Fundamentele Informatica 1 (I&E)

najaar 2015

`http://www.liacs.leidenuniv.nl/~vlietrvan1/fi1ie/`

Rudy van Vliet
kamer 124 Snellius, tel. 071-527 5777
rvvliet(at)liacs(dot)nl

college 2, vrijdag 30 oktober 2015

2.1 Finite Automata: Examples and Definitions

2.2 Accepting the Union, Intersection, or Difference of Two Languages

reg. languages	FA	reg. grammar	reg. expression
determ. cf. languages	DPDA		
cf. languages	PDA	cf. grammar	
re. languages	TM	unrestr. grammar	

Binary representations of 0,1,2,. . . ,10. . .

Binary representations of 0,1,2,. . . ,10. . .

Example 2.7.

A finite automaton for accepting

$L_4 = \{x \in \{0,1\}^* \mid$ x is binary representation of integer divisible by 3$\}$

Example 2.9.

A finite automaton for lexical analysis:
accepting strings that consist of one or more consecutive ‘tokens’ in a programming language

Possible tokens:

identifier   ; $=$ if   numeric_literal

Definition 2.11. A Finite Automaton

A *finite automaton* (FA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where
Q is a finite set of *states*
Σ is a finite *input alphabet*
$q_0 \in Q$ is the *initial* state
$A \subseteq Q$ is the set of *accepting* states
$\delta : \ldots$ is the *transition* function

Example...

Definition 2.11. A Finite Automaton

A *finite automaton* (FA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where
Q is a finite set of *states*
Σ is a finite *input alphabet*
$q_0 \in Q$ is the *initial* state
$A \subseteq Q$ is the set of *accepting* states
$\delta : Q \times \Sigma \to Q$ is the *transition* function

For any state q of Q and any symbol $\sigma \in \Sigma$,
we interpret $\delta(q, \sigma)$ as the state to which the FA moves,
if it is in state q and receives the input σ.

Definition 2.12. The Extended Transition Function δ^*

Where do we go to from state q,
if we receive a string $x \in \Sigma^*$ as input?

Let $M = (Q, \Sigma, q_0, A, \delta)$ be a finite automaton. We define the extended transition function

$$\delta^* : Q \times \Sigma^* \to Q$$

as follows:

1. For every $q \in Q$, $\delta^*(q, \Lambda) = q$
2. For every $q \in Q$, every $y \in \Sigma^*$, and every $\sigma \in \Sigma$,
   $\delta^*(q, y\sigma) = \ldots$

Definition 2.12. The Extended Transition Function δ^*

Where do we go to from state q,
if we receive a string $x \in \Sigma^*$ as input?

Let $M = (Q, \Sigma, q_0, A, \delta)$ be a finite automaton. We define the extended transition function

$$\delta^* : Q \times \Sigma^* \to Q$$

as follows:

1. For every $q \in Q$, $\delta^*(q, \Lambda) = q$
2. For every $q \in Q$, every $y \in \Sigma^*$, and every $\sigma \in \Sigma$,
   $$\delta^*(q, y\sigma) = \delta(\delta^*(q, y), \sigma)$$

Recursive definition

Definition 2.12. The Extended Transition Function δ^*

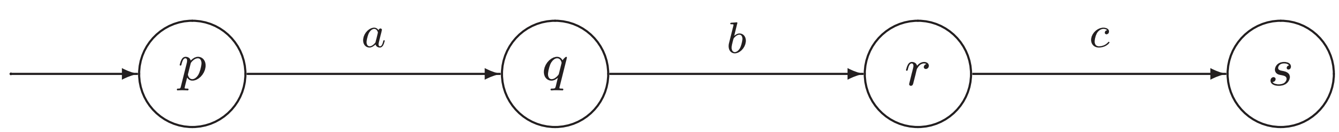
Where do we go to from state q,
if we receive a string $x \in \Sigma^*$ as input?

