Automater

Matematik för språkteknologer

Mattias Nilsson

Automater

- Beräkningsmodeller
  - Beräkning - (eng) Computation
- Inom automatateorin studeras flera olika beräkningsmodeller med olika egenskaper och olika beräkningsförmåga
- *Finita automater* är en typ av beräkningsmodell som ligger till grund för många, mer avancerade, beräkningsmodeller som t.ex. *stackautomater* eller *pushdown-automater*, och *Turingmaskiner*.

Tillämpning av automatateori

- Viktigt hjälpmedel för design, implementation, och modellering av många typer av system.
- Mönsterigenkänning:
  - Textsökning (t.ex. grep)
  - Stavnings- och grammatikkontroll
  - Kompilatorteknik
  - Optisk teckenigenkänning (OCR)

Inledande språktekniska begrepp

- Alfabet
- Strängar
- Språk

Inledande språktekniska begrepp

Alfabet: Σ

- En ändlig mängd av *ett eller flera* tecken (symboler)

Exempel

- Det binära alfabetet: $\Sigma = \{0, 1\}$
- Det svenska alfabetet: $\Sigma = \{a, \ldots, \text{ö}, A, \ldots, \text{Ö}\}$
- Mängden av små boktäver (Latin): $\Sigma = \{a, b, \ldots, z\}$
- Mängden av ASCII-tecken

Inledande språktekniska begrepp

Strängar

- En sträng w över ett alfabet är en ändlig sekvens av *noll eller flera* tecken från alfabetet.
  - $\Sigma_1 = \{0, 1\} \qquad w_1 = 01101$
  - $\Sigma_2 = \{a, b, c\} \qquad w_2 = babca$

Den tomma strängen: ϵ

- En sträng utan realiserade tecken (dvs. som består av noll tecken)

Längden av en sträng w: $|w|$

- $|w_1| = 5 \qquad |w_2| = 5 \qquad |\epsilon| = 0$

Inledande språktekniska begrepp

Sammansättning av strängar (strängkonkatenering)

$\Sigma = \{a, b, \ldots, z\}$

$w_1 = king \qquad w_2 = kong$

$w_1 w_2 = kingkong \qquad w_2 w_1 = kongking \qquad w_1 \epsilon = king$

$\Sigma = \{0, 1\}$

$x = 01101 \qquad y = 110$

$xy = 01101110 \qquad yx = 11001101 \qquad \epsilon y = 110$

$w\epsilon = \epsilon w = w$

Inledande språktekniska begrepp

Språk: L

- Ett språk L är *en mängd av strängar*

Kleenstjärna: Σ^*

- Mängden av alla strängar över ett alfabet Σ
- $\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
- $\{a, b\}^* = \{\epsilon, a, b, aa, bb, ab, ba, aab, \ldots\}$

Finita automater

Vad är finita automater?

- En abstrakt (idealiserad) beräkningsmodell
- Ett system som har ett ändligt antal möjliga tillstånd och vars beteende kan beskrivas med enkla transitionsregler, t.ex. om systemet är i tillstånd p och det får input a då går det till tillstånd q.
- En automat läser en given indatasträng från vänster till höger, en symbol i taget.
- Beroende på vilket tillstånd automaten befinner sig i när hela strängen är läst, accepteras eller avvisas strängen, så varje automat definierar en mängd av godkända strängar - automatens *språk*.

Finita automater

Deterministisk finit automata (definition)

En *deterministisk finit automat* (DFA) är en fem-tupel $(Q, \Sigma, \delta, q_0, F)$, där

- Q är en ändlig mängd *tillstånd*,
- Σ är en ändlig mängd symboler - ett *alfabet*,
- $\delta : Q \times \Sigma \longmapsto Q$ är en *transitionsfunktion*,
- q_0 är det initiala tillståndet (*starttillstånd*),
- F är en delmängd av Q; *mängden accepterande tillstånd*.

Determinism

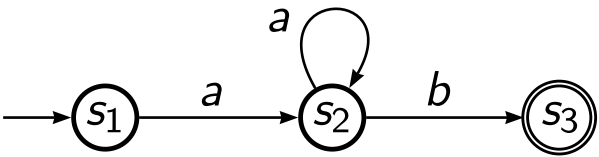
Deterministisk finit automata (DFA)

- Givet ett tillstånd och en inputsymbol finns det exakt ett möjligt måltillstånd (exakt en tillämpbar transition).

Ickedeterministisk finit automata (NFA)

- Givet ett tillstånd och en inputsymbol kan det finnas flera möjliga måltillstånd (mer än en tillämpbar transition).
- För varje NFA finns det en ekvivalent DFA. En ickedeterministisk finit automat kan således alltid *determiniseras*, dvs. konverteras till en determinitisk finit automat.

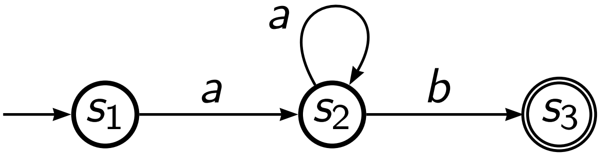
Finita automater som grafer



Finita automater representeras ofta som grafer (tillståndsdiagram):

- noder = tillstånd
- riktade bågar (pilar) med symboler = transitioner
- inkommande pil utan avsändare = starttillstånd
- dubbla cirklar = accepterande tillstånd

Finita automater som grafer



Automaten ovan definieras formellt:

$$Q = \{s_1, s_2, s_3\}$$

$$\Sigma = \{a, b\}$$

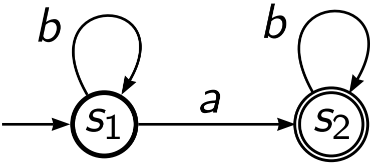
$$q_0 = s_1$$

$$\delta = \{< s_1, a >= s_2, < s_2, a >= s_2, < s_2, b >= s_3\}$$

$$F = \{s_3\}$$

Övning

Hur definieras automaten nedan formellt? Vilket språk accepterar den?



