Principles of Model Checking

Christel Baier
Technische Universität Dresden

Joost-Pieter Katoen
RWTH Aachen University

Chapter 4: Regular Properties

Principles of Model Checking

Christel Baier and Joost-Pieter Katoen

Overview

- Automata on finite words
  - Regular safety property's bad prefixes constitute a regular language that can be recognized as a finite automaton (NFA or DFA)
- Model-checking regular safety properties
  - Reduce the safety property check problem to the invariant-checking problem in a product construction of TS with a finite automaton that recognized the bad prefixes of the safety property
- Automata on infinite words
  - Generalize the verification algorithm to a larger class of linear time properties: ω-regular properties
- Model-checking ω-regular properties
  - ω-regular properties can be represented by Buchi automata that is the key concept to verify ω-regular properties via a reduction to persistence checking

Overview

- Automata on finite words
  - Regular safety property's bad prefixes constitute a regular language that can be recognized as a finite automaton (NFA or DFA)
- Model-checking regular safety properties
  - Reduce the safety property check problem to the invariant-checking problem in a product construction of TS with a finite automaton that recognized the bad prefixes of the safety property
- Automata on infinite words
  - Generalize the verification algorithm to a larger class of linear time properties: ω-regular properties
- Model-checking ω-regular properties
  - ω-regular properties can be represented by Buchi automata that is the key concept to verify ω-regular properties via a reduction to persistence checking

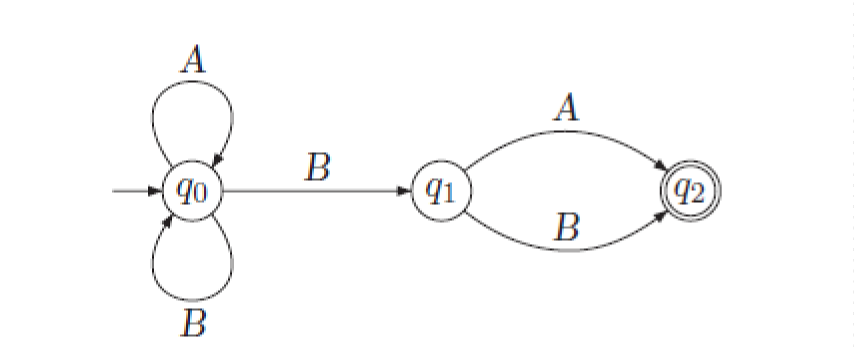
Automata on Finite Words

Definition 4.1. Nondeterministic Finite Automaton (NFA)

A nondeterministic finite automaton (NFA) A is a tuple A = (Q,Σ, δ,Q0, F) where

- • Q is a finite set of states,
- • Σ is an alphabet,
- • $\delta : Q \times \Sigma \to 2^{Q}$ is a transition function,
- • Q0 ⊆ Q is a set of initial states, and
- • F ⊆ Q is a set of accept (or: final) states.

An Example of a Finite-State Automaton



- $Q = \{ q0, q1, q2 \}$, $\Sigma = \{A,B\}$,
- $Q0 = \{ q0 \}$, $F = \{ q2 \}$,
- The transition function δ is defined by

  $\delta(q0,A) = \{q0\}$, $\delta(q0,B) = \{ q0, q1 \}$,

  $\delta(q1,A) = \{q2\}$, $\delta(q1,B) = \{ q2 \}$,

  $\delta(q2,A) = \emptyset$, $\delta(q2,B) = \emptyset$

Automata on Finite Words

Definition 4.3. Runs, Accepted Language of an NFA

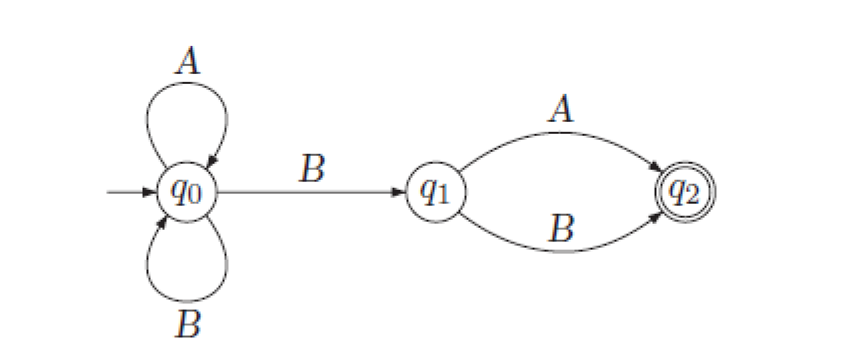
Let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NFA and $w = A_1 \ldots A_n \in \Sigma^*$ a finite word. A *run* for w in $\mathcal{A}$ is a finite sequence of states $q_0 \, q_1 \, \ldots \, q_n$ such that

- $q_0 \in Q_0$ and
- $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $0 \leqslant i < n$.