Let $M = (Q, \Sigma, q_0, A, \delta)$ be a finite automaton. We define the extended transition function

$\delta^* : Q \times \Sigma^* \to Q$

as follows:

1. For every $q \in Q$, $\delta^*(q, \Lambda) = q$
2. For every $q \in Q$, every $y \in \Sigma^*$, and every $\sigma \in \Sigma$,
   $\delta^*(q, y\sigma) = \delta(\delta^*(q, y), \sigma)$



$\delta^*(p, abc) = \ldots$

Definition 2.13. Acceptance by a Finite Automaton

Let $M = (Q, \Sigma, q_0, A, \delta)$ be a finite automaton, and let $x \in \Sigma^*$.

The string x is *accepted* by M if

$$\delta^*(q_0, x) \in A$$

and is *rejected* by M otherwise.

The *language* accepted by M is the set

$$L(M) = \{x \in \Sigma^* \mid \ x \text{ is accepted by } M\}$$

If L is a language over Σ,
L is accepted by M if and only if $L = L(M)$.
M accepts L and nothing more!

2.2 Accepting the Union, Intersection, or Difference of Two Languages

Example 2.16.

Let

$$L_1 = \{x \in \{a,b\}^* \mid \ aa \text{ is not a substring of } x\}$$
$$L_2 = \{x \in \{a,b\}^* \mid \ x \text{ ends with } ab\}$$

FAs for: L_1, L_2,

$L_1 \cup L_2$

Theorem 2.15.
Suppose $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ are finite automata accepting L_1 and L_2, respectively.
Let M be the FA $(Q, \Sigma, q_0, A, \delta)$, where

$Q = Q_1 \times Q_2$

$q_0 = (q_1, q_2)$

and the transition function δ is defined by the formula

$\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$

for every $p \in Q_1$, every $q \in Q_2$, and every $\sigma \in \Sigma$.

Then

1. If $A = \{(p, q) |\ p \in A_1$ or $q \in A_2\}$,
   M accepts the language $L_1 \cup L_2$.
2. If . . .
   M accepts the language $L_1 \cap L_2$.
3. If . . .
   M accepts the language $L_1 - L_2$.

Theorem 2.15.
Suppose $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ are finite automata accepting L_1 and L_2, respectively.
Let M be the FA $(Q, \Sigma, q_0, A, \delta)$, where

$Q = Q_1 \times Q_2$

$q_0 = (q_1, q_2)$

and the transition function δ is defined by the formula

$\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$

for every $p \in Q_1$, every $q \in Q_2$, and every $\sigma \in \Sigma$.

Then
1. If $A = \{(p, q) |\ p \in A_1$ or $q \in A_2\}$,
   M accepts the language $L_1 \cup L_2$.
2. If $A = \{(p, q) |\ p \in A_1$ and $q \in A_2\}$,
   M accepts the language $L_1 \cap L_2$.
3. If $A = \{(p, q) |\ p \in A_1$ and $q \notin A_2\}$,
   M accepts the language $L_1 - L_2$.

Example 2.16.

Let

$$L_1 = \{x \in \{a,b\}^* \mid \ aa \text{ is not a substring of } x\}$$
$$L_2 = \{x \in \{a,b\}^* \mid \ x \text{ ends with } ab\}$$

FAs for: L_1, L_2,

$L_1 \cup L_2$, $L_1 \cap L_2$, and $L_1 - L_2$

simplification possible

Example 2.18.

Let

$$L_1 = \{x \in \{a,b\}^* \mid \ x \text{ contains the substring } ab\}$$
$$L_2 = \{x \in \{a,b\}^* \mid \ x \text{ contains the substring } bba\}$$

FAs for: L_1, L_2,

and $L_1 \cup L_2 = \{x \in \{a,b\}^* \mid \ x \text{ contains either } ab \text{ or } bba\}$

(two versions)

Corollary.

For every finite language . . .

Corollary.
For every finite language $L = \{x_1, x_2, \ldots, x_n\}$ for some $n \geq 0$, we can construct an FA accepting L.

Proof

$L = \{x_1\} \cup \{x_2\} \cup \ldots \cup \{x_n\}$