Overview

Introduction
Modelling parallel systems
Linear Time Properties
Regular Properties
Linear Temporal Logic
Computation-Tree Logic
Equivalences and Abstraction
Regular LT properties
Idea: define regular LT properties to be those languages of infinite words over the alphabet $2^{\text{AP}}$ that have a representation by a finite automata.
Regular LT properties

Idea: define regular LT properties to be those languages of infinite words over the alphabet $2^{AP}$ that have a representation by a finite automata.

- regular safety properties:
  \textbf{NFA}-representation for the bad prefixes
Idea: define regular LT properties to be those languages of infinite words over the alphabet $2^{AP}$ that have a representation by a finite automata

- regular safety properties:
  \textbf{NFA}-representation for the bad prefixes

- other regular LT properties:
  representation by $\omega$-automata, i.e., acceptors for infinite words
Overview

Introduction
Modelling parallel systems
Linear Time Properties

Regular Properties
  regular safety properties
  \(\omega\)-regular properties
  model checking with Büchi automata

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction
Recall: definition of safety properties

Let $E$ be a LT property over $AP$, i.e., $E \subseteq (2^{AP})^\omega$.

$E$ is called a safety property if for all words

$$\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \setminus E$$

there exists a finite prefix $A_0 A_1 \ldots A_n$ of $\sigma$ such that none of the words $A_0 A_1 \ldots A_n B_{n+1} B_{n+2} B_{n+3} \ldots$ belongs to $E$, i.e.,

$$E \cap \{ \sigma' \in (2^{AP})^\omega : A_0 \ldots A_n \text{ is a prefix of } \sigma' \} = \emptyset$$

Such words $A_0 A_1 \ldots A_n$ are called bad prefixes for $E$.

$$BadPref \overset{\text{def}}{=} \text{set of bad prefixes for } E \subseteq (2^{AP})^+$$
Regular safety properties
Regular safety properties

Let $E \subseteq (2^{AP})^\omega$ be a safety property.

$E$ is called regular iff the language

$BadPref = \text{set of all bad prefixes for } E$

is regular.
Let $E \subseteq (2^{AP})^\omega$ be a safety property.

$E$ is called regular iff the language

$$\text{BadPref} = \text{set of all bad prefixes for } E \subseteq (2^{AP})^+$$

is regular.
Let $E \subseteq (2^{AP})^\omega$ be a safety property.

$E$ is called regular iff the language

$$\text{BadPref} = \text{set of all bad prefixes for } E \subseteq (2^{AP})^+$$

is regular.

$$\text{BadPref} = \mathcal{L}(A) \text{ for some NFA } A \text{ over the alphabet } 2^{AP}$$
Nondeterministic finite automata (NFA)
Nondeterministic finite automata (NFA)

NFA $A = (Q, \Sigma, \delta, Q_0, F)$

- $Q$ finite set of states
- $\Sigma$ alphabet
- $\delta : Q \times \Sigma \rightarrow 2^Q$ transition relation
- $Q_0 \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states, also called accept states
Nondeterministic finite automata (NFA)

NFA \( \mathcal{A} = (Q, \Sigma, \delta, Q_0, F) \)

- \( Q \) finite set of states
- \( \Sigma \) alphabet
- \( \delta : Q \times \Sigma \to 2^Q \) transition relation
- \( Q_0 \subseteq Q \) set of initial states
- \( F \subseteq Q \) set of final states, also called accept states

Run for a word \( A_0A_1 \ldots A_{n-1} \in \Sigma^* \):

State sequence \( \pi = q_0 q_1 \ldots q_n \) where \( q_0 \in Q_0 \) and \( q_{i+1} \in \delta(q_i, A_i) \) for \( 0 \leq i < n \)
Nondeterministic finite automata (NFA)

NFA $A = (Q, \Sigma, \delta, Q_0, F)$

- $Q$ finite set of states
- $\Sigma$ alphabet
- $\delta : Q \times \Sigma \rightarrow 2^Q$ transition relation
- $Q_0 \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states, also called accept states

run for a word $A_0A_1 \ldots A_{n-1} \in \Sigma^*$:

state sequence $\pi = q_0q_1 \ldots q_n$ where $q_0 \in Q_0$ and $q_{i+1} \in \delta(q_i, A_i)$ for $0 \leq i < n$

run $\pi$ is called accepting if $q_n \in F$
Nondeterministic finite automata (NFA)

NFA $A = (Q, \Sigma, \delta, Q_0, F)$

- $Q$ finite set of states
- $\Sigma$ alphabet
- $\delta : Q \times \Sigma \rightarrow 2^Q$ transition relation
- $Q_0 \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states, also called accept states

accepted language $L(A) \subseteq \Sigma^*$ is given by:

$$L(A) = \text{set of finite words over } \Sigma \text{ that have an accepting run in } A$$
Nondeterministic finite automata (NFA)

NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$

- $Q$ finite set of states
- $\Sigma$ alphabet
  - here: $\Sigma = 2^{AP}$
- $\delta : Q \times \Sigma \rightarrow 2^Q$ transition relation
- $Q_0 \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states, also called accept states

accepted language $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ is given by:

$\mathcal{L}(\mathcal{A}) = \text{set of finite words over } \Sigma \text{ that have an accepting run in } \mathcal{A}$
Notations in pictures for NFA

- **Initial state**: arrow pointing out from a circle
- **Nonfinal state**: circle
- **Final state**: square

Diagram:

- **Vertices**: $q_0$, $q_F$
- **Edges**:
  - $q_0$ to $q_F$ with label $B$
  - $q_F$ to $q_0$ with label $A$
  - $q_0$ to itself labeled $B$
  - $q_F$ to itself labeled $A$
Notations in pictures for NFA

NFA $\mathcal{A}$ with state space $\{q_0, q_F\}$

- $q_0$ initial state
- $q_F$ final state

alphabet $\Sigma = \{A, B\}$
Notations in pictures for NFA

initial state

nonfinal state

final state

accepted language $\mathcal{L}(A)$:

set of all finite words over $\{A, B\}$
ending with letter $A$
Symbolic notations

for transitions in **NFA** over the alphabet $\Sigma = 2^{AP}$
Symbolic notations