Run $q_0 \, q_1 \, \ldots \, q_n$ is called *accepting* if $q_n \in F$. A finite word $w \in \Sigma^*$ is called *accepted* by $\mathcal{A}$ if there exists an accepting run for w. The *accepted language* of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of finite words in Σ^* accepted by $\mathcal{A}$, i.e.,

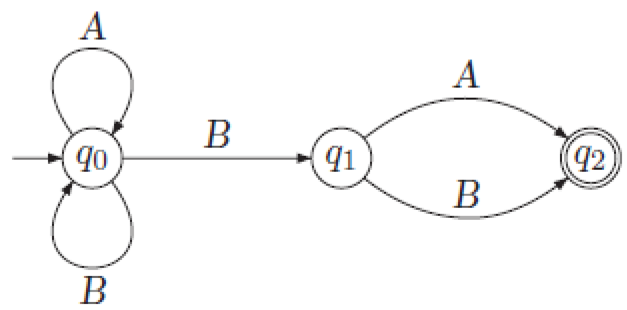
$$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^* \mid \text{ there exists an accepting run for } w \text{ in } \mathcal{A} \}.$$

Runs and Accepted Words



Runs	Words
q0	ε
q0 q0 q0 q0	ABA, BBA, ABA, BBB, AAA…
q0 q1 q2	BA, BB
q0 q0 q1 q2	ABB, ABA, BBA, BBB…

Runs and Accepted Words



- Accepting runs: runs that finish in the final state. (e.g., q0q1q2)
- Accepting words: words that can be represented by accepting runs. (e.g., ABA, BBB)
- Accepting words belong to the accepted language L(A) that is given by the regular expression (A+B)*B(A+B).

Alternative Characterization of the Accepted Language

Lemma 4.5. ***Alternative Characterization of the Accepted Language***

Let $\mathcal{A}$ be an NFA. Then:

$$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \varnothing \text{ for some } q_0 \in Q_0 \}.$$

An equivalent alternative characterization of the accepted language of an NFA $\mathcal{A}$ is as follows. Let $\mathcal{A}$ be an NFA as above. We extend the transition function δ to the function $\delta^* : Q \times \Sigma^* \to 2^Q$ as follows: $\delta^*(q, \varepsilon) = \{ q \}$, $\delta^*(q, A) = \delta(q, A)$, and

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

Stated in words, $\delta^*(q, w)$ is the set of states that are reachable from q for the input word w. In particular, $\bigcup_{q_0 \in Q_0} \delta^*(q_0, w)$ is the set of all states where a run for w in $\mathcal{A}$ can end.

Properties in NFA

Definition 4.6. Equivalence of NFAs

Let $\mathcal{A}$ and $\mathcal{A}'$ be NFAs with the same alphabet. $\mathcal{A}$ and $\mathcal{A}'$ are called *equivalent* if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. ■

Theorem 4.7. Language Emptiness is Equivalent to Reachability

Let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NFA. Then, $\mathcal{L}(\mathcal{A}) \neq \varnothing$ if and only if there exists $q_0 \in Q_0$ and $q \in F$ such that $q \in Reach(q_0)$.

Definition 4.8. Synchronous Product of NFAs

For NFA $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, Q_{0,i}, F_i)$, with i=1, 2, the *product automaton*

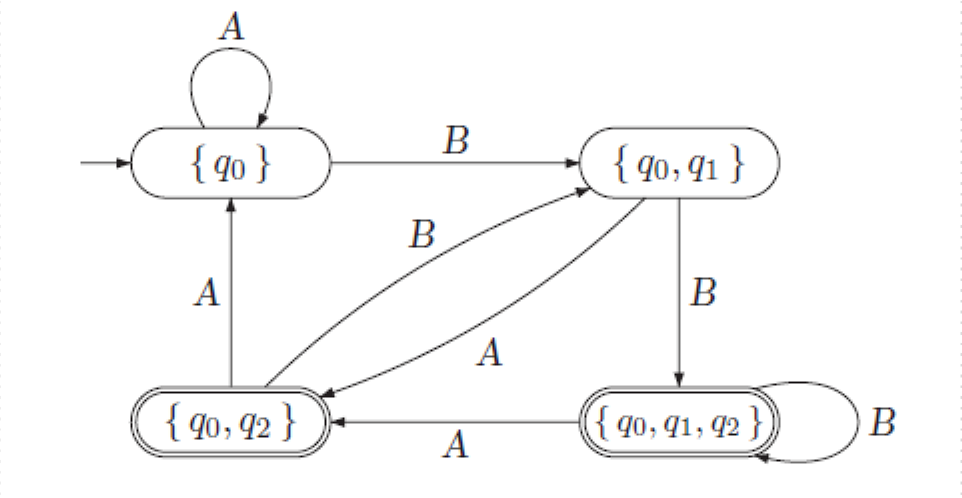
$$\mathcal{A}_1 \otimes \mathcal{A}_2 \;=\; (Q_1 \times Q_2, \Sigma, \delta, Q_{0,1} \times Q_{0,2}, F_1 \times F_2)$$

