Verifying Regular Linear-Time Properties

Lecture #2 of Principles of Model Checking

Joost-Pieter Katoen

Software Modeling and Verification Group

affiliated to University of Twente, Formal Methods and Tools

University of Twente, September 5, 2012

Content of this lecture

- Automata on finite words
  - refresh your memory
- Verifying regular safety properties
  - product construction, counterexamples
- Automata on infinite words
  - (generalised) Büchi automata, ω-regular languages
- Verifying ω-regular properties
  - nested depth first search

Content of this lecture

⇒ Automata on finite words

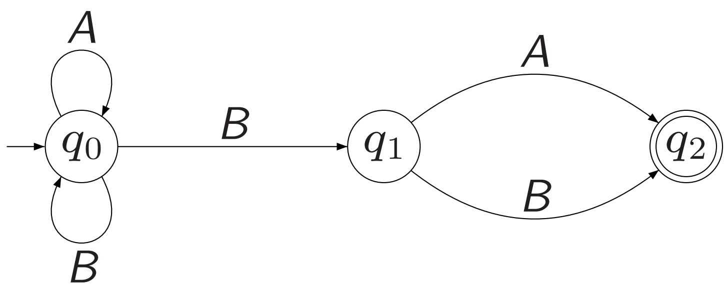
– refresh your memory

- Verifying regular safety properties
  – product construction, counterexamples
- Automata on infinite words
  – (generalised) Büchi automata, ω-regular languages
- Verifying ω-regular properties
  – nested depth first search

Refresh your memory: Finite automata

A *nondeterministic finite automaton* (NFA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, Q_0, F)$ where:

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



Language of an NFA

- NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ and word $w = A_1 \ldots A_n \in \Sigma^*$
- An accepted run for w in $\mathcal{A}$ is a finite sequence $q_0 \, q_1 \, \ldots \, q_n$ such that:
  - $q_0 \in Q_0$ and $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $0 \leqslant i < n$, and $q_n \in F$
- $w \in \Sigma^*$ is accepted by $\mathcal{A}$ if there exists an accepting run for w
- $\mathcal{L}(\mathcal{A}) = \left\{ w \in \Sigma^* \mid \text{ there exists an accepting run for } w \text{ in } \mathcal{A} \right\}$
- NFA $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$

Facts about finite automata

- They are as expressive as regular languages
- They are closed under $\cap$ and complementation
  - NFA $\mathcal{A} \otimes B$ (= cross product) accepts $\mathcal{L}(A) \cap \mathcal{L}(B)$
  - Total DFA $\overline{\mathcal{A}}$ (= swap all accept and normal states) accepts $\overline{\mathcal{L}(A)} = \Sigma^* \setminus \mathcal{L}(\mathcal{A})$
- They are closed under determinization (= removal of choice)
  - although at an exponential cost.....
- $\mathcal{L}(\mathcal{A}) = \varnothing$? = check for a reachable accept state in $\mathcal{A}$
  - this can be done using a simple depth-first search
- For regular language $\mathcal{L}$ there is a unique minimal DFA accepting $\mathcal{L}$

Content of this lecture

- Automata on finite words
  - refresh your memory

⇒ Verifying regular safety properties
  - product construction, counterexamples

- Automata on infinite words
  - (generalised) Büchi automata, ω-regular languages

- Verifying ω-regular properties
  - nested depth first search

Safety properties

- LT property P_{safe} over *AP* is a *safety property* if
  - for all $\sigma \notin P_{safe}$ there exists a finite prefix $\widehat{\sigma}$ of σ such that:

$$P_{safe} \cap \left\{ \sigma' \in \left(2^{AP}\right)^{\omega} \mid \widehat{\sigma} \in \mathit{pref}(\sigma) \right\} \;=\; \varnothing$$

- The set *BadPref* of *bad prefixes* for P_{safe}:

$$\mathit{BadPref}(P_{safe}) \;=\; \left(2^{AP}\right)^{*} \setminus \mathit{pref}(P_{safe})$$

- The set *MinBadPref* of *minimal bad prefixes* for P_{safe}:

$$\mathit{MinBadPref}(P_{safe}) \;=\; \{\, \sigma \in \left(2^{AP}\right)^{*} \mid \mathit{pref}(\sigma) \,\cap\, \mathit{BadPref}(P_{safe}) = \{\, \sigma \,\}\,\}$$

Regular safety properties

- Definition:

  Safety property P_{safe} is regular if *BadPref*(P_{safe}) is a regular language

- Or, equivalently:

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

Some regular safety properties

- Every invariant (over AP) is a regular safety property
  - bad prefixes have form $\Phi^*(\neg\Phi)\text{true}^*$ for invariant condition Φ
  - ... where Φ stands for any $A \subseteq AP$ with $A \models \Phi$
- 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 number of inserted coins is at least the number of dispensed drinks"

Peterson's banking system

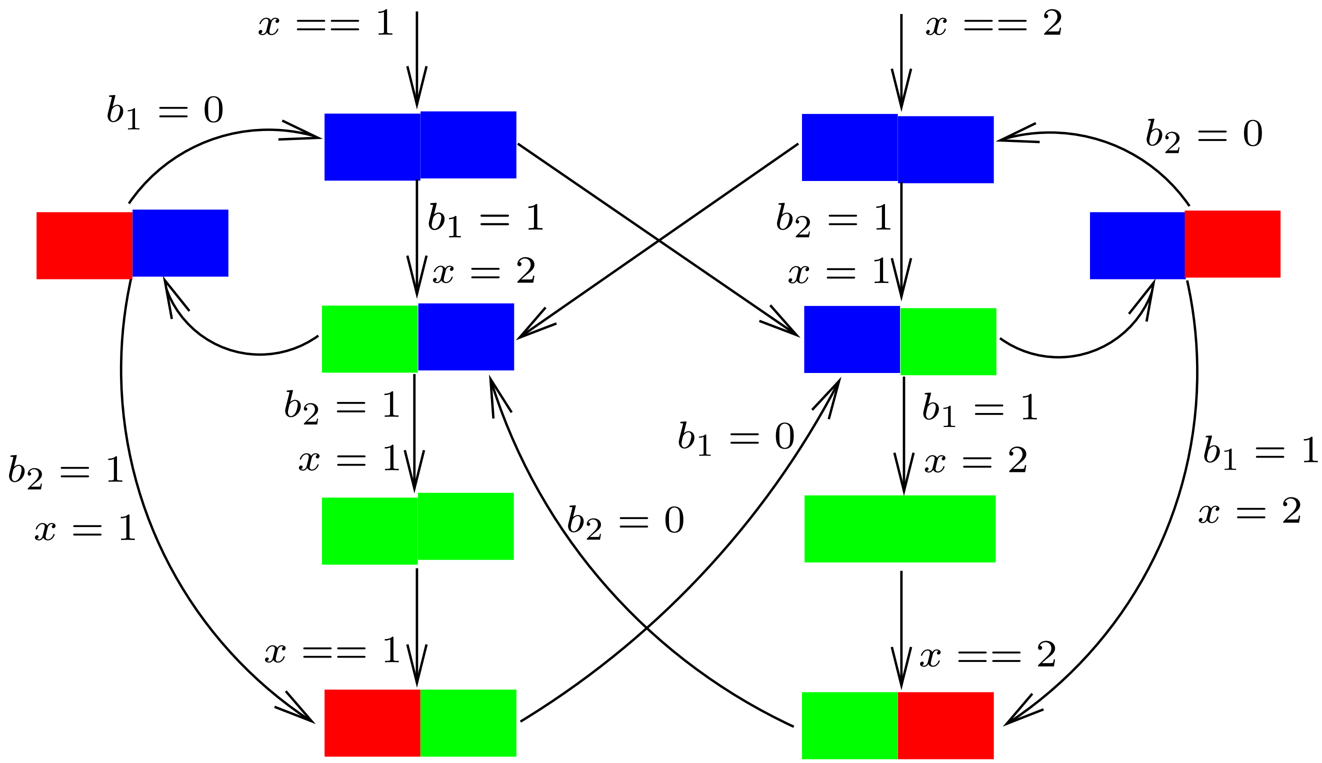
Person Left behaves as follows:

