Alternating Finite Automata

Matthias Hengel

Seminar: Automata Theory
Software Engineering
Albert-Ludwigs-Universität Freiburg

24. July, 2013

Motivation

- Problem: DFA occurring in practice are often very big with a lot of states
- How can they be represented efficiently?
- Using alternating finite automata a DFA with 2^k states can be represented as a automaton with $k+1$ states
- Problem: The "complexity" of the automaton is shifted to the transition function
- How can the transition function be represented efficiently?

Motivation

- Problem: DFA occurring in practice are often very big with a lot of states
- How can they be represented efficiently?
- Using alternating finite automata a DFA with 2^k states can be represented as a automaton with $k+1$ states
- Problem: The "complexity" of the automaton is shifted to the transition function
- How can the transition function be represented efficiently?

Content

Basic Definitions

AFA: Definition

Definition

A h-AFA is a tuple (Q, Σ, g, h, f), where

- Q is a finite set of states,
- Σ is the input alphabet,
- $g : Q \times \Sigma \times B^Q \to B$ is the transition function, where B denotes the two-element Boolean algebra,
- $h : B^Q \to B$ is the accepting function, and
- $F \subseteq Q$ is the set of final states.

AFA: Further definitions

Definition

The transition function $g : Q \times \Sigma \times B^Q \to B$ is extended to a function $g : Q \times \Sigma^* \times B^Q \to B$ as follows:

- $g(s, \lambda, u) = u_s$, and
- $g(s, aw, u) = g(s, a, g(s, w, u))$.

Definition

A word $w \in \Sigma^*$ is accepted by an AFA iff $h(g(w, f)) = 1$, where

- $f \in B^Q$ and $f_q = 1$ iff $q \in F$, and
- $g(w, f) = g(s, w, f)_{s \in Q}$.

AFA: Further definitions

Definition

The transition function $g : Q \times \Sigma \times B^Q \to B$ is extended to a function $g : Q \times \Sigma^* \times B^Q \to B$ as follows:

- $g(s, \lambda, u) = u_s$, and
- $g(s, aw, u) = g(s, a, g(s, w, u))$.

Definition

A word $w \in \Sigma^*$ is accepted by an AFA iff $h(g(w, f)) = 1$, where

- $f \in B^Q$ and $f_q = 1$ iff $q \in F$, and
- $g(w, f) = g(s, w, f)_{s \in Q}$.

Example

Example

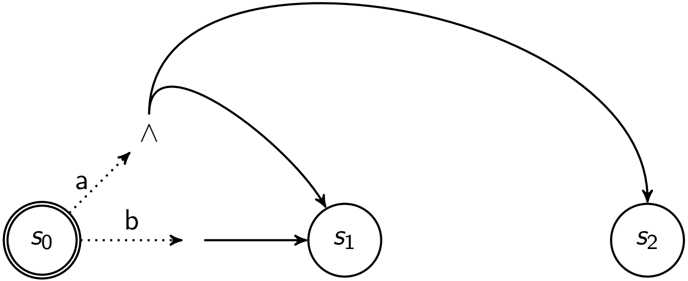
Consider the automata $A = (Q_A, \Sigma, g, h, F_A)$ where

- $Q_A = \{s_0, s_1, s_2\}$
- $\Sigma = \{a, b\}$
- $h(s_0, s_1, s_2) = s_0$
- $F_A = \emptyset$

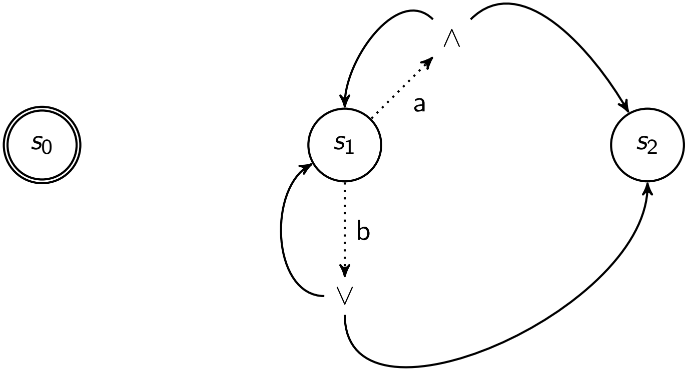
and g is defined by:

State	a	b
s_0	$u_1 \wedge u_2$	u_1
s_1	$u_1 \wedge u_2$	$u_1 \vee u_2$
s_2	1	u_1

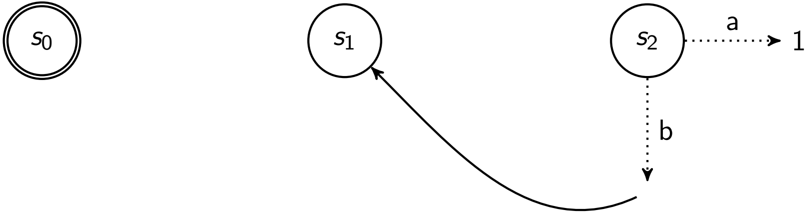
Example



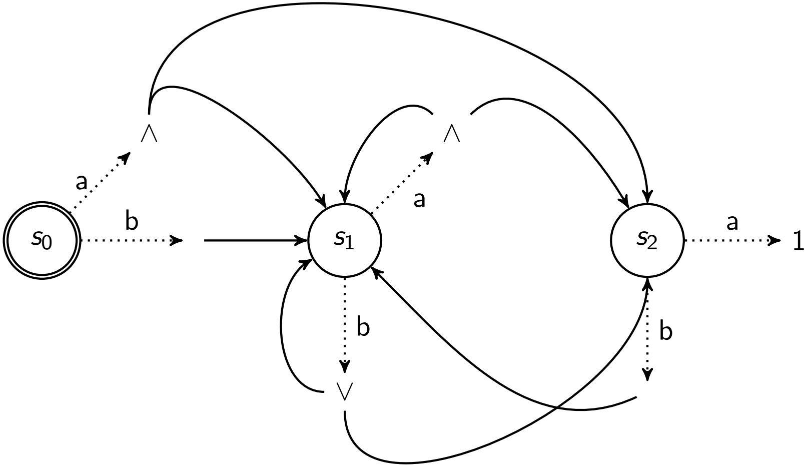
Example



Example



Example



Example run

Example

State	a	b
s_0	$u_1 \wedge u_2$	u_1
s_1	$u_1 \wedge u_2$	$u_1 \vee u_2$
s_2	1	u_1

Consider the word *bba*.

