Uebungsstunde Theoretische Informatik

Thomas Briner, brinerth@iiic.ethz.ch, http://briner.computerscience.ch/ti

11.4.03

1 Organisatorisches

- Website zur Uebung: http://briner.computerscience.ch/ti

Ablauf

Uebung wird am Donnerstag der Woche i in der Vorlesung verteilt und am Freitag in der Uebungsstunde vorbesprochen.

Variante 1:

- Abgabe bis Donnerstag der Woche i+1 in der Vorlesung, korrigiert und Rückgabe am Freitag der Woche i+1

Variante 2:

- Abgabe am Freitag in Uebungsstunde Woche i+1, korrigiert und Rückgabe am Freitag der Woche i+2

Testat:

Alle Uebungen abgeben, allein oder in 2er Gruppen

Literatur:

- Michael Sipser: Introduction to the theory of computation, PWS Publishing Company, Boston 1997
- U. Schöning: Theoretische Informatik - kurzgefasst, Spektrum Akademischer Verlag, Heidelberg 1997

2 Besprechung Uebung 1

HW 1.2

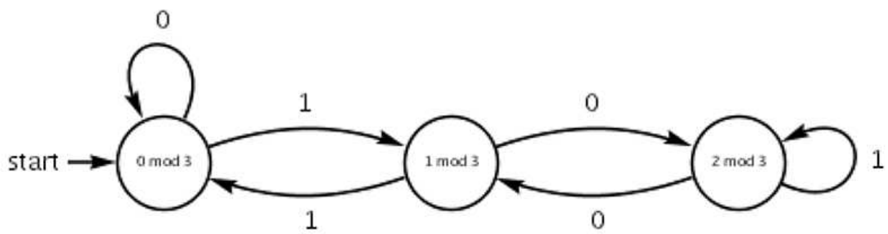
Prove a sorting network using only adjacent comparisons must have $\geq n - choose - 2$ comparators
Grundidee: Transpositionen

1	2	3	4	5	6	7	8	9	10

3 Vorbesprechung Uebung 2

HW 2.1

- in Vorlesung: Mod 3 einer binären Zahl in Notation MSB first: (z.B. $23_{10} \cong 10111_2$)

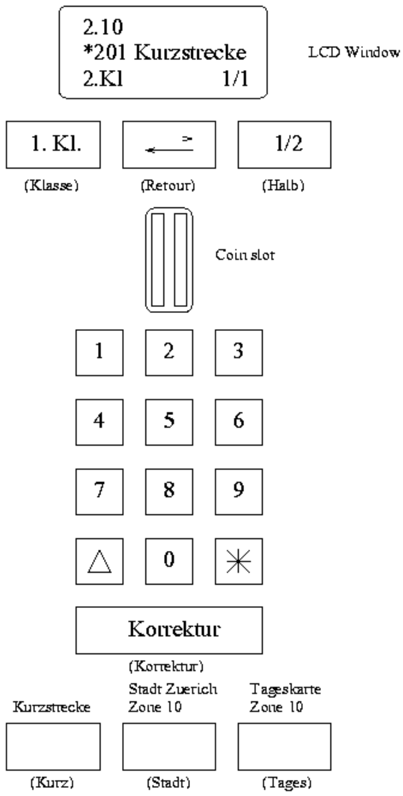


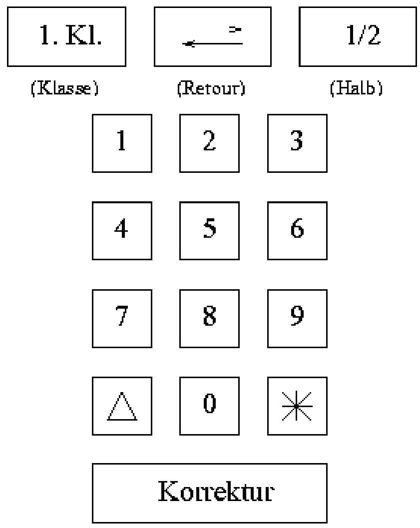
- als Uebung: Mod 3 einer binären Zahl in Notation LSB first: (z.B. $23_{10} \cong 11101_2 = 1*1 + 1*2 + 1*4 + 0*8 + 1*16$)

