Overview

1. Regular Safety Properties
2. Refresher: Finite Automata
3. Verifying Regular Safety Properties
4. \( \omega \)-Regular Properties
5. Büchi Automata
6. Outlook

Finite transition system \( T \) and regular safety property \( E \):

Does \( T \models E \) hold?
Safety Properties

**Definition: Safety Property**

LT property $E_{safe}$ over $AP$ is a **safety property** if for all $\sigma \in (2^{AP})^\omega \setminus E_{safe}$:

$$E_{safe} \cap \{ \sigma' \in (2^{AP})^\omega | \bar{\sigma} \text{ is a prefix of } \sigma' \} = \emptyset.$$ 

for some prefix $\bar{\sigma}$ of $\sigma$.

- Path fragment $\bar{\sigma}$ is called a **bad prefix** of $E_{safe}$
- Let $BadPref(E_{safe})$ denote the set of bad prefixes of $E_{safe}$
- $\bar{\sigma} \in E_{safe}$ is **minimal** if no proper prefix of it is in $BadPref(E_{safe})$
- Let $MinBadPref(E_{safe})$ denote the set of minimal bad prefixes of $E_{safe}$

Regular Safety Properties

**Definition: regular safety property**

Safety property $E_{safe}$ is **regular** if $BadPref(E_{safe})$ is a regular language.

Or, equivalently:

Safety property $E_{safe}$ is **regular** if there exists a finite automaton over the alphabet $2^{AP}$ recognizing $BadPref(E_{safe})$
Finite Automata

A nondeterministic finite automaton (NFA) \( \mathcal{A} = (Q, \Sigma, \delta, Q_0, F) \) with:

- \( Q \) is a finite set of states
- \( \Sigma \) 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

Language of an Finite Automaton

- NFA \( \mathcal{A} = (Q, \Sigma, \delta, Q_0, F) \) and finite word \( w = A_1 \ldots A_n \in \Sigma^* \)
- A run for \( w \) in \( \mathcal{A} \) is a finite sequence \( q_0 q_1 \ldots q_n \in Q^* \) such that:
  - \( q_0 \in Q_0 \) and \( q_i \xrightarrow{A_i} q_{i+1} \) for all \( 0 \leq i < n \)
- Run \( q_0 q_1 \ldots q_n \) is accepting if \( q_n \in F \)
- The accepted language of \( \mathcal{A} \):
  \[ L(\mathcal{A}) = \{ w \in \Sigma^* \mid \mathcal{A} \text{ has an accepting run for } w \} \]
  - \( w \in \Sigma^* \) is accepted by \( \mathcal{A} \) if \( \mathcal{A} \) has an accepting run for \( w \)
- NFA \( \mathcal{A} \) and \( \mathcal{A}' \) are equivalent if \( L(\mathcal{A}) = L(\mathcal{A}') \)

Facts about Finite Automata

- They are as expressive as regular languages (Kleene’s theorem)
- They are closed under \( \cup, \cap, \) and complementation
  - NFA \( \mathcal{A} \otimes B \) (= cross product) accepts \( L(\mathcal{A}) \cap L(B) \)
  - Total DFA \( \overline{\mathcal{A}} \) (= swap all accept and normal states) accepts \( L(\overline{\mathcal{A}}) = \Sigma^* \setminus L(\mathcal{A}) \)
- They are closed under determinization (= powerset construction)
  - although at an exponential cost ...
- \( L(\mathcal{A}) = \emptyset \)? = check for a reachable accept state in NFA \( \mathcal{A} \)
  - this can be done using a classical depth-first search
  - in linear-time complexity in the size of \( \mathcal{A} \)
- For regular language \( \mathcal{L} \) there is a unique minimal DFA accepting \( \mathcal{L} \)

Regular Safety Properties Revisited

Definition: regular safety property

Safety property \( E_{\text{safe}} \) is regular if \( \text{BadPref}(E_{\text{safe}}) \) is a regular language.