$$\begin{aligned}
h(g(bba, f)) &= h(g(b, g(b, g(a, f)))) = h(g(b, g(b, g(a, (0, 0, 0)))))\\
&= h(g(b, g(b, (0, 0, 1))))\\
&= h(g(b, (0, 1, 0)))\\
&= h(1, 1, 1)\\
&= 1
\end{aligned}$$

Example run

Example

State	a	b
s_0	$u_1 \wedge u_2$	u_1
s_1	$u_1 \wedge u_2$	$u_1 \vee u_2$
s_2	1	u_1

Consider the word *bba*.

$$\begin{aligned} h(g(bba, f)) &= h(g(b, g(b, g(a, f)))) = h(g(b, g(b, g(a, (0, 0, 0))))) \\ &= h(g(b, g(b, (0, 0, 1)))) \\ &= h(g(b, (0, 1, 0)) \\ &= h(1, 1, 1) \\ &= 1 \end{aligned}$$

Construction from DFA to an equivalent AFA

Idea

- Consider a DFA with 2^k states
- 2^k states can be encoded by Boolean vectors of length k
- **Idea:** Every state of the DFA is represented as an assignment of states of the AFA
- This corresponds to an encoding of the states of the DFA as Boolean vectors
- The transition function must be build accordingly
- The AFA accepts the reverse language

Idea

- Consider a DFA with 2^k states
- 2^k states can be encoded by Boolean vectors of length k
- **Idea:** Every state of the DFA is represented as an assignment of states of the AFA
- This corresponds to an encoding of the states of the DFA as Boolean vectors
- The transition function must be build accordingly
- The AFA accepts the reverse language

Idea

- Consider a DFA with 2^k states
- 2^k states can be encoded by Boolean vectors of length k
- **Idea:** Every state of the DFA is represented as an assignment of states of the AFA
- This corresponds to an encoding of the states of the DFA as Boolean vectors
- The transition function must be build accordingly
- The AFA accepts the reverse language

Idea

- Consider a DFA with 2^k states
- 2^k states can be encoded by Boolean vectors of length k
- **Idea:** Every state of the DFA is represented as an assignment of states of the AFA
- This corresponds to an encoding of the states of the DFA as Boolean vectors
- The transition function must be build accordingly
- The AFA accepts the reverse language

Idea

- Consider a DFA with 2^k states
- 2^k states can be encoded by Boolean vectors of length k
- **Idea:** Every state of the DFA is represented as an assignment of states of the AFA
- This corresponds to an encoding of the states of the DFA as Boolean vectors
- The transition function must be build accordingly
- The AFA accepts the reverse language

Idea

- Consider a DFA with 2^k states
- 2^k states can be encoded by Boolean vectors of length k
- **Idea:** Every state of the DFA is represented as an assignment of states of the AFA
- This corresponds to an encoding of the states of the DFA as Boolean vectors
- The transition function must be build accordingly
- The AFA accepts the reverse language

Construction

Theorem

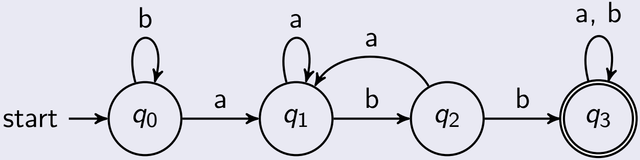
A language L is accepted by a DFA with 2^k states if and only if it's reversed language L^R is accepted by an AFA with $k+1$ states.

Construction

Let $A = (Q_D, \Sigma, q_0, F_D, \delta)$, $Q_D = \{q_0, \ldots, q_{2^k-1}\}$ be an DFA with 2^k states.

Example

Consider a DFA A as following:



State	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

Construction: States

The set of states is constructed as $Q_A = \{s_0, s_1, \ldots, s_k\}$.

The state s_0 has a special role as will be seen later.

Example

$Q_A = \{s_0, s_1, s_2\}$

Construction: States

The set of states is constructed as $Q_A = \{s_0, s_1, \ldots, s_k\}$.
The state s_0 has a special role as will be seen later.

Example

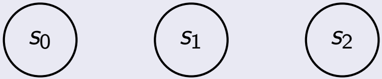
$Q_A = \{s_0, s_1, s_2\}$

Construction: States

The set of states is constructed as $Q_A = \{s_0, s_1, \ldots, s_k\}$.
The state s_0 has a special role as will be seen later.

Example

$Q_A = \{s_0, s_1, s_2\}$



Construction: Accepting function

The accepting function is constructed as $h(s_0, s_1, \ldots, s_k) = s_0$.

Example

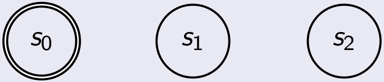
$h(s_0, s_1, s_2) = s_0$

Construction: Accepting function

The accepting function is constructed as $h(s_0, s_1, \ldots, s_k) = s_0$.

Example

$h(s_0, s_1, s_2) = s_0$



Construction: Final states

The final states are constructed as $F_A = \begin{cases} \{s_0\} & \text{if } q_0 \in F_D, \\ \emptyset & \text{otherwise.} \end{cases}$.

Example

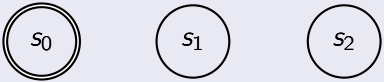
Start state is q_0 and $F_D = \{q_3\}$, therefore $F_A = \emptyset$. The characteristic vector now is $(0, 0, 0)$.

Construction: Final states

The final states are constructed as $F_A = \begin{cases} \{s_0\} & \text{if } q_0 \in F_D, \\ \emptyset & \text{otherwise.} \end{cases}$.

Example

Start state is q_0 and $F_D = \{q_3\}$, therefore $F_A = \emptyset$. The characteristic vector now is $(0, 0, 0)$.