	while true $\{$
	$\ldots\ldots$
rq :	$b_1, x =$ true$, 2;$
wt :	**wait until**$(x == 1 \,||\, \neg\, b_2)\ \{$
cs :	$\ldots$ @account$_L \ldots\}$
	$b_1 =$ false;
	$\ldots\ldots$
	$\}$

Person Right behaves as follows:

	while true $\{$
	$\ldots\ldots$
rq :	$b_2, x =$ true$, 1;$
wt :	**wait until**$(x == 2 \,||\, \neg\, b_1)\ \{$
cs :	$\ldots$ @account$_R \ldots\}$
	$b_2 =$ false;
	$\ldots\ldots$
	$\}$

Is the banking system safe?



Can we guarantee that only one person at a time has access to the bank account?

"always $\neg$ (@account$_L$ $\wedge$ @account$_R$)"

Is the banking system safe?

- Safe = at most one person may have access to the account
- Unsafe: two persons have access to the account simultaneously
  - unsafe behaviour can be characterized by bad prefix
  - alternatively (in this case) by the finite automaton:

- $\mathit{Traces}(\mathit{TS}_{Pet}) \cap \mathit{BadPref}(P_{safe}) = \varnothing$?
  - intersection, complementation and emptiness of languages . . .

Problem statement

Let

- P_{safe} be a *regular* safety property over *AP*
- $\mathcal{A}$ be an NFA recognizing the bad prefixes of P_{safe}
  - assume that $\varepsilon \notin \mathcal{L}(\mathcal{A})$

  $\Rightarrow$ otherwise all finite words over 2^{AP} are bad prefixes and $P_{safe} = \varnothing$
- *TS* be a *finite* transition system (over *AP*) without terminal states

How to establish whether $TS \models P_{safe}$?

Basic idea of the algorithm

$$\mathit{TS} \models P_{safe} \quad \text{if and only if} \quad \mathit{Traces}_{\mathit{fin}}(\mathit{TS}) \cap \mathit{BadPref}(P_{safe}) = \varnothing$$

$$\text{if and only if} \quad \mathit{Traces}_{\mathit{fin}}(\mathit{TS}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$$

$$\text{if and only if} \quad \mathit{TS} \otimes \mathcal{A} \models \text{"always" } \Phi$$

But . . . . . . this amounts to invariant checking on $\mathit{TS} \otimes \mathcal{A}$

$\Rightarrow$ *checking regular safety properties can be done by depth-first search!*

Synchronous product

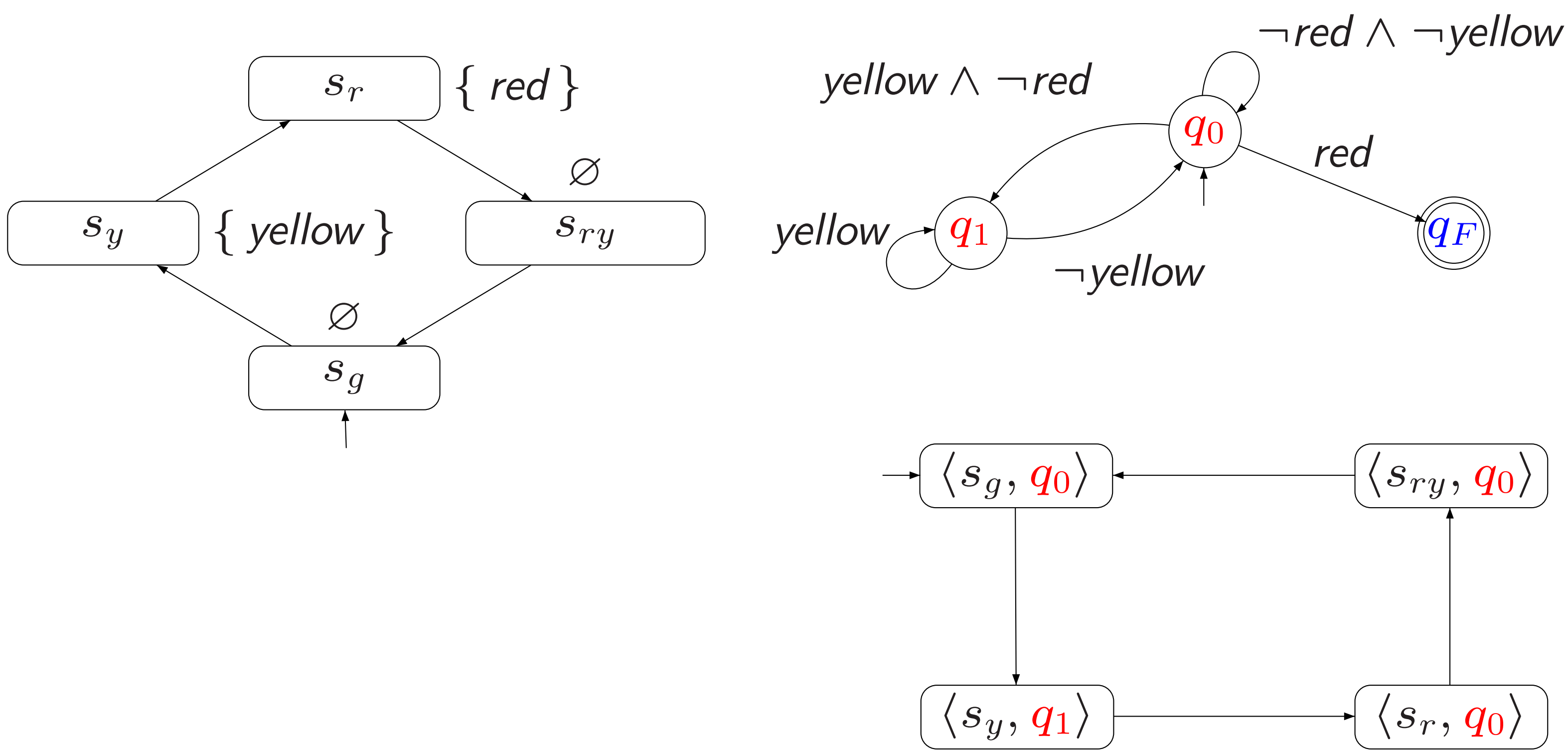
For transition system $\mathit{TS} = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ without terminal states and $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ an NFA with $\Sigma = 2^{\mathit{AP}}$ and $Q_0 \cap F = \varnothing$, let:

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

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

without loss of generality it may be assumed that $\mathit{TS} \otimes \mathcal{A}$ has no terminal states

Example product



Verification of regular safety properties

Let TS over AP, NFA $\mathcal{A}$, and P a regular safety property with $\mathcal{L}(\mathcal{A}) = \mathit{BadPref}(P)$

The following statements are equivalent:

(a) $\mathit{TS} \models P$

(b) $\mathit{Traces}_{\mathit{fin}}(\mathit{TS}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$

(c) $\mathit{TS} \otimes \mathcal{A} \models P_{inv(A)} = \bigwedge_{q \in F} \neg\, q$

Counterexamples

For each initial path fragment $\langle s_0, q_1 \rangle \ldots \langle s_n, q_{n+1} \rangle$ of $\mathit{TS} \otimes \mathcal{A}$:

$$q_1, \ldots, q_n \notin F \text{ and } q_{n+1} \in F \quad \Rightarrow \quad \underbrace{\mathit{trace}(s_0\, s_1 \ldots s_n)}_{\text{bad prefix for } P_{safe}} \in \mathcal{L}(\mathcal{A})$$

Time complexity

The time and space complexity of checking $TS \models P_{safe}$ is in:

$$\mathcal{O}(|TS| \cdot |\mathcal{A}|)$$

where $\mathcal{A}$ is an NFA with $\mathcal{L}(\mathcal{A}) = \mathit{MinBadPref}(P_{safe})$

Content of this lecture

- Automata on finite words
  - refresh your memory
- Verifying regular safety properties
  - product construction, counterexamples

$\Rightarrow$ Automata on infinite words
  - (generalised) Büchi automata, ω-regular languages

- Verifying ω-regular properties
  - nested depth first search

Peterson's banking system

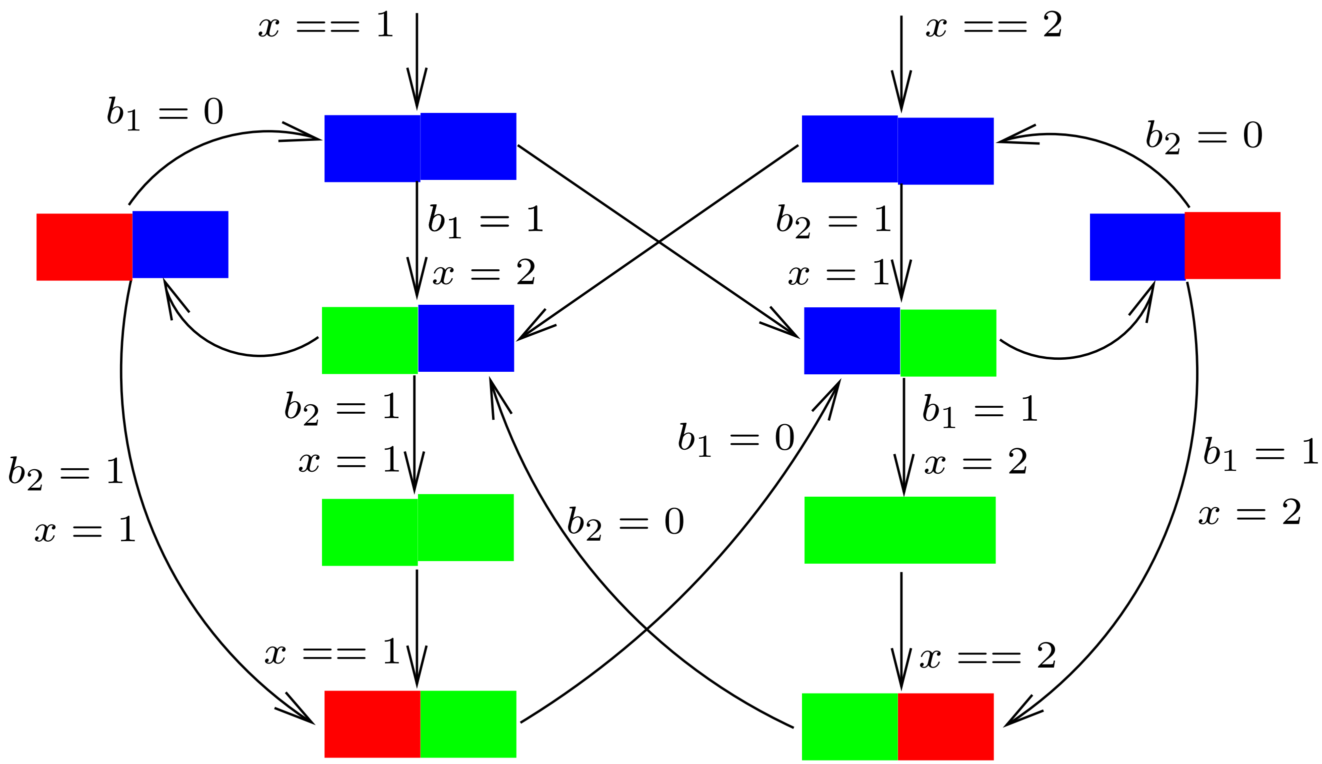
Person Left behaves as follows:

$$
\begin{array}{ll}
 & \textbf{while } \mathsf{true} \;\; \{ \\
 & \quad \ldots\ldots \\
{\color{blue} rq}: & \quad b_1, x = \mathsf{true}, 2; \\
{\color{green} wt}: & \quad \textbf{wait until}(x == 1 \,||\, \neg\, b_2)\, \{ \\
{\color{red} cs}: & \qquad \ldots \mathsf{@account}_L \ldots \} \\
 & \quad b_1 = \mathsf{false}; \\
 & \quad \ldots\ldots \\
 & \}
\end{array}
$$

Person Right behaves as follows:

$$
\begin{array}{ll}
 & \textbf{while } \mathsf{true} \;\; \{ \\
 & \quad \ldots\ldots \\
{\color{blue} rq}: & \quad b_2, x = \mathsf{true}, 1; \\
{\color{green} wt}: & \quad \textbf{wait until}(x == 2 \,||\, \neg\, b_1)\, \{ \\
{\color{red} cs}: & \qquad \ldots \mathsf{@account}_R \ldots \} \\
 & \quad b_2 = \mathsf{false}; \\
 & \quad \ldots\ldots \\
 & \}
\end{array}
$$

Is the banking system live?

