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: $\omega$-regular properties

- **Model-checking $\omega$-regular properties**
  - $\omega$-regular properties can be represented by Buchi automata that is the key concept to verify $\omega$-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 \( A = (Q, \Sigma, \delta, Q_0, F) \) where

- \( Q \) is a finite set of states,
- \( \Sigma \) is an alphabet,
- \( \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q) \) 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.
An Example of a Finite-State Automaton

- $Q = \{ q_0, q_1, q_2 \}$, $\Sigma = \{ A, B \}$,
- $Q_0 = \{ q_0 \}$, $F = \{ q_2 \}$,
- The transition function $\delta$ is defined by
  $\delta(q_0, A) = \{ q_0 \}$, $\delta(q_0, B) = \{ q_0, q_1 \}$,
  $\delta(q_1, A) = \{ q_2 \}$, $\delta(q_1, B) = \{ q_2 \}$,
  $\delta(q_2, A) = \emptyset$, $\delta(q_2, B) = \emptyset$
Automata on Finite Words

Definition 4.3. Runs, Accepted Language of an NFA
Let \( A = (Q, \Sigma, \delta, Q_0, F) \) be an NFA and \( w = A_1 \ldots A_n \in \Sigma^* \) a finite word. A run for \( w \) in \( A \) 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 \( 0 \leq i < n \).

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 \( A \) if there exists an accepting run for \( w \). The accepted language of \( A \), denoted \( \mathcal{L}(A) \), is the set of finite words in \( \Sigma^* \) accepted by \( A \), i.e.,

\[
\mathcal{L}(A) = \{ w \in \Sigma^* \mid \text{there exists an accepting run for } w \text{ in } A \}.
\]
# Runs and Accepted Words

![Diagram of a finite automaton]

<table>
<thead>
<tr>
<th>Runs</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>q0 q0 q0 q0</td>
<td>ABA, BBA, ABA, BBB, AAA...</td>
</tr>
<tr>
<td>q0 q1 q2</td>
<td>BA, BB</td>
</tr>
<tr>
<td>q0 q0 q1 q2</td>
<td>ABB, ABA, BBA, BBB...</td>
</tr>
</tbody>
</table>
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 $(A+B)^*B(A+B)$. 

![Diagram](image-url)
Lemma 4.5. Alternative Characterization of the Accepted Language

Let $A$ be an NFA. Then:

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

An equivalent alternative characterization of the accepted language of an NFA $A$ is as follows. Let $A$ be an NFA as above. We extend the transition function $\delta$ to the function $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$ 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 $A$ can end.
Definition 4.6. Equivalence of NFAs
Let $A$ and $A'$ be NFAs with the same alphabet. $A$ and $A'$ are called equivalent if $L(A) = L(A')$.

Theorem 4.7. Language Emptiness is Equivalent to Reachability
Let $A = (Q, \Sigma, \delta, Q_0, F)$ be an NFA. Then, $L(A) \neq \emptyset$ if and only if there exists $q_0 \in Q_0$ and $q \in F$ such that $q \in \text{Reach}(q_0)$.

Definition 4.8. Synchronous Product of NFAs
For NFA $A_i = (Q_i, \Sigma, \delta_i, Q_{0,i}, F_i)$, with $i = 1, 2$, the product automaton $A_1 \otimes A_2 = (Q_1 \times Q_2, \Sigma, \delta, Q_{0,1} \times Q_{0,2}, F_1 \times F_2)$
where $\delta$ is defined by

\[
\frac{q_1 \overset{A_1}{\rightarrow} q'_1 \land q_2 \overset{A_2}{\rightarrow} q'_2}{(q_1, q_2) \overset{A}{\rightarrow} (q'_1, q'_2)}.
\]
Deterministic Finite Automaton (DFA)

Let \( A = (Q, \Sigma, \delta, Q_0, F) \) be an NFA. \( A \) is called deterministic if \( |Q_0| \leq 1 \) and \( |\delta(q, A)| \leq 1 \) 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 \( |Q_0| = 1 \) 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: \(\omega\)-regular properties

- **Model-checking \(\omega\)-regular properties**
  - \(\omega\)-regular properties can be represented by Buchi automata that is the key concept to verify \(\omega\)-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 $A \in \Sigma$ of the NFA are now sets of atomic propositions $AP$.