Construction: Encoding

Choose an arbitrary bijection $\pi : Q_D \to B^k$ such that $\pi(q_0) = (0, \ldots, 0)$.

Identify $\pi(a)$, $a \in Q_D$, with an assignment of the states $s_1, \ldots, s_k$.

- This represents an encoding scheme of the states of the DFA A in B^k
- $\pi(s)$ is chosen as $(0, \ldots, 0)$ because of the definition of F_A
- The state s_0 is not considered by π

Example

State of A	0	1	2	3
Assignment of s_1 and s_2 under π	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$

Construction: Encoding

Choose an arbitrary bijection $\pi : Q_D \to B^k$ such that $\pi(q_0) = (0, \dots, 0)$.

Identify $\pi(a)$, $a \in Q_D$, with an assignment of the states $s_1, \dots, s_k$.

- This represents an encoding scheme of the states of the DFA A in B^k
- $\pi(s)$ is chosen as $(0, \dots, 0)$ because of the definition of F_A
- The state s_0 is not considered by π

Example

State of A	0	1	2	3
Assignment of s_1 and s_2 under π	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$

Construction: Encoding

Choose an arbitrary bijection $\pi : Q_D \to B^k$ such that $\pi(q_0) = (0, \dots, 0)$.

Identify $\pi(a)$, $a \in Q_D$, with an assignment of the states $s_1, \dots, s_k$.

- This represents an encoding scheme of the states of the DFA A in B^k
- $\pi(s)$ is chosen as $(0, \dots, 0)$ because of the definition of F_A
- The state s_0 is not considered by π

Example

State of A	0	1	2	3
Assignment of s_1 and s_2 under π	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$

Construction: Encoding

Choose an arbitrary bijection $\pi : Q_D \to B^k$ such that $\pi(q_0) = (0, \ldots, 0)$.

Identify $\pi(a)$, $a \in Q_D$, with an assignment of the states $s_1, \ldots, s_k$.

- This represents an encoding scheme of the states of the DFA A in B^k
- $\pi(s)$ is chosen as $(0, \ldots, 0)$ because of the definition of F_A
- The state s_0 is not considered by π

Example

State of A	0	1	2	3
Assignment of s_1 and s_2 under π	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$

Construction: Encoding

Choose an arbitrary bijection $\pi : Q_D \to B^k$ such that $\pi(q_0) = (0, \ldots, 0)$.

Identify $\pi(a)$, $a \in Q_D$, with an assignment of the states $s_1, \ldots, s_k$.

- This represents an encoding scheme of the states of the DFA A in B^k
- $\pi(s)$ is chosen as $(0, \ldots, 0)$ because of the definition of F_A
- The state s_0 is not considered by π

Example

State of A	0	1	2	3
Assignment of s_1 and s_2 under π	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$

Construction: Encoding

Choose an arbitrary bijection $\pi : Q_D \to B^k$ such that $\pi(q_0) = (0, \ldots, 0)$.

Identify $\pi(a)$, $a \in Q_D$, with an assignment of the states $s_1, \ldots, s_k$.

- This represents an encoding scheme of the states of the DFA A in B^k
- $\pi(s)$ is chosen as $(0, \ldots, 0)$ because of the definition of F_A
- The state s_0 is not considered by π

Example

State of A	0	1	2	3
Assignment of s_1 and s_2 under π	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$

Construction: Transition function

Let $\theta_1(x) = x$ and $\theta_0(x) = \overline{x}$.

For s_i, $i \neq 0$, the transition function is constructed as:

$$g(s_i, a, u) = \bigvee_{v \in B^k} (\pi(\delta(\pi^{-1}(v), a))_i \wedge \theta_{v_1}(u_1) \wedge \cdots \wedge \theta_{v_k}(u_k)).$$

Construction: Transition function

Let $\theta_1(x) = x$ and $\theta_0(x) = \overline{x}$.

For s_i, $i \neq 0$, the transition function is constructed as:

$$g(s_i, a, u) = \bigvee_{v \in B^k} (\pi(\delta(\pi^{-1}(v), a))_i \wedge \theta_{v_1}(u_1) \wedge \cdots \wedge \theta_{v_k}(u_k)).$$

Construction: Lemma

Lemma

Let $z, x \in B$. Then $\theta_z(x) = 1$ if and only if $z = x$.

Proof.

„$\Rightarrow$“: Let $\theta_z(x)$ be 1.

- $z = 1$: Then $\theta_z(x) = x$ and therefore $x = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$, therefore $\overline{x} = 1$ and thus $x = 0$.

„$\Leftarrow$“: Let $z = x$:

- $z = 1$: Then $\theta_z(x) = x$ and therefore $\theta_z(x) = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$ and therefore $\theta_z(x) = \overline{0} = 1$.

□

Construction: Lemma

Lemma

Let $z, x \in B$. Then $\theta_z(x) = 1$ if and only if $z = x$.

Proof.

„$\Rightarrow$“: Let $\theta_z(x)$ be 1.

- $z = 1$: Then $\theta_z(x) = x$ and therefore $x = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$, therefore $\overline{x} = 1$ and thus $x = 0$.

„$\Leftarrow$“: Let $z = x$:

- $z = 1$: Then $\theta_z(x) = x$ and therefore $\theta_z(x) = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$ and therefore $\theta_z(x) = \overline{0} = 1$.

$\square$

Construction: Lemma

Lemma

Let $z, x \in B$. Then $\theta_z(x) = 1$ if and only if $z = x$.

Proof.

„$\Rightarrow$“: Let $\theta_z(x)$ be 1.

- $z = 1$: Then $\theta_z(x) = x$ and therefore $x = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$, therefore $\overline{x} = 1$ and thus $x = 0$.

„$\Leftarrow$“: Let $z = x$:

- $z = 1$: Then $\theta_z(x) = x$ and therefore $\theta_z(x) = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$ and therefore $\theta_z(x) = \overline{0} = 1$.

□

Construction: Lemma

Lemma

Let $z, x \in B$. Then $\theta_z(x) = 1$ if and only if $z = x$.