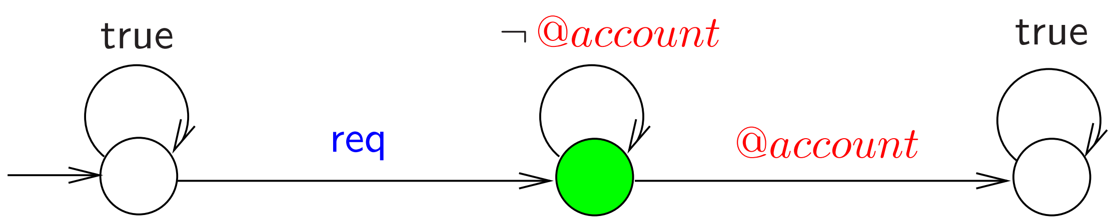


If someone wants to update the account, does he ever get the opportunity to do so?

"always (req_L $\Rightarrow$ eventually @account$_L$) $\wedge$ always (req_R $\Rightarrow$ eventually @account$_R$)"

Is the banking system live?

- Live = when you want access to account, you eventually get it
- Not live: once you want access to the account, you never get it
  - unlive behaviour can be characterized as a (set of) infinite traces
  - or, equivalently, by a Büchi-automaton *Live*:



- Checking liveness: $\mathit{Traces}(\mathit{TS}_{Pet}) \cap \mathcal{L}_\omega(\overline{\mathit{Live}}) = \varnothing$?
  - (explicit) complementation, intersection and emptiness of Büchi automata!

Regular expressions

- Let Σ be an alphabet with $A \in \Sigma$
- Regular expressions over Σ have *syntax*:

$$\mathsf{E} \ ::= \ \underline{\varnothing} \ \mid \ \underline{\varepsilon} \ \mid \ \underline{A} \ \mid \ \mathsf{E} + \mathsf{E}' \ \mid \ \mathsf{E}.\mathsf{E}' \ \mid \ \mathsf{E}^*$$

- The *semantics* of regular expression E is a language $\mathcal{L}(\mathsf{E}) \subseteq \Sigma^*$:

$$\mathcal{L}(\underline{\varnothing}) = \varnothing, \quad \mathcal{L}(\underline{\varepsilon}) = \{\, \varepsilon \,\}, \quad \mathcal{L}(\underline{A}) = \{\, A \,\}$$

$$\mathcal{L}(\mathsf{E}+\mathsf{E}') = \mathcal{L}(\mathsf{E}) \cup \mathcal{L}(\mathsf{E}') \quad \mathcal{L}(\mathsf{E}.\mathsf{E}') = \mathcal{L}(\mathsf{E}).\mathcal{L}(\mathsf{E}') \quad \mathcal{L}(\mathsf{E}^*) = \mathcal{L}(\mathsf{E})^*$$

Syntax of ω-regular expressions

- Regular expressions denote languages of finite words
- ω-Regular expressions denote languages of infinite words
- An ω-*regular expression* G over Σ has the form:

$$\mathsf{G} \;=\; \mathsf{E}_1.\mathsf{F}_1^\omega + \ldots + \mathsf{E}_n.\mathsf{F}_n^\omega \quad \text{for } n > 0$$

where $\mathsf{E}_i, \mathsf{F}_i$ are regular expressions over Σ with $\varepsilon \notin \mathcal{L}(\mathsf{F}_i)$

- Some examples: $(A+B)^*.B^\omega$, $(B^*.A)^\omega$, and $A^*.B^\omega + A^\omega$

Semantics of ω-regular expressions

- For $\mathcal{L} \subseteq \Sigma^*$ let $\mathcal{L}^\omega \;=\; \{\, w_1 w_2 w_3 \ldots \mid \forall i \geqslant 0.\, w_i \in \mathcal{L} \,\}$
- Let ω-regular expression $\mathsf{G} \;=\; \mathsf{E}_1.\mathsf{F}_1^\omega + \ldots + \mathsf{E}_n.\mathsf{F}_n^\omega$
- The *semantics* of G is a language $\mathcal{L}(\mathsf{G}) \subseteq \Sigma^\omega$:

$$\mathcal{L}_\omega(\mathsf{G}) \;=\; \mathcal{L}(\mathsf{E}_1).\mathcal{L}(\mathsf{F}_1)^\omega \;\cup\; \ldots \;\cup\; \mathcal{L}(\mathsf{E}_n).\mathcal{L}(\mathsf{F}_n)^\omega$$

- G_1 and G_2 are *equivalent*, denoted $\mathsf{G}_1 \equiv \mathsf{G}_2$, if $\mathcal{L}_\omega(\mathsf{G}_1) = \mathcal{L}_\omega(\mathsf{G}_2)$

ω-Regular languages

- $\mathcal{L}$ is ω-regular if $\mathcal{L} = \mathcal{L}_\omega(\mathsf{G})$ for some ω-regular expression G
- Examples over $\Sigma = \{ A, B \}$:
  - language of all words with infinitely many As:

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

  - language of all words with finitely many As:

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

  - the empty language

$$\varnothing^\omega$$

- ω-Regular languages are closed under $\cup$, $\cap$, and complementation

ω-regular properties

- Definition:

LT property P over AP is ω-regular if

P is an ω-regular language over the alphabet 2^{AP}

- Or, equivalently:

LT property P over AP is ω-regular if P is a language

accepted by a nondeterministic Büchi automaton over 2^{AP}

Example ω-regular properties

- Any invariant P is an ω-regular property
  - as Φ^ω describes P with invariant condition Φ

- Any regular safety property P is an ω-regular property
  - as $\overline{P} \;=\; \mathit{BadPref}(P).\left(2^{AP}\right)^\omega$ is ω-regular
  - and the fact that ω-regular languages are closed under complement

- Many liveness properties P are ω-regular properties

Nondeterministic Büchi automata

- NFA (and DFA) are incapable of accepting infinite words
- Automata on infinite words
  - suited for accepting ω-regular languages
  - we consider nondeterministic Büchi automata (NBA)
- Accepting runs have to "check" the entire input word $\Rightarrow$ are infinite