Or, equivalently:

- if there exists a regular expression \( D \) over \( 2^{AP} \) with \( L(D) = \text{BadPref}(E_{\text{safe}}) \)

Or, equivalently:

Safety property \( E_{\text{safe}} \) is regular if there exists

an NFA \( \mathcal{A} \) over the alphabet \( 2^{AP} \) with \( L(\mathcal{A}) = \text{BadPref}(E_{\text{safe}}) \)

Or, equivalently:

\[ \ldots \text{ if there exists a DFA } \mathcal{A} \text{ over } 2^{AP} \text{ with } L(\mathcal{A}) = \text{BadPref}(E_{\text{safe}}) \]
Sets as Formulas

Let $\mathfrak{A} = (Q, \Sigma, \delta, Q_0, F)$ over the alphabet $\Sigma = 2^{AP}$.

We adopt a shorthand notation for the transitions using propositional logic. If $\Phi$ is a propositional logic formula over $AP$ then:

$$p \Phi q \quad \text{stands for the set of transitions} \quad p \xrightarrow{A} q \quad \text{with} \ A \subseteq AP \text{ and } A \models \Phi$$

where $A \subseteq AP$ such that $A \models \Phi$.

Examples. Let $A = \{ a, b, c \}$. Then:

- $p \quad a \wedge \neg b \rightarrow q$ stands for $\{ p \{ a \} \rightarrow q, p \{ a, c \} \rightarrow q \}$
- $p \quad \text{false} \rightarrow q$ stands for $\{ p \emptyset \rightarrow q \}$, i.e., $\{ p \neg (a \lor b \lor c) \rightarrow q \}$
- $p \quad \text{true} \rightarrow q$ stands for $\{ p \xrightarrow{A} q \mid \forall A \subseteq AP \}$

Examples

- Every invariant (over $AP$) is a regular safety property
  - bad prefixes are of the form $\Phi^* (\neg \Phi) \text{true}^*$
  - where $\Phi$ is the invariant condition
- A regular safety property which is not an invariant:
  - “a red light is immediately preceded by a yellow light”
- A non-regular safety property:
  - “the # inserted coins is at least the # of dispensed drinks”

Example

“Every red phase is preceded by a yellow phase”
set of all infinite words $A_0 A_1 A_2 \ldots$ s.t. for all $i \geq 0$:

- $red \in A_i \implies i \geq 1$ and $yellow \in A_{i-1}$

DFA for all (possibly non-minimal) bad prefixes
Overview

1. Regular Safety Properties
2. Refresher: Finite Automata
3. Verifying Regular Safety Properties
4. \( \omega \)-Regular Properties
5. Büchi Automata
6. Outlook

### Büchi Automata

#### Verifying Regular Safety Properties

Peterson’s Algorithm

\[ P_1 \]

```
loop forever
  (* non-critical actions *)
  \( b_1 := \text{true}; x := 2 \); (* request *)
  wait until \( (x = 1 \lor \neg b_2) \)
  do critical section od
  \( b_2 := \text{false} \) (* release *)
  (* non-critical actions *)
end loop
```

\( b_i \) is true if and only if process \( P_i \) is waiting or in critical section

if both threads want to enter their critical section, \( x \) decides who gets access.

### Accessing a Bank Account

Thread Left behaves as follows:

```
while true {
  ....
  nc : \( (b_1, x = \text{true}, 2;) \)
  wt : wait until \( x == 1 || \neg b_2 \) \{ \\
  cs : ....@account ....
  \}
  b_1 = \text{false};
  ....
}
```

Thread Right behaves as follows:

```
while true {
  ....
  nc : \( (b_2, x = \text{true}, 1;) \)
  wt : wait until \( x == 2 || \neg b_1 \) \{
  cs : ....@account ....
  \}
  b_2 = \text{false};
  ....
}
```

Does only one thread at a time has access to the bank account?

### Peterson’s Transition System

Manual inspection reveals that mutual exclusion is guaranteed.
Verifying Mutual Exclusion

- Mutual exclusion = no simultaneous access to the account

- Bad prefix NFA $\mathcal{A}$:

<table>
<thead>
<tr>
<th>$q_0$</th>
<th>$\neg \text{crit}_1 \lor \neg \text{crit}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
</tr>
</tbody>
</table>

- Checking mutual exclusion:

$$\text{Traces}_{\text{fin}}(TS_{\text{Pet}}) \cap \overline{\text{BadPref}(E_{\text{safe}})} = \emptyset?$$

- Intersection, complementation and emptiness of finite automata accept finite words

Problem Statement

Let

1. $E_{\text{safe}}$ be a regular safety property over $AP$

2. $\mathcal{A}$ be an NFA (or DFA) recognizing the bad prefixes of $E_{\text{safe}}$
   - with $\varepsilon \notin L(\mathcal{A})$
   - otherwise all finite words over $2^{AP}$ are bad prefixes and $E_{\text{safe}} = \emptyset$

3. $TS$ be a finite transition system (over $AP$) without terminal states

How to establish whether $TS \not\models E_{\text{safe}}$?