where δ is defined by

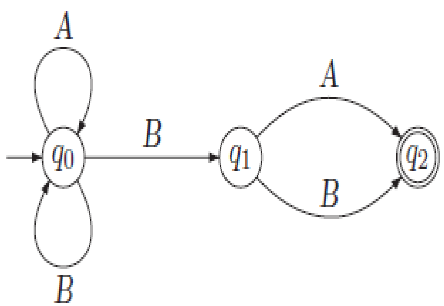
$$\frac{q_1 \xrightarrow{A}_1 q_1' \;\wedge\; q_2 \xrightarrow{A}_2 q_2'}{(q_1, q_2) \xrightarrow{A} (q_1', q_2')}.$$

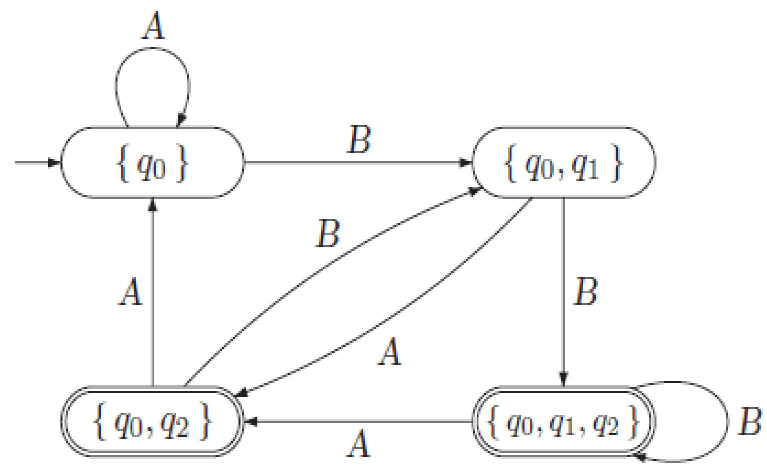
Deterministic Finite Automaton (DFA)

Let A = (Q,Σ, δ,Q0, F) be an NFA. A is called deterministic if $|Q0| <= 1$ and $|\delta(q,A)| <= 1$
for all states $q \in Q$ and all symbols $A \in \Sigma$. We will use the abbreviation DFA for a deterministic finite automaton.
DFA A is called total if $|Q0| = 1$ and $|\delta(q,A)| = 1$ for all $q \in Q$ and all $A \in \Sigma$.



Powerset Construction





Overview

- Automata on finite words
  - Regular safety property's bad prefixes constitute a regular language that can be recognized as a finite automaton (NFA or DFA)
- Model-checking regular safety properties
  - Reduce the safety property check problem to the invariant-checking problem in a product construction of TS with a finite automaton that recognized the bad prefixes of the safety property
- Automata on infinite words
  - Generalize the verification algorithm to a larger class of linear time properties: ω-regular properties
- Model-checking ω-regular properties
  - ω-regular properties can be represented by Buchi automata that is the key concept to verify ω-regular properties via a reduction to persistence checking

Regular Safety Properties

Every trace that violates a safety property has a bad prefix that causes a refutation.

The set of bad prefixes constitutes a language of finite words over the alphabet $\Sigma = 2^{AP}$.

The input symbols A ∈ Σ of the NFA are now sets of atomic propositions AP.

E.g., AP={a, b}, then Σ={{}, {a}, {b}, {a, b}}

Regular Safety Property

Definition 4.11. **Regular Safety Property**

Safety property P_{safe} over AP is called *regular* if its set of bad prefixes constitutes a regular language over 2^{AP}. ■

Every invariant is a regular safety property. If Φ is the state condition (propositional formula) of the invariant that should be satisfied by all reachable states, then the language of bad prefixes consists of the words $A_0 A_1 \ldots A_n$ such that $A_i \not\models \Phi$ for some $0 \leqslant i \leqslant n$. Such languages are regular, since they can be characterized by the (casually written) regular notation

$$\Phi^* (\neg \Phi)\, \text{true}^*.$$

Here, Φ stands for the set of all $A \subseteq AP$ with $A \models \Phi$, $\neg\Phi$ for the set of all $A \subseteq AP$ with $A \not\models \Phi$, while true means the set of all subsets A of AP. For instance, if $AP = \{a, b\}$ and $\Phi = a \vee \neg b$, then

- Φ stands for the regular expression $\{\} + \{a\} + \{a, b\}$,
- $\neg\Phi$ stands for the regular expression consisting of the symbol $\{b\}$,
- true stands for the regular expression $\{\} + \{a\} + \{b\} + \{a, b\}$.