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

*symbolic notation* for the labels of transitions:

If $\Phi$ is a propositional formula over $AP$ then $q \xrightarrow{\Phi} p$ stands for the set of transitions $q \xrightarrow{A} p$ where $A \subseteq AP$ such that $A \models \Phi$
Symbolic notations

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

*symbolic notation* for the labels of transitions:

If $\Phi$ is a propositional formula over $AP$ then

$q \xrightarrow{\Phi} p$ stands for the set of transitions $q \xrightarrow{A} p$

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

Example: if $AP = \{a, b, c\}$ then

$q \xrightarrow{a \land \neg b} p \equiv \{ q \xrightarrow{A} p : A = \{a, c\} \text{ or } A = \{a\} \}$
Symbolic notations

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

symbolic notation for the labels of transitions:

If $\Phi$ is a propositional formula over $AP$ then

$q \xrightarrow{\Phi} p$ stands for the set of transitions $q \xrightarrow{A} p$

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

Example: if $AP = \{a, b, c\}$ then

$q \xrightarrow{a \land \neg b} p \equiv \{ q \xrightarrow{A} p : A=\{a, c\} \text{ or } A=\{a\} \}$

$q \xrightarrow{\text{true}} p \equiv \{ q \xrightarrow{A} p : A \subseteq AP \}$
A safety property $E \subseteq (2^{\text{AP}})^\omega$ is called regular iff

$pref = \text{set of all bad prefixes for } E \subseteq (2^{\text{AP}})^+$

$pref = \mathcal{L}(A)$ for some NFA $A$

over the alphabet $2^{\text{AP}}$

is regular.
A safety property $E \subseteq (2^\text{AP})^\omega$ is called regular iff

$$\text{BadPref} = \text{set of all bad prefixes for } E \subseteq (2^\text{AP})^+$$

$$\text{BadPref} = \mathcal{L}(A) \text{ for some NFA } A$$

over the alphabet $2^\text{AP}$ is regular.

![Diagram](image)

$\mathcal{L}(A)$ for some NFA $A$

$AP = \{a, b\}$
Regular safety properties

A safety property $E \subseteq (2^{AP})^\omega$ is called regular iff

\[ \text{BadPref} = \text{set of all bad prefixes for } E \subseteq (2^{AP})^+ \]

\[ \text{BadPref} = \mathcal{L}(A) \text{ for some NFA } A \]

is regular.

\[ AP = \{a, b\} \]

symbolic notation:

\[ a \land \neg b \equiv \{a\} \]
A safety property \( E \subseteq (2^{AP})^\omega \) is called regular iff

\[
\text{BadPref} = \text{set of all bad prefixes for } E \subseteq (2^{AP})^+ \text{ is regular.}
\]

\[
\text{BadPref} = \mathcal{L}(A) \text{ for some NFA } A
\]

over the alphabet \( 2^{AP} \) is regular.

safety property \( E \): “\( a \land \neg b \) never holds twice in a row”
“Every red phase is preceded by a yellow phase”
“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 $i \geq 0$:

\[ \text{red} \in A_i \iff i \geq 1 \text{ and yellow} \in A_{i-1} \]
Example: regular safety property

“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 $i \geq 0$:

\[ \text{red} \in A_i \implies i \geq 1 \text{ and } \text{yellow} \in A_{i-1} \]

DFA for all (possibly non-minimal) bad prefixes
Example: regular safety property

“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 $i \geq 0$:

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

**DFA** for minimal bad prefixes
Bad prefixes vs minimal bad prefixes

Let $E \subseteq (2^\text{AP})^\omega$ be a safety property.

$\text{BadPref} = \text{set of all bad prefixes for } E$

$\text{MinBadPref} = \text{set of minimal bad prefixes for } E$

Claim: $\text{BadPref}$ is regular $\iff$ $\text{MinBadPref}$ is regular
Bad prefixes vs minimal bad prefixes

Let $E \subseteq (2^{AP})^\omega$ be a safety property.

$\text{BadPref} = \text{set of all bad prefixes for } E$

$\text{MinBadPref} = \text{set of minimal bad prefixes for } E$

Claim: $\text{BadPref}$ is regular $\iff$ $\text{MinBadPref}$ is regular

“$\iff$”: Let $\mathcal{A}$ be an NFA for $\text{MinBadPref}$.
Bad prefixes vs minimal bad prefixes

Let $E \subseteq (2^{AP})^\omega$ be a safety property.

- $\text{BadPref} = \text{set of all bad prefixes for } E$
- $\text{MinBadPref} = \text{set of minimal bad prefixes for } E$

\textbf{Claim:} $\text{BadPref}$ is regular $\iff$ $\text{MinBadPref}$ is regular

$\iff$: Let $\mathcal{A}$ be an NFA for $\text{MinBadPref}$. An NFA $\mathcal{A}'$ for $\text{BadPref}$ is obtained from $\mathcal{A}$ by adding self-loops $p \xrightarrow{\text{true}} p$ to all final states $p$. 
Let $E \subseteq (2^\mathcal{AP})^\omega$ be a safety property.

$\text{BadPref} = \text{set of all bad prefixes for } E$

$\text{MinBadPref} = \text{set of minimal bad prefixes for } E$

Claim: $\text{BadPref}$ is regular $\iff \text{MinBadPref}$ is regular

“$\iff$”: Let $\mathcal{A}$ be an NFA for $\text{MinBadPref}$.

An NFA $\mathcal{A}'$ for $\text{BadPref}$ is obtained from $\mathcal{A}$ by adding self-loops $p \xrightarrow{\text{true}} p$ to all final states $p$.

“$\implies$”: Let $\mathcal{A}$ be a DFA for $\text{BadPref}$. 

Bad prefixes vs minimal bad prefixes

Let $E \subseteq (2^{AP})^\omega$ be a safety property.

