Kap 01

Fra Skolems artikkel ønsker vi tre ting for å beskrive syntaktiske strukturer

- En datastruktur
- Et språk for å beskrive objektene i datastrukturen
- En kalkyle for å regne ut hvilke beskrivelser som er sanne og gale

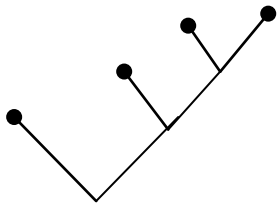
Datastrukturen – endelige par

Strukturen kunne også ha hett endelige binære trær. Vi kan definere strukturen slik

$\mathcal{P}$ ::= **nil** | ($\mathcal{P}$. $\mathcal{P}$)

Den er bygd opp av par-dannelse fra **nil –** en konstant og en binær konstruktor

Valget av de endelige parene – vi vet fra lisp og prolog at det er en datastruktur som mye kan uttrykkes i. Vi skriver dem opp slik



Dette er (**nil** . (**nil** . (**nil** . **nil**))) og inneholder delene (**nil** . (**nil** . **nil**)) (**nil** . **nil**) **nil**

Gitt en syntaktisk struktur – som la oss si utsagnslogiske utsagn bygd opp fra de tre atomære utsagnene A B C ved hjelp av konnektiver og parenteser på vanlig måte, da er det en enkel (men litt omstendelig) oppgave å kode dette inn som par. (Gjør dette!)

Det er ikke noe annet spesielt med par enn at slik koding kan gjøres på en enkel måte. Andre datastrukturer der en også kan gjøre det – hereditært endelige mengder, endelige stringer i 2 symboler. Men en kan ikke kode inn syntaks på samme måte inn i de unære tall eller endelige stringer i 1 symbol.

Språket over de endelige parene

- Konstanten nil
- Variable x y z ….
- Den binære funksjonen (x . y) - pardannelse, også kalt cons
- Likhet x = y
- Del x < y x er et delpar av y, x er konstruert før y
- Konnektiver
- Begrensete kvantorer $\forall x < y$ $\exists x < y$

- Vanlige kvantorer $\forall x \quad \exists x$

Vi skal legge merke til de begrensete kvantorene, og tenke på dem som en mellomting mellom konnektiv og vanlig kvantor. Vi kan tenke oss at datastrukturen endelige par er konstruert fra konstanten nil ved å bruke den binære konstruktoren (x . y). Da svarer den begrensete kvantoren $\forall x < y$ til at vi kan kvantifisere over alle elementer som er konstruert før y . Når vi har setninger (dvs utsagn uten frie variable) så vil de begrensete kvantorene svare til endelige konjunksjoner eller disjunksjoner.

Klassifikasjon av utsagn

- $\Delta_0 = \Pi_0 = \Sigma_0$ – utsagn uten vanlige kvantorer (eller ekvivalent med et slikt)
- Π_{i+1} – utsagn med $\forall$-kvantorer utenfor et Σ_i-utsagn (eller ekvivalent med et slikt)
- Σ_{i+1} – utsagn med $\exists$ -kvantorer utenfor et Π_i-utsagn (eller ekvivalent med et slikt)

Typiske eksempler for setninger

- Δ_0 - syntaks, at noe er syntaktisk
- Σ_1 - at noe er bevisbart, at noe er beregnbart
- Π_1 – en invariansegenskap for programmer
- Π_2 - en spesifikasjon av et program

Den vesentlige vanskeligheten er å vise at en endelige sekvens av par kan uttrykkes som et par. Til det trenger vi noen enkle utvidelser av språket

$x \leq y \ : \ x < y \ \vee x = y$

$x = hd(y) \ : \quad \exists z < y \ . \ y = (x \ . \ z)$

$x = tl(y) \ : \quad \exists z < y \ . \ y = (z \ . \ x)$

Disse utvidelsene kan elimineres i kontekst.

Nå koder vi inn ”x er en endelig sekvens av par, og y er et av parene” ved følgende

$y < x \ \wedge \ \exists z \leq x \ (\ y = hd(z) \ \wedge \ \forall u < x \ (\ z < u \rightarrow z \leq tl \ (u) \ ) \ )$

Unære tall

Vi klarer oss ikke med bare datastrukturen. Vi må ha med ekstra funksjoner som addisjon og multiplikasjon for å få til innkoding av syntaks. I monografien er dette gjennomført, men i forelesningene har vi nøyd oss med å peke på at her har vi en ekstra vanskelighet. Selve innkodingen er rett fram ved å bruke primtallene til koding om vi også har eksponensiering med som en funksjon. Uten den trenger vi ekstra argumenter.