E.g., $AP=\{a, b\}$, then $\Sigma=\{\{\}, \{a\}, \{b\}, \{a, b\}\}$
Definition 4.11. Regular Safety Property

Safety property $P_{\text{safe}}$ 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 $\Phi$ 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 $0 \leq i \leq n$. Such languages are regular, since they can be characterized by the (casually written) regular notation

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

Here, $\Phi$ 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 $\Phi = a \lor \neg b$, then

- $\Phi$ 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

\[ \Phi^* (\neg \Phi) \text{ true}^* \quad \Phi = a \lor \neg b \]

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

The bad prefixes of the invariant over condition \( a \lor \neg b \) are given by the regular expression:

\[ E = (\{\} + \{a\} + \{a, b\})^* \{b\} (\{\} + \{a\} + \{b\} + \{a, b\})^*. \]

Thus, \( L(E) \) consists of all words \( A_1 \ldots A_n \) such that \( A_i = \{b\} \) for some \( 1 \leq i \leq n \). Note that, for \( A \ Subseteq AP = \{a, b\} \), we have \( A \ NotEqual a \lor \neg b \) if and only if \( A = \{b\} \). Hence, \( L(E) \) agrees with the set of bad prefixes for the invariant induced by the condition \( \Phi \).

\[ \begin{array}{c}
q_0 \\
\Phi \\
\neg \Phi \\
q_1 \\
\text{true}
\end{array} \]
Consider a mutual exclusion algorithm such as the semaphore-based one or Peterson’s algorithm. The bad prefixes of the safety property $P_{\text{mutex}}$ (“there is always at most one process in its critical section”) constitute the language of all finite words $A_0 A_1 \ldots A_n$ such that
\[
\{ \text{crit1, crit2} \} \subseteq A_i
\]
for some index $i$ with $0 \leq i \leq n$.
A regular expression representing all bad prefixes is
\[
(\neg(\text{crit1} \land \text{crit2}))^*(\text{crit1} \land \text{crit2}).
\]
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 $A_i \subseteq \{\text{red, yellow}\}$ such that for all $i \geq 0$ we have that $\text{red} \in A_i$ implies $i > 0$ and $\text{yellow} \in A_{i-1}$.

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 \{ \text{pay}, \text{drink} \}. Minimal bad prefixes for this safety property constitute the language

\[ \{ \text{pay}^n \text{drink}^{n+1} \mid n \geq 0 \} \]

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 $P_{safe}$:

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

Therefore, we need to check whether

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

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

$$\mathcal{L}(A_1) \cap \mathcal{L}(A_2) = \emptyset \text{ if and only if } \mathcal{L}(A_1 \otimes A_2) = \emptyset.$$
Verifying Regular Safety Properties

Definition 4.16. Product of Transition System and NFA

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

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

where

- $S' = S \times Q$,
- $\rightarrow'$ is the smallest relation defined by the rule
  $$\frac{s \xrightarrow{\alpha} t \land q \xrightarrow{L(t)} p}{\langle s, q \rangle \xrightarrow{\alpha, I} \langle t, p \rangle},$$
- $I' = \{ \langle s_0, q \rangle \mid s_0 \in I \land \exists q_0 \in Q_0. q_0 \xrightarrow{L(s_0)} q \}$,
- $AP' = Q$, and
- $L' : S \times Q \rightarrow 2^Q$ is given by $L'((s, q)) = \{ q \}$. 
Example: a product automaton

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

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 $P_{\text{inv}}(\mathcal{A})$ 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 non-accept states.
Example: a product automaton

- A product automaton combines two separate automata into a single model.

- The automaton on the left represents states `red`, `yellow`, and `green`.

- The automaton on the right represents a transition system with states `q0`, `q1`, and `q_F`.

- The transitions include conjunctions like `yellow ∧ ¬red` and `red ∧ ¬yellow`.
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 $A$ (with accept states $F$) be such that $\mathcal{L}(A) =$ bad prefixes of $P_{safe}$
Construct the product transition system $TS \otimes A$
Check the invariant $P_{inv(A)}$ with proposition $\neg F = \bigwedge_{q \in F} \neg q$ on $TS \otimes A$.

if $TS \otimes A \models P_{inv(A)}$ then
  return true
else
  Determine an initial path fragment $\langle s_0, q_1 \rangle \ldots \langle s_n, q_{n+1} \rangle$ of $TS \otimes A$ 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: \(\omega\)-regular properties

- Model-checking \(\omega\)-regular properties
  - \(\omega\)-regular properties can be represented by Buchi automata that is the key concept to verify \(\omega\)-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.