Regular Safety Property

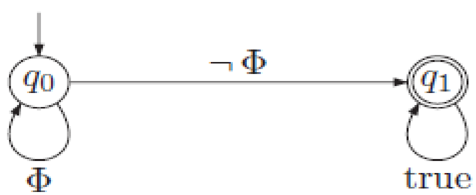
$$\Phi^*(\neg\,\Phi)\,\text{true}^*. \qquad \Phi = a \vee \neg b$$

- Φ stands for the regular expression $\{\} + \{a\} + \{a, b\}$,
- $\neg\Phi$ stands for the regular expression consisting of the symbol $\{b\}$,
- true stands for the regular expression $\{\} + \{a\} + \{b\} + \{a, b\}$.

The bad prefixes of the invariant over condition $a \vee \neg b$ are given by the regular expression:

$$\mathsf{E} = \underbrace{(\{\} + \{a\} + \{a, b\})^*}_{\Phi^*}\,\underbrace{\{b\}}_{\neg\Phi}\,\underbrace{(\{\} + \{a\} + \{b\} + \{a, b\})^*}_{\text{true}^*}.$$

Thus, $\mathcal{L}(\mathsf{E})$ consists of all words $A_1 \ldots A_n$ such that $A_i = \{b\}$ for some $1 \leqslant i \leqslant n$. Note that, for $A \subseteq AP = \{a, b\}$, we have $A \not\models a \vee \neg b$ if and only if $A = \{b\}$. Hence, $\mathcal{L}(\mathsf{E})$ agrees with the set of bad prefixes for the invariant induced by the condition Φ.



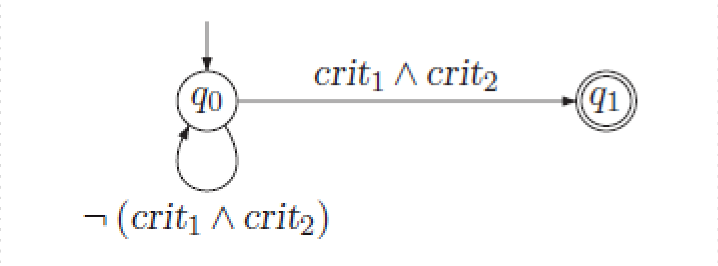
Example: Regular Safety Property for Mutual Exclusion Algorithms

Consider a mutual exclusion algorithm such as the semaphore-based one or Peterson's algorithm. The bad prefixes of the safety property P_mutex ("there is always at most one process in its critical section") constitute the language of all finite words A0 A1 . . .An such that

$$\{ crit1, crit2\} \subseteq Ai$$

for some index i with 0 <= i <= n.

A regular expression representing all bad prefixes is (~(crit1^crit2))*(crit1^crit2).

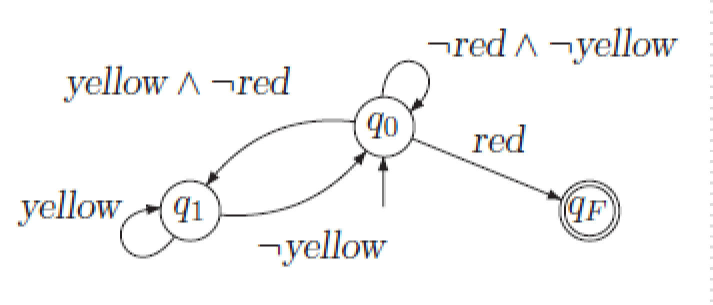


Example: Regular Safety Property for the Traffic Light

Consider a traffic light with three possible colors: red, yellow and green. The property “a red phase must be preceded immediately by a yellow phase” is specified by the set of infinite words $\sigma = A_0 A_1 \ldots$ with $A_i \subseteq \{red, yellow\}$ such that for all $i >= 0$ we have that

$$red \in A_i \text{ implies } i > 0 \text{ and } yellow \in A_{i-1}.$$

A NFA recognizing all bad prefixes of the property is shown as below:



A Nonregular Safety Property

Not all safety properties are regular. As an example of a nonregular safety property, consider:

"The number of inserted coins is always at least the number of dispensed drinks."

Let the set of propositions be { pay, drink }. Minimal bad prefixes for this safety property constitute the language

$$\{ \, pay^n \, drink^{n+1} \mid n \geqslant 0 \, \}$$

which is not a regular, but a context-free language.

Verifying Regular Safety Properties

Let P_{safe} be a regular safety property over the atomic propositions AP and A an NFA recognizing the bad prefixes of P_{safe}.

