omega \setminus E \neq \varnothing$

if and only if $Traces(TS) \cap \overline{E} \neq \varnothing$

if and only if $Traces(TS) \cap \mathfrak{L}_\omega(\mathfrak{A}) \neq \varnothing$

if and only if $TS \otimes \mathfrak{A} \not\models \underbrace{\text{"eventually for ever"} \neg F}_{\text{persistence property}}$

where $\mathfrak{A}$ is an NBA accepting the complement property $\overline{E} = \left(2^{AP}\right)^\omega \setminus E$

Persistence Property

Definition: persistence property

A persistence property over AP is an LT property $E_{pers} \subseteq \left(2^{AP}\right)^\omega$ of the form "eventually for ever Φ" for some propositional logic formula Φ over AP:

$$E_{pers} = \left\{ A_0 A_1 A_2 \ldots \in \left(2^{AP}\right)^\omega \mid \exists i \geq 0.\ \forall j \geq i.\ A_j \models \Phi \right\}$$

The formula Φ is called the persistence (or state) condition of E_{pers}.

"Φ is an invariant after a while"

Example

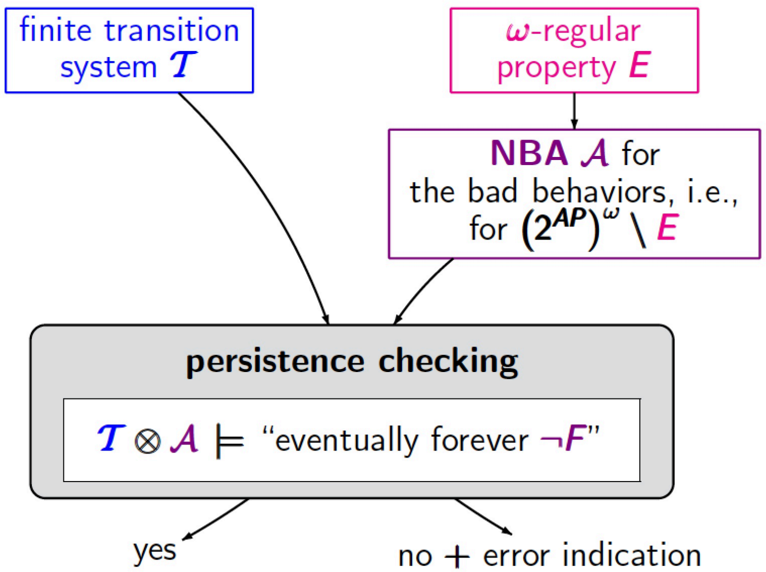
Problem Statement

Let

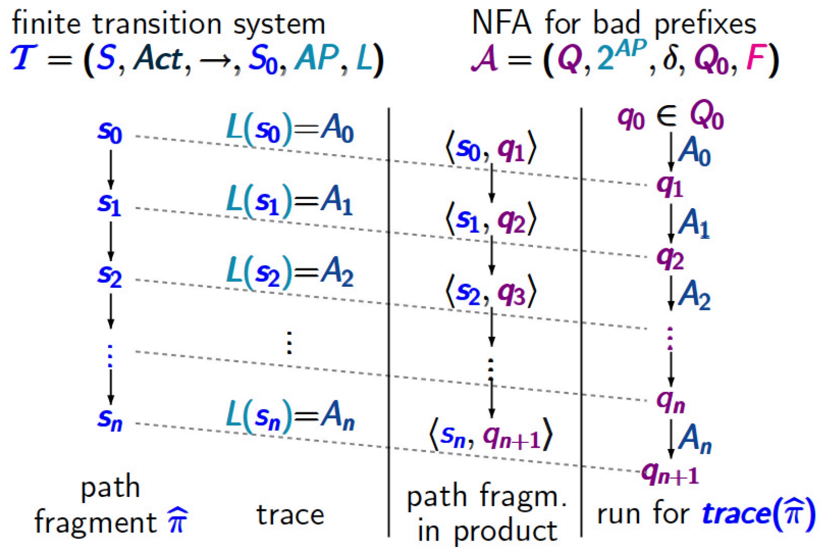
1. E be an ω-regular property over AP
2. $\mathfrak{A}$ be an NBA recognizing the complement of E
3. TS be a finite transition system (over AP) without terminal states

How to establish whether $TS \models E$?

Verifying Omega-Regular Properties



Product: Idea



Synchronous Product

Definition: synchronous product of TS and NBA

Let transition system $TS = (S, Act, \rightarrow, I, AP, L)$ without terminal states and $\mathfrak{A} = (Q, \Sigma, \delta, Q_0, F)$ a non-blocking NBA with $\Sigma = 2^{AP}$. The product of TS and $\mathfrak{A}$ is the transition system:

$$TS \otimes \mathfrak{A} \;=\; (S', Act, \rightarrow', I', AP', L') \qquad \text{where}$$

- $S' = S \times Q$, $AP' = Q$ and $L'(\langle s, q \rangle) \;=\; \{\, q \,\}$
- $\rightarrow'$ is the smallest relation defined by: $\dfrac{s \xrightarrow{\alpha} t \;\wedge\; q \xrightarrow{L(t)} p}{\langle s, q \rangle \xrightarrow{\alpha}{}' \langle t, p \rangle}$
- $I' = \{\, \langle s_0, q \rangle \mid s_0 \in I \;\wedge\; \exists q_0 \in Q_0.\; q_0 \xrightarrow{L(s_0)} q \,\}$.