Proof.

„$\Rightarrow$“: Let $\theta_z(x)$ be 1.

- $z = 1$: Then $\theta_z(x) = x$ and therefore $x = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$, therefore $\overline{x} = 1$ and thus $x = 0$.

„$\Leftarrow$“: Let $z = x$:

- $z = 1$: Then $\theta_z(x) = x$ and therefore $\theta_z(x) = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$ and therefore $\theta_z(x) = \overline{0} = 1$.

$\square$

Construction: Lemma

Lemma

Let $z, x \in B$. Then $\theta_z(x) = 1$ if and only if $z = x$.

Proof.

„$\Rightarrow$“: Let $\theta_z(x)$ be 1.

- $z = 1$: Then $\theta_z(x) = x$ and therefore $x = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$, therefore $\overline{x} = 1$ and thus $x = 0$.

„$\Leftarrow$“: Let $z = x$:

- $z = 1$: Then $\theta_z(x) = x$ and therefore $\theta_z(x) = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$ and therefore $\theta_z(x) = \overline{0} = 1$.

□

Construction: Lemma

Lemma

Let $z, x \in B$. Then $\theta_z(x) = 1$ if and only if $z = x$.

Proof.

„$\Rightarrow$“: Let $\theta_z(x)$ be 1.

- $z = 1$: Then $\theta_z(x) = x$ and therefore $x = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$, therefore $\overline{x} = 1$ and thus $x = 0$.

„$\Leftarrow$“: Let $z = x$:

- $z = 1$: Then $\theta_z(x) = x$ and therefore $\theta_z(x) = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$ and therefore $\theta_z(x) = \overline{0} = 1$.

$\square$

Construction: Lemma

Lemma

Let $z, x \in B$. Then $\theta_z(x) = 1$ if and only if $z = x$.

Proof.

„$\Rightarrow$“: Let $\theta_z(x)$ be 1.

- $z = 1$: Then $\theta_z(x) = x$ and therefore $x = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$, therefore $\overline{x} = 1$ and thus $x = 0$.

„$\Leftarrow$“: Let $z = x$:

- $z = 1$: Then $\theta_z(x) = x$ and therefore $\theta_z(x) = 1$.
- $z = 0$: Then $\theta_z(x) = \overline{x}$ and therefore $\theta_z(x) = \overline{0} = 1$.

□

Transition function: Details

Using the lemma the transition function can be rearranged as following:

$$
\begin{aligned}
g(s_i, a, u) &= \bigvee_{v \in B^k} (\pi(\delta(\pi^{-1}(v), a))_i \wedge \theta_{v_1}(u_1) \wedge \cdots \wedge \theta_{v_k}(u_k)) \\
&= \pi(\delta(\pi^{-1}(u_1, \ldots, u_k), a))_i
\end{aligned}
$$

- The transition function g directly represents the transitions of A in the encoding scheme!
- The reason for the initial notation is that in this way it can be represented more easily as a Boolean function.

Transition function: Details

Using the lemma the transition function can be rearranged as following:

$$\begin{aligned} g(s_i, a, u) &= \bigvee_{v \in B^k} (\pi(\delta(\pi^{-1}(v), a))_i \wedge \theta_{v_1}(u_1) \wedge \cdots \wedge \theta_{v_k}(u_k)) \\ &= \pi(\delta(\pi^{-1}(u_1, \ldots, u_k), a))_i \end{aligned}$$

- The transition function g directly represents the transitions of A in the encoding scheme!
- The reason for the initial notation is that in this way it can be represented more easily as a Boolean function.

Transition function: Details

Using the lemma the transition function can be rearranged as following:

$$\begin{aligned} g(s_i, a, u) &= \bigvee_{v \in B^k} (\pi(\delta(\pi^{-1}(v), a))_i \wedge \theta_{v_1}(u_1) \wedge \cdots \wedge \theta_{v_k}(u_k)) \\ &= \pi(\delta(\pi^{-1}(u_1, \ldots, u_k), a))_i \end{aligned}$$

- The transition function g directly represents the transitions of A in the encoding scheme!
- The reason for the initial notation is that in this way it can be represented more easily as a Boolean function.

Transition function: Details

For s_0 the transition function is constructed as:

$$g(s_0, a, u) = \bigvee_{q \in F_D} \theta_{\pi(q)_1}(g(s_1, a, u)) \wedge \cdots \wedge \theta_{\pi(q)_k}(g(s_k, a, u))$$

- Again we consider the lemma: $g(s_0, a, u)$ is true iff the encoding of at least one of the final states of A is the current assignment of the AFA.
- Because of $h(s_0, s_1, \ldots, s_k) = s_0$, the state s_0 is the only state which needs to be considered for acceptance.

Transition function: Details

For s_0 the transition function is constructed as:

$$g(s_0, a, u) = \bigvee_{q \in F_D} \theta_{\pi(q)_1}(g(s_1, a, u)) \wedge \cdots \wedge \theta_{\pi(q)_k}(g(s_k, a, u))$$

- Again we consider the lemma: $g(s_0, a, u)$ is true iff the encoding of at least one of the final states of A is the current assignment of the AFA.
- Because of $h(s_0, s_1, \ldots, s_k) = s_0$, the state s_0 is the only state which needs to be considered for acceptance.

Transition function: Details

For s_0 the transition function is constructed as:

$$g(s_0, a, u) = \bigvee_{q \in F_D} \theta_{\pi(q)_1}(g(s_1, a, u)) \wedge \cdots \wedge \theta_{\pi(q)_k}(g(s_k, a, u))$$

- Again we consider the lemma: $g(s_0, a, u)$ is true iff the encoding of at least one of the final states of A is the current assignment of the AFA.
- Because of $h(s_0, s_1, \ldots, s_k) = s_0$, the state s_0 is the only state which needs to be considered for acceptance.