HW 2.2

Problem bei realistischen Beispielen: # Zustände ist sehr gross

⇒nur beschränkte Funktionalität modellieren
⇒verschiedene Zustände mit gleichen Eigenschaften zusammenfassen





4 Chomsky's Sprachenhierarchie

Grammatik G (V,A,P,S)

Typen:

- Typ 0: phrase structure grammar: Keine Einschränkungen
- Typ 1: context sensitive grammar: |lhs|≤|rhs|, ausser bei S→ε
- Typ 2: context free grammar: lhs nur Variablen
- Typ 3: regular grammar: siehe nächste Stunden

5 Markov Algorithmen

A = {0 1}
M = {α}

0 : $\alpha\,0 \rightarrow 0\,\alpha$

1 : $\alpha\,1 \rightarrow 1\,\alpha$

2 : $\alpha \dashv 0$

3 : $\epsilon \rightarrow \alpha$

Inputstring: 101

Bestandteile:

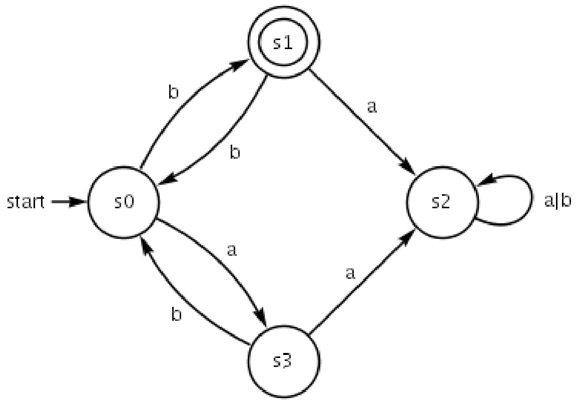
-
-
-
-

Ablauf der Abarbeitung:

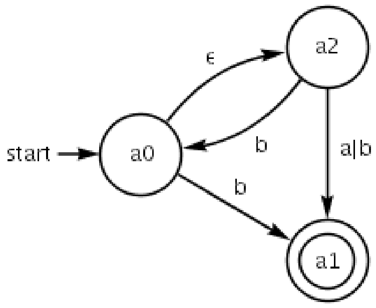
```
aktuelleRegel = 0;
loop {
    nimm die aktuelle Regel;
    scanne ganzen aktuellen String;
    if (gibt irgendwo einen Patternmatch mit der linken Seite der Produktion) {
        ersetze das Pattern durch die rhs;
        if (ist Stoppregel) { exit loop, Schluss; }
        else { beginne wieder mit erster Regel ganz von vorne, aktuelleRegel=0; }
    }
    else { nimm nächste Regel, aktuelleRegel=aktuelleRegel+1;
    }
    if (alle Regeln haben nicht gepasst) { exit loop, Schluss; }
}
```

6 Endliche Automaten

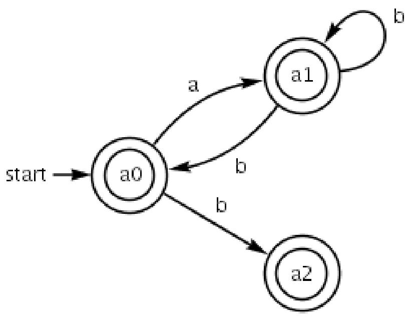
- Deterministic finite automaton DFA, M = (Q, Σ, δ, q_0, F)



- Non-deterministic finite automaton NFA with ε- transition, M = (Q, Σ, δ, q_0, F)

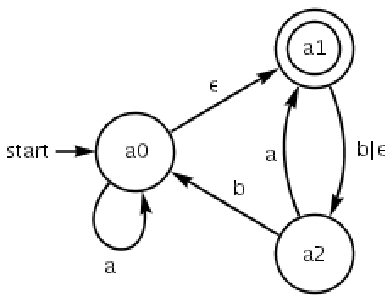


- Non-deterministic finite automaton NFA without ε- transition, M = (Q, Σ, δ, q_0, F)



7 Transformation NFA with ε $\Rightarrow$NFA without ε

Bsp1:



Bsp2: