

TDT4205 Compilers

Exercise 1

Stian Hvatum (hvatum)
MTDT

January 30, 2012

Task 1

Version info of *gcc*, *flex* and *bison*

- gcc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu7)
- flex 2.5.34
- bison (GNU Bison) 2.3

Task 2

A lexical analyzer is a program that accepts the source code of a program as a stream of characters, and identifying them as lexemes and outputs this as tokens.

An acceptor is a program or machine that inputs something, like the token stream generated by the lexical analyzer, and decides if it satisfies a given syntax, like a programming language. If the input is accepted, the acceptor returns *true*, else it returns *false*.

Task 3

3.1

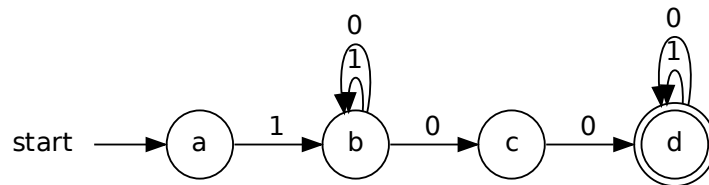


Figure 1: NDF for $1(0-1)^*00(0-1)^*$

3.2

This language includes:

- The word 100
- All words starting with 100
- All words starting with 1 and ending with 00
- All words starting with 1 and containing 00

given the alphabeth is the set of 0 1

Examples are

- 11111111111111110011111111111111
- 10011111111111111111
- 10011001100110011001
- 10001

3.3

	a	b	c	d
1	b	b	-	d
0	-	{b c}	d	d

Figure 2: Transition table for $1(0-1)^*00(0-1)^*$

Task 4

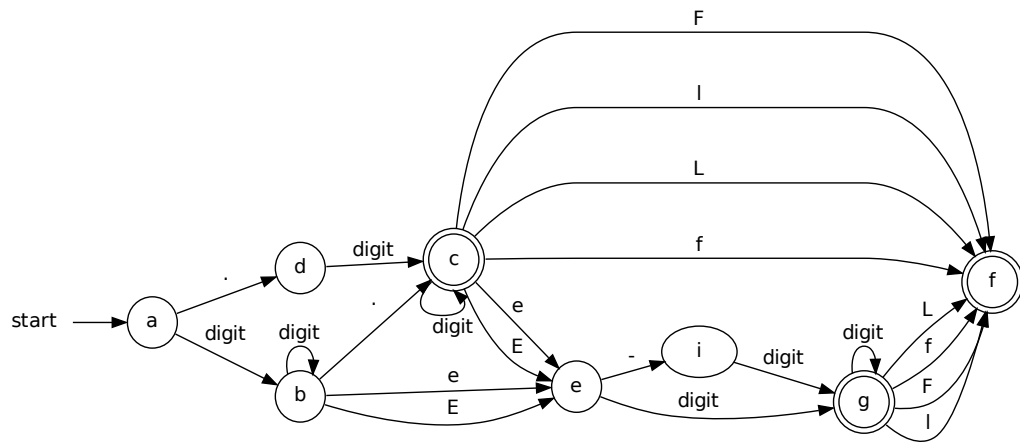


Figure 3: DFA for float

Appendix

DOT-code for NFA 3.1

```
1 //NFA for lang 1(0|1)*00(0|1)*
2 digraph G {
3     rankdir=LR;
4     shape=circle;
5     start->a
6     a->b [label="1"];
7     b->b [label="1" dir=back];
8     b->b [label="0" dir=back];
9     b->c [label="0"];
10    c->d [label="0"];
11    d->d [label="1" dir=back];
12    d->d [label="0" dir=back];
13
14    a[shape=circle];
15    b[shape=circle];
16    c[shape=circle];
17    d[shape=doublecircle];
18    start[shape=plaintext];
19 }
```

DOT-code for NDFA 4

```
1 //NFA for lang 1(0|1)*00(0|1)*
2 digraph G {
3     rankdir=LR;
4     shape=circle;
5     start->a
6     a->b [label="digit"];
7     a->d [label="."];
8     d->c [label="digit"];
9     b->b [label="digit"];
10    b->c [label="."];
11    b->e [label="e"];
12    b->e [label="E"];
13    c->c [tailport=sw headport=s label="digit"];
14    c->e [label="e"];
15    c->e [label="E"];
16    e->g [label="digit"];
17    e->i [label="-"];
18    i->g [label="digit"];
19    g->g [label="digit"];
20    g->f [label="f"];
21    g->f [label="F"];
22    g->f [label="l"];
23    g->f [label="L"];
24    c->f [label="f"];
25    c->f [label="F"];
26    c->f [label="l"];
27    c->f [label="L"];
28
29
30    a[shape=circle];
31    b[shape=circle];
32    c[shape=doublecircle];
33    d[shape=circle];
34    e[shape=circle];
35    f[shape=doublecircle];
36    g[shape=doublecircle];
37    start[shape=plaintext];
38 }
```