$\text{BadPref} = \text{set of all bad prefixes for } E$

$\text{MinBadPref} = \text{set of minimal bad prefixes for } E$

Claim: $\text{BadPref}$ is regular $\iff$ $\text{MinBadPref}$ is regular

“$\iff$”: Let $A$ be an NFA for $\text{MinBadPref}$.

An NFA $A'$ for $\text{BadPref}$ is obtained from $A$ by adding self-loops $p \xrightarrow{\text{true}} p$ to all final states $p$.

“$\implies$”: Let $A$ be a DFA for $\text{BadPref}$.

A DFA $A'$ for $\text{MinBadPref}$ is obtained from $A$ by removing all outgoing transitions of final states.
Every **invariant** is regular.
Correct or wrong?

Every \textit{invariant} is regular.

correct.
Correct or wrong?

Every invariant is regular.

correct.

Let $E$ be an invariant with invariant condition $\Phi$. 
Every invariant is regular.

**Correct.**

Let $E$ be an invariant with invariant condition $\Phi$

is a DFA for the language of all bad prefixes
Correct or wrong?

Every invariant is regular.

correct.

Let $E$ be an invariant with invariant condition $\Phi$

is a DFA for the language of all minimal bad prefixes
Example: DFA for MUTEX

“The two processes are never simultaneously in their critical sections”
Example: DFA for MUTEX

“The two processes are never simultaneously in their critical sections”

DFA for minimal bad prefixes over the alphabet $2^{AP}$ where $AP = \{\text{crit}_1, \text{crit}_2\}$

$q_0$ $\rightarrow$ $q_1$

$\neg\text{crit}_1 \lor \neg\text{crit}_2$

$\text{crit}_1 \land \text{crit}_2$
Correct or wrong?

Every safety property is regular.
Correct or wrong?

Every safety property is regular.

wrong.
Correct or wrong?

Every safety property is regular.

wrong. e.g., $AP = \{\text{pay, drink}\}$

$$E = \text{set of all infinite words } A_0 A_1 A_2 \ldots \in (2^AP)^\omega$$

such that for all $j \in \mathbb{N}$:

$$\left| \{i \leq j : \text{pay} \in A_i \} \right| \geq \left| \{i \leq j : \text{drink} \in A_i \} \right|$$
Correct or wrong?

Every safety property is regular.

Wrong. e.g., $AP = \{\text{pay, drink}\}$

$E = \text{set of all infinite words } A_0 A_1 A_2 \ldots \in (2^{AP})^\omega$

such that for all $j \in \mathbb{N}$:

$|\{i \leq j : \text{pay} \in A_i\}| \geq |\{i \leq j : \text{drink} \in A_i\}|$

- $E$ is a safety property, but
- the language of (minimal) bad prefixes is \textit{not} regular
Verifying regular safety properties
Verifying regular safety properties

given: finite TS $\mathcal{T}$
regular safety property $E$
(represented by an NFA for its bad prefixes)

question: does $\mathcal{T} \models E$ hold?
Verifying regular safety properties

given: finite TS $\mathcal{T}$
regular safety property $E$
(represented by an NFA for its bad prefixes)

question: does $\mathcal{T} \models E$ hold?

method: relies on an analogy between the tasks:

- checking language inclusion for NFA
- model checking regular safety properties
<table>
<thead>
<tr>
<th>Language inclusion for NFA</th>
<th>Verification of regular safety properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L(A_1) \subseteq L(A_2)$ ?</td>
<td>$Traces(T) \subseteq E$ ?</td>
</tr>
</tbody>
</table>

- $L(A_1) \subseteq L(A_2)$: This checks if the language generated by $A_1$ is included in the language generated by $A_2$.
- $Traces(T) \subseteq E$: This checks if the set of traces of $T$ is included in the set $E$. 

These checks are fundamental in verifying properties in automata theory.
<table>
<thead>
<tr>
<th>language inclusion for NFA</th>
<th>verification of regular safety properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{L}(A_1) \subseteq \mathcal{L}(A_2) \ ? )</td>
<td>( \text{Traces}(T) \subseteq E \ ? )</td>
</tr>
<tr>
<td>check whether ( \mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2)) ) is empty</td>
<td></td>
</tr>
</tbody>
</table>

\( \mathcal{L}(A_1) \subseteq \mathcal{L}(A_2) \) ?

\( \mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2)) \) is empty
<table>
<thead>
<tr>
<th>language inclusion for NFA</th>
<th>verification of regular safety properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$?</td>
<td>$\text{Traces}(T) \subseteq E$?</td>
</tr>
<tr>
<td>check whether $\mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2))$ is empty</td>
<td></td>
</tr>
<tr>
<td>1. complement $\overline{A_2}$, i.e., construct NFA $\overline{A_2}$ with $\mathcal{L}(\overline{A_2}) = \Sigma^* \setminus \mathcal{L}(A_2)$</td>
<td></td>
</tr>
<tr>
<td><strong>language inclusion for NFA</strong></td>
<td><strong>verification of regular safety properties</strong></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>$L(A_1) \subseteq L(A_2)$ ?</td>
<td>$Traces(T) \subseteq E$ ?</td>
</tr>
<tr>
<td>check whether $L(A_1) \cap (\Sigma^* \setminus L(A_2))$ is empty</td>
<td></td>
</tr>
<tr>
<td>1. complement $A_2$, i.e., construct NFA $\overline{A_2}$ with $L(\overline{A_2}) = \Sigma^* \setminus L(A_2)$</td>
<td></td>
</tr>
<tr>
<td>2. construct NFA $\mathcal{A}$ with $L(\mathcal{A}) = L(A_1) \cap L(\overline{A_2})$</td>
<td></td>
</tr>
<tr>
<td>language inclusion for NFA</td>
<td>verification of regular safety properties</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ ?</td>
<td>$\text{Traces}(T) \subseteq E$ ?</td>
</tr>
</tbody>
</table>

check whether
$\mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2))$
is empty

