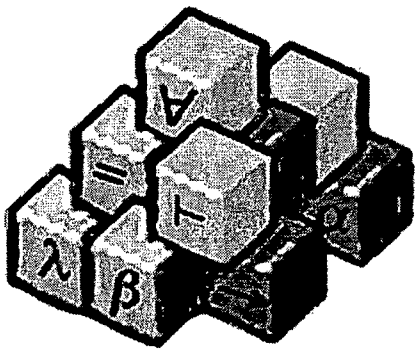
TEKST NR 427 2004

Fra Leibniz til *Isabelle*

Typeteoriens fremkomst og udvikling samt dens anvendelse i bevisføreren *Isabelle*



Ingunn Gunnarsdóttir
Uffe Thomas Volmer Jankvist
Bjørn Toldbod

Vejleder: Jørgen Villadsen

TEKSTER fra

IMFUFA ROSKILDE UNIVERSITETSCENTER
INSTITUT FOR STUDIET AF MATEMATIK OG FYSIK SAMT DERES FUNKTIONER I UNDERVISNING, FORSKNING OG ANVENDELSER

IMFUFA - Roskilde Universitetscenter - Postboks 260 - DK 4000 Roskilde
Tlf.: 46742263 - Fax: 46743020 - Mail: imfufa@ruc.dk
Ingunn Gunnarsdóttir, Uffe Thomas Volmer Jankvist og Bjørn Toldbod
'Fra Leibniz til *Isabelle* - Typeteoriens fremkomst og udvikling
samt dens anvendelse i bevisføreren *Isabelle*'
IMFUFA tekst nr. 427 - 158 sider - ISBN 0106-6242

Abstract

Nærværende projekt er dels et studie af typeteoriens fremkomst og udvikling og dels et studie af dens anvendelse i den moderne bevisfører *Isabelle.*

Rapporten er i to dele. Den første del indeholder en historisk gennemgang af matematikkens formalisering med udviklingen af typeteorien som omdrejningspunkt. Der redegøres for relevante arbejder af Frege, Russell, Gödel, Turing og Church. Denne redegørelse følges af en beskrivelse af den simple typeteori eksemplificeret ved en præsentation af Andrews' formelle system, $\mathcal{Q}_0^\infty$.

I anden del af rapporten præsenteres og forklares bevisføreren *Isabelle* og beviset for korrektheden af sorteringsalgoritmen Quicksort i logikken *HOL*. Med udgangspunkt i beviset undersøges *HOL*s brug af den simple typeteori og lighederne imellem *HOL* og $\mathcal{Q}_0^\infty$. Desuden undersøges om de aksiomer, der optræder i *HOL*, kan spores tilbage til de arbejder, vi har behandlet i første del af rapporten. Til sidst undersøges den betydning matematikkens formalisering har haft for bevisførere generelt.

I rapporten konkluderes, at der er så store ligheder imellem *HOL* og $\mathcal{Q}_0^\infty$, at *HOL* må opfattes som værende baseret på $\mathcal{Q}_0^\infty$. Den konkrete formulering af aksiomerne i *HOL* er dog ikke mulig at finde i de arbejder, der er gennemgået i rapportens første del. Derimod har mange af de ideer, der har drevet matematikkens formalisering, været med til at forme bevisførerne.

'All propositions' must be in some way limited before it becomes a legitimate totality, and any limitation which makes it legitimate must make any statement about the totality fall outside the totality.

Whitehead & Russell 1927

Forord

Nærværende IMFUFA-tekst er en revideret udgave af en projektrapport, der omhandler bevisføreren *Isabelle*[1] samt et bevis for korrektheden af sorteringsalgoritmen Quicksort foretaget i denne. Som grundlag for forståelsen af bevisføreren og dens formelle systemer gives en fremstilling af den simple typeteori samt en gennemgang af typeteoriens fremkomst og udvikling. I forhold til den oprindelige rapport er der indført rettelser af slåfejl samt en række småændringer, der blev foreslået til eksamen.

Projektet, der ligger til grund for IMFUFA-teksten, indgik som en del af overbygningsuddannelsen på Datalogi ved Roskilde Universitetscenter. Projektet hørte under 3. modul og talte $^3/_4$ semesterværk (22,5 ECTS). Rapporten er skrevet i LaTeX.

Vi vil gerne takke vores vejleder lektor Jørgen Villadsen for et yderst engageret samarbejde, hans entusiasme i forhold til projektet samt hans villighed til at dele ud af sin viden om hvad som helst. Vi vil også gerne takke lektor Jørgen Larsen (IMFUFA) for hans besvarelser af LaTeX-relaterede spørgsmål. Tilsidst ønsker vi at takke vores eksamenscensor professor Jørgen Fischer Nilsson (DTU) for hans forslag til forbedringer af rapporten.

Lad os afslutte dette forord med at bringe et citat fra Kleenes *Mathematical Logic* (1967), som udtrykker hans holdning til uvidende studerende som os selv:

> *It will be very important as we proceed to keep in mind this distiction between the logic we are studying (the object logic) and our use of logic in studying it (the observer's logic). To any student who is not ready to do so, we suggest that he close the book now, and pick some other subject instead, such as acrostics*[2] *or beekeeping.* [Kleene, 1967, s. 3-4]

Bjørn Toldbod
Ingunn Gunnarsdóttir
Uffe Thomas Volmer Jankvist

Datalogi OB, Roskilde, den 17. februar 2004

[1] Forsidebilledet er en del af *Isabelles* logo. Det stammer fra `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/logics.html`. Citatet på side i stammer fra Whitehead og Russells anden udgave af *Principia mathematica*: [Whitehead and Russell, 1927, s. 38].

[2] En 'acrostic' er et digt, hvori udvalgte bogstaver danner et nyt ord. Et eksempel på dette er ifølge Jørgen Villadsen: *Panthers growl, Orioles sing, Eagles soar, Monkeys swing.* De store bogstaver danner her ordet *POEM*.

Indhold

1 Indledning

I dette projekt har vi arbejdet med fremkomsten af typeteorien inden for matematikken og denne teoris nuværende anvendelse i datalogien i forbindelse med bevisførere.

Typeteorien blev oprindeligt udviklet som en teori, i hvilken man kunne undgå visse paradokser, som var opstået i matematikken. I dag er typeteori en fællesbetegnelse for en række systemer, i hvilke matematisk ræsonnering kan formaliseres. I et sådant system skelner man mellem objekter med forskellige egenskaber ved at tildele hvert objekt en *type* og opstille regler for, hvordan objekter af en given type må anvendes. Dette vil blive klart senere.

Formaliseret matematik kan ikke påstås at være et udbredt interesseområde blandt matematikere, og de forholdsvisfå matematikere, der gennem det sidste halve århundrede har beskæftiget sig med matematikkens formalisering og fundament, har i stort omfang ignoreret typeteorien, og istedet baseret matematikken på aksiomatisk mængdelære.

Imidlertid har typeteorien (også kendt som højereordens-logik) fundet anvendelse indenfor datalogien. Det er specielt bevisførere vi her tænker på. Mange af de systemer, der optræder i bevisførere i dag, er typeteoretiske systemer. Der findes også bevisførere med systemer, der er baseret på aksiomatisk mængdelære, men indenfor en række områder er det typeteorien, man er kommet længst med.

Det er ikke kun matematiske resultater, bevisførere er i stand til at bevise, snarere tværtimod. De benyttes i mange sammenhænge, som for eksempel i verifikationen af computersoftware og -hardware, herunder til at bevise korrektheden af datalogiske algoritmer. Det er netop dette sidste område, beviser for korrektheden af algoritmer, vi i dette projekt skal kigge nærmere på.

Vi har valgt at begrænse os til en bestemt bevisfører og et bestemt bevis; bevisføreren *Isabelle*, og et bevis for korrektheden af sorteringsalgoritmen Quicksort i denne. *Isabelle* er en generisk bevisfører, det vil sige en bevisfører, der kan opstille beviser i forskellige formelle systemer. Sorteringsalgoritmen Quicksort er eksemplarisk som repræsentant for datalogiske algoritmer, idet den overordnede ide bag algoritmen er forholdsvis simpel uden dog at være triviel.

Som datalog er man – eller bør man være – optaget af, om de algoritmer, man arbejder med, er korrekte. Ofte præsenteres man enten for et uformelt bevis for en algoritmes korrekthed, eller også overbeviser man sig selv om korrektheden af algoritmen ved at betragte en række eksempler. Imidlertid kan man også være interesseret i at give et formelt bevis for en algoritmes korrekthed. Her kommer de automatiske bevisførere ind i billedet. I en automatisk bevisfører er det muligt at opstille et formelt bevis for algoritmens korrekthed.

For at bevise korrektheden af en sorteringsalgoritme kan man vise, (1) at algoritmen returnerer et sorteret resultat, (2) at algoritmen ikke 'mister' elementer under sorteringen og (3) at algoritmen terminerer, det vil sige kører færdig i endelig tid. I *Isabelle* har man blandt andet bevist korrektheden af sorteringsalgoritmerne Insertion-sort, Mergesort og Quicksort.

De aksiomer, som bevisførerne i dag er bygget op omkring, er ikke bare tilfældige postulater. Det er aksiomer som igennem arbejdet med matematikkens formalisering, herunder udviklingen af typeteorien, har vist sig anvendelige som et fornuftigt fundament for et logisk system. Det vil altså sige at disse aksiomer, i en eller anden udgave, muligvis kan spores tilbage i historien til det system, hvori de første gang er blevet formuleret. Dette vil vi forsøge at gøre for det formelle system *HOL* i bevisføreren *Isabelle*.

For at have et grundlag for forståelsen af en bevisførers beviser må man kende til en vis mængde matematisk logik og typeteori. En del af denne projektrapport omhandler derfor en beskrivelse af et logisk system, betegnet $\mathcal{Q}_0^\infty$, der kaldes simpel typeteori. Dette system vil vi sammenligne med *Isabelles* system *HOL*.

Fra starten var vi af den opfattelse, at en vis indsigt i historien bag matematikkens formalisering samt fremkomsten og udviklingen af typeteorien ville øge vores forståelse for den moderne fremstilling af matematisk logik og typeteori. Samtidig var et kendskab til denne historie essentiel, hvis vi skulle realisere vores ide om at spore *Isabelles* aksiomer tilbage i tiden.

En større del af denne projektrapport beskæftiger sig derfor med en historisk gennemgang af matematikkens formalisering med typeteorien som omdrejningspunkt.

I en sådan gennemgang er det umuligt at komme uden om Bertrand Russell (1872-1970), der må betragtes som typeteoriens grundlægger. Russells forsøg på at formalisere matematikken førte til udviklingen af typeteorien, der var en måde, hvorpå man kunne undgå de paradokser, der plagede matematikken omkring år 1900. Russell opstillede sammen med Alfred North Whitehead (1861-1947) i det store tre-binds værk *Principia mathematica* fra 1910-13 et system, i hvilket matematikkens formalisering var mulig.

Russell var imidlertid ikke den første, der arbejdede seriøst med matematikkens formalisering. I værket *Grundsetze der Arithmetik*, som udkom 1893-1903, forsøgte Gottlob Frege (1848-1925) at skabe et fundament for matematikken (mere præcist aritmetikken). I værket anvendte Frege sit *Begriffsschrift* – et formelt sprog, som han allerede i 1879 havde opstillet. Freges system viste sig imidlertid at være inkonsistent[1]. Inkonsistensen blev påpeget af Russell i 1902 – historien om hvorledes Russell påpegede problemet, da andet bind af *Grundsetze* var i trykken, er en berømt anekdote. På trods af inkonsistensen er det Frege, der i dag tilskrives ideen om et formelt system – et system med aksiomer formuleret helt i systemets syntaks og præcise slutningsregler, således at resultater kan udledes på rent 'mekanisk' vis.

En af de matematikere, der arbejdede med matematikkens formalisering, var David Hilbert (1862-1943). Hilbert stillede ved en kongres i 1928 tre spørgsmål,

[1] At et system er inkonsistent vil her sige, at man både kan vise P og $\neg P$ i systemet – dermed kan man vise hvad som helst.

som han mente var essentielle for matematikkens formalisering. Det skulle undersøges (1) om der for ethvert matematisk udsagn gjaldt, at man enten kunne bevise dette eller dets negation, det vil sige om matematikken var *fuldstændig*[2], (2) om matematikken var *konsistent*, og (3) om der fandtes en bestemt metode til at afgøre om en påstand er sand eller falsk, med andre ord, om matematikken var *afgørlig*.