Lemma 3.25. Satisfaction Relation for Safety Properties

For transition system TS without terminal states and safety property P_{safe}:

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

Therefore, we need to check whether $Traces_{fin}(TS) \cap \mathcal{L}(\mathcal{A}) = \varnothing$

To check whether the NFAs A1 and A2 do intersect, it suffices to consider their product automaton, so

$$\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = \varnothing \quad \text{if and only if} \quad \mathcal{L}(\mathcal{A}_1 \otimes \mathcal{A}_2) = \varnothing.$$

Verifying Regular Safety Properties

Definition 4.16. Product of Transition System and NFA

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states and $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ an NFA with the alphabet $\Sigma = 2^{AP}$ and $Q_0 \cap F = \varnothing$. The product transition system $TS \otimes \mathcal{A}$ is defined as follows:

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

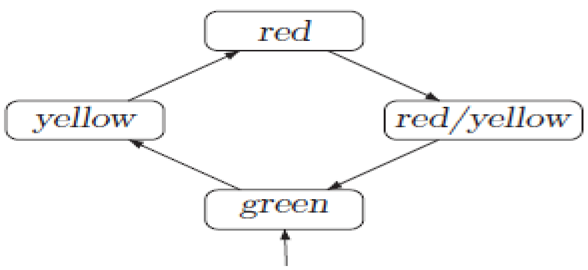
where

- $S' = S \times Q$,
- $\rightarrow'$ is the smallest relation defined by the rule

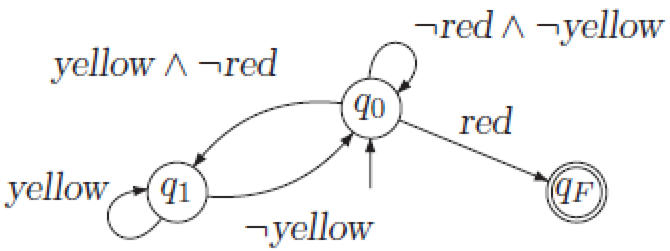
$$\frac{s \xrightarrow{\alpha} t \ \wedge \ q \xrightarrow{L(t)} p}{\langle s, q \rangle \xrightarrow{\alpha}{}' \langle t, p \rangle},$$

- $I' = \{ \langle s_0, q \rangle \mid s_0 \in I \ \wedge \ \exists q_0 \in Q_0.\ q_0 \xrightarrow{L(s_0)} q \}$,
- $AP' = Q$, and
- $L' : S \times Q \rightarrow 2^Q$ is given by $L'(\langle s, q \rangle) = \{ q \}$.

Example: a product automaton

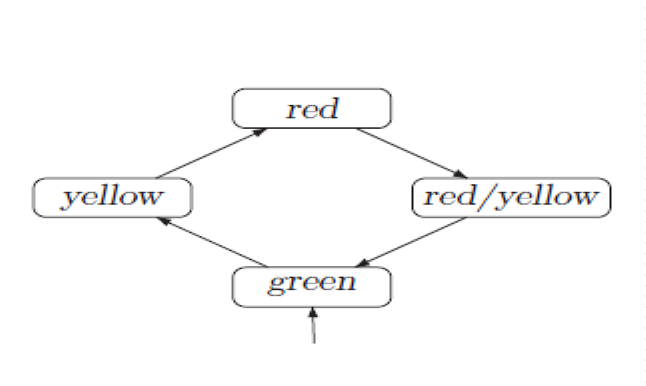


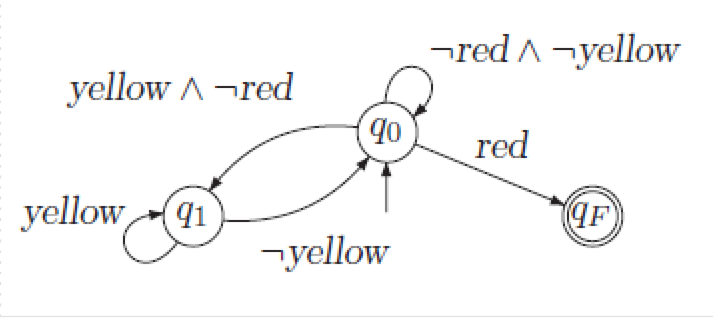
Consider a German traffic light, AP = { red, yellow } indicating the corresponding light phases.
The labeling is defined as follows: L(red) = { red }, L(yellow) = { yellow }, L(green) = ∅ = L(red+yellow).