Construction: Transition function

Example

$$
\begin{aligned}
g(s_1, a, u) = & \bigvee_{v \in B^k} (\pi(\delta(\pi^{-1}(v), a))_1 \wedge \theta_{v_1}(u_1) \wedge \theta_{v_2}(u_2)) \\
= & (\pi(\delta(\pi^{-1}(00), a))_1 \wedge \theta_0(u_1) \wedge \theta_0(u_2)) \\
& \vee (\pi(\delta(\pi^{-1}(01), a))_1 \wedge \theta_0(u_1) \wedge \theta_1(u_2)) \\
& \vee (\pi(\delta(\pi^{-1}(10), a))_1 \wedge \theta_1(u_1) \wedge \theta_0(u_2)) \\
& \vee (\pi(\delta(\pi^{-1}(11), a))_1 \wedge \theta_1(u_1) \wedge \theta_1(u_2)) \\
= & (0 \wedge \overline{u_1} \wedge \overline{u_2}) \vee (0 \wedge \overline{u_1} \wedge u_2) \\
& \vee (0 \wedge u_1 \wedge \overline{u_2}) \vee (1 \wedge u_1 \wedge u_2) \\
= & u_1 \wedge u_2
\end{aligned}
$$

Construction: Transition function

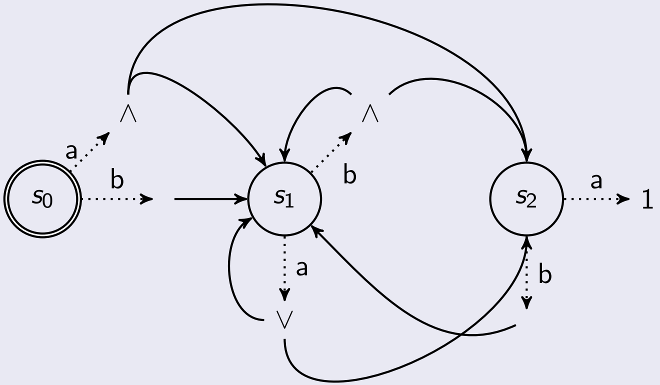
Example

Overall the transition function is:

g	a	b
s_0	$u_1 \wedge u_2$	u_1
s_1	$u_1 \wedge u_2$	$u_1 \vee u_2$
s_2	1	u_1

Construction: Transition function

Example



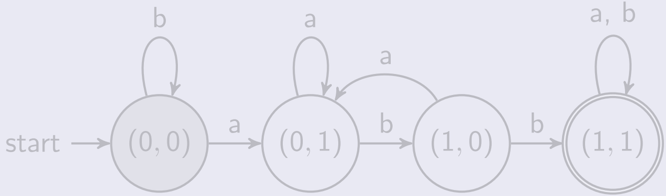
Example run

Example

Consider the word $w = abb$.

- w is accepted by $A \Leftrightarrow w^R$ is accepted by the constructed AFA
- The word w is accepted iff $h(g(w^R, f)) = 1$, where f is the characteristic vector $(0, 0, 0)$
- Only the last two numbers of a vector encode the state, the first represents the state of s_0

$h(g(bba, f)) = h(g(b, g(b, g(a, (0, 0, 0)))))$



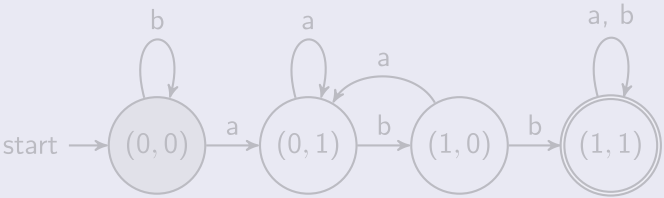
Example run

Example

Consider the word $w = abb$.

- w is accepted by $A \Leftrightarrow w^R$ is accepted by the constructed AFA
- The word w is accepted iff $h(g(w^R, f)) = 1$, where f is the characteristic vector $(0, 0, 0)$
- Only the last two numbers of a vector encode the state, the first represents the state of s_0

$$h(g(bba, f)) = h(g(b, g(b, g(a, (0, 0, 0)))))$$



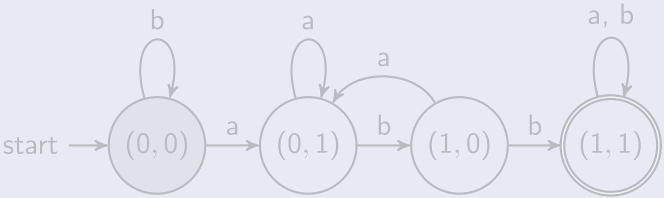
Example run

Example

Consider the word $w = abb$.

- w is accepted by $A \Leftrightarrow w^R$ is accepted by the constructed AFA
- The word w is accepted iff $h(g(w^R, f)) = 1$, where f is the characteristic vector $(0, 0, 0)$
- Only the last two numbers of a vector encode the state, the first represents the state of s_0

$h(g(bba, f)) = h(g(b, g(b, g(a, (0, 0, 0)))))$



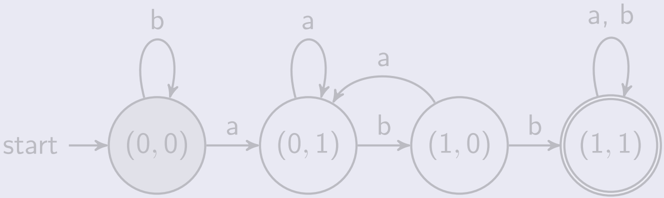
Example run

Example

Consider the word $w = abb$.

- w is accepted by $A \Leftrightarrow w^R$ is accepted by the constructed AFA
- The word w is accepted iff $h(g(w^R, f)) = 1$, where f is the characteristic vector $(0, 0, 0)$
- Only the last two numbers of a vector encode the state, the first represents the state of s_0

$h(g(bba, f)) = h(g(b, g(b, g(a, (0, 0, 0)))))$



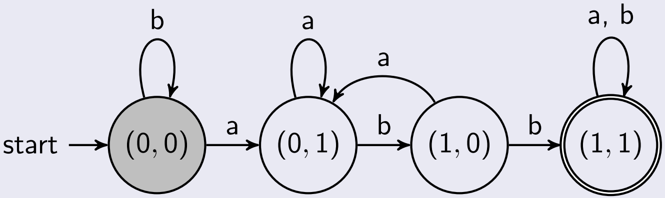
Example run

Example

Consider the word $w = abb$.