$\Rightarrow$ acceptance criteria for infinite runs are needed

- NBA are like NFA, but have a distinct *acceptance criterion*
  - one of the accept states must be visited infinitely often

Language of an NBA

- NBA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ and word $\sigma = A_0 A_1 A_2 \ldots \in \Sigma^\omega$

- An accepted run for σ in $\mathcal{A}$ is an infinite sequence $q_0\, q_1\, q_2 \ldots$ such that:
  - $q_0 \in Q_0$ and $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $0 \leqslant i$, and
  - $q_i \in F$ for infinitely many i

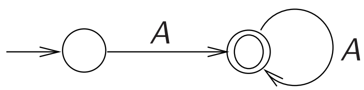
- $\sigma \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if there exists an accepting run for σ

- The accepted language of $\mathcal{A}$:

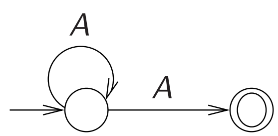
$$\mathcal{L}_\omega(\mathcal{A}) = \big\{ \sigma \in \Sigma^\omega \mid \text{ there exists an accepting run for } \sigma \text{ in } \mathcal{A} \big\}$$

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

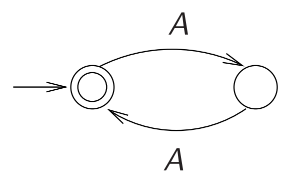
NBA versus NFA



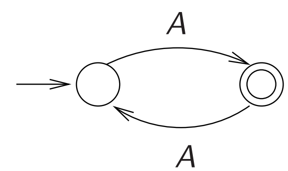
$\mathcal{A}_1$



$\mathcal{A}_2$



$\mathcal{A}_1$



$\mathcal{A}_2$

finite equivalence $\not\Rightarrow$ ω-equivalence

$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$, but $\mathcal{L}_\omega(\mathcal{A}) \neq \mathcal{L}_\omega(\mathcal{A}')$

ω-equivalence $\not\Rightarrow$ finite equivalence

$\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{A}')$, but $\mathcal{L}(\mathcal{A}) \neq \mathcal{L}(\mathcal{A}')$

NBA and ω-regular languages

The class of languages accepted by NBA
agrees with the class of ω-regular languages

(1) any ω-regular language is recognized by an NBA

(2) for any NBA $\mathcal{A}$, the language $\mathcal{L}_\omega(\mathcal{A})$ is ω-regular

For any ω-regular language there is an NBA

- How to construct an NBA for the ω-regular expression:

$$\mathsf{G} \;=\; \mathsf{E}_1.\mathsf{F}_1^{\omega} + \ldots + \mathsf{E}_n.\mathsf{F}_n^{\omega} \;?$$

where E_i and F_i are regular expressions over alphabet Σ with $\varepsilon \notin \mathsf{F}_i$

- Use operators on NBA mimicking operators on ω-regular expressions:

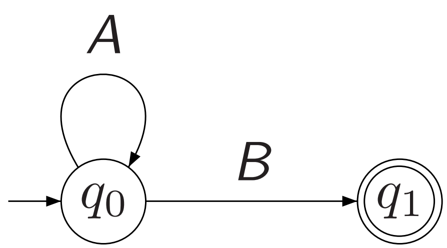
(1) for NBA $\mathcal{A}_1$ and $\mathcal{A}_2$ there is an NBA accepting $\mathcal{L}_\omega(\mathcal{A}_1) \cup \mathcal{L}_\omega(\mathcal{A}_2)$
(2) for any regular language $\mathcal{L}$ with $\varepsilon \notin \mathcal{L}$ there is an NBA accepting $\mathcal{L}^\omega$
(3) for regular language $\mathcal{L}$ and NBA $\mathcal{A}'$ there is an NBA accepting $\mathcal{L}.\mathcal{L}_\omega(\mathcal{A}')$

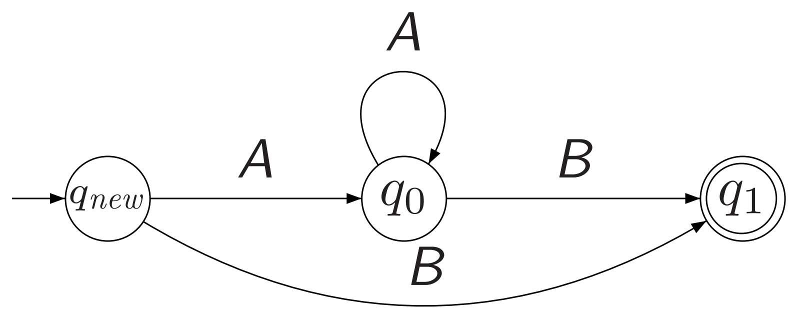
- We will discuss these three operators in detail

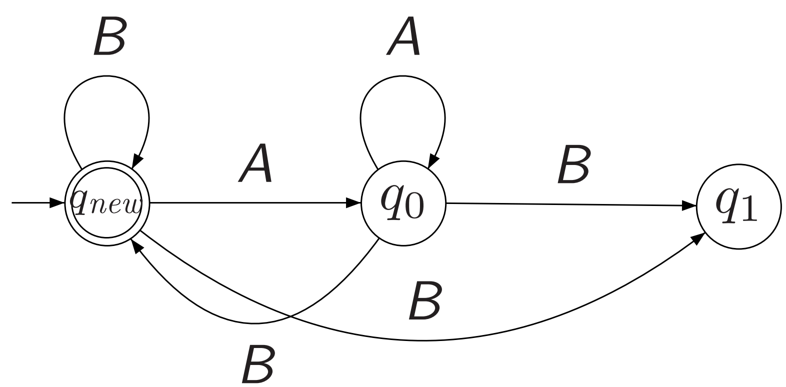
Definition of ω-operator for NFA

- Let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NFA with $\varepsilon \notin \mathcal{L}(\mathcal{A})$.
- Assume no initial state in $\mathcal{A}$ has incoming transitions and $Q_0 \cap F = \varnothing$
  - otherwise introduce a new initial state $q_{new} \notin F$
  - let $q_{new} \xrightarrow{A} q$ iff $q_0 \xrightarrow{A} q$ for some $q_0 \in Q_0$
  - keep all transitions in $\mathcal{A}$
- Construct an NBA $\mathcal{A}' = (Q, \Sigma, \delta', Q_0', F')$ as follows
  - if $q \xrightarrow{A} q' \in F$ then add $q \xrightarrow{A} q_0$ for any $q_0 \in Q_0$
  - keep all transitions in $\mathcal{A}$
  - $Q_0' = Q_0$ and $F' = Q_0$.