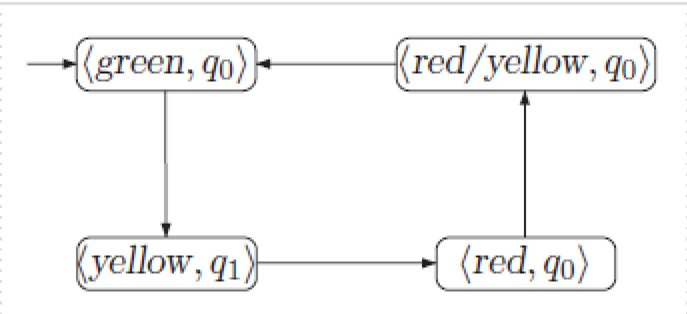


The language of the minimal bad prefixes of the safety property "each red light phase is preceded by a yellow light phase" is accepted by the DFA A indicated here.

Example: a product automaton







Verifying Regular Safety Properties

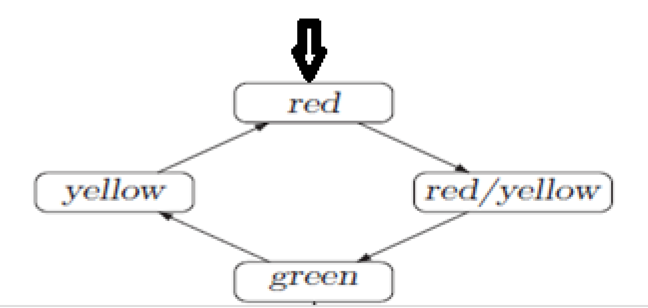
The following theorem shows that the verification of a regular safety property can be reduced to checking an invariant in the product.

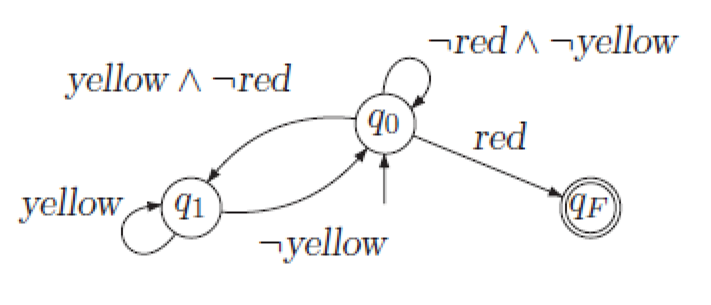
Let TS and $\mathcal{A}$ be as before. Let $P_{inv(\mathcal{A})}$ be the invariant over $AP' = 2^Q$ which is defined by the propositional formula

$$\bigwedge_{q \in F} \neg\, q.$$

In the sequel, we often write $\neg\, F$ as shorthand for $\bigwedge_{q \in F} \neg\, q$. Stated in words, $\neg F$ holds in all nonaccept states.

Example: a product automaton





Verifying Regular Safety Properties

Algorithm 5 Model-checking algorithm for regular safety properties

Input: finite transition system TS and regular safety property P_{safe}
Output: true if $TS \models P_{safe}$. Otherwise false plus a counterexample for P_{safe}.

Let NFA $\mathcal{A}$ (with accept states F) be such that $\mathcal{L}(\mathcal{A})$ = bad prefixes of P_{safe}
Construct the product transition system $TS \otimes \mathcal{A}$
Check the invariant $P_{inv(\mathcal{A})}$ with proposition $\neg F = \bigwedge_{q \in F} \neg q$ on $TS \otimes \mathcal{A}$.

if $TS \otimes \mathcal{A} \models P_{inv(\mathcal{A})}$ **then**
  return true
else
  Determine an initial path fragment $\langle s_0, q_1 \rangle \ldots \langle s_n, q_{n+1} \rangle$ of $TS \otimes \mathcal{A}$ with $q_{n+1} \in F$
  return (false, $s_0\, s_1 \ldots s_n$)
fi

Overview

- Automata on finite words
  - Regular safety property's bad prefixes constitute a regular language that can be recognized as a finite automaton (NFA or DFA)
- Model-checking regular safety properties
  - Reduce the safety property check problem to the invariant-checking problem in a product construction of TS with a finite automaton that recognized the bad prefixes of the safety property
- Automata on infinite words
  - Generalize the verification algorithm to a larger class of linear time properties: ω-regular properties
- Model-checking ω-regular properties
  - ω-regular properties can be represented by Buchi automata that is the key concept to verify ω-regular properties via a reduction to persistence checking

ω-Regular Languages and Properties

Infinite words over the alphabet Σ are infinite sequences A0 A1 A2 . . . of symbols Ai ∈ Σ.

Σ^ω denotes the set of all infinite words over Σ.

Any subset of Σ^ω is called a language of infinite words, called an ω-language.

For instance, the infinite repetition of the finite word AB yields the infinite word ABABABABAB. . . (ad infinitum) and is denoted by $(AB)^\omega$.