Övning

- Hur ser tillståndsdiagrammet ut för automaten $M = (Q, \Sigma, \delta, q_0, F)$, där $Q = \{s_1, s_2\}, \Sigma = \{1, 0\}, q_0 = s_1, F = \{s_1\}$, och δ ges av tabellen nedan?

	0	1
s_1	s_1	s_2
s_2	s_2	s_1

- Vilket språk accepterar M?

Accepterande automater

Accepterande automater kan användas för att avgöra om en viss sträng ingår i automatens språk:

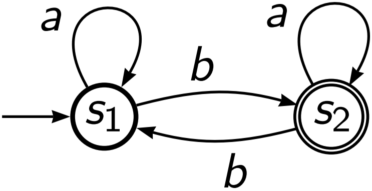
1. Starta i starttillståndet
2. Läs nästa symbol i strängen

   Om strängen inte innehåller fler symboler:

   Acceptera strängen om tillståndet är ett accepterande tillstånd
   Annars: avvisa strängen
3. Följ bågen markerad med inläst symbol

   Om ingen sådan finns avvisas strängen
4. Gå till steg 2.

Övning



- Vilka av följande strängar känns igen (accepteras) av automaten ovan?

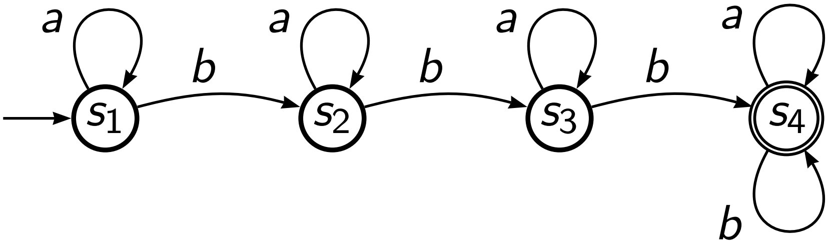
  $w_1 = bababa$; $w_2 = ababab$; $w_3 = aabbaa$; $w_4 = bbaabb$;
  $w_5 = bbbaaa$; $w_6 = aaabbb$;

- Vilket språk accepterar automaten ovan?

- Definera en sträng w, där $|w| = 8$, som känns igen av automaten

Övning

- Vilket språk accepterar automaten?



Övning

- Konstruera en automat som känner igen språket av alla binära strängar som slutar på 1.
- Konstruera en automat som känner igen språket av alla binära strängar som innehåller noll eller fler förekomster av 0 följt av noll eller fler förekomster av 1.
- Konstuera en automat som känner igen språket av alla binära strängar som innehåller exakt tre ettor.

Automater med utmatning

Mealyautomater

- En deterministisk finit automat (DFA) kompletterad med en funktion som lämnar utdata vid varje tillståndsövergång.

Mooreautomater

- En deterministisk finit automat (DFA) som lämnar utdata vid varje tillståndsbesök (även i starttillståndet).

Reguljära språk

- Varje språk som accepteras av en finit automat är ett *reguljärt* språk
- Reguljära språk är en klass av formella språk
  - Alla språk är alltså inte reguljära
- Är naturliga språk reguljära?

Reguljära operationer

Låt A och B vara (reguljära) språk. De reguljära operationerna *union*, *konkatenering* och *Kleenstjärna* definieras som följer:

- Union: $A \cup B = \{x \mid x \in A \textit{ eller } x \in B\}$
- Konkatenering: $A \cdot B = \{xy \mid x \in A \textit{ och } y \in B\}$
- Kleenstjärna: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \textit{ och } x_i \in A\}$

Reguljära operationer: exempel

Låt $L_1 = \{001, 10, 111\}$ och $L_2 = \{\epsilon, 001\}$

- $L_1 \cup L_2 = \{\epsilon, 001, 10, 111\}$
- $L_1 \cdot L_2 = \{001, 10, 111, 001001, 10001, 111001\}$
- ${L_2}^* = \{\epsilon, 001, 001001, 001001001, \ldots\}$

Reguljära uttryck

- Matematiskt uttryckssätt för att beskriva klassen av reguljära språk
- *Varje språk som accepteras av en finit automat är ett reguljärt språk*
- Går att översätta mellan reguljära uttryck och finita automater
- Reguljära uttryck och finita automater är två olika beräkningsmodeller som båda beskriver klassen av reguljära språk

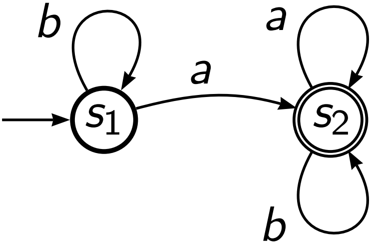
Övning

Konstruera automater som känner igen följande reguljära språk

- ab
- ab+
- a|b
- ab*
- (ab)*

Övning

Formulera ett reguljärt uttryck för det språk som känns igen av följande automat:



Övningar (Eriksson & Gavel)

Sektion 9.1

- Övning 9.1 - 9.7

Sektion 9.2

- Övning 9.9, 9.16, 9.19, 9.20

Sektion 9.3

- Övning 9.32-9.33