- w is accepted by $A \Leftrightarrow w^R$ is accepted by the constructed AFA
- The word w is accepted iff $h(g(w^R, f)) = 1$, where f is the characteristic vector $(0,0,0)$
- Only the last two numbers of a vector encode the state, the first represents the state of s_0

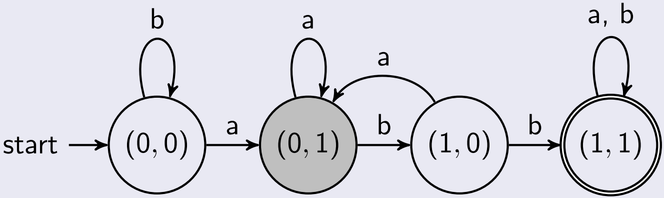
$$h(g(bba, f)) = h(g(b, g(b, g(a, (0,0,0)))))$$



Example run

Example

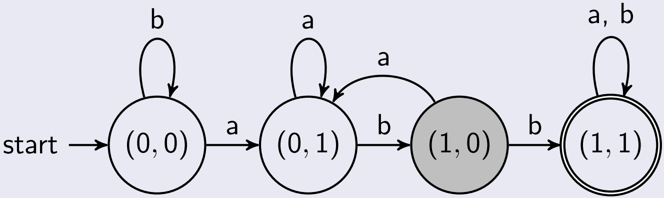
$h(g(bba, f)) = h(g(b, g(b, (0, 0, 1))))$



Example run

Example

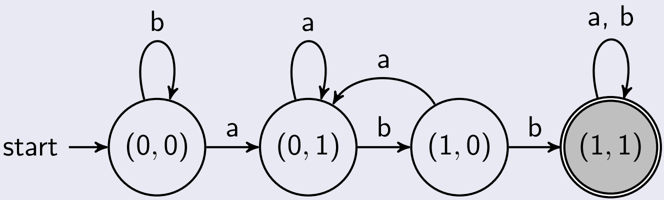
$h(g(bba, f)) = h(g(b, (0, 1, 0)))$



Example run

Example

$h(g(bba, f)) = h(1, 1, 1) = 1$



Bit-wise implementation

Transformation DFA to AFA: Observations

- Complexity of states of the DFA is transformed to complexity of the transition function of the AFA
- How can the transition function be represented efficiently?
- Is there a efficient representation of Boolean functions?

Transformation DFA to AFA: Observations

- Complexity of states of the DFA is transformed to complexity of the transition function of the AFA
- How can the transition function be represented efficiently?
- Is there a efficient representation of Boolean functions?

Transformation DFA to AFA: Observations

- Complexity of states of the DFA is transformed to complexity of the transition function of the AFA
- How can the transition function be represented efficiently?
- Is there a efficient representation of Boolean functions?

Basic definitions

Let $S = \{x_1, \ldots, x_n\}$ a set of Boolean variables, and $\overline{S} = \{\overline{x_1}, \ldots, \overline{x_n}\}$.

Definition

A term t defined on $S \cup \overline{S}$ is a conjunction

$$t = y_1 \wedge \cdots \wedge y_k,\ 1 \leq k \leq n$$

where $y_i \in S \cup \overline{S}$, $y_i \neq y_j$, $y_i \neq \overline{y_j}$ for $1 \leq i < j \leq k$, or t is constant.

Definition

A Boolean expression f is said to be in disjunctive normal form if $f = \bigvee_{i=1}^{k} t_i$, where t_i, $i = 1, \ldots, k$, is a term defined on $S \cup \overline{S}$.

Basic definitions

Let $S = \{x_1, \ldots, x_n\}$ a set of Boolean variables, and $\overline{S} = \{\overline{x_1}, \ldots, \overline{x_n}\}$.

Definition

A term t defined on $S \cup \overline{S}$ is a conjunction

$$t = y_1 \wedge \cdots \wedge y_k,\ 1 \leq k \leq n$$

where $y_i \in S \cup \overline{S}$, $y_i \neq y_j$, $y_i \neq \overline{y_j}$ for $1 \leq i < j \leq k$, or t is constant.

Definition

A Boolean expression f is said to be in disjunctive normal form if $f = \bigvee_{i=1}^{k} t_i$, where t_i, $i = 1, \ldots, k$, is a term defined on $S \cup \overline{S}$.

Theorem: Bit-wise representation of Boolean functions

Theorem

For every Boolean function f defined on S that can be expressed as a single term, there exist two n-bit vectors α and β such that for all $u \in B^n$

$$f(u) = 1 \Leftrightarrow (\alpha \& u) \uparrow \beta = 0$$

where & is the bit-wise AND operator, $\uparrow$ the bit-wise exclusive-or operator, and 0 is the zero vector $(0, \ldots, 0) \in B^n$.

Using this theorem we can represent a term of a Boolean function as two n-bit integers.

Theorem: Bit-wise representation of Boolean functions

Theorem

For every Boolean function f defined on S that can be expressed as a single term, there exist two n-bit vectors α and β such that for all $u \in B^n$

$$f(u) = 1 \Leftrightarrow (\alpha \& u) \uparrow \beta = 0$$

where & is the bit-wise AND operator, $\uparrow$ the bit-wise exclusive-or operator, and 0 is the zero vector $(0, \ldots, 0) \in B^n$.

Using this theorem we can represent a term of a Boolean function as two n-bit integers.

Proof

Proof.

Let $f = y_{i_1} \wedge \cdots \wedge y_{i_k}$, where $y_{i_j} = x_{i_j}$ or $\overline{x_{i_j}}$, $i_j \neq i_{j'}$ for $j \neq j'$.

Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be defined as follows:

- $\alpha_i = 1$ iff x_i or $\overline{x_i}$ appears in f
- $\beta_i = 1$ iff x_i appears in f

Then $(\alpha \& u)_i = 1 \Leftrightarrow u_i = 1$ and (x_i or $\overline{x_i}$ appears in f).