Σω denotes the set of all infinite words over Σ.

Any subset of Σω 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 ABABABABABAB . . . (ad infinitum) and is denoted by (AB)ω.

For the special case of the empty word, we have εω = ε. For an infinite word, infinite repetition has no effect, that is, σω = σ if σ ∈ Σω.
ω-Regular Expression

Definition 4.23. ω-Regular Expression

An ω-regular expression $G$ over the alphabet $\Sigma$ has the form

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

where $n \geq 1$ and $E_1, \ldots, E_n, F_1, \ldots, F_n$ are regular expressions over $\Sigma$ such that $\varepsilon \notin \mathcal{L}(F_i)$, for all $1 \leq i \leq n$.

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

$$\mathcal{L}_\omega(G) = \mathcal{L}(E_1).\mathcal{L}(F_1)^\omega \cup \ldots \cup \mathcal{L}(E_n).\mathcal{L}(F_n)^\omega$$

where $\mathcal{L}(E) \subseteq \Sigma^*$ 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

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

Definition 4.24. \( \omega \)-Regular Language

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

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

$$P_{inv} = \left\{ A_0A_1A_2\ldots \in (2^{AP})^\omega \mid \forall i \geq 0. (a \in A_i \text{ or } b \notin A_i) \right\}$$

$$= \left\{ A_0A_1A_2\ldots \in (2^{AP})^\omega \mid \forall i \geq 0. (A_i \in \{\}, \{a\}, \{a, b\}) \right\}$$

is given by the ω-regular expression $E = (\{\} + \{a\} + \{a, b\})^\omega$ over the alphabet $Σ = 2^{AP} = \{\}, \{a\}, \{b\}, \{a, b\}$.
Example: Mutual Exclusion

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

$$(\{ \} + \{ \text{wait} \})^*.(\{ \text{crit} \} + \{ \text{wait, crit} \})^\omega.$$  

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

$$(\neg \text{wait})^* . \text{wait.true}^* . \text{crit}^\omega + (\neg \text{wait})^* . \text{wait.true}^* . \text{crit}^* . (\neg \text{wait})^\omega.$$
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,
- $\Sigma$ 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 $i \geq 0$. 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 $\omega$-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 \( AP = \{ \text{green, red} \} \) or any other set containing the proposition green. The language of words \( \sigma = A_0 A_1 \ldots \in 2^{AP} \) satisfying the LT property “infinitely often green” is accepted by the NBA A depicted below.

Accepting runs: \( (q_0 q_1)^w, (q_0 q_1)^n q_1^w \ldots \)

Non accepting runs: \( q_1^w, q_0^w \ldots \)
NBA Properties

Theorem 4.32. NBA\(\)s and \(\omega\)-Regular Languages
The class of languages accepted by NBA\(\)s agrees with the class of \(\omega\)-regular languages.

Lemma 4.33. Union Operator on NBA
For NBA \(A_1\) and \(A_2\) (both over the alphabet \(\Sigma\)) there exists an NBA \(A\) such that:
\[
\mathcal{L}_\omega(A) = \mathcal{L}_\omega(A_1) \cup \mathcal{L}_\omega(A_2) \quad \text{and} \quad |A| = \mathcal{O}(|A_1| + |A_2|).
\]

Lemma 4.34. \(\omega\)-Operator for NFA
For each NFA \(A\) with \(\varepsilon \notin \mathcal{L}(A)\) there exists an NBA \(A'\) such that
\[
\mathcal{L}_\omega(A') = \mathcal{L}(A)^\omega \quad \text{and} \quad |A'| = \mathcal{O}(|A|).
\]
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.
In the sequel, we assume that $A = (Q, \Sigma, \delta, Q_0, F)$ is an NFA such that the states in $Q_0$ do not have any incoming transitions and $Q_0 \cap F = \emptyset$. We now construct an NBA $A' = (Q, \Sigma, \delta', Q'_0, F')$ with $L_\omega(A') = L(A)^\omega$. The basic idea of the construction of $A'$ is to add for any transition in $A$ that leads to an accept state new transitions leading to the initial states of $A$. Formally, the transition relation $\delta'$ in the NBA $A'$ is given by

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

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