Example for ω-operator for NFA







From an NFA accepting A^*B to an NBA accepting $(A^*B)^\omega$

Extended transition function

Extend the transition function δ to $\delta^* : Q \times \Sigma^* \to 2^Q$ by:

$$\delta^*(q, \varepsilon) = \{ q \} \quad \text{and} \quad \delta^*(q, A) = \delta(q, A)$$

$$\delta^*(q, A_1 A_2 \ldots A_n) = \bigcup_{p \in \delta(q, A_1)} \delta^*(p, A_2 \ldots A_n)$$

$\delta^*(q, w)$ = set of states reachable from q for the word w

Checking non-emptiness

$$\mathcal{L}_\omega(\mathcal{A}) \neq \varnothing$$

if and only if

$$\underbrace{\exists q_0 \in Q_0.\ \exists q \in F.\ \exists w \in \Sigma^*.\ \exists v \in \Sigma^+.\ \ q \in \delta^*(q_0, w)\ \wedge\ q \in \delta^*(q, v)}_{\text{there is a reachable accept state on a cycle}}$$

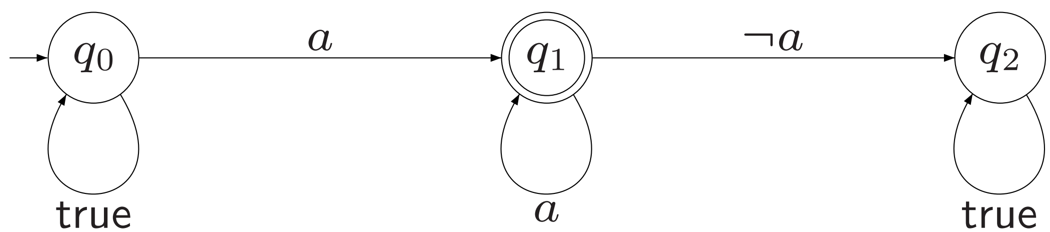
The emptiness problem for NBA $\mathcal{A}$ can be solved in $\mathcal{O}(|\mathcal{A}|)$

NBA are more expressive than DBA

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

There is no DBA that accepts $\mathcal{L}_\omega((A + B)^* B^\omega)$

An LT property requiring nondeterminism



let $\{\ a\ \} = AP$, i.e., $2^{AP} = \{A, B\}$ where $A = \{\}$ and $B = \{a\}$

"eventually for ever a" equals $(A + B)^* B^\omega \ = \ (\{\} + \{a\})^* \{a\}^\omega$

Generalized Büchi automata

- NBA are as expressive as ω-regular languages
- Variants of NBA exist that are equally expressive
  - Muller, Rabin, Streett automata, and eneralized Büchi automata (GNBA)
- GNBA are like NBA, but have a distinct acceptance criterion
  - a GNBA requires to visit several sets $F_1, \ldots, F_k$ $(k \geqslant 0)$ infinitely often
  - for $k{=}0$, all runs are accepting; for $k{=}1$ it behaves like an NBA
- GNBA are useful to relate temporal logic and automata

Generalized Büchi automata

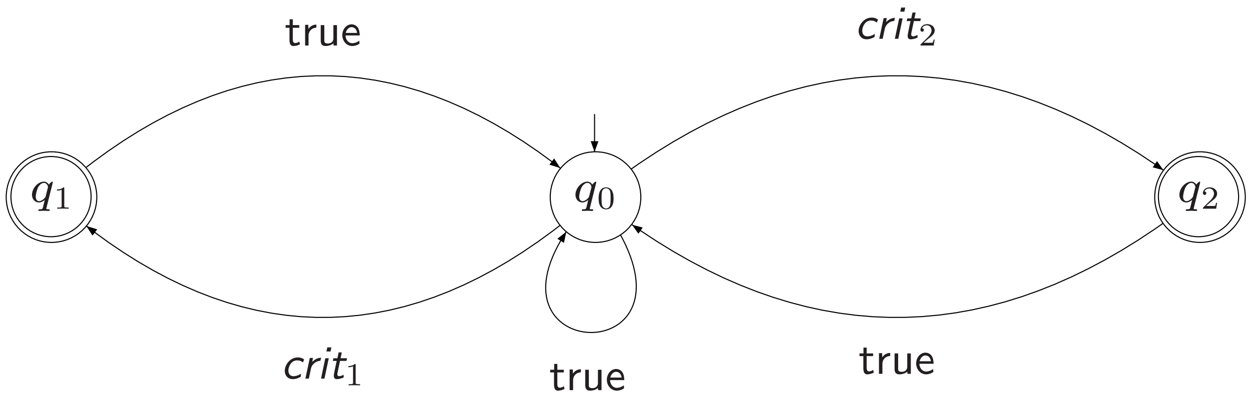
A generalized NBA (GNBA) $\mathcal{G}$ is a tuple $(Q, \Sigma, \delta, Q_0, \mathcal{F})$ where:

- Q, Σ, δ and Q_0 are as before, and
- $\mathcal{F} = \{ F_1, \ldots, F_k \}$ is a (possibly empty) subset of 2^Q

Language of a GNBA

- GNBA $\mathcal{G} = (Q, \Sigma, \delta, Q_0, \mathcal{F})$ and word $\sigma = A_0 A_1 A_2 \ldots \in \Sigma^\omega$
- A accepted run for σ in $\mathcal{G}$ is an infinite sequence $q_0\, q_1\, q_2 \ldots$ such that:
  - $q_0 \in Q_0$ and $q_i \xrightarrow{A_i} q_{i+1}$ for all $0 \leqslant i$, and
  - for all $F \in \mathcal{F}$: $q_i \in F$ for infinitely many i
- $\mathcal{L}_\omega(\mathcal{G}) = \{\sigma \in \Sigma^\omega \mid \text{ there exists an accepting run for } \sigma \text{ in } \mathcal{G}\}$

Example



A GNBA for the property "both processes are infinitely often in their critical section"

