Obligatorisk oppgave 1 i INF2080, våren 2012

1 Regulære språk og endelige automater

Vi tar utgangspunkt i det regulære uttrykket $((10^*(1 \vee 0)0) \vee 0)^*$.

1. Lag en NFA som aksepterer strenger av dette språket. Bruk algoritmen og vis trinnene.
2. Konverter NFAen om til en DFA. Bruk algoritmen og vis trinnene.

2 Simulering av endelige automater

I figur 1 ser vi pseudokode som simulerer kjøringen av en DFA M gitt en inputstreng w. Algoritmen starter ved å initialisere variabelen st til å være starttilstanden. Metoden *neste-symbol(string)* henter neste symbol fra strengen som er gitt (første gang vi kaller på metoden henter den første symbol, andre gang vi kaller den henter den andre symbol, etc.). Funksjonen $\delta(tilstand, symbol)$ returnerer den tilstanden man kommer til når man står i den gitte tilstanden og følger kanten merket med det gitte symbolet. Hvis st refererer til en final tilstand når algoritmen er ferdig med løkken, vet vi at automaten skal akseptere.
Forsøk å se for deg hvordan algoritmen burde bli hvis du vil simulere en NFA. I en NFA er det noen ting man må være oppmerksom på:

- Automaten kan være i flere tilstander samtidig, altså burde st være en mengde tilstander, og ikke bare én tilstand.
- Nå som st inneholder en mengde tilstander kan man ikke sende denne som argument til δ, man må introdusere en ny løkke.
- I stedenfor en funksjon δ har vi en relasjon $\Delta : Q \times (\mathcal{A} \cup \epsilon) \times Q$, hvor $\mathcal{A}$ er input-alfabetet og Q tilstandene i NFAen.
- Man må undersøke alle muligheter med tanke på ϵ-transisjoner. For å hjelpe til med dette definerer vi følgende funksjon: $eps(a)$ vil returnere en ikke-tom mengde med tilstander, som består av tilstanden a samt alle tilstander som kan nås fra a ved å følge ϵ-transisjoner. Vi kan beregne $eps(q)$ slik.

  1. $resultat = q$

2. Så lenge det finnes en tilstand $p \in resultat$ og en tilstand $r \notin resultat$ og en transisjon $< p, \epsilon, r > \in \Delta$, sett $resultat = resultat \cup \{r\}$.
3. Returner $resultat$.

Første trinn i algoritmen for å simulere en NFA vil altså være:

1. $st = eps(start)$

$dfsimulate(M : DFA, w : string) =$

1. $st = start$
2. Repeter mens det er flere symboler igjen på strengen:
   (a) c = neste-symbol(w)
   (b) $st = \delta(st, c)$
3. **HVIS** st er final, aksepter strengen.

Figur 1: En algoritme for å simulere en DFA M på en string w.

Oppgaven går ut på følgende.

1. Beskriv trinn for trinn hva algoritmen i figur 1 vil gjøre når den simulerer DFAen du kom frem til i oppgave 1.2 på strengen 110.
2. Forsøk å gi resten av algoritmen for å simulere en NFA.
3. Skriv innholdet til st for hver iterasjon i hovedløkken når algoritmen din simulerer NFAen du kom frem til i oppgave 1.1. Start med å beskrive innholdet i st etter trinn 1. ($st = eps(start)$).
4. Hva burde man være oppmerksom på hvis man vil simulere en AFA? Skisser hvordan algoritmen burde utvides for en AFA.

3 Kontekst-frie språk

Vi tar utgangspunkt i språket $L = L_1 \cap L_2$ hvor:

- $L_1 = \{ww^R | w \in \{a, b\}^n\}$ (palindromer av partall lengde, w^R betyr w reversert, hvis $w = abc$ så er $w^R = cba$.)
- $L_2 = \{a^n b^* a^n | n \in \mathbb{N}\}$

1. Vis de fire første strengene i L i leksikografisk rekkefølge.
2. Vis at L ikke er regulær.
3. Lag en PDA for L.