1. complement $A_2$, i.e.,
   construct NFA $\overline{A_2}$ with
   $\mathcal{L}(\overline{A_2}) = \Sigma^* \setminus \mathcal{L}(A_2)$

2. construct NFA $A$ with
   $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(\overline{A_2})$

3. check if $\mathcal{L}(A) = \emptyset$
<table>
<thead>
<tr>
<th>Language inclusion for NFA</th>
<th>Verification of regular safety properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ ?</td>
<td>$\text{Traces}(T) \subseteq E$ ?</td>
</tr>
<tr>
<td>check whether $\mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2))$ is empty</td>
<td>check whether $\text{Traces}_{\text{fin}}(T) \cap \text{BadPref}$ is empty</td>
</tr>
</tbody>
</table>

1. complement $A_2$, i.e.,
   construct NFA $\overline{A_2}$ with
   $\mathcal{L}(\overline{A_2}) = \Sigma^* \setminus \mathcal{L}(A_2)$
2. construct NFA $A$ with
   $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(\overline{A_2})$
3. check if $\mathcal{L}(A) = \emptyset$
<table>
<thead>
<tr>
<th>language inclusion for NFA</th>
<th>verification of regular safety properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ ?</td>
<td>$Traces(T) \subseteq E$ ?</td>
</tr>
<tr>
<td>check whether $\mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2))$ is empty</td>
<td>check whether $Traces_{\text{fin}}(T) \cap \text{BadPref}$ is empty</td>
</tr>
</tbody>
</table>

1. complement $A_2$, i.e., construct NFA $\overline{A_2}$ with $\mathcal{L}(\overline{A_2}) = \Sigma^* \setminus \mathcal{L}(A_2)$

2. construct NFA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(A_1) \cap \mathcal{L}(\overline{A_2})$

3. check if $\mathcal{L}(\mathcal{A}) = \emptyset$

1. construct NFA $\mathcal{A}$ for the bad prefixes $\mathcal{L}(\overline{\mathcal{A}}) = \text{BadPref}$
<table>
<thead>
<tr>
<th>Language inclusion for NFA</th>
<th>Verification of regular safety properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$?</td>
<td>$\text{Traces}(T) \subseteq E$?</td>
</tr>
<tr>
<td>check whether $\mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2))$ is empty</td>
<td>check whether $\text{Traces}_{\text{fin}}(T) \cap \text{BadPref}$ is empty</td>
</tr>
<tr>
<td>1. complement $A_2$, i.e., construct NFA $\overline{A_2}$ with $\mathcal{L}(\overline{A_2}) = \Sigma^* \setminus \mathcal{L}(A_2)$</td>
<td>1. construct NFA $A$ for the bad prefixes $\mathcal{L}(\overline{A}) = \text{BadPref}$</td>
</tr>
<tr>
<td>2. construct NFA $A$ with $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(\overline{A_2})$</td>
<td>2. construct TS $T'$ with $\text{Traces}_{\text{fin}}(T') = \ldots$</td>
</tr>
<tr>
<td>3. check if $\mathcal{L}(A) = \emptyset$</td>
<td></td>
</tr>
</tbody>
</table>
### Language Inclusion for NFA

<table>
<thead>
<tr>
<th>Language Inclusion for NFA</th>
<th>Verification of Regular Safety Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{L}(A_1) \subseteq \mathcal{L}(A_2) ) ?</td>
<td>( \text{Traces}(T) \subseteq E ) ?</td>
</tr>
<tr>
<td>check whether ( \mathcal{L}(A_1) \cap (\Sigma^* \setminus \mathcal{L}(A_2)) ) is empty</td>
<td>check whether ( \text{Traces}_{\text{fin}}(T) \cap \text{BadPref} ) is empty</td>
</tr>
<tr>
<td>1. complement ( A_2 ), i.e., construct NFA ( \overline{A_2} ) with ( \mathcal{L}(\overline{A_2}) = \Sigma^* \setminus \mathcal{L}(A_2) )</td>
<td>1. construct NFA ( A ) for the bad prefixes with ( \mathcal{L}(\overline{A}) = \text{BadPref} )</td>
</tr>
<tr>
<td>2. construct NFA ( A ) with ( \mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(\overline{A_2}) )</td>
<td>2. construct TS ( T' ) with ( \text{Traces}_{\text{fin}}(T') = \ldots )</td>
</tr>
<tr>
<td>3. check if ( \mathcal{L}(A) = \emptyset )</td>
<td>3. invariant checking for ( T' )</td>
</tr>
</tbody>
</table>
Checking regular safety properties

finite transition system $\mathcal{T}$

regular safety property $E$

safety checking

does $\mathcal{T} \models E$ hold?

yes

no
Checking regular safety properties

finite transition system $\mathcal{T}$

regular safety property $E$

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

safety checking

does $\mathcal{T} \models E$ hold? 

yes

no
Checking regular safety properties

finite transition system $\mathcal{T}$

regular safety property $E$

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

safety checking via invariant checking

$\mathcal{T} \otimes \mathcal{A} \models \text{“never final state”}$

yes

no
Checking regular safety properties

finite transition system $\mathcal{T}$

regular safety property $E$

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

safety checking

via invariant checking

$\mathcal{T} \otimes \mathcal{A} \models \text{“never final state”}$

yes

no $\pm$ error indication
Product of a TS and an NFA

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

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

path fragment \( \hat{\pi} \)
Product of a TS and an NFA

finite transition system

\[ T = (S, \text{Act}, \rightarrow, S_0, AP, L) \]