$$\mathcal{F} = \{ \{ q_1 \}, \{ q_2 \} \}$$

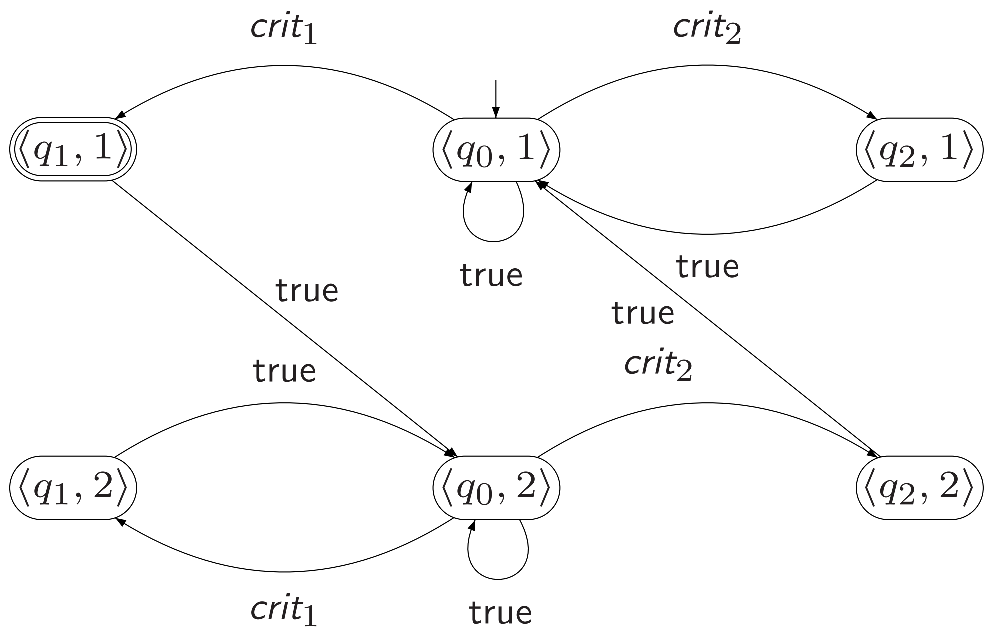
From GNBA to NBA

For any GNBA $\mathcal{G}$ there exists an NBA $\mathcal{A}$ with:

$\mathcal{L}_\omega(\mathcal{G}) = \mathcal{L}_\omega(\mathcal{A})$ and $|\mathcal{A}| = \mathcal{O}(|\mathcal{G}| \cdot |\mathcal{F}|)$

where $\mathcal{F}$ denotes the set of acceptance sets in $\mathcal{G}$

Example


crit1
crit2
⟨q1, 1⟩
⟨q0, 1⟩
⟨q2, 1⟩
true
true
true
true
true
crit2
⟨q1, 2⟩
⟨q0, 2⟩
⟨q2, 2⟩
crit1
true

Content of this lecture

- Automata on finite words
  - refresh your memory
- Verifying regular safety properties
  - product construction, counterexamples
- Automata on infinite words
  - (generalised) Büchi automata, ω-regular languages

$\Rightarrow$ Verifying ω-regular properties
  - nested depth first search

ω-regular properties

- Definition:

LT property P over AP is ω-regular if
P is an ω-regular language over the alphabet 2^{AP}

- Or, equivalently:

LT property P over AP is ω-regular if P is a language
accepted by a nondeterministic Büchi automaton over 2^{AP}

Basic idea of the algorithm

$$
\begin{aligned}
TS \not\models P \quad & \text{if and only if} \quad Traces(TS) \not\subseteq P \\
& \text{if and only if} \quad Traces(TS) \cap \left(2^{AP}\right)^{\omega} \setminus P \neq \varnothing \\
& \text{if and only if} \quad Traces(TS) \cap \overline{P} \neq \varnothing \\
& \text{if and only if} \quad Traces(TS) \cap \mathcal{L}_{\omega}(\mathcal{A}) \neq \varnothing \\
& \text{if and only if} \quad TS \otimes \mathcal{A} \not\models \underbrace{\text{“eventually for ever”} \ \neg F}_{\text{persistence property}}
\end{aligned}
$$

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

Persistence property

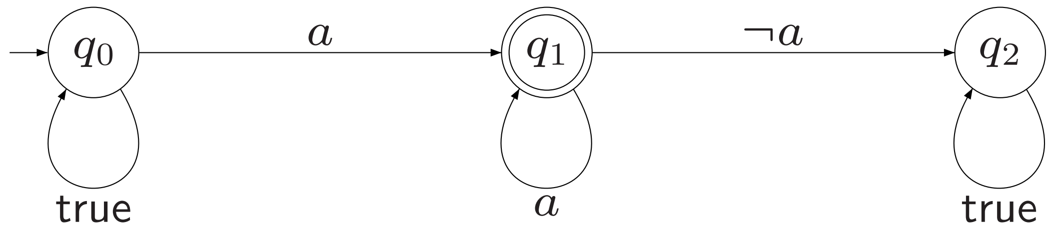
A *persistence property* over *AP* is an LT property $P_{pers} \subseteq \left(2^{AP}\right)^{\omega}$ "eventually for ever Φ" for some propositional logic formula Φ over *AP*:

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

Φ is called a persistence (or state) condition of P_{pers}

"Φ is an invariant after a while"

Example persistence property



let $\{ a \} = AP$, i.e., $2^{AP} = \{A, B\}$ where $A = \{\}$ and $B = \{a\}$

"eventually for ever a" equals $(A + B)^* B^\omega \;=\; (\{\} + \{a\})^* \{a\}^\omega$

Recall synchronous product

For transition system $\mathit{TS} = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ without terminal states and $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ a non-blocking NBA with $\Sigma = 2^{\mathit{AP}}$, let:

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

- $S' = S \times Q$, $\mathit{AP}' = Q$ 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

Let:

- TS be a transition system over AP
- P be an ω-regular property over AP, and
- $\mathcal{A}$ a non-blocking NBA such that $\mathcal{L}_\omega(\mathcal{A}) = \overline{P}$.

The following statements are equivalent:

(a) $TS \models P$

(b) $Traces(TS) \cap \mathcal{L}_\omega(\mathcal{A}) = \varnothing$

(c) $TS \otimes \mathcal{A} \models P_{pers(A)}$

where $P_{pers(A)}$ = *"eventually for ever* $\neg F$*"*

$\Rightarrow$ *checking ω-regular properties is reduced to persistence checking!*

Persistence checking

- Aim: establish whether *TS* $\not\models P_{pers}$ = "eventually for ever Φ"
- Let state s be reachable in *TS* and $s \not\models \Phi$
  - *TS* has an initial path fragment that ends in s
- If s is on a *cycle*
  - this path fragment can be continued by an infinite path
  - ...... by traversing the cycle containing s infinitely often

$\Rightarrow$ *TS* may visit the $\neg\Phi$-state s infinitely often and so: *TS* $\not\models P_{pers}$

- If not such s is found then: *TS* $\models P_{pers}$

Persistence checking and cycle detection

Let

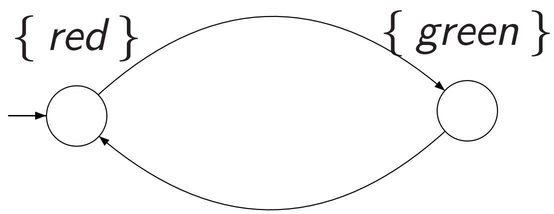
- TS be a finite transition system without terminal states over AP
- Φ a propositional formula over AP, and
- P_{pers} the persistence property "eventually for ever Φ"