Verifying Regular Safety Properties

finite transition system $T$

regular safety property $E$

NFA $\mathcal{A}$ for the bad prefixes of $E$

Product: Idea (1)

finite transition system $T = (S, \text{Act}, \rightarrow, S_0, AP, L)$

NFA for bad prefixes $\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$

$$s_0 \xrightarrow{L(s_0)=A_0} q_0 \xrightarrow{L(s_1)=A_1} q_1 \xrightarrow{L(s_2)=A_2} q_2 \cdots \xrightarrow{L(s_n)=A_n} q_n \xrightarrow{q_{n+1}}$$

run for $trace(\hat{\pi})$
Product: Idea (2)

**Synchronous Product**

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

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

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

where

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

**Example**

A Note on Terminal States

It may be safely assumed that $TS \otimes \mathfrak{A}$ has no terminal states

- Although $TS$ has no terminal state, $TS \otimes \mathfrak{A}$ may have one
- This can only occur if $\delta(q, A) = \emptyset$ for some $A \subseteq AP$
- Let NFA $\mathfrak{A}$ with some reachable state $q$ with $\delta(q, A) = \emptyset$

Obtain an equivalent NFA $\mathfrak{A}'$ as follows:

- introduce new state $q_{\text{trap}} \notin Q$
- if $\delta(q, A) = \emptyset$ let $\delta'(q, A) = \{ q_{\text{trap}} \}$
- set $\delta'(q_{\text{trap}}, A) = \{ q_{\text{trap}} \}$ for all $A \subseteq AP$
- keep all other transitions that are present in $\mathfrak{A}$

It follows $L(\mathfrak{A}) = L(\mathfrak{A}')$
Verifying Regular Safety Properties

**Theorem**

Let $TS$ over $AP$, $E_{safe}$ a safety property such that $L(\mathfrak{A}) = \text{BadPref}(E_{safe})$ for some NFA $\mathfrak{A}$. Then:

$$TS \models E_{safe} \iff \text{Traces}_{\text{fin}}(TS) \cap L(\mathfrak{A}) = \emptyset \iff TS \otimes \mathfrak{A} \models \underbrace{\text{always } \neg F}_{\text{invariant}}$$

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

**Proof**

**Example**

**Counterexamples**

For each initial path fragment $(s_0, q_1) \ldots (s_n, q_{n+1})$ of $TS \otimes \mathfrak{A}$:

$q_1, \ldots, q_n \notin F$ and $q_{n+1} \in F \Rightarrow \text{trace}(s_0, s_1, \ldots, s_n) \in L(\mathfrak{A})$.

bad prefix for $E_{safe}$
Complexity of Verifying Regular Safety Properties

The time and space complexity of checking $TS \models E_{safe}$ is in $O(|TS| \cdot |A|)$ where $A$ is an NFA with $L(A) = BadPref(E_{safe})$ and $|A|$ is the size of $A$.

The size of NFA $A$ is the number of states and transitions in $A$:

$$|A| = |Q| + \sum_{q \in Q} \sum_{A \in \Sigma} |\delta(q, A)|$$

Peterson’s Transition System

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

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:
  
  $$Traces(TS_{pet}) \cap L(E_{live}) = \emptyset?$$

  infinite traces

- Intersection, complementation and emptiness of Büchi automata accept infinite words
**ω-Regular Expressions: Syntax**

**Definition: ω-regular expression**

An ω-regular expression $G$ over the alphabet $Σ$ has the form:

$$G = E_1.F_1^ω + \ldots + E_n.F_n^ω$$

for $n \in \mathbb{N}_{>0}$

where $E_i, F_i$ are regular expressions over $Σ$ with $ε \notin L(F_i)$.