- Case 1: Neither x_i nor $\overline{x_i}$ appear in f, then $((\alpha \& u) \uparrow \beta)_i = 0$
- Case 2: x_i appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 1$
- Case 3: $\overline{x_i}$ appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 0$

All in all $f(u) = 1$ iff $(\alpha \& u) \uparrow \beta = 0$. □

Proof

Proof.

Let $f = y_{i_1} \wedge \cdots \wedge y_{i_k}$, where $y_{i_j} = x_{i_j}$ or $\overline{x_{i_j}}$, $i_j \neq i_{j'}$ for $j \neq j'$.
Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be defined as follows:

- $\alpha_i = 1$ iff x_i or $\overline{x_i}$ appears in f
- $\beta_i = 1$ iff x_i appears in f

Then $(\alpha \& u)_i = 1 \Leftrightarrow u_i = 1$ and (x_i or $\overline{x_i}$ appears in f).

- Case 1: Neither x_i nor $\overline{x_i}$ appear in f, then $((\alpha \& u) \uparrow \beta)_i = 0$
- Case 2: x_i appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 1$
- Case 3: $\overline{x_i}$ appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 0$

All in all $f(u) = 1$ iff $(\alpha \& u) \uparrow \beta = 0$.

□

Proof

Proof.

Let $f = y_{i_1} \wedge \cdots \wedge y_{i_k}$, where $y_{i_j} = x_{i_j}$ or $\overline{x_{i_j}}$, $i_j \neq i_{j'}$ for $j \neq j'$.
Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be defined as follows:

- $\alpha_i = 1$ iff x_i or $\overline{x_i}$ appears in f
- $\beta_i = 1$ iff x_i appears in f

Then $(\alpha \& u)_i = 1 \Leftrightarrow u_i = 1$ and (x_i or $\overline{x_i}$ appears in f).

- Case 1: Neither x_i nor $\overline{x_i}$ appear in f, then $((\alpha \& u) \uparrow \beta)_i = 0$
- Case 2: x_i appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 1$
- Case 3: $\overline{x_i}$ appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 0$

All in all $f(u) = 1$ iff $(\alpha \& u) \uparrow \beta = 0$. □

Proof

Proof.

Let $f = y_{i_1} \wedge \cdots \wedge y_{i_k}$, where $y_{i_j} = x_{i_j}$ or $\overline{x_{i_j}}$, $i_j \neq i_{j'}$ for $j \neq j'$.
Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be defined as follows:

- $\alpha_i = 1$ iff x_i or $\overline{x_i}$ appears in f
- $\beta_i = 1$ iff x_i appears in f

Then $(\alpha \& u)_i = 1 \Leftrightarrow u_i = 1$ and (x_i or $\overline{x_i}$ appears in f).

- Case 1: Neither x_i nor $\overline{x_i}$ appear in f, then $((\alpha \& u) \uparrow \beta)_i = 0$
- Case 2: x_i appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 1$
- Case 3: $\overline{x_i}$ appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 0$

All in all $f(u) = 1$ iff $(\alpha \& u) \uparrow \beta = 0$.

□

Proof

Proof.

Let $f = y_{i_1} \wedge \cdots \wedge y_{i_k}$, where $y_{i_j} = x_{i_j}$ or $\overline{x_{i_j}}$, $i_j \neq i_{j'}$ for $j \neq j'$.
Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be defined as follows:

- $\alpha_i = 1$ iff x_i or $\overline{x_i}$ appears in f
- $\beta_i = 1$ iff x_i appears in f

Then $(\alpha \& u)_i = 1 \Leftrightarrow u_i = 1$ and (x_i or $\overline{x_i}$ appears in f).

- Case 1: Neither x_i nor $\overline{x_i}$ appear in f, then $((\alpha \& u) \uparrow \beta)_i = 0$
- Case 2: x_i appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 1$
- Case 3: $\overline{x_i}$ appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 0$

All in all $f(u) = 1$ iff $(\alpha \& u) \uparrow \beta = 0$.

□

Proof

Proof.

Let $f = y_{i_1} \wedge \cdots \wedge y_{i_k}$, where $y_{i_j} = x_{i_j}$ or $\overline{x_{i_j}}$, $i_j \neq i_{j'}$ for $j \neq j'$.
Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be defined as follows:

- $\alpha_i = 1$ iff x_i or $\overline{x_i}$ appears in f
- $\beta_i = 1$ iff x_i appears in f

Then $(\alpha \& u)_i = 1 \Leftrightarrow u_i = 1$ and (x_i or $\overline{x_i}$ appears in f).

- Case 1: Neither x_i nor $\overline{x_i}$ appear in f, then $((\alpha \& u) \uparrow \beta)_i = 0$
- Case 2: x_i appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 1$
- Case 3: $\overline{x_i}$ appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 0$

All in all $f(u) = 1$ iff $(\alpha \& u) \uparrow \beta = 0$.

□

Proof

Proof.

Let $f = y_{i_1} \wedge \cdots \wedge y_{i_k}$, where $y_{i_j} = x_{i_j}$ or $\overline{x_{i_j}}$, $i_j \neq i_{j'}$ for $j \neq j'$.
Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be defined as follows:

- $\alpha_i = 1$ iff x_i or $\overline{x_i}$ appears in f
- $\beta_i = 1$ iff x_i appears in f

Then $(\alpha \& u)_i = 1 \Leftrightarrow u_i = 1$ and (x_i or $\overline{x_i}$ appears in f).

- Case 1: Neither x_i nor $\overline{x_i}$ appear in f, then $((\alpha \& u) \uparrow \beta)_i = 0$
- Case 2: x_i appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 1$
- Case 3: $\overline{x_i}$ appears in f, then $((\alpha \& u) \uparrow \beta)_i = 0$ iff $u_i = 0$

All in all $f(u) = 1$ iff $(\alpha \& u) \uparrow \beta = 0$.