Verifying ω-Regular Properties

Theorem

Let TS over AP, E an ω-regular property and NBA $\mathfrak{A}$ with $\mathfrak{L}(\mathfrak{A}) = \overline{E}$. Then:

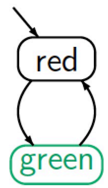
$$TS \models E \text{ iff } Traces(TS) \cap \mathfrak{L}_\omega(\mathfrak{A}) \;=\; \varnothing \text{ iff } TS \otimes \mathfrak{A} \models \underbrace{\text{eventually forever} \neg F}_{\text{persistence property}}$$

where F stands for $\bigvee_{q \in F} q$.

Proof

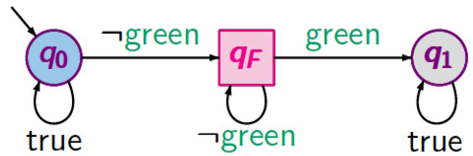
Example (1)

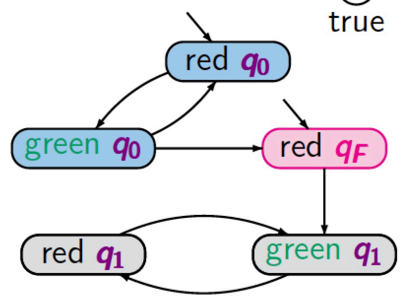
TS $\mathcal{T}$



LT property: "infinitely often green"

NBA $\mathcal{A}$ for the complement "from some moment on $\neg$green"



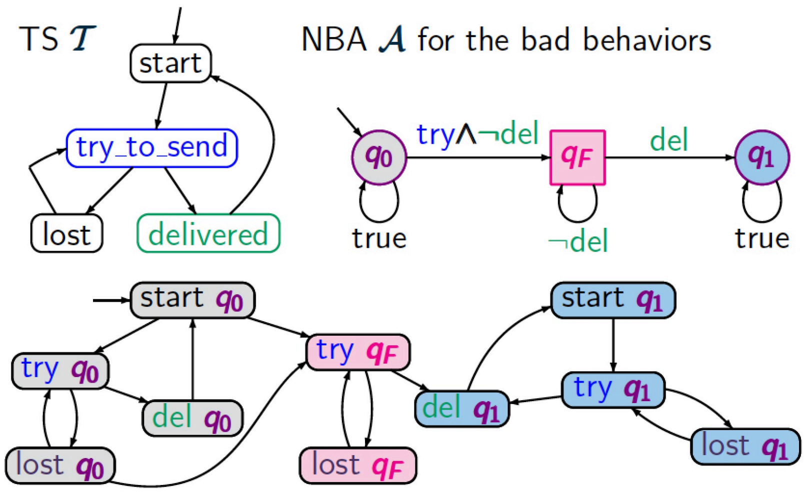


atomic propositions $AP' = \{q_0, q_F, q_1\}$

obvious labeling function

$\mathcal{T} \otimes \mathcal{A} \models$ "eventually forever $\neg F$"

Example (2)



set of atomic propositions $AP' = \{q_0, q_1, q_F\}$

Overview

Persistence Checking and Cycle Detection

Let

- TS be a finite transition system over AP without terminal states
- Φ a propositional formula over AP, and
- E the persistence property "eventually forever Φ"

$$TS \not\models E$$

if and only if

$$\exists s \in Reach(TS).\, s \not\models \Phi \;\wedge\; s \text{ is on a cycle in } TS$$

if and only if

there exists a non-trivial reachable SCC C with $C \cap \{ s \in S \mid s \models \neg\Phi \} \neq \varnothing$

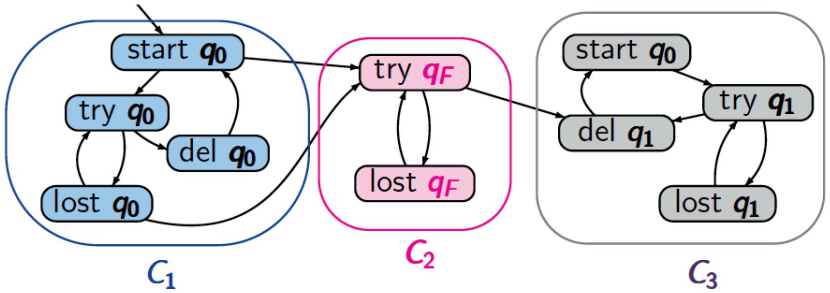
Persistence Checking

How to check for a reachable cycle containing a $\neg\Phi$-state?