- ω-Regular expressions denote languages of infinite words
- Examples over the alphabet $Σ = \{A, B\}$:
  - language of all words with infinitely many $A$s: $(B^*.A)^ω$
  - language of all words with finitely many $A$s: $(A + B)^*.B^ω$
  - the empty language $∅^ω$

**ω-Regular Properties**

**Definition: ω-regular language**

The set $L$ of infinite works over the alphabet $Σ$ is ω-regular if $L = L^ω(G)$ 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 $2^AP$.

We will see that 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 $2^AP$).

But not by a deterministic Büchi automaton.

**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
  - $E = BadPref(E).2^ω$ is ω-regular
  - and ω-regular languages are closed under complement

- Let $Σ = \{a, b\}$ Then:
  - Infinitely often $a$:
    - $((∅ + \{b\})^*.\{(a) + \{a, b\})^ω$
  - Eventually $a$:
    - $2^ω.(\{(a) + \{a, b\})\cdot2^ω$
**Shorthand Notation**

*Examples for $AP = \{a, b\}$*

- invariant with invariant condition $a \lor \neg b$
  \[(a \lor \neg b)^\omega \equiv (\emptyset + \{a\} + \{a, b\})^\omega\]
- “infinitely often $a$”
  \[(\neg a^* . a)^\omega \equiv ((\emptyset + \{b\})^* . \{a\}^* + \{a, b\})^\omega\]
- “from some moment on $a$”:
  \[true^* . a^\omega\]
- “whenever $a$ then $b$ will hold somewhen later”
  \[(\neg a^* . a . true^* . b)^* . (\neg a)^\omega + (\neg a^* . a . true^* . b)^\omega\]

**Overview**

1. Regular Safety Properties
2. Refresher: Finite Automata
3. Verifying Regular Safety Properties
4. $\omega$-Regular Properties
5. Büchi Automata
6. Outlook

**Verifying $\omega$-Regular Properties**

- Finite transition system $T$
- $\omega$-regular property $E$
- NBA $A$ for the bad behaviors, i.e., for $(2^{AP})^\omega \setminus E$

- **Persistence Checking**
  \[T \otimes A \models \text{“eventually forever } \neg F\]
  - yes
  - no + error indication

**Julius Richard Büchi**

Julius Richard Büchi (1924 – †1984)
Nondeterministic Büchi automata

Definition: Nondeterministic Büchi automaton

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

- $Q$ is a finite set of states
- $\Sigma$ 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 an Büchi Automaton

- NBA $A = (Q, \Sigma, \delta, Q_0, F)$ and infinite word $w = A_1 A_2 \ldots \in \Sigma^\omega$
- A run for $w$ in $A$ 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 $A$:
  \[ L_\omega(A) = \{ w \in \Sigma^\omega \mid A \text{ has an accepting run for } w \} \]
- $w \in \Sigma^*$ is accepted by $A$ if $A$ has an accepting run for $w$
- NBA $A$ and $A'$ are equivalent if $L_\omega(A) = L_\omega(A')$

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 for LT Properties
Deterministic Büchi Automata

**Definition: Deterministic Büchi automaton**

A Büchi automaton $\mathfrak{A}$ is deterministic if

$$|Q_0| \leq 1 \quad \text{and} \quad |\delta(q, A)| \leq 1 \quad \text{for all } q \in Q \text{ and } A \in \Sigma.$$ 

A DBA is total if both inequalities are equalities.

A total DBA has a unique run for each input word.

DBA Are Less Expressive Than NBA

There is no DBA that accepts $\omega((A + B)^*B^\omega)$.

NFA and DFA are equally expressive but NBA and DBA are not!
Büchi Automata

Overview

1. Regular Safety Properties
2. Refresher: Finite Automata
3. Verifying Regular Safety Properties
4. \( \omega \)-Regular Properties
5. Büchi Automata
6. Outlook

Büchi Automata

NBA and \( \omega \)-Regular Languages

Theorem

1. For every NBA \( A \), the language \( L_\omega(A) \) is \( \omega \)-regular.
2. For every \( \omega \)-regular language \( L \), there is an NBA \( A \) with \( L = L_\omega(A) \).

Proof.

Next lecture.

Verifying \( \omega \)-Regular Safety Properties

Theorem

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

\[ TS \models E \iff \text{Traces}(TS) \cap L_\omega(A) = \emptyset \iff TS \otimes A \models \text{eventually forever} \neg F \]

where \( F \) stands for

\[ \bigvee_{q \in F} q \]

Proof.

Next lecture.