For the special case of the empty word, we have ε^ω = ε.
For an infinite word, infinite repetition has no effect, that is, σ^ω = σ if σ ∈ Σ^ω.

ω-Regular Expression

Definition 4.23. **ω-Regular Expression**

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

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

where $n \geqslant 1$ and $\mathsf{E}_1, \ldots, \mathsf{E}_n, \mathsf{F}_1, \ldots, \mathsf{F}_n$ are regular expressions over Σ such that $\varepsilon \notin \mathcal{L}(\mathsf{F}_i)$, for all $1 \leqslant i \leqslant n$.

The semantics of the ω-regular expression G is a language of infinite words, defined by

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

where $\mathcal{L}(\mathsf{E}) \subseteq \Sigma^*$ denotes the language (of finite words) induced by the regular expression E (see page 914).

Examples for ω-regular expressions over the alphabet $\Sigma = \{ A, B, C \}$ are

$$(A+B)^* A(AAB+C)^\omega \quad \text{or} \quad A(B+C)^* A^\omega + B(A+C)^\omega.$$

ω-Regular Language

Definition 4.24. **ω-Regular Language**

A language $\mathcal{L} \subseteq \Sigma^\omega$ is called ω-regular if $\mathcal{L} = \mathcal{L}_\omega(\mathsf{G})$ for some ω-regular expression G over Σ. ■

For instance, the language consisting of all infinite words over $\{A, B\}$ that contain infinitely many A's is ω-regular since it is given by the ω-regular expression $(B^*A)^\omega$. The language consisting of all infinite words over $\{A, B\}$ that contain only finitely many A's is ω-regular too. A corresponding ω-regular expression is $(A+B)^*B^\omega$. The empty set is ω-regular since it is obtained, e.g., by the ω-regular expression $\varnothing^\omega$. More generally, if $\mathcal{L} \subseteq \Sigma^*$ is regular and $\mathcal{L}'$ is ω-regular, then $\mathcal{L}^\omega$ and $\mathcal{L}.\mathcal{L}'$ are ω-regular.

ω-Regular Properties

Definition 4.25. **ω-Regular Properties**

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

For instance, for $AP = \{a, b\}$, the invariant P_{inv} induced by the proposition $\Phi = a \vee \neg b$ is an ω-regular property since

$$\begin{aligned} P_{inv} &= \Big\{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid \forall i \geqslant 0. \, (a \in A_i \text{ or } b \notin A_i) \Big\} \\ &= \Big\{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid \forall i \geqslant 0. \, (A_i \in \{\{\}, \{a\}, \{a, b\}\} \Big\} \end{aligned}$$

is given by the ω-regular expression $\mathsf{E} = (\{\} + \{a\} + \{a, b\})^\omega$ over the alphabet $\Sigma = 2^{AP} = \{\{\}, \{a\}, \{b\}, \{a, b\}\}$.

Example: Mutual Exclusion

An example of an ω-regular property is the property given by the informal statement "process P visits its critical section infinitely often" which, for AP = { wait, crit }, can be formalized by the ω-regular expression:

$$((\underbrace{\{\} + \{\,wait\,\}}_{\text{negative literal } \neg crit})^*.(\underbrace{\{\,crit\,\} + \{\,wait, crit\,\}}_{\text{positive literal } crit}))^\omega.$$

Starvation freedom in the sense of "whenever process P is waiting then it will enter its critical section eventually later" is an ω-regular property as it can be described by

$$((\neg wait)^*.wait.\mathrm{true}^*.crit)^\omega \;+\; ((\neg wait)^*.wait.\mathrm{true}^*.crit)^*.(\neg wait)^\omega$$

Nondeterministic Buchi Automata

Definition 4.27. Nondeterministic Büchi Automaton (NBA)

A *nondeterministic Büchi automaton* (NBA) $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where

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

A run for $\sigma = A_0 A_1 A_2 \ldots \in \Sigma^\omega$ denotes an infinite sequence $q_0\, q_1\, q_2 \ldots$ of states in $\mathcal{A}$ such that $q_0 \in Q_0$ and $q_i \xrightarrow{A_i} q_{i+1}$ for $i \geqslant 0$. Run $q_0\, q_1\, q_2 \ldots$ is *accepting* if $q_i \in F$ for infinitely many indices $i \in \mathbb{N}$. The *accepted language* of $\mathcal{A}$ is

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

NFA v.s. NBA

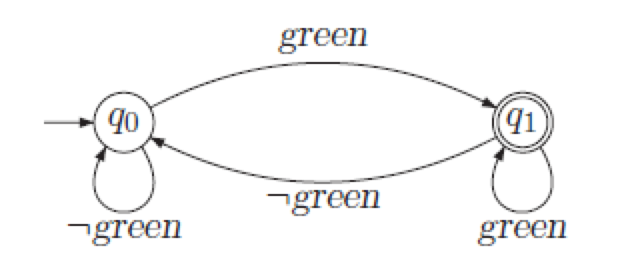
Syntax differences between NFA and NBA : None

Semantics differences between NFA and NBA: the accepted language of an NFA A is a language of **finite words**, whereas the accepted language of NBA A is an **ω-language**.