NFA for bad prefixes

\[ \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \]

\[
\begin{align*}
L(s_0) &= A_0 \\
L(s_1) &= A_1 \\
L(s_2) &= A_2 \\
& \vdots \\
L(s_n) &= A_n
\end{align*}
\]

path fragment \( \hat{\pi} \)  

trace
Product of a TS and an NFA

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

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

path fragment \( \hat{\pi} \)

trace

run for \( \text{trace}(\hat{\pi}) \)
Product of a TS and an NFA

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

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

\[ L(s_0) = A_0 \]
\[ L(s_1) = A_1 \]
\[ L(s_2) = A_2 \]
\[ \vdots \]
\[ L(s_n) = A_n \]

\[ \langle s_0, q_1 \rangle \]
\[ \langle s_1, q_2 \rangle \]
\[ \langle s_2, q_3 \rangle \]
\[ \vdots \]
\[ \langle s_n, q_{n+1} \rangle \]

path fragment \( \hat{\pi} \)

trace

path fragm. in product

run for \( trace(\hat{\pi}) \)
Product transition system
Product transition system

\[ T = (S, Act, \rightarrow, S_0, AP, L) \] transition system

\[ A = (Q, 2^{AP}, \delta, Q_0, F) \] NFA
Product transition system

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

\[ A = (Q, 2^{AP}, \delta, Q_0, F) \] NFA

product-TS \[ T \otimes A \overset{\text{def}}{=} (S \times Q, \text{Act}, \rightarrow', S'_0, AP', L') \]
Product transition system

\[ T \quad = \quad (S, \text{Act}, \rightarrow, S_0, AP, L) \quad \text{transition system} \]

\[ \mathcal{A} \quad = \quad (Q, 2^{AP}, \delta, Q_0, F) \quad \text{NFA} \]

product-TS \( T \otimes \mathcal{A} \overset{\text{def}}{=} (S \times Q, \text{Act}, \rightarrow', S'_0, AP', L') \)

\[
\begin{align*}
    s \xrightarrow{\alpha} s' & \quad \land \quad q' \in \delta(q, L(s')) \\
    \langle s, q \rangle \xrightarrow{\alpha}' \langle s', q' \rangle
\end{align*}
\]
Product transition system

\[ \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \quad \text{transition system} \]

\[ \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \quad \text{NFA} \]

product-TS \( \mathcal{T} \otimes \mathcal{A} \overset{\text{def}}{=} (S \times Q, \text{Act}, \rightarrow', S'_0, AP', L') \)

\[
\begin{align*}
s \xrightarrow{\alpha} s' & \land q' \in \delta(q, L(s')) \\
\langle s', q' \rangle & \xrightarrow{\alpha'} \langle s', q' \rangle
\end{align*}
\]

initial states: \( S'_0 = \{ \langle s_0, q \rangle : s_0 \in S_0, q \in \delta(Q_0, L(s_0)) \} \)
Product transition system

\[ \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \] transition system

\[ \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \] NFA

product-TS \( \mathcal{T} \otimes \mathcal{A} \overset{\text{def}}{=} (S \times Q, \text{Act}, \rightarrow', S'_0, AP', L') \)

\[
\begin{align*}
    s & \xrightarrow{\alpha} s' \land q' \in \delta(q, L(s')) \\
    \langle s, q \rangle & \xrightarrow{\alpha} ' \langle s', q' \rangle
\end{align*}
\]

initial states: \( S'_0 = \{ \langle s_0, q \rangle : s_0 \in S_0, q \in \delta(Q_0, L(s_0)) \} \)

for \( P \subseteq Q \) and \( A \subseteq AP \):

\[ \delta(P, A) = \bigcup_{p \in P} \delta(p, A) \]
Product transition system

\[ \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \] transition system

\[ \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \] NFA

product-TS \( \mathcal{T} \otimes \mathcal{A} \) \( \overset{\text{def}}{=} \ (S \times Q, \text{Act}, \rightarrow', S'_0, AP', L') \)

\[
\begin{align*}
\langle s, q \rangle \xrightarrow{\alpha}\langle s', q' \rangle & \land q' \in \delta(q, L(s')) \\
\langle s, q \rangle \xrightarrow{\alpha} & \langle s', q' \rangle
\end{align*}
\]

initial states: \( S'_0 = \{ \langle s_0, q \rangle : s_0 \in S_0, q \in \delta(Q_0, L(s_0)) \} \)

set of atomic propositions: \( AP' = Q \)
Product transition system

\[ \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \] transition system

\[ \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \] NFA

product-TS \( \mathcal{T} \otimes \mathcal{A} \overset{\text{def}}{=} (S \times Q, \text{Act}, \rightarrow', S_0', AP', L') \)

\[ s \xrightarrow{\alpha} s' \land q' \in \delta(q, L(s')) \]

\[ \langle s, q \rangle \xrightarrow{\alpha} \langle s', q' \rangle \]

initial states: \( S_0' = \{ \langle s_0, q \rangle : s_0 \in S_0, q \in \delta(Q_0, L(s_0)) \} \)

set of atomic propositions: \( AP' = Q \)

labeling function: \( L'(\langle s, q \rangle) = \{ q \} \)
Example: product-TS

transition system $\mathcal{T}$ over $AP = \{\text{red, yellow}\}$
Example: product-TS

transition system $\mathcal{T}$ over
$AP = \{\text{red}, \text{yellow}\}$

$\mathcal{T}$ satisfies the safety property $E$
“every red phase is preceded by a yellow phase”
Example: product-TS

Transition system $\mathcal{T}$ over $AP = \{red, yellow\}$ satisfies the safety property $E$

"every red phase is preceded by a yellow phase"

DFA $A$ for the bad prefixes for $E$
Example: product-TS

\[ T \otimes A \]

