Overview

1. Omega-Regular Properties
2. Refresher: Büchi Automata
3. Verifying Omega-Regular Safety Properties
4. Nested Depth-First Search
5. Summary

Omega-Regular Properties

finite transition system \( T \)

\( \omega \)-regular property \( E \)

model checking

does \( T \models E \) hold?

yes

no + error indication
**ω-Regular Properties**

**Definition: ω-regular language**
The set $\mathcal{L}$ of infinite works over the alphabet $\Sigma$ is $\omega$-regular if $\mathcal{L} = L_\omega(G)$ for some $\omega$-regular expression $G$ over $\Sigma$.

**Definition: ω-regular properties**
LT property $E$ over $AP$ is $\omega$-regular if $E$ is an $\omega$-regular language over $2^{AP}$.

This is equivalent to:

LT property $E$ over $AP$ is $\omega$-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 $\omega$-regular property
  - $\Phi^\omega$ describes $E$ with invariant condition $\Phi$

- Any regular safety property $E$ is an $\omega$-regular property
  - $\overline{E} = \text{BadPref}(E).\overline{\{a\}^\omega}$ is $\omega$-regular
  - and $\omega$-regular languages are closed under complement

- Let $\Sigma = \{a, b\}$ Then:
  - Infinitely often $a$:
    $$\left((\emptyset + \{b\})^*.(\{a\} + \{a, b\})\right)^\omega$$
  - Eventually $a$:
    $$\left(2^{AP}^*.\{\{a\} + \{a, b\}\} \cdot \overline{\{a\}^\omega}\right)$$

---

**Overview**

1. Omega-Regular Properties
2. Refresher: Büchi Automata
3. Verifying Omega-Regular Safety Properties
4. Nested Depth-First Search
5. Summary

**Nondeterministic Büchi automata**

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

- $Q$ is a finite set of states
- $\Sigma$ is an alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is a transition function
- $Q_0 \subseteq Q$ a set of initial states
- $F \subseteq Q$ 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 $q_i \xrightarrow{A_i} q_{i+1}$ 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 $\mathfrak{A}$:
  $$\mathcal{L}_\omega(\mathfrak{A}) = \{w \in \Sigma^\omega \mid \mathfrak{A} \text{ has an accepting run for } w\}$$
- NBA $\mathfrak{A}$ and $\mathfrak{A}'$ are equivalent if $\mathcal{L}_\omega(\mathfrak{A}) = \mathcal{L}_\omega(\mathfrak{A}')$

Examples

NBA and $\omega$-Regular Languages

Theorem

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

Proof.

Previous lecture.
Verifying Omega-Regular Safety Properties

Peterson’s Transition System

If a thread wants to update the account, does it ever get the opportunity to do so?
“always (reqL ⇒ eventually @accountL) ∧ always (reqR ⇒ eventually @accountR)”

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:

  \[ \text{Traces}(TS_{Pet}) \cap L_\omega(E_{live}) = \emptyset? \]

  infinite traces

  Intersection, complementation and emptiness of Büchi automata accept infinite words

Basic Idea

\[
TS \not\models E \quad \text{if and only if} \quad \text{Traces}(TS) \not\subseteq E \\
\quad \text{if and only if} \quad \text{Traces}(TS) \cap (2^{\mathcal{AP}})^\omega \setminus E \neq \emptyset \\
\quad \text{if and only if} \quad \text{Traces}(TS) \cap \overline{E} \neq \emptyset \\
\quad \text{if and only if} \quad \text{Traces}(TS) \cap \bigcup_{i=0}^{\infty} (A_i) \neq \emptyset \\
\quad \text{if and only if} \quad TS \otimes \mathcal{A} \not\models \text{“eventually for ever” } \neg F \\
\text{persistence property}
\]

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

Persistence Property

Definition: persistence property

A persistence property over \( AP \) is an LT property \( E_{pers} \subseteq (2^{\mathcal{AP}})^\omega \) of the form “eventually for ever \( \Phi \)” for some propositional logic formula \( \Phi \) over \( AP \):

\[
E_{pers} = \{ A_0A_1A_2 \ldots \in (2^{\mathcal{AP}})^\omega \mid \exists i \geq 0. \forall j \geq i. A_j \models \Phi \}
\]

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

“\( \Phi \) is an invariant after a while”
Problem Statement

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

How to establish whether \( TS \models E \)?
### Synchronous Product

**Definition: synchronous product of TS and NBA**

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

$$TS \otimes A = (S', Act, \rightarrow', I', AP', L')$$

where

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

### Proof

**Theorem**

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

$$TS \models E \iff \text{Traces}(TS) \cap L_ω(A) = \emptyset \iff TS \otimes A \models \text{eventually forever} \neg F$$

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

### Example (1)

![Diagram](image_url)

**LT property:** “infinitely often green”

**NBA A for the complement “from some moment on ¬green”**

**atomic propositions**

- $AP' = \{ q_0, q_F, q_1 \}$
- obvious labeling function
- $T \otimes A \models \text{“eventually forever} \neg F’$
Example (2)

Let

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

$TS \not\models E$

if and only if

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

if and only if

there exists a non-trivial reachable SCC $C$ with $C \cap \{ s \in S | s \not\models \neg \Phi \} \neq \emptyset$

Persistence Checking and Cycle Detection

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

Persistence Checking
**Example SCC Algorithm**

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)

**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: $\Theta(N \cdot (N+M) \log M)$

   ▶ where $N$ is the number of states and $M$ the number of transitions

   ▶ where it is assumed that checking $\Phi$ 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 DFS: Example**
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 \( TS \models \text{eventually forever } \Phi \) is an initial path fragment of the form

\[ s_0 \ldots s_{n-1} s_n s_{n+1} \ldots s_{n+m-1} s_0 \]

for \( m > 0 \).

Using nested depth-first search:

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

1. Omega-Regular Properties
2. Refresher: Büchi Automata
3. Verifying Omega-Regular Safety Properties
4. Nested Depth-First Search
5. Summary

Summary

- Checking a regular safety property $E = $ checking invariant on product
  - with an NFA $A$ for the bad prefixes of $E$
  - "never reach an accept state of $A$"

- Checking an $\omega$-regular property $E = $ checking persistence on a product
  - with an NBA for the complement of $E$
  - "eventually forever no accept state of $A$"

- 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