En del af svaret kom allerede i 1931. Den unge Kurt Gödel (1906-1978) offentliggjorde på dette tidspunkt sine to ufuldstændighedssætninger. Sætningerne siger overordnet set, at (1) der i ethvert konsistent formelt system, som indeholder en vis mængde talteori, altid vil eksistere sande udsagn hvorom gælder, at hverken de eller negationen af dem kan bevises i systemet samt, at (2) konsistensen af et sådant system ikke kan bevises inden for systemet selv. Disse to sætninger var nogen af de mest overraskende og uventede resultater indenfor matematikken i nyere tid, og de rev mildest talt tæppet væk under fødderne på de matematikere, der arbejdede med matematikkens formalisering. Det var altså ikke muligt indenfor et system som *Principia mathematicas* at bevise systemets egen konsistens. For at bevise konsistensen af systemet måtte man gå uden for systemet.

Gödels sætninger besvarer imidlertid ikke Hilberts tredje spørgsmål, det såkaldte *Entscheidungsproblem*. Dette spørgsmål blev senere besvaret af Alonzo Church (1903-1995) og (uafhængigt af Churchs svar) af Alan Turing (1912-1954) i 1936. De viser, henholdsvis med udgangspunkt i Churchs λ-kalkyle og Turingmaskinen, at der ikke findes en metode, som kan afgøre om et vilkårligt udsagn er et teorem eller ej, for de formelle systemer.

Resultaterne blev dødsstødet for Gottfried von Leibniz' (1646-1716) gamle vision fra cirka 1665; den såkaldte Leibniz' drøm. Denne drøm gik ud på at skabe en generel metode, hvori udledningen af alle sandheder, som man ellers skulle deducere sig frem til ved menneskelig ræsonnering, kunne reduceres til en form for beregning eller 'mekanisk' udledning. Ifølge Leibniz skulle man med denne metode, som indeholdt et specielt sprog, kunne besvare alle former for spørgsmål såvel matematiske som filosofiske og juridiske. Men med det negative svar på afgørlighedsspørgsmålet brast drømmen, der på dette tidspunkt var næsten 300 år gammel. Drømmen er dog i dag ikke helt uden betydning, idet den har været inspirationskilde for mangen en matematisk logiker.

Besvarelsen af afgørlighedspørgsmålet er ikke Turings eneste fortjeneste. Ifølge Gödel er det Turings arbejde, der har gjort det muligt at give en præcis og tilstrækkelig definition af et formelt system. Turing gav en definition af begrebet beregnelighed[3] og introducerede herigennem Turingmaskinen, som har en central betydning for datalogiens teoretiske fundament.

Church gav i forbindelse med sit svar på afgørlighedsspørgsmålet, gennem sin λ-kalkyle, ligeledes en definition af beregnelighed. Af stor vigtighed er Churchs simple typeteori fra 1940, som bygger på λ-kalkylen. Det er i bund og grund denne typeteori, som mange moderne bevisførere, eksempelvis *Isabelle*, bygger på.

[2]Fuldstændighed betyder, kort fortalt, at alle de udsagn, som er sande indenfor et system, kan vises i systemet. Ufuldstændighed betyder, at der eksisterer sande udsagn i systemet, som man ikke er i stand til at vise indenfor systemet.

[3]Dansk for *computability*.

Vi har i projektrapporten beskrevet ovenstående historie om matematikkens formalisering samt typeteoriens fremkomst og udvikling i større detalje. Især er vi gået i dybden med de ovenfor nævnte personer: Frege, Russell, Gödel, Turing og Church.

Man kan således se denne projektrapport som et forsøg på at beskrive, hvorledes behovet for typeteorien opstod, udviklede sig og til sidst fandt sin anvendelse inden for bevisførere.

I det efterfølgende gives vores motivation for at arbejde med de ovenfor skitserede problemstillinger. Derefter præsenteres projektets problemformulering samt en uddybning og afgrænsning af denne. Dernæst følger en beskrivelse af de metoder og konventioner, der anvendes i projektrapporten. Til sidst gives en beskrivelse af rapportens opbygning.

1.1 Motivation

Vi ønskede fra starten at beskæftige os med et grænseområde imellem datalogien og matematikken; bevisførere og deres anvendelse af typeteorien var et sådant. Som ovenfor nævnt opstod typeteorien som et løsningsforslag til problemstillinger i matematikken og blev siden hen udviklet som et fundament for matematikken. I dag er det dog ikke en særlig udbredt disciplin iblandt matematikere, der ofte er mere bekendte med den aksiomatiske mængdelære, især ZFC[4]. Peter Andrews kommenterer dette således:

> *One of the basic tasks of mathematical logic is the formalization of mathematical reasoning. Both type theory (otherwise known as higher-order logic) and axiomatic set theory can be used as formal languages for this purpose, and it is really an accident of intellectual history that at present most logicians and mathematicians are more familiar with axiomatic set theory than with type theory.* [Andrews, 2002, s. xi-xii]

Typeteorien har derimod gjort sit indtog i datalogien. Her har den fundet anvendelse i forskellige discipliner blandt andet indenfor lingvistik (Montague-grammatik). Med det samme fængede det os at en disciplin, der er opstået og udviklet indenfor matematikken i dag fortrinsvist udøves indenfor datalogien.

Et andet aspekt, vi fandt interessant i denne sammenhæng, var, at man ved hjælp af bevisførerne kan bevise korrektheden af datalogiske algoritmer. Groft sagt er datalogien jo udsprunget af matematikken som en selvstændig disciplin, der beskæftiger sig med algoritmer – altså udgør algoritmerne i mange sammenhænge datalogiens teoretiske fundament. Beviserne kan altså opfattes som en formalisering af den datalogiske ræsonnering. Den for os interessante observation var, at man således kan anvende teorier, hvori matematikken kan formaliseres, til også at formalisere datalogien.

Disse observationer førte til at vi bestemte os for at undersøge typeteoriens fremkomst og udvikling samt dens anvendelse i moderne bevisførere. Derudover

[4] Zermelo-Fraenkels aksiomer samt udvalgsaksiomet.

ønskede vi at undersøge bevisførernes beviser for korrektheden af datalogiske algoritmer.

I og med at bevisførerne gør brug af typeteoretiske aksiomer, tænkte vi, at det for en konkret bevisfører måtte være muligt at spore aksiomerne eller ideen bag dem tilbage i historien til der, hvor de første gang blev formuleret.

Alt dette udmøntede sig i følgende problemformulering.

1.2 Projektets problemformulering

Vi ønsker i denne projektrapport at beskæftige os med følgende:

- En redegørelse for matematikkens formalisering med typeteorien som omdrejningspunkt, det vil sige en redegørelse for typeteoriens fremkomst og udvikling i perioden fra Frege til Church.
- En kort beskrivelse af den simple typeteori gennem en præsentation af et formelt system betegnet $\mathcal{Q}_0^\infty$.
- En præsentation af bevisføreren *Isabelle* og beviset for korrektheden af sorteringsalgoritmen Quicksort i *Isabelles* logik *HOL*.

Med udgangspunkt i ovenstående ønsker vi at besvare følgende spørgsmål:

- Hvilke ligheder er der imellem *HOL* og systemet $\mathcal{Q}_0^\infty$?
- Kan aksiomerne i *HOL* eller ideen bag disse spores tilbage til de arbejder, vi har gennemgået i beskrivelsen af matematikkens formalisering?
- Hvordan har matematikkens formalisering bidraget til udformningen af *Isabelle* og andre bevisførere?

1.2.1 Uddybning og afgrænsning

Vi har valgt at lægge hovedvægten af gennemgangen af matematikkens formalisering på perioden fra Frege til Church. Frege var den første, som introducerede ideen om et formelt system, så det er her vores undersøgelse begynder. Undersøgelsen slutter med Churchs formulering af den simple typeteori, som er grundlaget for adskillige bevisføreres systemer i dag.

Oversigten over den simple typeteori samt opstillingen af det formelle system $\mathcal{Q}_0^\infty$ skal ses som et nødvendigt skridt på vejen til at forstå bevisførerne.

Vi har begrænset os til at se på én bevisfører. Valget faldt af flere grunde på bevisføreren *Isabelle*. *Isabelle* anvender typeteori i form af en logik kaldet *HOL* og samtidig er *Isabelle* en bevisfører, der tillader bevisførelse indenfor mange forskellige discipliner såvel matematiske som datalogiske. Tilmed er *Isabelle* en bevisfører, der gør sig i at bevise korrektheden af sorteringsalgoritmer, specielt algoritmen Quicksort. Da vi havde lagt os fast på at undersøge et korrekthedsbevis for netop Quicksort, fandt vi, at *Isabelle* var en oplagt kandidat.

Begrundelsen for valget af netop algoritmen Quicksort er at vi finder den eksemplarisk – det vil sige, at den bygger på en forholdsvis simpel ide uden at være triviel.

I vores besvarelse af første punkt i problemformuleringen tager vi fortrinsvist udgangspunkt i beviset for korrektheden af Quicksort i *Isabelles* logik *HOL*.

I andet punkt af problemformuleringen har vi begrænset vores sporing af aksiomer eller ideerne bag disse til de arbejder, vi har gennemgået i beskrivelsen af matematikkens formalisering. Muligheden foreligger dog, at aksiomerne er formuleret tidligere end perioden behandlet i vores gennemgang eller i en helt anden sammenhæng.

Det sidste punkt i problemformuleringen er af en lidt mere perspektiverende art. Vi ønsker her at se på, hvorledes de tidligere behandlede personer har spillet ind på udformningen af de moderne bevisførere. Specielt ønsker vi at relatere disse personers arbejder til bevisføreren *Isabelle*.

Dette leder frem til en beskrivelse af vores metode.

1.3 Projektets metode

De personer, hvis arbejder vi gennemgår i rapportens del I, tillægger vi forholdsvis stor betydning. Dette gør vi på baggrund af den viden vi har tilegnet os i sekundærlitteraturen. Her er der almindelig enighed om, at de personer, vi har beskæftiget os med, har haft den betydning, vi tilskriver dem i rapporten. Matematikkens formalisering, herunder typeteorien, er et emne, der er behandlet mange steder i litteraturen af anerkendte personer som for eksempel Kleene, van Heijenoort og Davis. Vi mener således, at vi kan retfærdiggøre vores lidt 'naive' tilgang til den historiske del af rapporten.

De værker vi har valgt at tage udgangspunkt i til beskrivelsen af matematikkens formalisering er som regel de værker, hvori den ide, vi er ude efter, optræder første gang. For eksempel benytter vi fortrinsvist Russells 1908-artikel til beskrivelse af ideen bag typeteorien i stedet for det senere værk *Principia mathematica*, hvori denne også optræder. Ligeledes er det Freges 1879-arbejde *Begriffsschrift* vi kigger på, ikke hans senere *Grundsetze*. Med hensyn til Frege og Gödel er vi klar over at det mest korrekte nok havde været at kigge på deres værker på originalsproget – tysk. Disse har vi dog ikke været i besiddelse af, hvorfor vi har valgt de engelske oversættelser i van Heijenoorts *From Frege to Gödel*.

Ofte, når vi i har forsøgt at gengive en specifik ide fra et værk, har vi gjort dette ved at citere fra værket. Dette har vi gjort dels fordi vi nogen gange har fundet gengivelsen af ideen vanskelig, dels fordi vi nogen gange har fundet det mere tro mod den pågældende ide at citere.

Vi har så vidt muligt i den historiske gennemgang af matematikkens formalisering og typeteorien forsøgt at gå til originalkilder, hvilket bør fremgå af kildehenvisningerne. Hvis vi har benyttet en senere udgave af et originalværk vil dette, foruden årstallet i selve henvisningen, fremgå af litteraturlisten eller blive nævnt i forbindelse med omtalen af værket i projektrapporten. Anvendes

optryk af artikler eller værker, som forefindes i senere udgivne samlinger, vil dette blive nævnt i fodnoter i forbindelse med omtalen af værket i rapporten.

1.4 Anlagte konventioner

Vi har i projektrapporten valgt at benytte os af forskellige konventioner. Vi benytter både udtrykket en 'sætning' og udtrykket et 'teorem'. Udtrykket sætning anvender vi, når der er tale om en matematisk sætning i den gængse forstand – altså i den uformelle matematik. Udtrykket et teorem benytter vi, når der er tale om et udtryk, der kan bevises i et logisk system.

Vi skelner i rapporten ikke strengt imellem vores brug af ordene 'udsagn' og 'proposition', rent faktisk benytter vi dem lidt i flæng. Det skulle dog gerne fremgå af konteksten, hvilken mening vi tillægger dem.