(4 * 3 = 12 states)
Example: product-TS

\[ L(\text{green}) = \emptyset \]
Example: product-TS

Initial state:
\[ \langle \text{green}, \delta(q_0, \emptyset) \rangle = q_0 \]
Example: product-TS

lifting the transition

\[ \text{green} \rightarrow \text{yellow} \]
Example: product-TS

\[
\begin{align*}
green &\rightarrow \text{red} \\
\text{yellow} &\rightarrow \text{red/yellow} \\
green &\rightarrow \text{yellow} \\
\text{red/yellow} &\rightarrow \text{yellow} \\
\end{align*}
\]

lifting the transition
\[
\langle \text{green}, q_0 \rangle \rightarrow \langle \text{yellow}, ? \rangle
\]
Example: product-TS

lifting the transition

\[ \langle \text{green}, q_0 \rangle \]
\[ \rightarrow \]
\[ \langle \text{yellow}, \delta(q_0, \{\text{yellow}\}) \rangle \]
\[ = q_1 \]
Example: product-TS

lifting the transition

\[ \langle \text{yellow}, q_1 \rangle \]

\[ \langle \text{red}, \delta(q_1, \{ \text{red} \}) \rangle \]

\[ = q_0 \]
Example: product-TS

lifting the transition
red $\rightarrow$ red/yellow

$\langle \text{red}, q_0 \rangle$  
$\downarrow$  
$\langle \text{red/yellow}, \delta(q_0, \emptyset) \rangle$  
$= q_0$
Example: product-TS

lifting the transition

\[
\text{red/yellow} \rightarrow \text{green}
\]

\[
\langle \text{red/yellow}, q_0 \rangle
\]

\[
\langle \text{green}, \delta(q_0, \emptyset) \rangle
\]

\[
= q_0
\]
Example: product-TS

\[ T \otimes A \]

\[ 4 \times 3 = 12 \text{ states, but just 4 reachable states} \]
Example: product-TS

\[ \text{set of propositions} \]

\[ AP' = \{ q_0, q_1, q_F \} \]
Example: product-TS

![Diagram showing transitions between states labeled as red, yellow, green, and set of propositions $AP' = \{q_0, q_1, q_F\}$]

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

invariant condition $\neg q_F$ holds for all reachable states
Technical remark on the product-TS

Definition of the product of

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

- an NFA $\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$

Then the product $\mathcal{T} \otimes \mathcal{A} = (S \times Q, \text{Act}, \rightarrow', \ldots)$ is a TS
definition of the product of

- a transition system \( \mathcal{T} = (S, Act, \rightarrow, S_0, AP, L) \)
  - without terminal states

- an NFA \( \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \)

then the product \( \mathcal{T} \otimes \mathcal{A} = (S \times Q, Act, \rightarrow', \ldots) \) is a TS
Technical remark on the product-TS

Definition of the product of

- a transition system \( \mathcal{T} = (S, Act, \rightarrow, S_0, AP, L) \)

- an NFA \( \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \)

then the product \( \mathcal{T} \otimes \mathcal{A} = (S \times Q, Act, \rightarrow', \ldots) \) is a TS

without terminal states
Technical remark on the product-TS

Definition of the product of

- a transition system $T = (S, Act, \rightarrow, S_0, AP, L)$

without terminal states

- an NFA $A = (Q, 2^{AP}, \delta, Q_0, F)$

then the product $T \otimes A = (S \times Q, Act, \rightarrow', \ldots)$ is a TS

without terminal states

Assumptions on the NFA $A$:
Technical remark on the product-TS

definition of the product of

  • a transition system $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$

  without terminal states

  • an NFA $\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$

then the product $\mathcal{T} \otimes \mathcal{A} = (S \times Q, Act, \rightarrow', \ldots)$ is a TS

  without terminal states

assumptions on the NFA $\mathcal{A}$:

  • $\mathcal{A}$ is non-blocking, i.e.,

    $Q_0 \neq \emptyset \land \forall q \in Q \forall A \in 2^{AP}. \delta(q, A) \neq \emptyset$
Technical remark on the product-TS

definition of the product of

• a transition system \( T = (S, Act, \rightarrow, S_0, AP, L) \)

\[ \text{without terminal states} \]

• an NFA \( \mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F) \)

then the product \( T \otimes \mathcal{A} = (S \times Q, Act, \rightarrow', \ldots) \) is a TS

\[ \text{without terminal states} \]

assumptions on the NFA \( \mathcal{A} \):

• \( \mathcal{A} \) is non-blocking, i.e.,

\[ Q_0 \neq \emptyset \land \forall q \in Q \forall A \in 2^{AP}. \delta(q, A) \neq \emptyset \]

• no initial state of \( \mathcal{A} \) is final, i.e., \( Q_0 \cap F = \emptyset \)
Non-blocking NFA

\[
\text{alphabet } \Sigma = 2^{\text{AP}} \text{ where } \text{AP} = \{a, b\}
\]
Non-blocking NFA

\[ \Sigma = \{a, b\} \]

blocks for input
\[ \{a\} \cap \{a\} \]

alphabet \( \Sigma = 2^{AP} \) where \( AP = \{a, b\} \)
Non-blocking NFA

NFA $\mathcal{A}$ $\ideal{	ext{equivalent}} \mathcal{A}'$

blocks for input $\{a\} \varnothing \{a\}$

add a trap state $\text{stop}$
Non-blocking NFA

NFA $\mathcal{A}$

$\neg a \land b \rightarrow p \rightarrow b \rightarrow u$

$q \rightarrow a \land \neg b$

$r \leftrightarrow b$

$\neg a$

blocks for input

$\{a\} \varnothing \{a\}$

equivalent NFA $\mathcal{A}'$

$\neg a \land b \rightarrow p \rightarrow b \rightarrow u$

$q \rightarrow a \land \neg b$

$r \leftrightarrow b$

$\neg a$

true

stop

add a trap state $\textit{stop}$
Non-blocking NFA

NFA $\mathcal{A}$

Equivalent NFA $\mathcal{A}'$

blocks for input $\{a\} \not\in \{a\}$

non-blocking

true

true

stop
NFA where no initial state is final

NFA $\mathcal{A}$ with $Q_0 \cap F \neq \emptyset$
NFA where no initial state is final

NFA $A$ with $Q_0 \cap F \neq \emptyset \quad \rightsquigarrow \quad$ NFA $A'$ with $Q_0 \cap F = \emptyset$
NFA where no initial state is final

NFA $A$ with $Q_0 \cap F \neq \emptyset \implies$ NFA $A'$ with $Q_0 \cap F = \emptyset$

$L(A') = L(A) \setminus \{\epsilon\}$
NFA where no initial state is final

\[ \text{NFA } \mathcal{A} \text{ with } Q_0 \cap F \neq \emptyset \quad \mapsto \quad \text{NFA } \mathcal{A}' \text{ with } Q_0 \cap F = \emptyset \]

\[
\begin{align*}
\text{NFA } \mathcal{A} & \quad \text{NFA } \mathcal{A}' \\
q_0 & \quad q'_0 \\
a \land b & \quad a \land b \\
b & \quad a \land b \\
r & \quad b \\
\neg a & \quad \neg a \\
\end{align*}
\]

\[ \mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A}) \setminus \{\varepsilon\} \]

\textit{note:} if \( \mathcal{A} \) is an NFA for the bad prefixes of a safety property then

\[ \varepsilon \notin \mathcal{L}(\mathcal{A}) = \text{BadPref} \]
Model checking regular safety properties
Model checking regular safety properties

... via a reduction to invariant checking .....
Model checking regular safety properties

Let $T = (S, Act, \rightarrow, S_0, AP, L)$ be a transition system

$A = (Q, 2^{AP}, \delta, Q_0, F)$ be an NFA

for the bad prefixes of a regular safety property $E$
Model checking regular safety properties

Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be a transition system (without terminal states)

$\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$ be an NFA for the bad prefixes of a regular safety property $E$
Model checking regular safety properties

Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be a transition system (without terminal states)

$\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$ be an NFA for the bad prefixes of a regular safety property $E$ (non-blocking and $Q_0 \cap F = \emptyset$)
Model checking regular safety properties

Let \( \mathcal{T} = (S, Act, \rightarrow, S_0, AP, L) \) be a transition system (without terminal states)

\[ A = (Q, 2^{AP}, \delta, Q_0, F) \]

be an NFA for the bad prefixes of a regular safety property \( E \)
(non-blocking and \( Q_0 \cap F = \emptyset \))

The following statements are equivalent:

(1) \( \mathcal{T} \models E \)

(2) \( \text{Traces}_{\text{fin}}(\mathcal{T}) \cap L(A) = \emptyset \)
Model checking regular safety properties

Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be a transition system (without terminal states)

$\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$ be an NFA

for the bad prefixes of a regular safety property $E$ (non-blocking and $Q_0 \cap F = \emptyset$)

The following statements are equivalent:

(1) $\mathcal{T} \models E$

(2) $\text{Traces}_{\text{fin}}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \emptyset$

(3) $\mathcal{T} \otimes \mathcal{A} \models \text{invariant \ "always } \neg F\text{"}$
Model checking regular safety properties

Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be a transition system (without terminal states)

$\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$ be an NFA for the bad prefixes of a regular safety property $E$ (non-blocking and $Q_0 \cap F = \emptyset$)

The following statements are equivalent:

1. $\mathcal{T} \models E$
2. $\text{Traces}_{\text{fin}}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \emptyset$
3. $\mathcal{T} \otimes \mathcal{A} \models \text{invariant “always } \neg F”$

where “$\neg F$” denotes $\bigwedge_{q \in F} \neg q$
Product transition system

\[ \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, \text{AP}, L) \quad \text{transition system} \]

\[ \mathcal{A} = (Q, 2^{\text{AP}}, \delta, Q_0, F) \quad \text{NFA} \]

product-TS \( \mathcal{T} \otimes \mathcal{A} \overset{\text{def}}{=} (S \times Q, \text{Act, } \rightarrow', S'_0, \text{AP}', L') \)

\[ s \xrightarrow{\alpha} s' \quad \land \quad q' \in \delta(q, L(s')) \]

\[ \langle s, q \rangle \xrightarrow{\alpha} ' \langle s', q' \rangle \]

initial states: \( S'_0 = \{ \langle s_0, q \rangle : s_0 \in S_0, q \in \delta(Q_0, L(s_0)) \} \)

set of atomic propositions: \( \text{AP}' = Q \)

labeling function: \( L'(\langle s, q \rangle) = \{ q \} \)
Example: sequential circuit

![Sequential Circuit Diagram]

\[ \lambda_y = \delta_r = x \oplus r \]
Example: sequential circuit

\[ \lambda_y = \delta_r = x \oplus r \]

initially \( r = 0 \)

over \( AP = \{y\} \)
Example: sequential circuit

\[ \lambda_y = \delta_r = x \oplus r \]

Initially \( r = 0 \)

Over \( AP = \{ y \} \)

Safety property \( E \)

The circuit will never output two ones after each other
Example: sequential circuit

\[ \lambda_y = \delta_r = x \oplus r \]

initially \( r = 0 \)

\[ \mathcal{T} \not\models E \]

safety property \( E \)

The circuit will never output two ones after each other
Example: sequential circuit

\[ \lambda_y = \delta_r = x \oplus r \]

initially \( r = 0 \)

\( T \not\models E \)

error indication, e.g., \( \langle 10 \rangle \langle 01 \rangle \langle 10 \rangle \langle 01 \rangle \langle 10 \rangle \langle 01 \rangle \)

safety property \( E \)

The circuit will never output two ones after each other
Example: sequential circuit

\[
\lambda_y = \delta_r = x \oplus r
\]

initially \( r = 0 \)

transition system \( \mathcal{T} \)

error indication, e.g., \( \langle 10 \rangle \langle 01 \rangle \)

bad prefix: \( \{y\} \{y\} \)

safety property \( E \)

\( The \ circuit \ will \ never \ output \ two \ ones \ after \ each \ other \)
Example: sequential circuit

\[ \lambda_y = \delta_r = x \oplus r \]

initially \( r = 0 \)

\[ \mathcal{T} \not\models E \]

error indication, e.g., \( \langle 10 \rangle \langle 01 \rangle \)

bad prefix: \( \{ y \} \{ y \} \)

safety property \( E \)

The circuit will never output two ones after each other
Example: product-TS

transition system $\mathcal{T}$

safety property $E$

... never two ones in a row ...
Example: product-TS

transition system $\mathcal{T}$

safety property $E$

$\ldots$ never two ones in a row $\ldots$

$\mathcal{T} \otimes A \not\vDash \text{“never } q_F \text{”}$
Example: product-TS

Transition system $T$

- States: 10, 01, 00, 11
- Transitions:
  - 10 → 01, $\{y\}$
  - 00 → 01, $\emptyset$
  - 01 → 11, $\emptyset$

Safety property $E$

- ... never two ones in a row ...

Error indication for $T \otimes A \not\models \text{“never } q_F \text{”}$

- Transition $10q_1 \rightarrow 01q_F$
- Transition $00q_F \rightarrow 10q_F$
- Transition $01q_F \rightarrow 00q_F$
Example: product-TS

transition system $\mathcal{T}$

safety property $E$

... never two ones in a row ...

error indication for $T \otimes A \not\models \text{“never } q_F \text{”}$

error indication for $T \not\models E$

true
Model checking regular safety properties
Model checking regular safety properties

input: finite TS $\mathcal{T}$, NFA $\mathcal{A}$ for the bad prefixes of $E$

output: “yes” if $\mathcal{T} \models E$ otherwise “no”
Model checking regular safety properties

input: finite TS $\mathcal{T}$, NFA $\mathcal{A}$ for the bad prefixes of $E$

output: “yes” if $\mathcal{T} \models E$
otherwise “no”

construct product transition system $\mathcal{T} \otimes \mathcal{A}$
check whether $\mathcal{T} \otimes \mathcal{A} \models \text{“always } \neg F\text{”}$

where $F =$ set of final states in $\mathcal{A}$
Model checking regular safety properties

**input:** finite TS $\mathcal{T}$, NFA $\mathcal{A}$ for the bad prefixes of $E$

**output:** “yes” if $\mathcal{T} \models E$
otherwise “no”

- construct product transition system $\mathcal{T} \otimes \mathcal{A}$
- check whether $\mathcal{T} \otimes \mathcal{A} \models \text{“always } \neg F\text{”}$
  - if so, then return “yes”
  - if not, then return “no”

where $F = \text{set of final states in } \mathcal{A}$
**Model checking regular safety properties**

*input:* finite TS $\mathcal{T}$, NFA $\mathcal{A}$ for the bad prefixes of $E$

*output:* “yes” if $\mathcal{T} \models E$
otherwise “no” + error indication

construct product transition system $\mathcal{T} \otimes \mathcal{A}$
check whether $\mathcal{T} \otimes \mathcal{A} \models \text{“always } \neg F\text{”}$
if so, then return “yes”
if not, then return “no” ← and an error indication

where $F = \text{set of final states in } \mathcal{A}$
Model checking regular safety properties

construct product transition system $T \otimes A$

IF $T \otimes A \models \text{"always } \neg F\text{"}$

THEN return "yes"

ELSE

FI
Model checking regular safety properties

construct product transition system $T \otimes A$

IF $T \otimes A \models \text{"always } \neg F\text{"}$
  THEN return “yes”
ELSE compute a counterexample for $T \otimes A$ and the invariant “always $\neg F$”,

FI
Model checking regular safety properties

construct product transition system $\mathcal{T} \otimes \mathcal{A}$

IF $\mathcal{T} \otimes \mathcal{A} \models \text{“always } \neg \mathcal{F} \text{”}$

THEN return “yes”

ELSE compute a counterexample for $\mathcal{T} \otimes \mathcal{A}$ and the invariant “always $\neg \mathcal{F}$”,

i.e., an initial path fragment in the product $\langle s_0, p_0 \rangle \langle s_1, p_1 \rangle \ldots \langle s_n, p_n \rangle$ where $p_n \in F$

FI
Model checking regular safety properties

construct product transition system \( T \otimes A \)

IF \( T \otimes A \models \text{"always } \neg F \text{"} \)

THEN return “yes”

ELSE compute a counterexample for \( T \otimes A \) and the invariant “always \( \neg F \)”,

i.e., an initial path fragment in the product

\[ \langle s_0, p_0 \rangle \langle s_1, p_1 \rangle \ldots \langle s_n, p_n \rangle \text{ where } p_n \in F \]

return “no” and \( s_0 s_1 \ldots s_n \)
Model checking regular safety properties

construct product transition system $T \otimes A$

IF $T \otimes A \models \text{“always } \neg F\text{”}$

THEN return “yes”

ELSE compute a counterexample for $T \otimes A$ and the invariant “always $\neg F$”,
i.e., an initial path fragment in the product

$\langle s_0, p_0 \rangle \langle s_1, p_1 \rangle \ldots \langle s_n, p_n \rangle$ where $p_n \in F$

return “no” and $s_0 s_1 \ldots s_n$

FI

time complexity: $O(\text{size}(T) \cdot \text{size}(A))$
Correct or wrong?

If $T$ is a finite transition system then $\text{Traces}_{\text{fin}}(T)$ is regular.
If $\mathcal{T}$ is a finite transition system then $\text{Traces}_\text{fin}(\mathcal{T})$ is regular.

correct.
If $\mathcal{T}$ is a finite transition system then $\text{Traces}_{\text{fin}}(\mathcal{T})$ is regular.

correct. $\mathcal{T}$ can be transformed into an NFA.
Correct or wrong?

If $\mathcal{T}$ is a finite transition system then $\text{Traces}_{\text{fin}}(\mathcal{T})$ is regular.

**correct.** $\mathcal{T}$ can be transformed into an NFA.
If $T$ is a finite transition system then $\text{Traces}_{\text{fin}}(T)$ is regular.

correct. $T$ can be transformed into an NFA.
If $\mathcal{T}$ is a finite transition system then $\text{Traces}_{\text{fin}}(\mathcal{T})$ is regular.

**Correct.** $\mathcal{T}$ can be transformed into an **NFA**.
If $T$ is a finite transition system then $\text{Traces}_{\text{fin}}(T)$ is regular.

correct. $T$ can be transformed into an NFA.