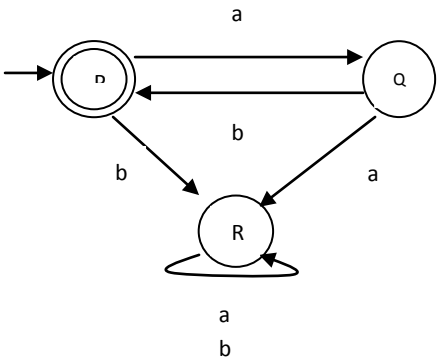
Uke 40

Simulering av automater med logikk



En kjøring av denne automaten med stringen <abba> kan simuleres med: START $\wedge$ TRANSISJON $\rightarrow$ FINAL , der

START: P(a:b:b:a:Λ)

TRANSISJON: $\forall x(P(a:x) \rightarrow Q(x)) \wedge \forall x(P(b:x) \rightarrow R(x)) \wedge \forall x(Q(a:x) \rightarrow R(x)) \wedge \forall x(Q(b:x) \rightarrow R(x)) \wedge \forall x(R(a:x) \rightarrow R(x)) \wedge \forall x(R(b:x) \rightarrow R(x))$

FINAL: P(Λ)

Vi skal senere vise at utsagnet er gyldig hvis og bare hvis stringen blir akseptert av automaten. Her har vi skrevet a:x for stringen vi får ved å sette a foran stringen x. Vi har brukt all-kvantoren $\forall$ som vi senere skal behandle når vi kommer til første ordens logikk.

Parentesspråket som kontekstfri grammatikk (CFG)

Parentesspråket er gitt ved stringer som: Λ () ()() (()(()))

Terminal symboler : ( )

Ikke-terminal symbol : S

Start: S

Regler: $S \rightarrow \Lambda$

$S \rightarrow (S)$

$S \rightarrow S\ S$

På neste side er en utledning av den siste stringen .

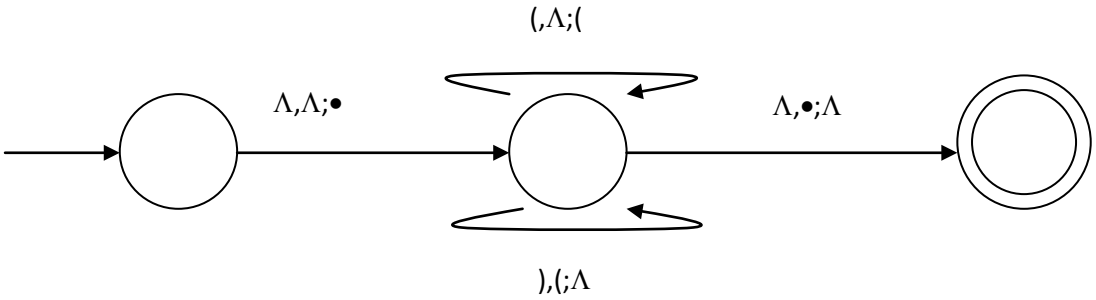
S
(S)
(SS)
((S)S)
((Λ)S)
((Λ)(S))
((Λ)((S)))
((Λ)((Λ)))

Som er det samme som (()(()))

Parentesspråket gitt ved push down automat (PDA, stakkautomat)

Vi har et nytt symbol • som markerer bunnen av stakken. Vi bruker notasjonen fra JFLAP for PDA'er. Det betyr at

- Vi får akseptering om vi kommer til en aksepterende tilstand
- Transisjonene inneholder 3 stringer skilt slik S,T;U - her er S og T betingelser som må være oppfylt (guards, voktere) og etter semikolon har vi aksjonen U
  - S T og U er stringer
  - S sier hvilke symboler somblir tatt fra inputstringen
  - T sier hvilke symboler som blir tatt fra stakken
  - U sier hvilke symboler som blir lagt til stakken



- Vi starter automaten ved å legge symbolet • i stakken for å markere en bunn
- Vi avslutter automaten om vi har kommet til bunnsymbolet på stakken og tar vekk dette
- Om vi leser (, så legger vi dette på toppen av stakken – vi pusher (
- Om vi leser) og det er (på toppen av stakken, så tar vi vekk det – vi popper (
- Selve automaten er ikke-deterministisk. Det er tilfeller der vi kan velge om vi vil fortsette i tilstand 2 eller gå over i tilstand 3.

CFG og (ikke-deterministiske) PDA gir samme språk

I JFLAP er det gitt en prosedyre for hvordan en fra en PDA kan konstruere en ekvivalent CFG. Den andre veien går også.