The intuitive meaning of the acceptance criterion named after Buchi is that the accept set of A has to be visited infinitely often. Thus, the accepted language $L\omega(A)$ consists of all infinite words that have a run in which some accept state is visited infinitely often.

Example: Infinitely Often Green

Let AP = { green, red } or any other set containing the proposition green. The language of words σ = A0 A1 . . . ∈ 2^{AP} satisfying the LT property "infinitely often green" is accepted by the NBA A depicted below.



Accepting runs: $(q0q1)^{w}$, $(q0q1)^{n}q1^{w}$ …

Non accepting runs: $q1^{w}$, $q0^{w}$ …

NBA Properties

Theorem 4.32. ***NBAs and ω-Regular Languages***

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

Lemma 4.33. ***Union Operator on NBA***

For NBA $\mathcal{A}_1$ and $\mathcal{A}_2$ (both over the alphabet Σ) there exists an NBA $\mathcal{A}$ such that:

$$\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{A}_1) \cup \mathcal{L}_\omega(\mathcal{A}_2) \quad \textit{and} \quad |\mathcal{A}| = \mathcal{O}(|\mathcal{A}_1| + |\mathcal{A}_2|).$$

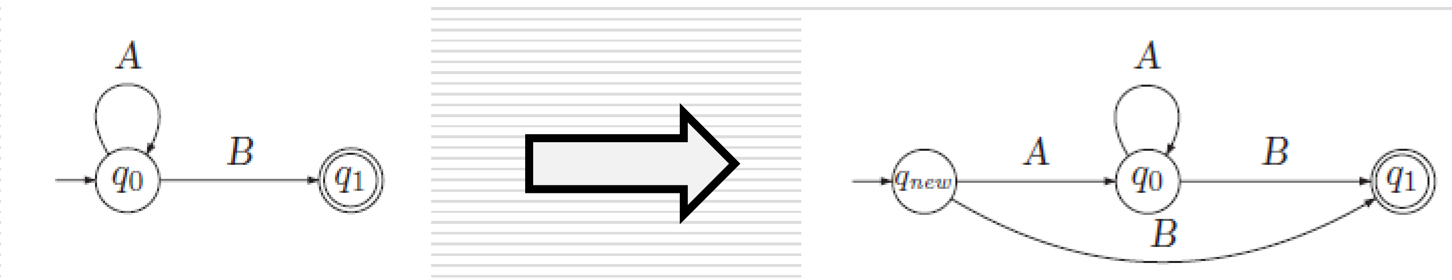
Lemma 4.34. ***ω-Operator for NFA***

For each NFA $\mathcal{A}$ with $\varepsilon \notin \mathcal{L}(\mathcal{A})$ there exists an NBA $\mathcal{A}'$ such that

$$\mathcal{L}_\omega(\mathcal{A}') = \mathcal{L}(\mathcal{A})^\omega \quad \textit{and} \quad |\mathcal{A}'| = \mathcal{O}(|\mathcal{A}|).$$

Constructing a NBA from a NFA

Add a new initial (nonaccept) state q_{new} to Q with the transitions $q_{new} \xrightarrow{A} q$ if and only if $q_0 \xrightarrow{A} q$ for some initial state $q_0 \in Q_0$. All other transitions, as well as the accept states, remain unchanged.



Constructing a NBA from a NFA

In the sequel, we assume that $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ is an NFA such that the states in Q_0 do not have any incoming transitions and $Q_0 \cap F = \varnothing$. We now construct an NBA $\mathcal{A}' = (Q, \Sigma, \delta', Q'_0, F')$ with $\mathcal{L}_\omega(\mathcal{A}') = \mathcal{L}(\mathcal{A})^\omega$. The basic idea of the construction of $\mathcal{A}'$ is to add for any transition in $\mathcal{A}$ that leads to an accept state new transitions leading to the initial states of $\mathcal{A}$. Formally, the transition relation δ' in the NBA $\mathcal{A}'$ is given by

$$\delta'(q, A) \;=\; \begin{cases} \delta(q, A) & \text{if } \delta(q, A) \cap F = \varnothing \\ \delta(q, A) \cup Q_0 & \text{otherwise.} \end{cases}$$

The initial states in the NBA $\mathcal{A}'$ agree with the initial states in $\mathcal{A}$, i.e., $Q'_0 = Q_0$. These are also the accept states in $\mathcal{A}'$, i.e., $F' = Q_0$.