Uke 38

Tre formuleringer av automater

1. DFA – deterministisk endelig tilstands automat
2. NFA – ikke-deterministisk endelig tilstands automat
3. REG – regulære uttrykk

Enhver DFA er også en NFA. Ellers har vi algoritmer for overgang fra en formulering til en annen. Vi skal lære tre ulike algoritmer som vil til sammen vise at de tre formuleringene – DFA, NFA og REG – behandler samme sak.

Kriterium på at vi har endelig tilstands automat

Vi kan lage en endelig mengde av tilstander Q slik at om automaten blir stoppet, så trenger vi bare vite hvilken tilstand fra Q den er i for å sette den i gang igjen.

Regulære uttrykk

Bygd opp fra endelig alfabet A. Kan skrive dette i kortform (her er $a \in A$) :

$$\text{Reg} = \varnothing \mid \Lambda \mid a \mid \text{Reg} + \text{Reg} \mid \text{Reg Reg} \mid \text{Reg}^*$$

Et regulært uttrykk er en delmengde av stringer over A – og kan sees som et språk.

Fra NFA til DFA – delmengdekonstruksjonen

La A være en NFA. Da kan vi konstruere B, en DFA som gjør det samme som A.

- Tilstandene i B = Delmengder av tilstandene i A - eksponensielt flere tilstander i B enn A
- Starttilstand i B = { q} - om q er starttilstanden i A
- Final tilstand i B = delmengde som inneholder en final tilstand fra A
- Transisjon i B = se på hvordan JFLAP virker når en stepper gjennom A

Regulære uttrykk og automater

For algoritmene som forbinder automater og algoritmer vil vi bruke automater der vi i transisjonene har regulære uttrykk – og ikke bare symboler – på pilene. Det er opplagt hvordan vi skal tolke slike automater, og de er en stor fordel når vi skal lage algoritmer for overgangen mellom automater og regulære uttrykk.

Fra REG til NFA

- Gitt et regulært uttrykk R
- Lag automat som har start tilstand og final tilstand med en pil mellom dem merket med R

- Nå går vi gjennom oppbyggingen av det regulære uttrykket og viser hvordan vi lager en vanlig NFA som gjør det samme.
- Først behandler vi $\varnothing$ Λ a
- Så tar vi de mulige oppbyggingene Reg + Reg Reg Reg Reg*. I det første tilfellet får vi en parallell kobling, i det andre en serie kobling, i det siste en tilbake kobling
- Vi gjentar dette – og ender opp med en NFA der vi bare har piler med enkle regulære uttrykk

Fra NFA til REG

- Gitt A en NFA
- Først litt preprossering – lag automaten slik at en har bare en enkel start S og en enkel final tilstand F med ingen piler inn til S og ingen piler ut fra F. Dette får vi til ved å bruke den tomme transisjonen Λ
- Så litt mer preprossering – for to vilkårlige tilstander P og Q bruk + slik at vi får høyst en pil fra P til Q
- I hoveddelen av algoritmen tar vi vekk en og en tilstand mot å gjøre transisjonsuttrykkene mer og mer kompliserte inntil vi ender opp med en enkel transisjon fra S til F med det søkte regulære uttrykket på pila.
- La R være en tilstand forskjellig fra S og F. Vi skal ta vekk tilstanden R mot å gjøre noen transisjoner mer kompliserte. Da må vi gjøre noe for samtlige P og Q slik at det fins pil fra P til R og pil fra R til Q. Anta at vi har
    - Pil merket c fra P til Q
    - Pil merket d fra P til R
    - Pil merket e fra R til R
    - Pil merket f fra R til Q
    - Da erstatter vi c med følgende: c + d(e)*f
    - Om noen av pilene mangler erstatter vi c med et litt enklere uttrykk
    - Dette gjør vi for samtlige P og Q og deretter stryker tilstand R

Det vesentlige å huske her er uttrykket c + d(e)*f - vi kommer fra P til Q ved enten å gå direkte via c eller først til R via d, deretter rundt i løkka ved R og så fra R til Q ved f.

.