ECOLE POLYTECHNIQUE DE LOUVAIN

LINGI2132 - LANGUAGES AND TRANSLATORS

Assignement 2 - Report

Professor :

Pierre SCHAUS

Students :

Benoît BAUFAYS 22200900

Julien COLMONTS 41630800

Program :

SINF21MS

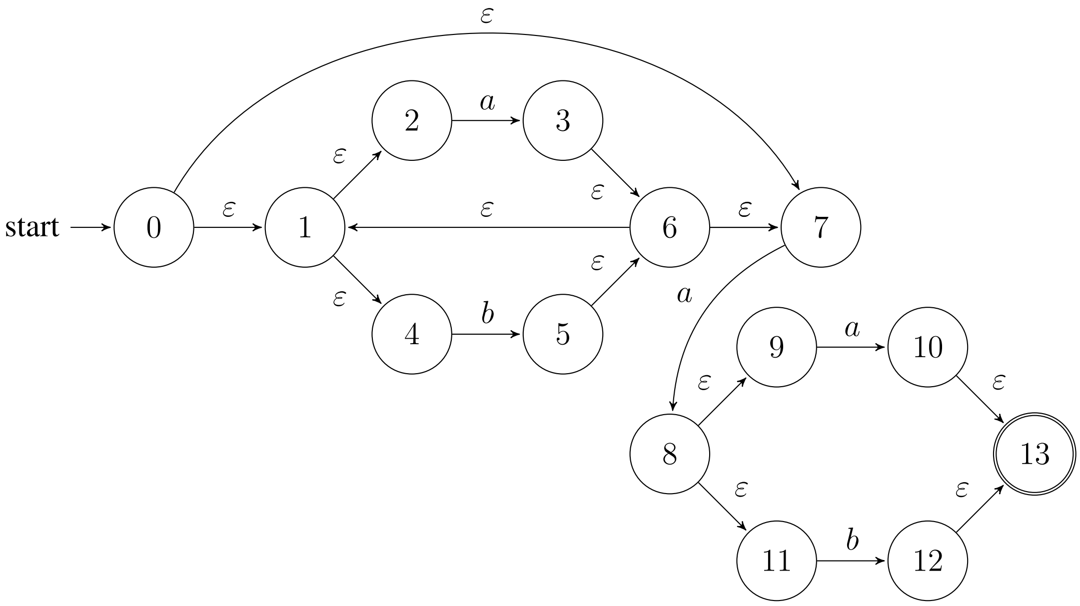
Academic Year 2013-2014

1 Lexical Analysis

1.1 Give 5 different strings belonging to the language described by this regExp.

1. aab
2. bbaa
3. ababab
4. aaaaaaa
5. bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbab

1.2 Construct the NFA from this regExp (Thompson Construction



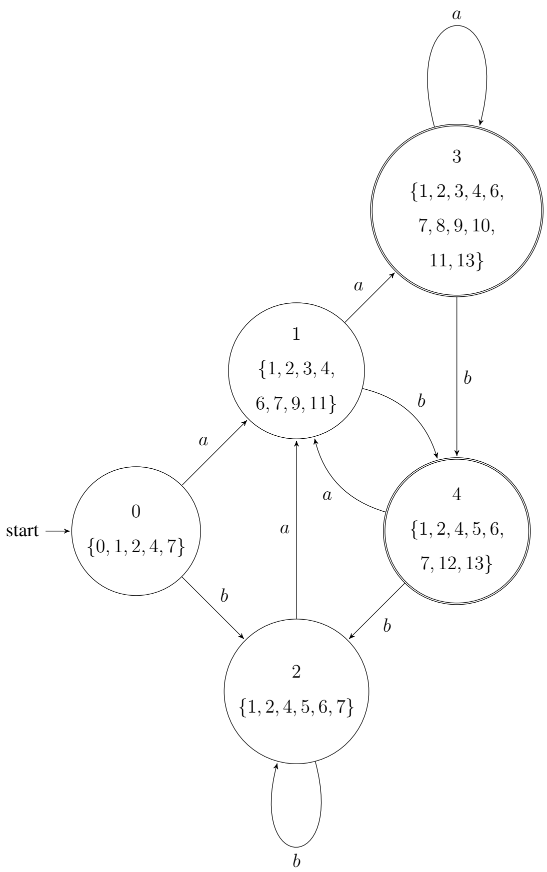
1.3 Transform the NFA into a DFA (justify important steps, ε-closures).