$\square$

Theorem: Example

Example

Consider $f(u_1, u_2) = u_1 \wedge \overline{u_2}$. Then:

- $\alpha = (1, 1)$
- $\beta = (1, 0)$

Therefore:

$$\begin{aligned} f(1,0) &= (\alpha \& (1,0)) \uparrow \beta \\ &= ((1,1)\&(1,0)) \uparrow (1,0) \\ &= (1,0) \uparrow (1,0) \\ &= (0,0) \end{aligned}$$

Theorem: Example

Example

Consider $f(u_1, u_2) = u_1 \wedge \overline{u_2}$. Then:

- $\alpha = (1, 1)$
- $\beta = (1, 0)$

Therefore:

$$\begin{aligned} f(1,0) &= (\alpha \& (1,0)) \uparrow \beta \\ &= ((1,1)\&(1,0)) \uparrow (1,0) \\ &= (1,0) \uparrow (1,0) \\ &= (0,0) \end{aligned}$$

Theorem: Example

Example

Consider $f(u_1, u_2) = u_1 \wedge \overline{u_2}$. Then:

- $\alpha = (1, 1)$
- $\beta = (1, 0)$

Therefore:

$$\begin{aligned} f(1,0) &= (\alpha \& (1,0)) \uparrow \beta \\ &= ((1,1)\&(1,0)) \uparrow (1,0) \\ &= (1,0) \uparrow (1,0) \\ &= (0,0) \end{aligned}$$

Theorem: Example

Example

Consider $f(u_1, u_2) = u_1 \wedge \overline{u_2}$. Then:

- $\alpha = (1,1)$
- $\beta = (1,0)$

Therefore:

$$\begin{aligned} f(1,0) &= (\alpha \& (1,0)) \uparrow \beta \\ &= ((1,1)\&(1,0)) \uparrow (1,0) \\ &= (1,0) \uparrow (1,0) \\ &= (0,0) \end{aligned}$$

Example

Example

Consider again the transition function (and the transition function of the DFA):

g	a	b
s_0	$u_1 \wedge u_2$	u_1
s_1	$u_1 \wedge u_2$	$u_1 \vee u_2$
s_2	1	u_1

State	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

This gives the following representation (compared to the DFA):

g	a	b
s_0	$((11),(11))$	$((10),(10))$
s_1	$((11),(11))$	$((10),(10)),((01),(01))$
s_2	$((00),(00))$	$((10),(10))$

Example

Example

Consider again the transition function (and the transition function of the DFA):

g	a	b
s_0	$u_1 \wedge u_2$	u_1
s_1	$u_1 \wedge u_2$	$u_1 \vee u_2$
s_2	1	u_1

State	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

This gives the following representation (compared to the DFA):

g	a	b
s_0	$((11),(11))$	$((10),(10))$
s_1	$((11),(11))$	$((10),(10)),((01),(01))$
s_2	$((00),(00))$	$((10),(10))$

Example: 2^{32} state DFA

- A DFA A with 2^{32} states can be represented as an AFA A' with 32 states
- The transition function g of A' can be represented as a $32 \times |\Sigma|$-Matrix, where Σ is the input alphabet of A and A'
- Every entry of the matrix representation of g can be represented as a List of pairs of integers
- All this results in an efficient way for representing DFAs

Example: 2^{32} state DFA

- A DFA A with 2^{32} states can be represented as an AFA A' with 32 states
- The transition function g of A' can be represented as a $32 \times |\Sigma|$-Matrix, where Σ is the input alphabet of A and A'
- Every entry of the matrix representation of g can be represented as a List of pairs of integers
- All this results in an efficient way for representing DFAs

Example: 2^{32} state DFA

- A DFA A with 2^{32} states can be represented as an AFA A' with 32 states
- The transition function g of A' can be represented as a $32 \times |\Sigma|$-Matrix, where Σ is the input alphabet of A and A'
- Every entry of the matrix representation of g can be represented as a List of pairs of integers
- All this results in an efficient way for representing DFAs

Example: 2^{32} state DFA

- A DFA A with 2^{32} states can be represented as an AFA A' with 32 states
- The transition function g of A' can be represented as a $32 \times |\Sigma|$-Matrix, where Σ is the input alphabet of A and A'
- Every entry of the matrix representation of g can be represented as a List of pairs of integers
- All this results in an efficient way for representing DFAs

Conclusion

Conclusion

- AFAs are an efficient way to represent DFAs
- It is even more efficient using a bit-wise representation of the transition function

Furthermore:

- Operations like the star operation, concatenation or reversal can also be implemented more efficiently

Conclusion

- AFAs are an efficient way to represent DFAs
- It is even more efficient using a bit-wise representation of the transition function

Furthermore:

- Operations like the star operation, concatenation or reversal can also be implemented more efficiently

Conclusion

- AFAs are an efficient way to represent DFAs
- It is even more efficient using a bit-wise representation of the transition function

Furthermore:

- Operations like the star operation, concatenation or reversal can also be implemented more efficiently

Literatur

Champarnaud, Jean-Marc, Denis Maurel und Djelloul Ziadi (Herausgeber): *Automata Implementation, Third International Workshop on Implementing Automata, WIA'98, Rouen, France, September 17-19, 1998, Revised Papers*, Band 1660 der Reihe *Lecture Notes in Computer Science*. Springer, 1999.

Huerter, Sandra, Kai Salomaa, Xiuming Wu und Sheng Yu: *Implementing Reversed Alternating Finite Automaton (r-AFA) Operations*.
In: Champarnaud, Jean-Marc et al. [CMZ99], Seiten 69–81.

Salomaa, Kai, Xiuming Wu und Sheng Yu: *Efficient Implementation of Regular Languages Using R-AFA*.
In: Wood, Derick und Sheng Yu [WY98], Seiten 176–184.

Literatur

- WOOD, DERICK und SHENG YU (Herausgeber): *Automata Implementation, Second International Workshop on Implementing Automata, WIA '97, London, Ontario, Canada, September 18-20, 1997, Revised Papers*, Band 1436 der Reihe *Lecture Notes in Computer Science*. Springer, 1998.

- YU, SHENG: *Regular languages*, Band 1 der Reihe *Handbook of formal languages*, Seiten 41–110.
  Springer-Verlag New York, Inc., New York, NY, USA, 1997.