Mekanisering av logikk

Det har vært en gammel drøm at en kunne lage et språk som all kunnskap skulle kunne uttrykkes innenfor (linga characterica) og så en metode til å regne ut hva som er riktig (calculus ratiocinator). La oss se hva vi får til :

Sekventkalkyle

Reglene er enkle og lett mekaniserbare. Vi ser at vi fort får mange kopier av formeldeler. Dette er jo et vanlig problem ved implementasjoner. Mye kan oppnås ved å bruke pekere i stedet for kopiering. Det er tre grunnleggende problem som gjenstår

- Valg av hvilken regel som skal brukes
- Valg av hvilken instansiering av term ved $\forall$-antesedent og $\exists$-suksedent
- Utvidelse med flere regler

I første ordens logikk med likhet vil alle innsettingene av termer ved analyse av =-antesedent gjøre at det blir lav effektivitet.

De to første problemene lar seg ofte løse – og vesentlige effektivitetsgevinst er oppnådd. Det siste er mye vanskeligere.

Nye regler – snitt, hjelpesetninger

De reglene vi har satt opp er tilstrekkelig til å bevise alt som kan bevises – men effektiviteten kan bli dårlig. Det er en regel som ofte tas med og kalles snitt-regel med F som snittformel:

$$\frac{\Gamma, F \vdash \Delta \qquad \Gamma \vdash F, \Delta}{\Gamma \vdash \Delta}$$

Denne regelen er overflødig – alt som kan vises med den, kan også vises uten. Men – og det er viktig – bevisene uten snitt kan bli forferdelig mye større enn bevis der snitt brukes. Jeg har lagd eksempler der bevisene uten snitt trenger hele universet og vel så det, mens de med snitt er på noen få linjer. Snittregelen brukes ofte i matematiske utledninger. Det er den vi bruker når vi har hjelpesetninger eller lemmaer – i stedet for et mer direkte bevis.

Simulering av beregninger

Vi har sett at beregninger lar seg simulere med logikk. Det er gode og dårlige nyheter. Den gode nyheten er at første ordens logikk har et språk der vi kan uttrykke noe så vanskelig som at vi har en beregning. Den dårlige nyheten er at vi har de samme problemene i logikk som vi har med beregninger. Vi er vant til å tenke oss at det er et gap mellom det intensjonale – det å lage et program. Og det ekstensjonale – finne ut run-time egenskapene ved programmet. Siden vi kan simulere beregninger med logikk, så får vi også tilsvarende problem mellom det å lage et utsagn – og det å finne ut om utsagnet er gyldig.

Utsagnskalkyle

Cooks formodning, $P \neq NP$, kan formuleres som et problem knyttet til mekanisering av utsagnskalkyle. Vi vet at for å finne ut om et utsagn er tilfredsstillbart så vil sannhetsverditabellen og vår mekanisering av sekventkalkyle gi algoritmer som bruker eksponensiell lang tid. (Tidsbruken som funksjon av størrelsen på input er en eksponensiell funksjon.) Cooks formodning blir vist om vi kan vise at det ikke fins noen polynomiell algoritme for tilfredsstillbarhet.

Første ordens logikk

Turing lagde sine turing maskiner for å vise at det ikke fins noen allmenn metode for å vise at et utsagn er gyldig enn det å prøve på å finne et bevis. Om en har flaks vil en finne det fort. Ellers kunne det tenkes at en lette og lette uten å kunne vite om en ville finne noe til slutt eller ei. Det fins ingen allmenn metode for å avgjøre om første ordens utsagn er gyldige.

Utvidelser av første ordens logikk

Den viktigste utvidelsen er for oss informatikere – logikk over en datastruktur. Det kan f eks være datastrukturen endelige stringer, endelige mengder, endelige lister. Her det store utfordinger.