Two linear-time algorithms:

- Alternative 1:
  - compute the maximal strongly connected components (SCCs) in TS
  - check whether some SCC is reachable from an initial state
  - ... that contains a $\neg\Phi$-state
- Alternative 2:
  - use a nested depth-first search
  - for each reachable $\neg\Phi$-state, check whether it belongs to a cycle
  - more adequate for on-the-fly verification algorithm
  - enables simple counterexample generation

Example SCC Algorithm



persistence property: "eventually forever $\neg q_F$"

3 reachable SCCs: C_1, C_2, C_3

C_2 non-trivial, and contains two states s with $s \not\models \neg q_F$

$\mathcal{T} \otimes \mathcal{A} \not\models$ "eventually forever $\neg q_F$"

A Naive Two-Phase Depth First-Search

1. Determine all $\neg\Phi$-states that are reachable from some initial state

   this is performed by a standard depth-first search

2. For each reachable $\neg\Phi$-state, check whether it belongs to a cycle
   - start a depth-first search in $\neg\Phi$-state s
   - to check whether s is reachable from itself

- Time complexity naive algorithm: $\Theta(N \cdot (\underbrace{N+M}_{\text{cycle check}}))$
  - where N is the number of states and M the number of transitions
  - where it is assumed that checking Φ is in $O(1)$
  - states reachable via K distinct $\neg\Phi$-states are searched K times

Time complexity nested DFS: $\Theta(N \cdot M)$.

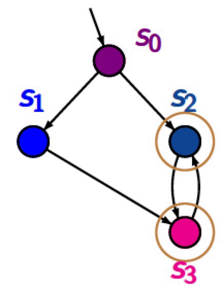
Nested Depth-First Search

- Idea: perform the two depth-first searches in an *interleaved* way
  - the outer DFS serves to encounter all reachable $\neg\Phi$-states
  - the inner DFS seeks for backward edges leading to a $\neg\Phi$-state
- Nested DFS
  - on full expansion of $\neg\Phi$-state s in the outer DFS, start inner DFS
  - in the inner DFS, visit all states reachable from s that have not been not visited in an inner DFS yet
  - backward edge found?

    a cycle containing $\neg\Phi$-state s found
  - no backward edge found to s?

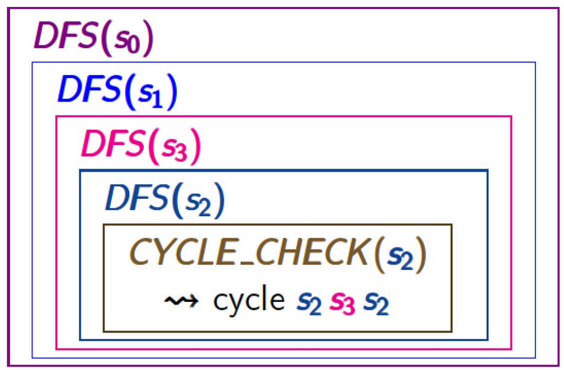
    continue the outer DFS (look for next $\neg\Phi$-state)

Nested DFS: Example



$s_1, s_2 \not\models a$

$s_0, s_3 \models a$



↑
returns correct answer "**no**"

$\mathcal{T} \not\models$ "eventually forever a"

Correctness of Nested DFS

Let:

- TS be a finite transition system over AP without terminal states and
- E a persistence property.

Then:

The nested DFS algorithm yields "no" if and only if $TS \not\models E$.

Proof

Time Complexity

The worst-case time complexity of nested DFS is in

$$\Theta(N+M)$$

where N is # states in TS, and M is # transitions in TS.

Counterexamples

A counterexample to $TS \models$ eventually forever Φ is an initial path fragment of the form

$$\underbrace{s_0}_{\in I} \ldots s_{n-1} \underbrace{s_n}_{\models \neg\Phi} s_{n+1} \ldots s_{n+m-1} \underbrace{s_n}_{\models \neg\Phi} \quad \text{for } m > 0.$$

Using nested depth-first search:

- Counterexample generation: use the DFS stacks
  - stack π_{out} for the outer DFS = path fragment $s_n\, s_{n-1} \ldots s_0$
  - stack π_{in} for the inner DFS = a cycle from state $s_n\, s_{n+m-1} \ldots s_n$
  - counterexample = *reverse* $(\pi_{in}.\pi_{out})$

Overview

Summary

- Checking a regular safety property E = checking invariant on product
  - with an NFA $\mathfrak{A}$ for the bad prefixes of E
  - "never reach an accept state of $\mathfrak{A}$"
- Checking an ω-regular property E = checking persistence on a product
  - with an NBA for the complement of E
  - "eventually forever no accept state of $\mathfrak{A}$"
- Persistence checking is solvable in linear time by a nested DFS
- Nested DFS = a DFS for reachable $\neg\Phi$-states + a DFS for cycle detection

Next Lecture

Friday November 8, 14:30