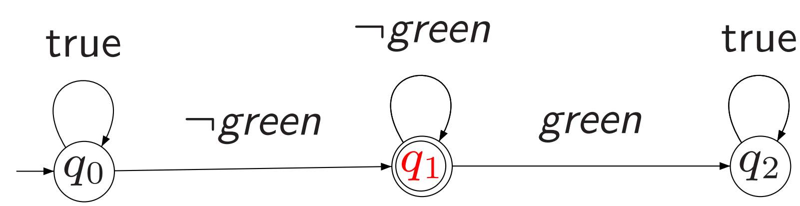
$$\mathit{TS} \not\models P_{pers}$$

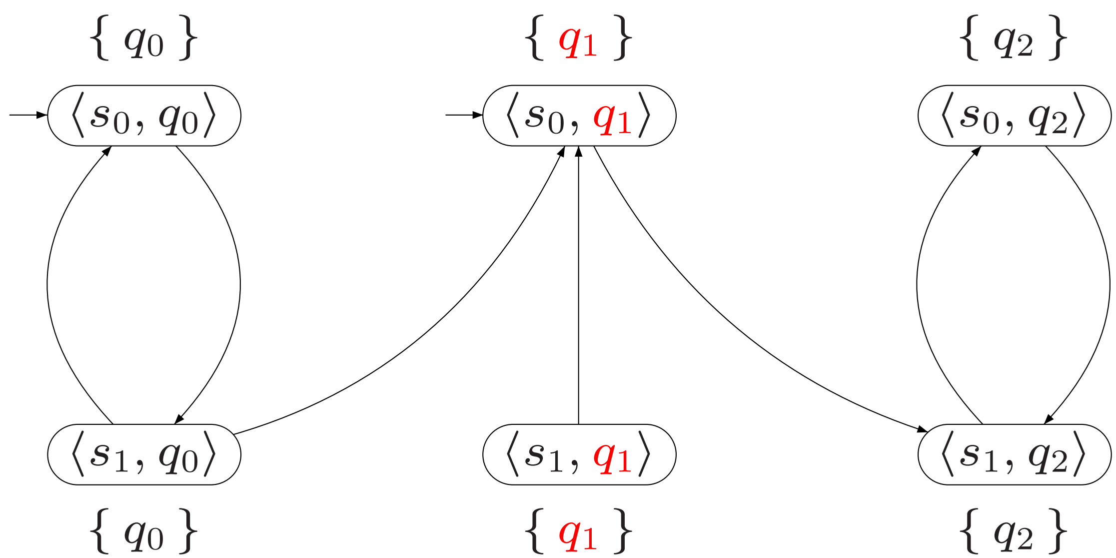
if and only if

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

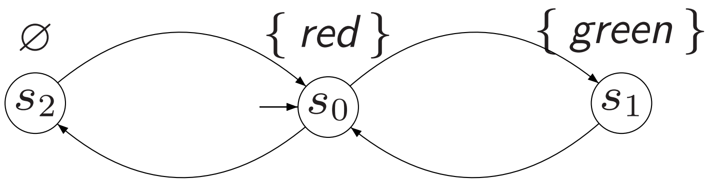
Infinitely often green?

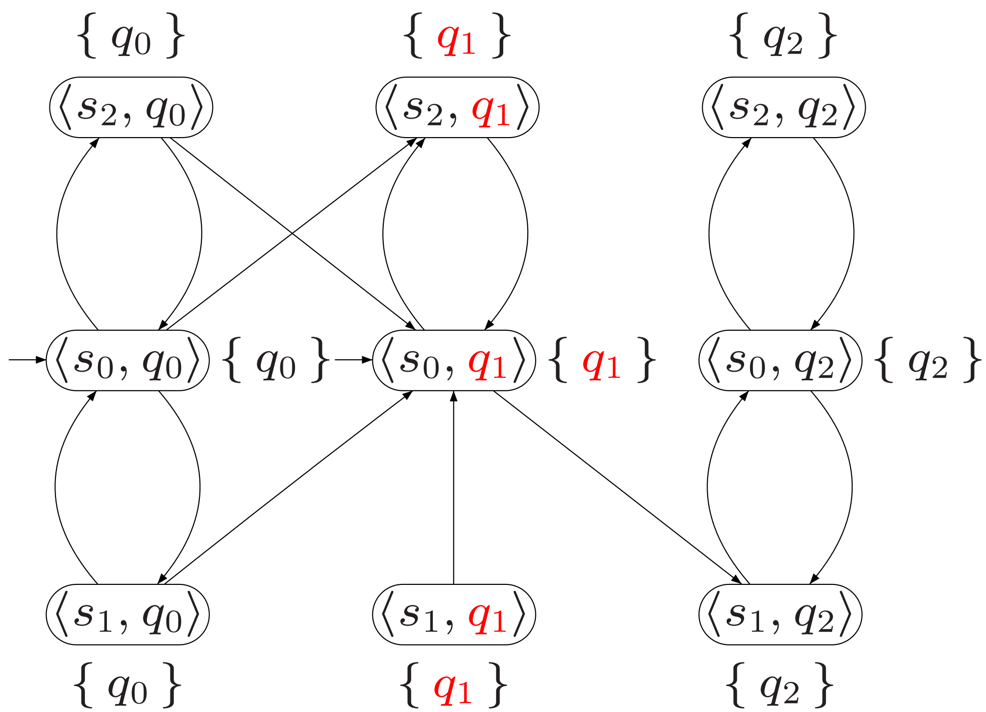






Infinitely often green?





Cycle detection

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

- Alternative 1:
  - compute the strongly connected components (SCCs) in $G(TS)$
  - check whether one such SCC is reachable from an initial state
  - . . . that contains a $\neg\Phi$-state
  - "eventually for ever Φ" is refuted if and only if such SCC is found

- Alternative 2:
  - *use a nested depth-first search*

  $\Rightarrow$ more adequate for an on-the-fly verification algorithm

  $\Rightarrow$ easier for generating counterexamples

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 inner DFS, visit all states reachable from s *not visited* in the inner DFS yet
  - no backward edge found to s? continue the outer DFS (look for next $\neg\Phi$ state)
- *Counterexample generation*: DFS stack concatenation
  - stack U for the outer DFS $=$ path fragment from $s_0 \in I$ to s (in reversed order)
  - stack V for the inner DFS $=$ a cycle from state s to s (in reversed order)

Correctness of nested DFS

Let:

- *TS* be a finite transition system over *AP* without terminal states and
- P_{pers} a persistence property

The nested DFS algorithm yields "no" if and only if $TS \not\models P_{pers}$

Time complexity

The worst-case time complexity of nested DFS is in

$$\mathcal{O}((N+M) + N \cdot |\, \Phi \,|)$$

where N is $\#$ reachable states in *TS*, and M is $\#$ transitions in *TS*