Visse ord i projektet er i *kursiv* og visse er i `skrivemaskineskrift`. Kursiv anvendes oftest til at fremhæve ord eller sætninger, for eksempel citater, men også navne på konkrete bevisførere eller konkrete teorier i *Isabelle*, for eksempel *HOL*. Skrivemaskineskriften har vi benyttet i forbindelse med koden fra beviset for korrektheden af Quicksort i *Isabelle*. Det har vi gjort for at gøre det nemt for læseren at skelne denne fra den øvrige rapport. Når vi i forbindelse med gennemgangen af koden omtaler selve beviset for algoritmen Quicksort eller teorier som beviset bygger på, har vi valgt også at bringe disse i skrivemaskineskrift, eksempelvis `Quicksort`, `Sort` og `Main`.

Når vi i projektet opremser kilder er vores strategi følgende: Vi opremser efter hvert afsnit (paragraf) samtlige kilder, vi har benyttet i det pågældende afsnit. Vi nævner enten de kilder, som er benyttet tidligst i det pågældende afsnit, eller de kilder som er benyttet mest i afsnittet, først. Når kilder har været benyttet som grundlag for hele afsnit, kommer henvisningerne til disse kilder i starten af de pågældende afsnit. Når der er tale om én specifik information og vi ønsker dette fremhævet skriver vi kildehenvisningen i forbindelse med informationen inde i afsnittet. Ved citater står kildehenvisningen umiddelbart efter citatet, dette er også tilfældet ved visse sætninger, beviser og deslige. Kildehenvisninger ser iøvrigt ud på følgende måde: [forfatter(e), udgivelsesår, sidetal].

1.5 Rapportens opbygning

Rapporten består af to dele. Første del omhandler matematikkens formalisering samt typeteoriens fremkomst og udvikling. Anden del omhandler bevisførere, i særdeleshed bevisføreren *Isabelle*, som bygger på matematikkens formalisering i form af typeteorien.

Vi gennemgår i første del relevante arbejder for typeteoriens tilblivelse og udvikling fra Frege og frem til Church. Vi har hver gang vi introducerer en ny matematiker valgt at bringe en biografi. Det har vi valgt at gøre af flere årsager, dels mener vi, at det kan være med til at fremme forståelsen af den pågældendes matematiske arbejde, at man har et indblik i hvilken tid denne levede i og hvilke

omstændigheder, der her gjorde sig gældende, dels var disse biografier noget af det, vi selv fandt underholdende i vores litteraturstudier.

I kapitel 2 gives en historisk introduktion til matematikkens formalisering i form af en beskrivelse af Leibniz' drøm. Der bringes også et diagram for den historiske udvikling af matematikkens formalisering med udgangspunkt i typeteorien.

Kapitel 3 omhandler Freges begrebsskrift, herunder de definitioner og ideer han introducerede i forbindelse med værket. Korrespondancen imellem Frege og Russell gennemgås. Russells, Cantors og Burali-Fortis paradokser præsenteres.

Beskrivelsen af Russells typeteori forefindes i kapitel 4. Formålet med kapitlet er at forsøge at gengive den ide, der ligger til grund for typeteorien.

Kapitel 5 omhandler Gödels ufuldstændighedssætninger samt de metoder, der anvendes i deres beviser. Derudover diskuteres betydningen af sætningerne kort.

Kapitel 6 indeholder en præsentation af Turings svar på Hilberts afgørlighedsspørgsmål. Herunder beskrives Turingmaskinen samt Turings definition af beregnelighed.

Kapitel 7 indeholder en historisk introduktion til λ-kalkylen samt den simple typeteori, som blev formuleret af Church i 1940.

I kapitel 8 gennemgås den simple typeteori overordnet. Der opstilles desuden et logisk system, $\mathcal{Q}_0^\infty$, som skal ligge til grund for diskussionen af bevisføreren *Isabelle* i anden del af projektet.

Kapitel 9 er en opsamling af de foregående kapitler i del I. Her forsøger vi at drage de vigtigste pointer frem i lyset.

Anden del af projektrapporten beskæftiger sig med bevisføreren *Isabelle*, herunder en beskrivelse af bevisførerens system, *HOL*, samt dens bevis for sorteringsalgoritmen Quicksorts korrekthed.

Der lægges ud med en gennemgang af bevisførere generelt i kapitel 10 samt en beskrivelse af deres systemer. Specielt behandles Freek Wiedijks 'Fifteen Provers of the World'.

I kapitel 11 behandles bevisførernes domæne, det vil sige områderne, de kan anvendes indenfor. Derudover bringes en gennemgang af sorteringsalgoritmen Quicksort.

Kapitel 12 omhandler *Isabelle*. Her findes en gennemgang af logikken i *Isabelle* samt en beskrivelse af, hvorledes *Isabelle* fungerer. Ydermere gives en kort beskrivelse af *Isabelles* metalogik.

Beviset for korrektheden af Quicksort i *Isabelle* præsenteres i kapitel 13. Denne præsentation ledsages af en grundig gennemgang af beviset.

I kapitel 14 forefindes undersøgelsen af beviset for korrektheden af Quicksort.

Kapitel 15 er en opsamling på projektrapportens del II.

I kapitel 16 forefindes projektrapportens samlede diskussion. Her diskuteres undersøgelsen i forhold til de forskellige sider af problemformuleringen

Kapitel 17 indeholder projektrapportens konklusion.

Kapitel 18 indeholder en epilog for projektet.

Appendiks A indeholder beviset for korrektheden af `Quicksort` og `Sort` i *Isabelle* (pretty-printed).

Appendiks B indeholder beviset for korrektheden af `Quicksort` og `Sort` i *Isabelle* (ASCII).

Appendiks C indeholder udklip fra *Isabelles* hjemmeside af de aksiomer, beviset for Quicksort i *Isabelle* bygger på.

Appendiks D indeholder screenshots, blandt andet et af *Isabelles* brugergrænseflade *Proof General.*

Litteraturlisten indeholder først og fremmest de kilder, vi har anvendt i rapporten. Efter hver kilde i litteraturlisten angives de sider i rapporten, hvor der henvises til kilden. Derudover indeholder den også de kilder, vi har benyttet os af i løbet af semesteret. Disse er: [Aleksandrov et al., 1969], [Bell, 1945], [Crossley et al., 1972], [Davis, 1958], [Frege, 1984], [Hatcher, 1982], [Hindley and Seldin, 1986], [Kleene, 1952], [Kline, 1980], [Meshckowiski, 1970-80], [Mostowski, 1966], [Nipkow, 2003], [Sluga, 1993], [Wussing and Arnold, 1975].

Del I

Matematikkens formalisering – Typeteoriens fremkomst og udvikling

2 Historie og motivation

Der er i dag udbredt enighed om at matematikkens formalisering starter med Gottlob Freges arbejde fra slutningen af det nittende århundrede. Frege tog arven op efter Aristoteles totusind år gamle logik, som indtil da havde været anset for at være et afsluttet kapitel. På trods af at vi i dag opfatter Frege som formaliseringens fader, er der dog visse indledende tiltag i denne retning inden Frege. Et af disse tiltag forefindes hos George Boole, som i midten af det nittende århundrede udviklede sin logiske (boolske) algebra. Et andet tiltag forefindes i Augustus De Morgans omtrent samtidige arbejde, hvori han blandt andet formulerede sine berømte love. Men allerede langt tidligere, i det syttende århundrede, havde Gottfried Leibniz tænkt nogle af de samme tanker. Disse ideer var for Leibniz et led i en langt større og mere vidtrækkende plan, som han som ganske ung havde udtænkt og som han forblev tro mod igennem hele sit liv. Denne plan er i dag kendt som *Leibniz' drøm*. Selv om ideen i dag betragtes som værende urealistisk, har den dog været en vigtig inspirationskilde for efterfølgende matematikere og logikere, for eksempel Frege, og der henvises ofte til den i litteraturen, hvorfor vi vil beskrive den her.

2.1 Leibniz' drøm

Leibniz var et geni og en af sin tids største tænkere. Han var uddannet fra universitetet i Leipniz med en grad i filosofi og jura. Det var dog indenfor den matematiske analyse, Leibniz blev mest kendt i kraft af sit arbejde om integration og differentiation. I en alder af cirka 20 år udtænkte Leibniz den ide, der senere er blevet kendt under navnet *Leibniz' drøm*. Lad os citere E. T. Bells beskrivelse af, hvad Leibniz' sigtede efter.

> *... a general method in which all truths of the reason would be reduced to a kind of calculation. At the same time this would be a sort of universal language or script, but infinitely different from all those projected hitherto; for the symbols and even the words in it would direct reason; and errors, except those of fact, would be mere mistakes in calculation. It would be very difficult to form or invent this language or characteristic, but very easy to understand it without any dictionaries.* [Bell, 1937, s. 123]

Der er altså tale om at Leibniz søgte et nyt slags alfabet, hvis elementer ikke skulle bestå af lyde, men af koncepter. Et sprog baseret på et sådant alfabet skulle da gøre det muligt ved symbolsk udregning at bestemme, hvilke udsagn i sproget, der var sande og hvilke, der var falske. Samtidig skulle det da også

være muligt at bestemme, hvilke logiske sammenhænge der eksisterede imellem disse sande udsagn. [Davis, 2000, s. 5], [Bell, 1937, s. 123]

Figur 2.1 Gottfried Wilhelm von Leibniz. Født 1. juli 1646 i Leipniz, Tyskland. Død 14. november 1716 i Hannover, Tyskland. [O'Connor and Robertson, 2003]

Leibniz' opdagelser (opfindelser) indenfor den matematiske analyse havde bekræftet ham i behovet for et universelt symbolsprog, eller konceptsprog om man vil. Som det muligvis er læseren bekendt havde Leibniz set sammenhængen imellem det at udregne arealer og det at bestemme retningsafledte[1], nemlig at de er hinandens inverse. Leibniz udviklede en bestemt symbolik til at beskrive disse operationer, rent faktisk den samme, som benyttes i dag[2],

$$\int \text{ for integration og } d \text{ for differentiation.}$$

Leibniz anså disse to symboler som indeholdende selve koncepterne om integration og differentiation, og det var på lignende vis, at tegnene i alfabetet til hans nye universalsprog skulle indeholde koncepter, blot koncepter af en anden karakter. [Davis, 2000, s. 11-12]

Leibniz tænkte, som citatet ovenfor antyder, på sit sprog som et universalsprog, det vil sige et sprog, hvori alle tænkelige spørgsmål om såvel naturvidenskab som filosofi og jura kunne besvares. Det var klart for ham, at de specielle tegn, som blev benyttet i aritmetisk algebra, kemi, astronomi og ikke mindst hans egne symboler for differential- og integralregning, alle viste hvor essentiel en velvalgt symbolisme var. Leibniz refererede til et sådant system af tegn som et *lingua*

[1] At integration og differentiation rent faktisk er inverse operationer er i dag kendt som analysens fundamentalsætning.

[2] Tegnet for integration $\int$ er et langt S, hvilket antyder at integration omhandler det at bestemme summer. Ligeledes antyder symbolet for differentiation d det at bestemme differenser.

characterica (et sprog af specielle symboler). Ulig de almindelige alfabetiske symboler var symbolerne i de ovenstående eksempler 'virkelige symboler', idet de repræsenterede bestemte ideer på en naturlig og velvalgt måde. Det der var brug for, vedblev Leibniz, var et universelt symbolsprog; et *lingua characterica universalis*. Det vil sige et sprog, hvori symbolerne ikke kun var virkelige, men som også omfattede den fulde rækkevidde af menneskets tanker. [Davis, 2000, s. 15], [Harrison, 1996, s. 5]

Leibniz så realiseringen af sin ide som bestående af tre dele. (1) For at kunne udvælge de rette symboler til alfabetet måtte der først skabes et kompendium eller en encyclopædi, som indeholdt menneskehedens samlede viden. (2) Når en sådan var udformet, ville det være klart, hvilke nøgle-koncepter der lå til grund for den samlede viden, hvorefter udvælgelsen af passende symboler for disse kunne foretages. (3) Endelig kunne slutningsreglerne da reduceres til manipulation med disse symboler, hvilket Leibniz kaldte en *calculus ratiocinator* (en form for symbolsk logik eller en 'regnemetode til at ræsonnere'). Leibniz var udmærket klar over, at opgaven ikke kunne løses af ham selv alene. Men han mente dog, at den ville kunne løses af nogle få dedikerede mænd over en femårig periode. At Leibniz kunne tro at hans plan var gennemførlig er for os i dag fuldstændig uforståeligt, for hvordan skulle hele universet og al dets kompleksitet kunne reduceres til en enkelt symbolsk logik. Men Leibniz levede i en anden tid, og for ham var intet i universet tilfældigt, alt fulgte en plan, som stod klart i Guds tanker. Gud havde trods alt skabt den bedste verden, det var muligt at skabe. [Davis, 2000, s. 16-17]

Leibniz gav visse forslag til hvordan hans såkaldte *calculus ratiocinator* skulle se ud. I denne forbindelse introducerede han et nyt tegn, $\oplus$, til repræsentation af sammenføjningen af helt arbitrære plurali af termer. Ideen var at sammenføje to samlinger af elementer til én samling indeholdende samtlige elementer fra de to.

