Kap 03

Kompleksitet av beregninger

Vi har følgende bilde av en beregning

Turingmannen har full oversikt over

- Startkonfigurasjon – gitt for eksempel ved inngangsdata
- Transisjonsregler – gitt for eksempel ved et program

Han ønsker å finne mer ut om hvordan beregningen arter seg – for eksempel om den terminerer. I kompleksitetsteori ser vi på forholdet mellom mer kunnskap fra turingmannen og mer kunnskap om atferden til beregningen. Kunnskapen til turingmannen er av to slag – compile time (kunnskap ved start) og run time (kunnskap underveis).

La oss liste opp noen kompleksitetsklasser:

P : Antall skritt i beregningen er polynomiell i størrelsen av inngangsdata

NP : Beregningen er et søk i et ELLER-tre med begrensete forgreninger og polynomiell høyde

PSPACE : Beregningen er et søk i et OG-ELLER-tre med begrensete forgreninger og polynomiell høyde

Primitiv rekursjon : Beregning med FOR-løkker. Turingmannen har compile-time kunnskap om løkkestrukturen og kunnskap underveis ved start av hver løkke om hvor mange ganger i løkka

Turingsumpen : Ingen spesiell kunnskap – vi regner og regner til vi er ferdig slik vi gjør med WHILE-løkker

Søket i **NP** og i **PSPACE** gjøres dybde først – da trenger turingmannen bare huske polynomielt mye underveis selv om han blir tvunget til å undersøke eksponensielt mange noder

Dette bildet av kompleksitetsklasser kan vi bygge ut i flere retninger. Her er vi spesielt interessert i klasser rundt **Primitiv rekursjon** – og spesielt hvilken logikk vi skal bruke for å gi kunnskap om atferden til beregninger til slike kompleksitetsklasser.

Først skal vi se beregninger og logikker knyttet til endelig tilstand automater. Det gir noen av de mest interessante logikker som er avgjørbare og der ufullstendighetsfenomener ikke forkommer.

Logikk for automater

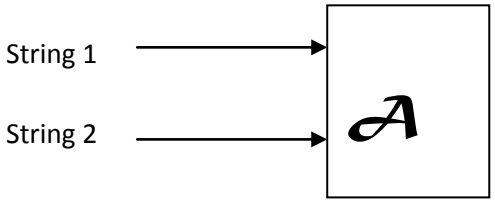
I en automat har vi en string inn. Selve beregningen kan turingmannen utføre på selve stringen. Vi kan tenke oss det som en fargeleggingsoppgave. Hver tilstand i automaten svarer til en farge – og beregningen foregår slik

String : s0 s1 s2 s3 s4 s5 s6 s7 s8 ……. S1000

Farger : f0 f1 f2 f3 f4 f5 f6 f7 f8 ……. f1000

Vi starter med selve stringen og den første fargen f0 som svarer til starttilstanden – vi har skrevet med svart det som er gitt og med rødt det turingmannen regner ut. Fra symbolet s0 og fargen f0 kan turingmannen finne fargen f1 . Deretter resten av fargene trinn for trinn. Stringen blir akseptert om den siste fargen er en aksepterende farge. Beregningen forgår som en dekorering og plassen vi bruker er som plassen til stringen og tiden er som lengden av stringen.

Mer allment kan vi tenke oss at vi har en automat med mange input stringer



Vi lager nå en logikk som beskriver hvilke stringer som blir akseptert. Som elementene i logikken tar vi stringer. Automaten over kan vi skrive som A(x,y) der x står for string 1 og y for string 2. Nå kan vi ved konstruksjoner på automatene konstruere sammensatte utsagn

Konjunksjon : Kartesisk produkt av automater – stringene virker uavhengig på automatene

Negasjon : Først lager deterministisk automat og deretter tar komplement

Eksistenskvantor : Bruker indeterminisme til å skjule en string-input

Men da kan vi lage automat som svarer til et hvilket som helst utsagn. Automatene kan bli veldig store – veksten skjer ved behandlingen av negasjon.

Eksempel – Presburger aritmetikk

Vi kan nå vise at teorien for naturlige tall med bare addisjon er avgjørbar. Det vesentlige er å lage en automat som avgør utsagn av typen X + Y = Z . Det får vi til ved å la tallene være i binær form og la addisjonen skje bakfra – fra høyre mot venstre. Med indeterminismen vi bruker i behandlingen av eksistenskvantor får vi både skjult hvilke symboler den skjulte stringen har og hvor lang stringen er.

Gitt en setning i Presburger aritmetikk. Lag en automat som svarer til setningen ved konstruksjonene over. Dette blir en stor automat og siden den er en setning har den ingen input. Vi får akseptering om starttilstanden er aksepterende.

Dette viser at vi får ikke ufullstendighet i datastrukturer over naturlige tall med bare addisjon.