Programmer som data trial exam

Albert Ross Johannessen

December 11, 2024

1 Opgave 1

Betragt den ikke-deterministiske endelige automat ("nondeterministic finite automaton", NFA) nedenfor. Det anvendte alfabet er {`a,b`}. Der er i alt 5 tilstande, hvor 5 er den eneste accept-tilstand.

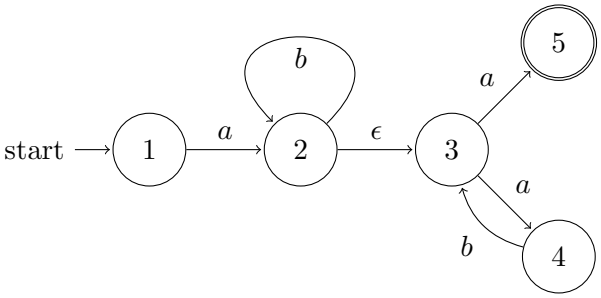

Figure 1: The labeled NFA

1.1 Angiv all årsager til at automaten er ikke-deterministisk

1. Der går en epsilon kant fra 2 til 3.
2. Der går 2 a kanter fra 3.

1.2 Giv tre eksempler på strenge der genkendes af automaten

1. aa
2. abbba
3. ababa

1.3 Giv en uformel beskrivelse af sproget (mængden af alle strenge) der beskrives af automate

Sproget der beskrives af automaten har følgende regler.

- Strenge skal starte og slutte på a
- Det første a kan være efterfulgt af et vilkårligt antal b'er
- I den efterfølgende streng, hvis man vil have mere end ét a så skal alle a'er være sepereret af b'er

1.4 Konstruer den tilsvarende DFA

	a	b	NFA State
S_1	$\{2,3\}^{S_2}$	$\{\}$	$\{1\}$
S_2	$\{4,5\}^{S_3}$	$\{\}^{S_2}$	$\{2,3\}$
S_3	$\{\}$	$\{3\}^{S_4}$	$\{4,5\}$
S_4	$\{4,5\}^{S_3}$	$\{\}$	$\{3\}$

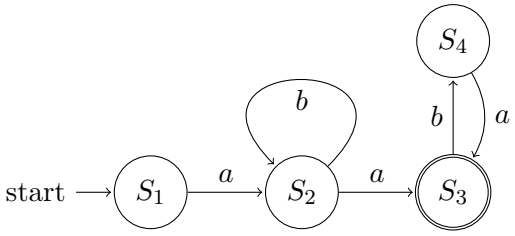

Figure 2: The labeled NFA

G_1	$\{S_1, S_2, S_4\}$
G_2	$\{S_3\}$

G_1	a	b
S_1	G_1	-
S_2	G_2	G_1
S_4	G_2	-

G_2	$\{S_3\}$
G_3	$\{S_1\}$
G_4	$\{S_2\}$
G_5	$\{S_4\}$

Siden der er én gruppe pr knude så er vores DFA så lille som den kan være.

1.5 Angiv et regulært udtryk for automaten

Hvis vi kigger på NFA'en som vi får givet i opgave beskrivelsen så kan vi splitte den op.

$$1 \xrightarrow{a} 2 = a$$

$$2 \xrightarrow{b} 2 = b*$$

$$3 \xrightarrow{a} 4 \xrightarrow{b} 3 = (ab)*$$

$$3 \xrightarrow{a} 5 = a$$

Hvis vi sætter dem sammen får vi følgende udtryk.

```
ab*(ab)*a
```

Det samme kan vi gør for DFA'en

$$S_1 \xrightarrow{a} S_2 = a$$
$$S_2 \xrightarrow{b} S_2 = b*$$
$$S_2 \xrightarrow{a} S_3 = a$$
$$S_3 \xrightarrow{b} S_4 \xrightarrow{a} S_3 = (ba)*$$

Hvilket producerer følgende udtryk.

```
ab*a(ba)*
```

Her er det indlysende at se at at det er det samme udtryk men skrevet på en anden måde, der er som udgangspunkt ikke nogen forskel mellem `a(ba)*` og `(ab)*a`.

2 Opgave 4

2.1 Opgave 4.1

```
[
-------------main-----------
  -999  old bp
  3     i
--------------f-------------
  24    ret addr PRINTI
  1     old bp
  3     arg
  42    i
]
```

2.2 Opgave 4.2

```
1   LDARGS;           Load args
2   CALL (0, "main"); Call Main with 0 args
3   STOP;             Return from Main and end program
4 Label "main";       Label for Main                          [-999]
5   INCSP 1;          Increase stackpointer(sp) with 1 sp = 1 [-999 0]
6   GETBP;            Get base pointer bp = 1                 [-999 0 1]
7   CSTI 3;           Put 3 on the stack                      [-999 0 1 3]
8   STI;              Store indirect                          [-999 3 3]
9   INCSP -1;         Decrease sp with 1                      [-999 3]
10  GETBP;            Get bp = 1                              [-999 3 1]
11  LDI;              Load indirect                           [-999 3 3]
12  CALL (1, "f");    Call function f with 1 argument         [-999 3 13 1 3]
13  PRINTI;           Print integer on top of stack           [-999 3 45]
14  RET 1;            Return and remove vals                  [ ]
15 Label "f";         Label for function f
16  INCSP 1;          Increase sp with 1 sp = 5               [-999 3 13 1 3 0]
17  GETBP;            Get bp = 4                              [-999 3 13 1 3 0 5]
18  CSTI 1;           Push 1 on the stack                     [-999 3 13 1 3 0 5 1]
19  ADD;              Add 1 and 4                             [-999 3 13 1 3 0 6]
```

```
20  CSTI 42;          Push 42 on the stack                    [-999 3 13 1 3 0 6 42]
21  STI;              Store indirect                          [-999 3 13 1 3 42 42]
22  INCSP -1;         Decrease sp by 1 sp = 5                 [-999 3 13 1 3 42]
23  GETBP;            Get bp = 4                              [-999 3 13 1 3 42 4]
24  LDI;              Load indirect, gets argument n          [-999 3 13 1 3 42 3]
25  GETBP;            Get bp = 4                              [-999 3 13 1 3 42 3 4]
26  CSTI 1;           Push 1 on the stack                     [-999 3 13 1 3 42 3 4 1]
27  ADD;              Add 4 and 1                             [-999 3 13 1 3 42 3 5]
28  LDI;              Load indirect, gets i                   [-999 3 13 1 3 42 3 42]
29  ADD;              Add 42 and 3                            [-999 3 13 1 3 42 45]
30  RET 2             Return to instruction 13 and remove args[-999 3 45]
```