Definition 2.1
A er i L, eller L indeholder A, er det samme som at sige, at L kan sammentræffe med et plurali af termer taget sammen, af hvilket A er et.

$$B \oplus N = L$$

betyder at B er i L og at B og N tilsammen udgør L. Det samme gør sig gældende for et større antal af termer.

Leibniz giver nu følgende aksiomer:

1. $B \oplus N = N \oplus B$
2. $A \oplus A = A$

Ud fra disse viser han en række resultater, for eksempel

Proposition 2.1
Hvis A er i M og B er i N, så er $A \oplus B$ i $M \oplus N$.

Det, der nok er noget af det mest bemærkelsesværdige er, at Leibniz, godt halvandet århundrede forud for sin tid, foreslog en algebra for logikken. En algebra som ville specificere reglerne for manipulation med logiske koncepter på samme måde, som den ordinære algebra gør det med reglerne for manipulation af tal. En anden ting, som vækker opsigt, er Leibniz' andet aksiom, $A \oplus A = A$, der siger at sammenføjningen af alle elementer i en given samling med den samme samling af elementer vil give samlingen selv. (Der er altså på sin vis tale om mængdelignende samlinger, ikke tal, idet der jo ikke gælder at $2+2=2$.) Netop aksiom 2, dog i en anderledes kontekst[3], blev en af grundpillerne i Booles logiske algebra cirka 150 år senere. [Davis, 2000, s. 18-19]

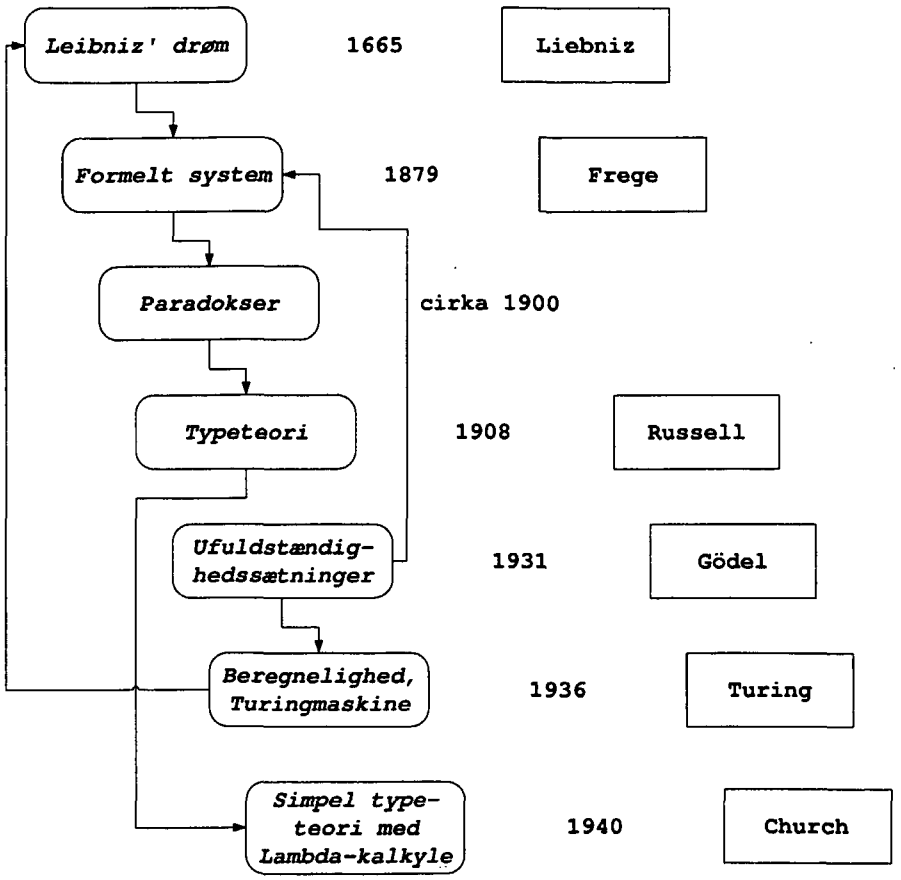
Benyttelsen af den boolske algebra som et system af regler for udregninger er ligefrem. En af Booles største bedrifter må anses at være, at han viste at logisk deduktion kan udvikles som en form for matematik. Boole kunne ikke have kendt til Leibniz' arbejder, idet disse kun forelå i private brevkorrespondancer. Selv om de begge benytter det ovenfor nævnte aksiom 2, er der specielt én væsentlig forskel på de to herrers 'systemer'. Leibniz forsøgte at fremstille sit deduktive system som et, hvor alle regler er udledt fra et lille antal aksiomer, hvilket er i overenstemmelse med moderne praksis. Det ubelejligede Boole sig ikke med, så på dette punkt må Leibniz siges, at have været længere fremme end Boole. [Davis, 2000, s. 39-40, 57-58]

2.2 Et diagram for typeteoriens udvikling

Typeteorien, som er omdrejningspunktet for denne projektrapport, er kun en del af matematikkens formalisering og udviklingen af den matematiske logik. Det tyvende århundredes resultater indenfor disse områder er relateret til hinanden på kryds og tværs. Skulle man forsøge at illustrere disciplinernes udvikling i et diagram, ville man formentlig få et meget indviklet og uoverskueligt netværk af pile imellem personer, relationer og resultater, som ikke ville bidrage videre til ens forståelse. Vi har derfor valgt at begrænse os til typeteoriens opståen og udvikling. På figur 2.2 forsøger vi således at billedliggøre udviklingen af, samt relationerne imellem vigtige resultater indenfor typeteorien. Vi er udmærkede klar over at et sådant diagram kunne have set ud på flere andre måder. Vi har dog fundet nedenstående fremstilling relevant for vores projekt.

Pilene imellem de afrundede kasser kræver en forklaring. Pilen fra *Leibniz' drøm* ned til *Formelt system* antyder, at drømmen på en vis måde har givet inspiration til opstillingen af et sådan system, hvilket Frege var den første som opstillede. Med *Paradokser* hentydes til de paradokser, Russell beskæftigede sig med, specielt hans eget. De formelle systemer gav anledning til disse paradokser, derfor pilen her imellem. Forsøg på at undgå paradokserne førte til udviklingen af typeteorien, hvorfor der er en pil imellem *Paradokser* og *Typeteori*. En videreudvikling af typeteorien var den simple typeteori som Church anvendte sammen med sin λ-kalkyle, derfor pilen imellem *Typeteori* og *Simpel typeteori med Lambda-kalkyle*. Gödels *Ufuldstændighedssætninger* kommer til udtryk

[3]Boole siger, at hvis x angiver en klasse, så er ligningen $xx = x$ altid sand. For hvis, for eksempel, x er klassen af får, så må xx være klassen af ting, som er får og ... som også er får. [Davis, 2000, s. 28]



Figur 2.2 Et diagram for udviklingen og tilblivelsen af typeteorien. Emneområder/discipliner/arbejder figurerer i kasser med afrundede hjørner. Ophavsmænd for disse arbejder/områder figurerer i kasser med skarpe hjørner på samme niveau i diagrammet. Årstallene står imellem disse.

i et *Formelt system*, for eksempel et system som *Principia mathematica*. Gödels sætninger implicerer på sin vis Turings svar på afgørlighedsspørgsmålet, derfor pilen fra disse ned til *Beregnelighed, Turingmaskine*. Det er Turings negative svar på afgørlighedsspørgsmålet som sætter en kæp i hjulet for *Leibniz' drøm*, derfor en pil op til denne.

Som sagt er vi klar over at dette diagram kunne have set anderledes ud, men vi har altså fundet ovenstående fremstilling fornuftig.

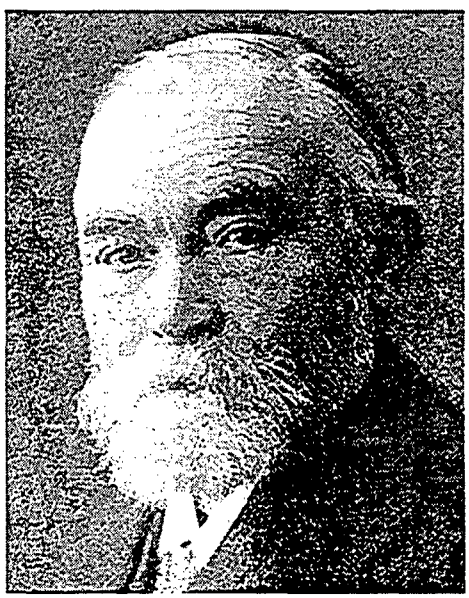
I de næste kapitler af rapportens del I skal vi se nærmere på de i diagrammet nævnte personer samt deres arbejder.

3 Freges begrebsskrift

Frege anses i dag for at være grundlæggeren af den moderne logik. Hans mest banebrydende arbejde i denne henseende er *Begriffsschrift, eine der aritmetiscen nachgebildete Formelsprache des reinen Denkens* fra 1879. Vi vil i dette kapitel give en gennemgang af dele af dette arbejde og derigennem komme ind på Russells paradoks og de andre paradokser, der opstod i tiden omkring år 1900. Først vil vi dog bringe en kort biografi for Frege.

3.1 Biografi

Gottlob Frege blev født i Wismar, Tyskland, i 1848. Han var søn af Alexander Frege, der var rektor på et gymnasium for piger, og Auguste Bialloblotzky. Efter gymnasiet studerede Frege i Jena fra 1869-1871, hvorefter han tog til Göttingen. Her fulgte han kurser i matematik, fysik, kemi og filosofi. I 1873 modtog Frege en doktorgrad i filosofi for afhandlingen *Über eine geometrische Darstellung der imäginaren Gebilde in der Ebene.* Året efter blev han privatdocent ved det



Figur 3.1 Friedrich Ludwig Gottlob Frege. Født 8. november 1848 i Wismar, Tyskland. Død 26. juli 1925 i Bad Kleinen, Tyskland. [O'Connor and Robertson, 2003]

filosofiske fakultet i Jena. Efter udgivelsen af *Begriffsschrift* i 1879 blev Frege udnævnt til 'ausserordentlicher Professor', og i 1896 blev han 'ordentlicher Honorar Professor'. Af andre væsentlige værker fra Freges tid i Jena kan nævnes *Grundlagen der Arithmetik* fra 1884 og ikke mindst hans to-binds værk *Grundsetze der Arithmetik*, der udkom 1893-1903. Dette to-binds værk var resultatet af Freges stædige arbejde mod hans mål – formaliseringen af aritmetikken. I *Grundsetze* udvidedes og forfinedes teorierne fra *Begriffsschrift* og *Grundlagen* og de tidligere udgivne forbedringer af hans begrebsskrift: *Funktion und Begriff* (1891), *Über Sinn und Bedeutung* (1892) og *Über Begriff und Gegenstand* (1892) blev også inkorporeret. Kort før udgivelsen af andet bind, faktisk mens det var i trykken, pointerede Russell i et brev til Frege, at hans system involverede en modstrid. Denne observation, senere kendt som Russells paradoks (se afsnit 3.3), ødelagde Freges teori for aritmetikken, og han så ingen måde hvorpå teorien kunne reddes. Freges videnskabelige arbejde i perioden efter 1903 kan ikke sammenlignes med det før 1903. Det var hovedsageligt en reaktion på de nye udviklinger inden for matematikken og dens formalisering, specielt med hensyn på Hilberts aksiomatisering. [van Rootselaar, 1970-80a, s. 152-154]

Figur 3.2 Gottlob Frege i en lidt yngre udgave. [Davis, 2000, s. 44]

Som det fremgår af ovennævnte var Freges liv i det ydre ganske udramatisk. Han levede, giftede sig, adopterede en søn og arbejdede herefter i Jena indtil sin pension i 1918, hvor han flyttede tilbage til en lille by nær sin fødeby. Frege fortsatte her sin forskningsmæssige produktivitet, blandt andet med *Logische*

Untersuchungen, skrevet fra 1918 til 1923, som er en udvidelse af hans tidligere arbejde. Frege døde den 26. juni 1925 i Bad Kleinen. [van Rootselaar, 1970-80a, s. 152-154], [Frege, 2002, s. 10]

Frege opnåede aldrig den helt store anerkendelse i det brede videnskabelige samfund i sin egen levetid, dog senere blandt en snæver kreds af ikke ubetydelige filosoffer og matematikere, som for eksempel Russell, Ludwig Wittgenstein, Rudolf Carnap, Edmund Husserl og Guiseppe Peano. Trods begrænset berømmelse og anerkendelse af sin samtid anses Frege i dag for at være grundlæggeren af den moderne analytiske filosofi såvel som den matematiske logik. [van Rootselaar, 1970-80a, s. 152-154], [Frege, 2002, s. 9-13]

Lars Binderup skriver følgende om Frege i introduktionen til den danske oversættelse af *Grundlagen*:

> *Han revolutionerede logikken, der praktisk talt havde stået stille i 2000 år, hvor den aristoteliske logik havde hersket. Han er tillige en af de mest betydningsfulde tænkere indenfor matematikkens filosofi og inden for filosofisk logik og meningsteori.* [Frege, 2002, s. 9]

Vi har i nærværende kapitel begrænset os til at kigge på Freges *Begriffsschrift*, da det er med dette han lægger grundlaget for sin senere *Grundsetze* og sin vision om at formalisere aritmetikken[1]. Det for os essentielle i Freges arbejde findes derfor i rigelig udstrækning i *Begriffsschrift*.