Steps from NFA to DFA :

- $s_0 = \epsilon - closure(\{0\}) = \{0, 1, 2, 4, 7\}$
- $m(s_0, a) = s_1$, where $s_1 = \epsilon - closure(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 9, 11\}$

- $m(s_0, b) = s_2$, where $s_2 = \epsilon - closure(\{5\}) = \{1, 2, 4, 5, 6, 7\}$
- $m(s_1, a) = s_3$, where $s_3 = \epsilon - closure(\{3, 8, 10\}) = \{1, 2, 3, 4, 6, 7, 8, 9, 11, 13\}$
- $m(s_1, b) = s_4$, where $s_4 = \epsilon - closure(\{5, 12\}) = \{1, 2, 4, 5, 6, 7, 12, 13\}$
- $m(s_2, a) = s_1$
- $m(s_2, b) = s_2$
- $m(s_3, a) = s_3$
- $m(s_3, b) = s_4$
- $m(s_4, a) = s_1$
- $m(s_4, b) = s_2$

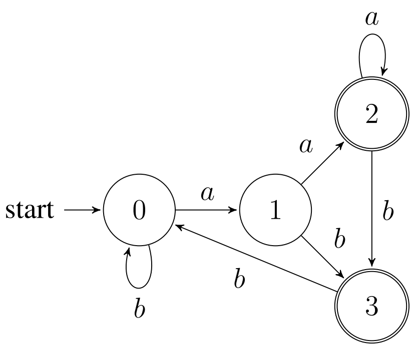
Since s_3 and s_4 contain state 13 which was final in the NFA, they are both final state too.



1.4 Minimize the DFA (Hopcroft Algorithm).

The DFA could be divided in four partitions :

- Two partitions for final states
- A partition which contains previous states $\{0, 2\}$ because :
  - $m(0, a) = 1$
  - $m(2, a) = 1$
  - $m(0, b) = 2$
  - $m(2, b) = 2$
- A partition which contains previous state $\{1\}$:
  - $m(1, a) = 3$
  - $m(1, b) = 4$



2 Parsing

2.1 LL(1) Grammar

This grammar is not LL(1) because it has a rule with left recursion: $B := Bv$ and $B := w$.

2.2 Modify the grammar

We replaced the left recursion with a right recursion to avoid the problem indicated in the first exercise.

$$B := wB'$$
$$B' := vB'$$
$$B' := \epsilon$$

Thus, we removed from the grammar the rules $B := Bv$ and $B := w$ and we introduced rules above.

2.3 Sets for this grammar

$first(u) = \{u\}$

$first(v) = \{v\}$

$first(w) = \{w\}$

$first(x) = \{x\}$

$first(y) = \{y\}$

$first(z) = \{z\}$

$\epsilon \in first(E), first(F), first(B')$

Rule 1:$S :== uBDz$

$first(uBDz) = \{u\}$

$\rightarrow u \in first(S)$

Rule 2: $B ::= wB'$

$first(wB') = \{w\}$

$\rightarrow w \in first(B)$

Rule 3: $B' ::= wB'$

$first(vB') = \{v\}$

$\rightarrow v \in first(B')$

Rule 5: $D ::= EF$

$first(EF) = first(E)\backslash\{\epsilon\} \cup first(F)$ (car $E \in first(E)$)= $\{x, y, E\}$

$\rightarrow x, y, E \subset first(D)$

Rule 6: $E ::= y$

$y \in first(E)$

Rule 8: $F ::= x$

$x \in first(F)$

$first(S) = \{u\}$

$first(B) = \{w\}$

$first(B') = \{v, \epsilon\}$

$first(D) = \{x, y, \epsilon\}$

$first(E) = \{y, \epsilon\}$

$first(F) = \{x, \epsilon\}$

Follow

$\#E\, follow(S)$

Rule 1: $S ::= uBDz$

$\rightarrow \{x, y, z\} \subset Bfollow(B)$

$and\, z \in follow(D)$

Rule 2: $B ::= wB'$

$\rightarrow follow(B) = \{x, y, z\} \subset follow(B')$

Rule 5: $D ::= EF$

$\rightarrow x \in follow(E)$

$follow(D) = \{z\} \subset follow(E)$

$and\, follow(D) = \{z\} \subset follow(F)$

$follow(S) = \{\#\}$

$follow(B) = \{x, y, z\}$

$follow(B') = \{x, y, z\}$

$follow(D) = \{z\}$

$follow(E) = \{x, z\}$

$follow(F) = \{z\}$

2.4 Parsing Table

Rule 1: $S ::= uBDz$

$\rightarrow table[S, u] = 1$

Rule 2: $B ::= wB'$

$\rightarrow table[B, w] = 2$

Rule 3: $B' ::= vB'$

$\rightarrow table[B', v] = 3$

Rule 4: $B' ::= \epsilon$

$\rightarrow table[B', x] = 4$

$\rightarrow table[B', y] = 4$

$\rightarrow table[B', z] = 4$

Rule 5: $D ::= EF$

$\rightarrow table[D, x] = 5$

$\rightarrow table[D, y] = 5$

$\rightarrow table[D, z] = 5$

Rule 6: $E ::= y$

$\rightarrow table[E, y] = 6$

Rule 7: $E ::= \epsilon$

$\rightarrow table[E, x] = 7$

$\rightarrow talbe[E, z] = 7$

Rule 8: $F ::= x$

$\rightarrow table[F, x] = 8$

Rule 9: $F ::= \epsilon$

$\rightarrow table[F, z] = 9$

The final parsing table is :

	u	v	w	x	y	z
S	1					
B			2			
B’		3		4	4	4
D				5	5	5
E				7	6	7
F				8		9

3 Programming Part

3.1 Recursive Descent

3.2 Programming directly in Java bytecode