3.2 *Begriffsschrift*

Med *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*[2], som vi vil kalde *Begriffsschrift*[3], bevæger Frege sig for første gang ind på logikkens område. van Heijenoort skriver om dette arbejde:

> *... although a mere booklet of eighty-eight pages, it is perhaps the most important single work ever written in logic. Its fundamental contributions, among lesser points, are the truth-functional propositional calculus, the analysis of the proposition into function and argument(s) instead of subject and predicate, the theory of quantification, a system of logic in which derivations are carried out exclusively according to the form of the expressions, and a logical definition of the notion of mathematical sequence.* [van Heijenoort, 1967, s. 1]

Freges intention med *Begriffsschrift* var at det skulle være et *formelsprog*, det vil sige et sprog skrevet med specielle symboler, *for rene tanker*, fri for retoriske konventioner. Sproget skulle være *modelleret ud fra aritmetikkens sprog*, hvilket

[1] Med aritmetik menes regning med hele tal ved hjælp af de basale regneoperationer; addition, subtraktion, multiplikation og division samt potensopløftning og roduddragning.

[2] Vi har benyttet en engelsk oversættelse i værket *From Frege to Gödel* (side 5-83) af Jean van Heijenoort, hvorfor henvisningerne til Freges arbejde vil se ud som følger: [van Heijenoort, 1967].

[3] Et begrebsskrift eller en ideografi.

vil sige, at det skulle være konstrueret udfra specifikke symboler, der skulle manipuleres i overensstemmelse med bestemte regler. Med et formelsprog mente Frege et *lingua characterica* i Leibniz' betydning. [van Heijenoort, 1967, s. 1, 5-8]

3.2.1 Freges propositionslogik

Som nævnt i citatet ovenfor introducerede Frege i *Begriffsschrift* den sandhedsfunktionelle propositionslogik. Denne havde sin spæde begyndelse i 1847 med publikationer af Boole såvel som af De Morgan, men tog først rigtig form med Hugh MacColls arbejde fra begyndelsen af 1877. Det som er skelsættende ved Freges introduktion af den sandhedsbaserede propositionslogik i forhold til det tidligere arbejde er, at han som den første formulerer sin logik som et logisk system, det vil sige, at han baserer det på en række aksiomer og slutningsregler samt sit præcise sprog. [Church, 1956, s. 155-156]

Noget af det første Frege foretager sig i sit begrebsskrift er at introducere tegnet $\vdash$, som følges af et symbol, hvis indhold kan bestemmes. Udtrykket

$$\vdash A$$

betyder 'A er et faktum'. Frege giver dernæst en ny notation, hvormed man kan skrive $B \rightarrow A$ som gør det er nemt at substituere formeludtryk med andre formeludtryk. Vi vil ikke gengive notationen her, da den er typografisk omstændelig. Frege siger, at udtrykket

$$B \rightarrow A$$

kan retfærdiggøres i tilfældene (1), (2) og (4) af følgende mulige:

1. A er sand, og B er sand
2. A er sand, og B er falsk
3. A er falsk, og B er sand
4. A er falsk, og B er falsk.

Frege benytter kun én slutningsregel, som består i at komme fra $\vdash B$ og $\vdash (B \rightarrow A)$ til $\vdash A$, hvilket er reglen *modus ponens*[4]. Ifølge van Heijenoort benytter Frege også en ikke-anført substitutionsregel. At 'A ikke er et faktum' skrives ved at sætte en lille lodret streg på midten af symbolet $\vdash$. Frege viser at 'og' og 'eller' kan udtrykkes udelukkende ved brug af negation og implikation. [van Heijenoort, 1967, s. 1-5, 11-21], [van Rootselaar, 1970-80a, s. 152-154]

Herefter udvikler han propositionslogikken på baggrund af følgende aksiomer[5] (de er ikke alle uafhængige):

1. $a \rightarrow (b \rightarrow a)$

[4] I moderne notation skrives *modus ponens* som $\frac{B \rightarrow A, B}{A}$. På engelsk kaldes reglen til tider *rule of detachment*.

[5] I *Begriffsschrift* har aksiomerne følgende formelnumre: (1), (2), (8), (28), (31) og (41).

2. $(c \to (b \to a)) \to ((c \to b) \to (c \to a))$
3. $(d \to (b \to a)) \to (b \to (d \to a))$
4. $(b \to a) \to (\neg a \to \neg b)$
5. $\neg(\neg a) \to a$
6. $a \to \neg(\neg a)$.

Visse af disse aksiomer er bevaret i præsentationer af moderne logik i dag. [van Heijenoort, 1967, s. 29, 36, 44-47]

3.2.2 Kvantorer og funktioner

Frege indfører nu kvantorer[6] og herigennem introducerer han kvantificeringsteorien. Dette er muligt for ham, fordi han anvender funktioner og argumenter i stedet for prædikater og subjekter. Et symbol i et udtryk, der anses for udskifteligt i nogle eller alle forekomster, kalder Frege for udtrykkets argument, mens den uudskiftelige del kaldes en funktion. Han vælger udtrykket

$$\Phi(A),$$

for en funktion af argumentet A. Funktioner af flere argumenter angiver han ved

$$\Psi(A, B).$$

Frege fortsætter

> *Since the sign* Φ *occurs in the expression* $\Phi(A)$ *and since we can imagine that it is replaced by other signs,* Ψ *and* X*, which would then express other functions of the argument* A*,* we can also regard $\Phi(A)$ as a function of the argument Φ. [van Heijenoort, 1967, s. 24]

Denne opfattelse af funktionsbegrebet kan opfattes således: Givet udsagnet *Sokrates er en mand*, kan dette opfattes som funktionen $\Phi(A)$, hvor Φ er funktionen, der udtrykker, at dens argument er en mand, og symbolet A repræsenterer Sokrates. Så kunne eksempelvis $\Phi(B)$ være udsagnet *Platon er en mand.* Vælger man at variere Φ og fastholde A opnås en funktion $\Psi(A)$, hvis værdier består af udsagn om Sokrates (A).

Frege pointerer, at den egenskab, at Φ kan betragtes som den udskiftelige del af $\Phi(A)$, gør hans funktionsbegreb mere generelt end funktionsbegrebet kendt fra den matematiske analyse. Desværre er det også denne egenskab, der senere skal give anledning til Russells paradoks (se afsnit 3.3 nedenfor). Den ovenfornævnte egenskab er ikke nødvendig for selve kvantificeringsteorien, men Frege tillader den for at kunne udlede nogle resultater om følger i tredje del af *Begriffsschrift.*

Frege introducerer yderligere tre aksiomer[7] i kvantificeringsteorien:

[6] Navnet 'kvantor' stammer dog fra C. S. Peirce, som i 1885 introducerede kvantorerne. Peirce var ubekendt med Freges arbejde og hævdede at ideen om kvantificering stammede fra O. H. Mitchells arbejde fra 1883. [Church, 1956, s. 288]

[7] I *Begriffsschrift* har disse aksiomer følgende formelnumrene: (52), (54), og (58).

7. $(c \equiv d) \rightarrow (f(c) \rightarrow f(d))$
8. $c \equiv c$
9. $(\forall a f(a)) \rightarrow f(c)$.

Aksiom 7 og 8 er identitetsaksiomer, hvorimod aksiom 9 udtaler sig om specialicering. [van Heijenoort, 1967, s. 5-8, 21-28, 50-51], [Harrison, 1996, s. 5]

Til trods for den inkonsistens Russell senere påviser i Freges system (her tænkes på hans *Grundtsetze*), er der næppe nogen tvivl om vigtigheden af det arbejde, Frege har udført. En af de største bedrifter er at Frege konstruerer sin logik som et sprog, der ikke skal suppleres af intuitive ræsonnementer. Han er tilmed meget omhyggelig med at beskrive sit sprog præcist, for eksempel taler han om bogstaver i stedet for variable, da 'variabel' er for upræcist. Frege er også fuldt ud klar over at et hvilket som helst system behøver regler, som ikke kan formuleres i systemet, i *Begriffsschrift's* tilfælde er det reglen *modus ponens* samt regler for håndtering af kvantorer. Dermed ikke sagt at disse regler er noget rent intuitivt, tværtimod er de helt klart formulerede, hvilket gør det muligt på 'mekanisk' vis at udlede resultater. [van Heijenoort, 1967, s. 3-4, 28-29]

Med hensyn til spørgsmålet om fuldstændighed og konsistens af *Begriffsschrift* skriver van Rootselaar

> *It should be observed that Frege did not construct his system for expressing pure thought as a formal system and therefore did not raise questions of completeness and consistency.* [van Rootselaar, 1970-80a, s. 153]

Vi vil nu kigge lidt nærmere på Russells paradoks og korrespondancen imellem Frege og Russell i 1902.

3.3 Russells paradoks

Som tidligere nævnt modtager Frege i 1902 et brev fra Russell, hvori Russell påpeger en modstrid i *Grundsetze*. Denne modstrid tager sit udgangspunkt i den definition af en funktion, som også findes i *Begriffsschrift* (se afsnit 3.2). I brevet[8] skriver Russell:

> *You state that a function, too, can act as the indeterminate element. This I formerly believed, but now this view seems doubtful to me because of the following contradiction. Let w be the predicate: to be a predicate that cannot be predicated of itself. Can w be predicated of itself? From each answer the opposite follows. Therefore we must conclude that w is not a predicate. Likewise there is no class (as a totality) of those classes which, each taken as a totality, do not belong*

[8] Vi har benyttet et genoptryk af korrespondancen fra *From Frege to Gödel* (side 124-128) af Jean van Heijenoort, hvorfor henvisningerne til denne vil se ud som følger: [van Heijenoort, 1967].

> *to themselves. From this I conclude that under certain circumstances a definable collection does not form a totality.* [van Heijenoort, 1967, s. 125]

Modstriden, som Russell pointerer, udspringer fra ideen om funktionen som den variable størrelse, idet denne ide gør det mere naturligt (og muligt) at overveje udtryk, som eksempelvis $\Phi(\Phi)$, det vil sige at funktionen kan tage sig selv som argument. I ovenstående citat lader Russell et prædikat w prædikere sig selv, hvilket er muligt i Freges teori. Denne diskussion føres videre i kapitel 4.

Russells paradoks (1902) kan formuleres på adskillige måder, hvilket Russell da også selv gør. I mængdeterminologi kan vi for eksempel have følgende: Givet mængden R, hvorom gælder, at

$$R = \{r | r \notin r\},$$

hvor r'erne også er mængder, tilhører R da R? Hvis $R \notin R$ har vi per definition af R at $R \in R$. Hvis derimod $R \in R$ følger igen af definitionen at $R \notin R$. Vi har altså to modstridende udsagn som er ækvivalente, mere præcist,

$$(R \in R) \Leftrightarrow (R \notin R).$$

En af de mere underholdende og lettest forståelige formuleringer som Russell giver af sit paradoks, er: En barber i en bestemt by har proklameret, at han vil klippe håret på de personer, og kun de personer, i byen som ikke klipper deres eget hår. Klipper barberen sit eget hår? [Katz, 1998, s. 809]

Men tilbage til korrespondancen imellem Russell og Frege. Seks dage efter at have afsendt sit brev, modtager Russell Freges svar. Frege skriver:

> *Your discovery of the contradiction caused me the greatest surprise and, I would almost say, consternation, since it has shaken the basis on which I intended to build arithmetic.* [van Heijenoort, 1967, s. 127]

Selv om Frege må have følt sig som ramt af lynet, indser og erkender han omgående problemet. På trods af at det forekommer ham at selve muligheden for at formalisere aritmetikken forsvinder, mener han dog, at det må være muligt at 'lappe' sit system, således at beviserne forbliver intakte. Afslutningsvis skriver han til Russell:

> *In any case your discovery is very remarkabel and will perhaps result in a great advance in logic, unwelcome as it may seem at first glance.* [van Heijenoort, 1967, s. 128]

Frege forsøger i et appendiks til 2. bind af *Grundtsetze der Arithmetik* at råde bod på sit systems inkonsistens. På et tidspunkt i løbet af de følgende år må han dog have konkluderet, at projektet ikke kunne gennemføres, i hvert fald arbejder han i de sidste år af sit liv på at grundlægge aritmetikken på geometrien. [van Heijenoort, 1967, s. 126-128], [Frege, 2002, s. 12-13]

Russells paradoks var nu ikke det eneste paradoks. Faktisk var der en hel række paradokser, som så dagens lys i tiden omkring år 1900. Nogle af disse vil kort blive gennemgået i det følgende.

3.4 Andre paradokser

Man kan dele paradokserne op i to typer; (1) de logiske matematiske paradokser, de som kun involverer notationer fra mængdelæren og (2) de semantiske paradokser, som gør brug af sproget og koncepter i dette.

Eksempler på semantiske paradokser er for eksempel Russells formulering af sit paradoks med barberen, eller det såkaldte *løgnerparadoks*, som er følgende: En mand siger: 'Jeg lyver'. Hvis han lyver, er sætningen sand, altså taler han sandt, hvorfor han ikke lyver. Hvis han ikke lyver, er sætningen falsk, altså lyver han. Det vil sige, at han både lyver og ikke lyver. Andre eksempler på semantiske paradokser er *Richards paradoks*, *Berrys paradoks*, *Grellings paradoks* og *Löbs paradoks*.[9]

I og med at de semantiske paradokser ikke er så interessante for matematikken, er det udelukkende de logiske matematiske paradokser som volder problemer for matematikkerne. Af disse kan nævnes fortrinsvist tre: *Russells paradoks*, som fremført i afsnit 3.3, *Cantors paradoks* og *Burali-Fortis paradoks*.

Vi gennemgår de to sidstnævnte her.

Cantors paradoks

Cantors paradoks (1899) involverer kardinaltal. Kardinaltallet $\overline{\overline{Y}}$ af en mængde Y er et udtryk for antallet af elementer i Y. Der gælder, at

$$\overline{\overline{Y}} = \overline{\overline{Z}},$$

hvis og kun hvis der findes en én-til-én-korrespondance imellem Y og Z. Der defineres, at

$$\overline{\overline{Y}} \leq \overline{\overline{Z}}$$

betyder, at Y står i én-til-én-korrespondance med en delmængde af Z, ydermere forstås

$$\overline{\overline{Y}} < \overline{\overline{Z}} \Leftrightarrow \overline{\overline{Y}} \leq \overline{\overline{Z}} \wedge \overline{\overline{Y}} \neq \overline{\overline{Z}}.$$

Cantor viste, at hvis $\mathcal{P}(V)$ er mængden af alle delmængder af V, så er

$$\overline{\overline{V}} < \overline{\overline{\mathcal{P}(V)}}.$$

Lad V være den universelle mængde, det vil sige mængden af alle mængder. Da $\mathcal{P}(V)$ selv er en mængde må den være indeholdt i V, som jo indeholder alle mængder. Så gælder, at $\overline{\overline{V}} \geq \overline{\overline{\mathcal{P}(V)}}$. På den anden side gælder der ifølge Cantors sætning, at $\overline{\overline{V}} < \overline{\overline{\mathcal{P}(V)}}$. Altså har vi:

$$\overline{\overline{V}} \geq \overline{\overline{\mathcal{P}(V)}} \wedge \overline{\overline{V}} < \overline{\overline{\mathcal{P}(V)}},$$

hvilket er en modstrid. [Mendelson, 1997, s. 2]

[9] En gennemgang af disse paradokser findes i [Mendelson, 1997, s. 2-3].

Burali-Fortis paradoks

Burali-Fortis paradoks (1897) er analogt til Cantors paradoks, blot udtaler dette sig om ordinaler. Den generelle definition af en ordinal er: Lad $(X, \leq)$ være en velordnet mængde og definer for hvert $a \in X$ mængden X_a ved $X_a = \{x \in X | x < a\}$. X_a kaldes det a'te *afsnit* af X. Hvis $(X, \leq)$ for alle $a \in X$ opfylder, at $a = X_a$, det vil sige, at det a'te element er lig det a'te afsnit i X, da kaldes den velordnede mængde $(X, \leq)$ en *ordinal*. Som mængde vil en ordinal have følgende udseende:

$$\alpha = \{\beta | \beta < \alpha\}.$$

Det vil sige, at en ordinal er mængden af alle mindre mængder. Det er klart, at der må findes en mindste (og første) ordinal, nemlig den tomme mængde, $\emptyset$ (betragtet som en velordnet mængde). Denne ordinal kaldes gerne for 0. De næste ordinaler skabes på analog vis

$$\begin{aligned} 0 &= \emptyset \\ 1 &= \{0\} \\ 2 &= \{0, 1\} \\ &\vdots \\ \omega &= \{0, 1, 2, \ldots, n, n+1, \ldots\} \\ \omega + 1 &= \{0, 1, 2, \ldots, n, n+1, \ldots, \omega\} \\ &\vdots \end{aligned}$$

hvor ω er den første uendelige ordinal[10] Generelt gælder, at givet en ordinal α vil dens efterfølger $\alpha + 1$ være givet ved

$$\alpha + 1 = \alpha \cup \{\alpha\}.$$

Burali-Fortis paradoks lyder nu: Givet et vilkårligt ordinaltal findes altid et større ordinaltal. Men ordinaltallet bestemt ud fra mængden af alle ordinaltal er det største ordinaltal. [Devlin, 1993, s. 18, 23-25], [Mendelson, 1997, s. 2]

3.4.1 Undgåelse af paradokserne

Analyse af paradokserne har ført til forskellige forslag til måder, hvorpå man kan undgå dem. Russell bemærkede at visse af paradokserne indeholder selvreferencer[11], for eksempel 'mængden af alle mængder', og foreslog typeteorien som løsning på problemet.

En anden kritik af de logiske paradokser går på antagelsen om, at der for enhver egenskab $P(x)$ eksisterer en tilsvarende mængde af alle elementer x der har egenskaben, altså som opfylder $P(x)$. Afvises denne antagelse, kan de logiske paradokser ikke længere udledes. Russells paradoks eksisterer ikke, da der

[10] Bemærk, at man som følge af definitionen af ordinaler kan definere de naturlige tal ved specifikke mængder, nemlig de endelige ordinaler. [Devlin, 1993, s. 23]

[11] Bemærk, at det ikke er noget generelt for paradokser, at de indeholder selvreferencer, der findes adskillige paradokser, logiske såvel som semantiske, hvorom dette ikke gør sig gældende.

ikke findes en mængde af alle mængder, der ikke tilhører sig selv. Ligeledes forekommer Cantors og Burali-Fortis paradokser ikke, da der ikke findes en universel mængde, eller en mængde, som indeholder alle ordinaltal. I stedet for denne antagelse skal der opstilles postulater, som muliggør påvisning af eksistensen af de mængder, som benyttes i den praktiserede matematik. En sådan aksiomatisk teori for mængdelæren blev opstillet af Zermelo i 1908. I den aksiomatiske mængdelære gør man ligesom i typeteorien brug af en hierarkisk struktur. Man indfører begrebet en klasse til at dække over 'store' samlinger af mængder. Således kan man ikke tale om mængden af alle mængder, istedet tales om klassen af alle mængder. [Mendelson, 1997, s. 1-5], [van Heijenoort, 1967, s. 199-200], [Devlin, 1993, s. 46-47]

Overordnet kan man sige, at de paradokser som opstod omkring år 1900 gav anledning til tre vigtige udviklinger inden for logikken: (1) bevisteori og jagten på et konsistensbevis, (2) typeteorien og (3) den aksiomatiske mængdelære. [van Heijenoort, 1967, s. vi]

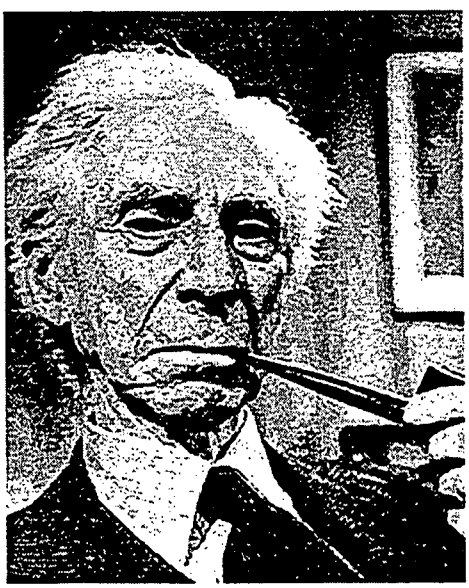
Typeteorien vil vi kigge nærmere på i næste kapitel.

4 Russells typeteori

Ligesom Frege betragtes som grundlæggeren af den moderne logik, anses Russell for at være grundlæggeren af typeteorien. Russell var nemlig, som tidligere nævnt, den første, der indførte begrebet en *type*, for på den måde at undgå de logiske matematiske paradokser. Vi vil i nærværende kapitel gennemgå Russells arbejde med typeteorien samt skitsere Whiteheads og hans arbejde med at formalisere matematikken. Men først en biografi for Russell.

4.1 Biografi

Bertrand Russell blev født i England i 1872. Hans familie var velhavende og spillede en stor rolle i det sociale, intellektuelle og politiske liv i England. Russell modtog privatundervisning i sit eget hjem, indtil han i 1890 blev optaget på Trinity College i Cambridge. Klasseundervisning huede ikke Russell, og først da han blev optaget i *The Apostles* i 1892, fik han den udfordring, han længtes efter. The Apostles var et lille samfund, som bestod af eliten af akademikerne ved Cambridge, hvis mål var at diskutere alle slags emner. Her stiftede Russell



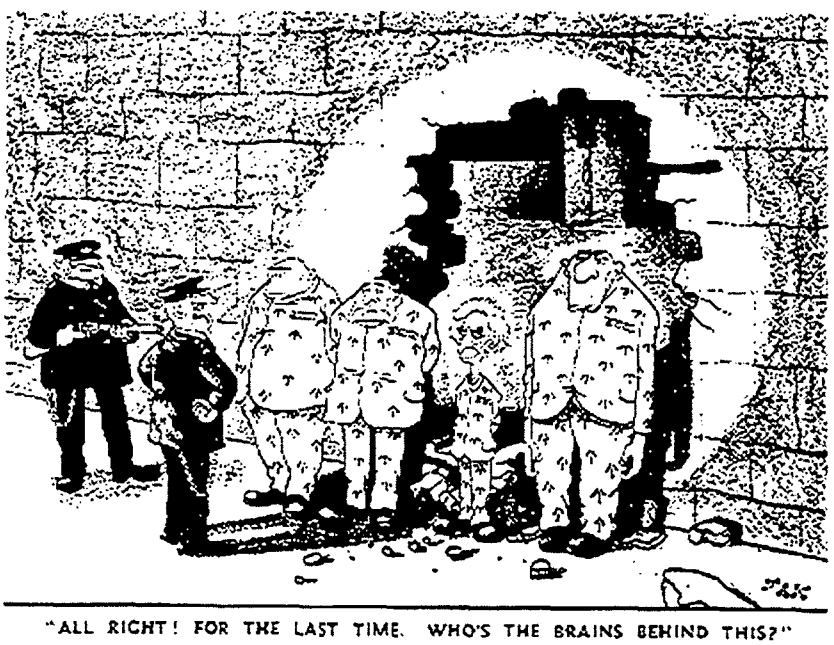
Figur 4.1 Bertrand Arthur William Russell. Født 18. maj 1872 i Monmouthshire, England. Død 2. februar 1970 i Plas Penrhyn, Wales. [O'Connor and Robertson, 2003]

bekendtskab med Alfred Whitehead, som på dette tidspunkt var ansat ved det matematiske institut i Cambridge. Whitehead nærede stor respekt for Russells evner, og de kom senere til at arbejde en hel del sammen.

I 1897 startede Russell på en omfattende afhandling om matematikkens principper, i hvilken han ønskede at fremstille sin ide om at matematikken kan udtrykkes og udledes ved hjælp af få logiske principper. Hans møde med Peano i 1900 havde stor indflydelse på hans arbejde selv kaldte han det *a turning point in my intellectual life*. Peano forsøgte systematisk at basere matematikken på et rent formelt og abstrakt fundament, hvortil han udviklede et nyt symbolsprog. Russell, der beundrede Peanos forståelse for matematisk logik, tilegnede sig hurtigt dette sprog. Efter mødet valgte Russell at skrive sit værk om matematikkens principper om. Første bind af *Principles of Mathematics* udkom derfor først tre år senere. [Broadent, 1970-80, s. 9-12]

Russell blev i 1910 tildelt en særlig stilling indenfor logik og matematisk filosofi ved Cambridge. Her arbejdede han sammen med Whitehead på andet bind af *Principles of Mathematics*. Bogen udkom dog aldrig, men blev i stedet erstattet af tre-binds værket *Principia mathematica*, som Whitehead og Russell udgav i henholdsvis 1910, 1912 og 1913. [Broadent, 1970-80, s. 10-12]

Under første verdenskrig tilkendegav Russell i flere artikler sin markante modstand imod krigen. Artiklerne førte til, at Russell i 1916 blev fyret fra sin stilling i Cambridge, og en senere artikel førte i 1918 til en seks måneder lang fængsels-



Figur 4.2 Denne vittighedstegning stammer fra avisen *Evening Standard*. Den refererer til Russells fængselsstraf i september 1961, som følge af hans dom for 'public order charges' efter en stor fredsdemonstration i anledning af Hiroshima-dagen (6. august) i det centrale London. [Barrette and Duncan, 2003]

dom, som han afsonede i Brixton fængsel. Efter krigen blev Russell atter inviteret til at holde foredrag ved Cambridge, hvilket han takkede ja til. Russells politiske aktivitet ophørte imidlertid ikke, og så sent som i 1961 modtog han endnu en fængselsdom som følge af demonstrationer mod atomkraft (se figur 4.2). Senere, i 1970, var Russell en stor forkæmper for kvinders valgrettigheder. Han nåede at skrive adskillige bøger, og modtog i 1950 Nobelprisen i litteratur. Han skrev i sin selvbiografi, at hans tre store lidenskaber i livet var *the longing for love, the search for happiness, and unbearable pity for the suffering of mankind.* [Broadent, 1970-80, s. 9-12], [Barrette and Duncan, 2003]

I sit matematiske arbejde beskæftigede Russell sig nærmest udelukkende med at undersøge og analysere matematikkens grundlag. Som nævnt ovenfor udgav han i 1903 værket *Principles of Mathematics.* Under udarbejdelsen af *Principles of Mathematics*, i forbindelse med arbejdet vedrørende klasser, formulerede han sit paradoks (se afsnit 3.3). I årene efter arbejdede han på forskellige teorier, som skulle være fri for de paradokser, der plagede matematikken. I 1908 udgav han så artiklen *Mathematical Logic as based on the Theory of Types*, i hvilken han redegjorde for sin typeteori. Teorien gjorde det muligt at undgå de nævnte paradokser, og Whitehead og Russell gjorde derfor brug af den i deres værk *Principia mathematica.* [van Heijenoort, 1967, s. 150-151]

I resten af dette kapitel vil vi forsøge at gengive de oprindelige ideer bag typeteorien.

4.2 Typeteorien

I diskussionen af Russells typeteori har vi valgt at tage udgangspunkt i hans artikel fra 1908, samt i anden udgave af *Principia mathematica*, da introduktionen til dette værk indeholder en række betragtninger vedrørende typeteorien, som ikke kommer til udtryk i 1908-artiklen. Vi har valgt ikke at gengive de aksiomer Russell opstiller for sit 1908-system, da disse optræder i kapitel 5 i forbindelse med gennemgangen af Gödels arbejde.

4.2.1 *Mathematical logic as based on the theory of types*

Russell angiver i indledningen til sin 1908-artikel[1] følgende som motivation for arbejdet:

> *The following theory of symbolic logic recommended itself to me in the first instance by its ability to solve certain contradictions, of which the one best known to mathematicians is Burali-Forti's concerning the greatest ordinal. But the theory in question seems not wholly dependent on this indirect recommendation; it has also, if I am not mistaken, a certain consonance with common sense which makes it inherently credible.* [van Heijenoort, 1967, s. 152-153]

[1] Vores udgave af 1908 artiklen er et genoptryk, der er at finde i værket *From Frege to Gödel* (side 150-182) af Jean van Heijenoort, hvorfor henvisningerne til denne artikel vil se ud som følger: [van Heijenoort, 1967].

Arbejdet med de matematiske og logiske paradokser havde fået Russell til at indse, at disse havde et fælles træk, som involverede en selvreference eller form for cirkelslutning. Problemet med paradokserne er, at de involverer samlinger af objekter, der indeholder elementer, som kun kan defineres udfra samlingen selv. Russell kommenterer dette således:

> *Thus all our contradictions have in common the assumption of a totality such that, if it where legitimate, it would at once be enlarged by new members defined in terms of itself.*
> *This leads us to the rule: 'Whatever involves* all *of a collection must not be one of the collection'; or, conversely: 'If, provided a certain collection had a total, it would have members only definable in terms of that total, then the said collection has no total.'* [van Heijenoort, 1967, s. 155]

Ovenstående regel kalder Russell *'the vicious-circle principle'*. Som eksempel nævnes udsagnet 'alle udsagn er enten sande eller falske'. Dette udsagn er meningsløst, da det refererer til *samlingen af alle udsagn*, som om denne samling var en fast størrelse eller helhed. Russell nævner i forbindelse med eksemplet, at man er nødt til at finde en erstatning for sådanne udsagn, fordi man ikke kan undvære udsagnene i forbindelse med logisk deduktion. [van Heijenoort, 1967, s. 155, 163]

Russells løsning af problemet involverer en definition af begrebet en type:

> *A* type *is defined as the range of significance of a propositional function, that is, as the collection of arguments for which the said function has values.* [van Heijenoort, 1967, s. 161-162]

Ovenstående definition af begrebet en type giver anledning til konstruktionen af et hierarki. Konstruktionen foregår således: Russell kalder en proposition uden variable for en elementær proposition. Som eksempel herpå kan nævnes *Sokrates er en mand.* Russell forklarer:

> *In an elementary proposition we can distinguish one or more* terms *from one or more* concepts*; the* terms *are whatever can be regarded as the subject of the proposition, while the concepts are the predicates or relations asserted of these terms. The terms of elementary propositions we will call* individuals*; these form the first or lowest type.* [van Heijenoort, 1967, s. 164]

I det simple eksempel ovenfor er individet altså *Sokrates*. Elementære propositioner og propositioner, der kun indeholder bundne variable af individ-typen, kalder Russell *førsteordens-propositioner*. Disse udgør så den anden logiske type. Propositioner i hvilke førsteordens-propositioner optræder som bundne variable kaldes så *andenordens-propositioner*. Disse udgør den tredje logiske type og så videre. Russell forklarer, at det i praksis kun er de *relative* typer af funktionernes argumenter, der er interessante. Den laveste type, der forekommer i en given sammenhæng, kan derfor betragtes som individtypen. [van Heijenoort, 1967, s. 164]

Russell konstruerer nu et nyt hierarki af *funktioner*, da han finder det mere bekvemt at arbejde med propositionsfunktioner end med propositioner. Dette hierarki defineres analogt til det første:

> *A function whose argument is an individual and whose value is always a first-order proposition will be called a first-order function. A function involving a first-order function or proposition as apparent variable will be called a second-order function, and so on. A function of one variable which is of the order next above that of its argument will be called a* predicative *function; the same name will be given to a function of several variables if there is one among these variables in respect of which the function becomes predicative when the values are assigned to all the other variables. Then the type of a function is determined by the type of its values and the number and type of its arguments.* [van Heijenoort, 1967, s. 164]

En funktion af orden n med ét argument kaldes altså prædikativ hvis dens argument er af orden $n-1$. En funktion af orden n med flere argumenter kaldes prædikativ hvis argumentet med højest orden er af orden $n-1$. Da det kun er argumenternes relative typer, der er væsentlige i praksis, kan argumentet for en prædikativ funktion med ét argument opfattes som et individ. På den måde kan en prædikativ funktion altså opfattes som førsteordens-funktion.

Det er interessant, at funktionernes orden i dette hierarki både afhænger af ordenen af funktionens argumenter og ordenen af de bundne variable. Dette adskiller Russells typeteori fra den simple typeteori, som vi skal stifte bekendtskab med senere i rapporten (se kapitel 7 og kapitel 8). Russells typeteori kaldes *ramificeret typeteori*[2]. [van Heijenoort, 1967, s. 151]

I den ramificerede typeteori undgås de matematiske paradokser, fordi den føromtalte selvreference ikke længere er mulig på grund af *'the vicious-circle principle'*. Russells paradoks, for eksempel, opstår ikke, fordi klassen af alle klasser, som ikke indeholder sig selv, ikke længere selv er en klasse. På tilsvarende vis undgås de andre paradokser. Russells restriktioner virker derfor umiddelbart som en god løsning på problemerne med paradokserne i matematikken. Restriktionerne er dog ikke helt uproblematiske.

Quine bemærker i forordet til vores udgave af artiklen, at der er en del uklarheder i Russells fremstiling. Derudover er der problemer, som vi vil beskrive i næste afsnit. [van Heijenoort, 1967, s. 151]

Reducerbarhedsaksiomet

De hierarkier, der opstilles i den ramificerede typeteori, skaber en række problemer, som tvinger Russell til at indføre et aksiom kaldet *the Axiom of Reducibility*[3]. Han indleder selv sit afsnit om dette aksiom således:

[2]Dette er en fordanskning af *ramified type theory*. *Ramified* kan eventuelt oversættes med 'forgrenet'.

[3]Vi har oversat *Axiom of Reducibility* til 'reducerbarhedsaksiomet'.

> *A propositional function of x may, as we have seen, be of any order; hence any statement about 'all properties of x' is meaningless. But it is absolutely necessary, if mathematics is to be possible, that we should have some method of making statements which will usually be equivalent to what we have in mind when we (inaccurately) speak of 'all properties of x'.* [van Heijenoort, 1967, s. 164]

Hierarkierne opstilles for at undgå de forskellige matematiske og logiske paradokser, men uheldigvis forhindrer de også i et vist omfang udledningen af matematiske teoremer. Som eksempel kan nævnes behandlingen af de reelle tal. De irrationale tal defineres ved hjælp af de rationale tal, som igen defineres ved hjælp af de naturlige tal, det vil sige, de tre talmængders elementer har forskellige typer. Dette gør det meget vanskeligt at behandle de reelle tal, som jo omfatter alle de førstnævnte talmængder. [Kline, 1972, s. 222]

På grund af besværlighederne indfører Russell det ovenfor nævnte aksiom. Aksiomet er en antagelse om, at enhver propositionsfunktion af orden n er ækvivalent med en prædikativ funktion af orden n. Som tidligere nævnt kan en prædikativ funktion altid opfattes som en førsteordens-funktion, så aksiomet udtrykker overordnet set, at enhver propositionsfunktion er ækvivalent med en førsteordens-funktion. Lidt forsimplet sagt har reducerbarhedsaksiomet den effekt, at de opstillede hierarkier jævnes med jorden. Opstillingen af hierarkierne synes derfor at være spildt arbejde. I næste afsnit skal vi se på Russells egne kommentarer til aksiomet.

4.3 *Principia mathematica*

Principia mathematica var et forsøg på at forene to udviklinger indenfor matematikken, nemlig aksiomatiseringen af forskellige dele af matematikken, herunder mængdelæren (ledet af blandt andre Georg Cantor), og den udvikling af symbolsk logik, som Peano arbejdede med. Hovedformålet med *Principia mathematica* var at opstille et konkret system, der gjorde brug af symbolsk logik, i hvilket den ovenfor nævnte aksiomatisering kunne finde sted. Et andet mål var, at det system, der skulle opstilles, skulle løse problemerne angående de mange paradokser, der plagede matematikken på dette tidspunkt. Whiteheads og Russells system gjorde derfor brug af Russells typeteori. I anden udgave af *Principia mathematica* fra 1927 knytter forfatterne følgende kommentar til reducerbarhedsaksiomet:

> *One point in regard to which improvements is obviously desirable is the axiom of reducibility. This axiom has a purely pragmatic justification: it leads to the desired results, and to no others. But clearly it is not the sort of axiom with which we can rest content.* [Whitehead and Russell, 1927, vol. I s. xiv]

Fra ovenstående citat er det tydeligt, at Whitehead og Russell ikke selv var tilfredse med reducerbarhedsaksiomet. Forsøg på at arbejde med den ramificerede typeteori uden at gøre brug af aksiomet resulterede dog ikke i de store gennembrud.

Den polske matematik-filosof Leon Chwistek (1884-1944) gjorde et sådan forsøg. Om hans arbejde skriver forfatterne i forordet til anden udgave af værket:

> *Dr Leon Chwistek took the heroic course of dispensing with the axiom without adopting any substitute; from his work, it is clear that this course compels us to sacrifice a great deal of ordinary mathematics.* [Whitehead and Russell, 1927, vol. I s. xiv]

Besværlighederne, der var forbundet med Russells typeteori, gjorde at den aldrig slog igennem. Det blev derfor foreslået af Chwistek allerede i 1921 og af Frank Ramsey i 1926, at den ramificerede typeteori skulle modificeres til den simple typeteori, som præsenteres senere i rapporten. [Whitehead and Russell, 1927, vol. I s. xiv], [Church, 1956, s. 56]

Principia mathematica og det system, der blev opstillet i værket, blev senere genstand for mange undersøgelser og kom på den måde til at spille en stor rolle for matematikkens formalisering. Blandt de matematikere, som underkastede systemet en nærmere analyse, var Kurt Gödel, som vi skal møde i næste kapitel.

5 Gödels ufuldstændighedssætninger

Gödel er den matematiker som i nyere tid har vist nogle af de mest uventede og mest usædvanlige resultater indenfor matematikken – ufuldstændighedssætningerne. Vi vil i dette kapitel give en beskrivelse af disse sætninger samt nogle af metoderne benyttet i beviserne for disse. Derudover vil vi forsøge at belyse sætningernes betydning for typeteorien og formaliseringen af matematikken. Men først Gödels biografi.

5.1 Biografi

Gödels fader, Rudolf Gödel, arbejdede størstedelen af sit liv for tekstilfabrikken 'Friedrich Redlich' i Brünn, hvor han var manager og medejer – Rudolf Gödel tilhørte den øvre middelklasse. Han giftede sig med Marianne Handschuh, datter af en væver, som oprindeligt kom fra Rhinelandet. Parret fik tilsammen to sønner; den ældste, Rudolf, som blev en radiolog i Wien; og den yngre, Kurt

Figur 5.1 Kurt Friedrich Gödel. Født 28. april 1906 i Brünn, Tjekkiet. Død 14. januar 1978 i Princeton, New Jersey. [O'Connor and Robertson, 2003]

Gödel. Ifølge Gödels broder, havde Gödel en lykkelig barndom, selvom han var generet og til tider lidt nærtagende. Hans familie kaldte ham for 'Herr Warum' på grund af hans kontinuerlige spørgsmål. I en alder af fem år oplevede Gödel en mild form for angstneurose, som han dog kom sig helt over. Tre eller fire år senere fik han en reumatisk feber. Til trods for at Gödel kom sig helt over denne sygdom, forblev han hele livet i den tro, at hans hjerte havde lidt permanent skade deraf. Ifølge Gödels broder var det også netop denne episode, der var årsagen til Gödels senere hypokondri. Gödel klarede sig fremragende i skolen og fik topkarakterer i alle fag. Som helt ung var han specielt interesseret i sprog og religion, men som 14-årig opstod interessen for matematik, og et par år senere interessen for filosofi. I 1924, da Gödel færdiggjorde gymnasiet, var han allerede fortrolig med store dele af den matematik, der blev undervist i ved universiteterne. [Moore, 1970-80, s. 348-349], [Shanker, 1988, s. 4-5]

I 1924 indskrev Gödel sig ved universitetet i Wien. Hans Hahn, en underviser hvis interesseområder var generel topologi og logik, blev snart Gödels fortrukne vejleder, og han introducerede Gödel for Wienerkredsen – en gruppe logiske positivister – som på det tidspunkt samledes omkring Moritz Schlick. Gödel kom her fra 1926 til 1933, hvor han holdt op med at deltage i møderne, fordi han i løbet af årene havde udviklet en anden filosofisk overbevisning end den her fremherskene. [Moore, 1970-80, s. 349-350]

Gödel fuldendte sin disputats i sommeren 1929, og i februar 1930 modtog han en doktorgrad i matematik fra universitetet i Wien. I oktober 1929 begyndte han at deltage i Mengers matematiske kollokvium, her mødte han blandt andre Alfred Tarski. Det var i denne kreds at Gödel i maj 1930 fremførte sit bevis for fuldstændigheden af prædikatlogikken. [Moore, 1970-80, s. 350]

I september 1930 nævnte Gödel sit første ufuldstændighedsbevis ved en konference i Königsberg. Kort efter fulgte hans andet ufuldstændighedsbevis (se afsnit 5.2), og i november samme år modtog *Monartshefte für Matehematik und Physik* hans artikel derom. Gödel indleverede artiklen som sit *Habilitationsschrift* ved universitetet i Wien i juni 1932. I marts 1933 blev han udnævnt til privatdocent. Det første kursus han underviste i handlede om aritmetikkens grundlag. Gödel var efter sigende en meget generet underviser og talte for det meste til tavlen i stedet for til de studerende. [Moore, 1970-80, s. 350], [Shanker, 1988, s. 5-7]

I 1933 kom Hitler til magten. I begyndelsen havde dette ingen betydning for Gödels liv i Wien, men efter mordet på hans ven Moritz Schlick, som blev myrdet af en nationalsocialistisk student, fik Gödel sit første psykiske sammenbrud. Da Gödel blev rask igen, modtog han et tilbud om et gæsteprofessorat ved *Institute of Advanced Study* (ISA) i Princeton, hvor han underviste i ufuldstændighedssætningerne i 1933-34. Efter at være vendt hjem til Europa i 1934 fik Gödel endnu et nervøst sammenbrud og tilbragte noget tid på et sanatorium. I årene frem til 1940, hvor Gödel emigrerede til USA viste han nye banebrydende resultater inden for et for ham nyt forskningsområde, nemlig mængdelære.

I 1935 kunne han under et besøg ved Princeton fortælle John von Neumann, at det var lykkedes ham at vise, at udvalgsaksiomet var konsistent relativt til aksiomerne i en særlig udgave af mængdelæren. Senere – i 1937 – gennemførte han et bevis relativt til et system, som i dag kaldes Bernays-Gödel-mængdelæren. Samme år viste han, at kontinuumshypotesen var konsistent relativt til en mængdelære, udviklet af von Neumann samt til Zermelo-Fraenkels system, og

det system Whitehead og Russell havde opstillet i *Principia mathematica.* Den 20. september samme år giftede Gödel sig med natklubdanserinden Adele Porkert Nimbursky, som han havde kendt i henimod ti år.

I 1939 blev han af nazisterne erklæret 'fit for garrison duty', og af frygt for at blive indkaldt anmodede han Oswald Veblen fra Princeton om at skaffe ham et amerikansk visa. Dette lod sig gøre i starten af 1940, men siden krigen havde gjort det farligt at krydse Atlanten, måtte hr. og fru Gödel først rejse med den transsibiriske jernbane og derefter sejle fra Yokohama til San Francisco. [Shanker, 1988, s. 5-7], [Moore, 1970-80, s. 351], [O'Connor and Robertson, 2003]

I Princeton fik Gödel et stille og roligt socialt liv og her blev han også venner med Albert Einstein og Oskar Morgenstern, der ligesom han selv kom fra Østrig. I 1943 begyndte han at forske i de mere filosofiske aspekter af matematikken, og få år senere gik han over til at forske i filosofi. Gödel skrev senere '*the greatest philosophical influence on me came from Leibniz, whom I studied about 1943-1946*'. I perioden 1947-1951 var det dog fra Immanuel Kant at Gödel hentede de største stimuli. Det er måske her på sin plads at nævne, at Gödel var platonist, hvilket var en næsten uhørt holdning i filosofisk matematik på det tidspunkt. I 1953 blev han udnævnt til professor i matematik ved *Institute of Advanced Study.* I den sidste tredjedel af sit liv blev han tildelt adskillige ærestitler, blandt andet modtog han i 1951 den første Einstein Award. [Moore, 1970-80, s. 351-352]

Figur 5.2 Kurt Gödel sammen med sin nære ven Albert Einstein i Princeton i 1950. [O'Connor and Robertson, 2003]

Moore skriver følgende om mennesket Gödel.

> *Gödels personality was idiosyncratic. Shy and solemn, short and slight of build, he was very courteous but lacked warmth and sensitivity. ... A hypochondriac whose health actually was relatively poor, he often complained of stomach trouble but remained distrustful of doctors, believing throughout his life that his medical judgment was better than theirs. ... Keenly sensitive to cold, he was often seen in Princeton wearing a large overcoat, even on warm summer days.* [Moore, 1970-80, s. 352]

I 1976 gik Gödel på pension. De sidste år af sit liv led han af depression og forfølgelsesvanvid. I slutningen af december 1977 lod Gödels kone ham indlægge; to uger senere døde han. Ifølge dødsattesten skyldes døden 'malnutrition and inanition' forårsaget af en 'personality disturbance'. Af frygt for at hans mad skulle være blevet forgiftet, nægtede Gödel at indtage føde, og sultede således sig selv ihjel. [Moore, 1970-80, s. 352]

5.2 Ufuldstændighedssætningerne

Gödel var ligesom mange andre i tiden inspireret af det såkaldte *Hilberts program*, som omhandlede udviklingen af formelle systemer for forskellige matematiske teorier. Selve udviklingen af systemerne var ikke det vanskeligste. Næste skridt i programmet var imidlertid, at disse systemer skulle bevises *konsistente*, det vil sige, at det ikke er muligt at udlede både P og P's negation, altså

$$\vdash P \quad \text{og} \quad \vdash \neg P.$$

Hilbert anså dette skridt i programmet som det vigtigste. Gennem sit forsøg på at efterstræbe Hilberts program indså Gödel, at konceptet beviselighed kunne defineres aritmetisk. Dette førte til hans ufuldstændighedssætninger, som ironisk nok kom til at kuldsejle Hilberts program (i hvert fald som oprindeligt fremført). [Shanker, 1988, s. 6]

5.2.1 Den første sætning

Gödel fremførte sit første resultat i en diskussion ved en konference i Königsberg i september 1930[1], hvor han dagen forinden havde forelæst over sit bevis for fuldstændigheden af prædikatlogikken. Reaktionerne (ikke altid baseret på forståelse) på Gödels ufuldstændighedssætning var umiddelbare, omend varierende fra dybsindig værdsættelse hos von Neumann over kraftig kritik fra Zermelo til påkaldelse af ejerskab fra Paul Finsler, hvilket Gödel dog ringeagtende afslog. Den første ufuldstændighedssætning lyder omtrent som følger:

[1] Gödel havde dog omtalt sit bevis privat overfor Carnap allerede den 26. august. [Moore, 1970-80, s. 353]

Sætning 5.1 (Gödels første ufuldstændighedssætning)
Givet et (konsistent) formelt system vil dette ikke engang kunne omfatte alle sandheder om elementær talteori; der vil altid være sande udsagn som ikke kan bevises indenfor systemet.

Det at 'der er *sande udsagn* som ikke kan bevises indenfor systemet' vil nogen måske betragte som lidt af en tilsnigelse. Lad os forklare hvorfor. Gödels sætning siger, at der i et sådant formelt system altid vil eksistere udsagn (propositioner), lad os kalde dem τ, for hvilke hverken τ eller $\neg\tau$ kan bevises. I og med at de fleste matematikere er af den opfattelse, at der om et arbitrært udsagn τ må gælde, at enten τ eller $\neg\tau$ er sand (lad os sige at det er τ der er sand), må der således eksistere sande udsagn τ, som ikke er beviselige teoremer inden for systemet. Tilsnigelsen fremkommer, hvis man ikke er af denne opfattelse, eksempelvis hvis man er intuitionist, og i så fald kan man kun konkludere at hverken τ eller $\neg\tau$ er beviselig. [Shanker, 1988, s. 6], [Harrison, 1996, s. 7-8], [van Heijenoort, 1967, s. 595]

Vi vil kort skitsere metoden (ideen) bag Gödels bevis for ovenstående sætning. Lad os starte med at definere en *Gödel-nummerering.*

Definition 5.1
En Gödel-nummerering *for et system er en funktion, som tildeler ethvert udtryk i systemet et tal kaldet* Gödel-tallet *for det pågældende udtryk, således at*

i. der findes en effektiv procedure til udregning af Gödel-tallet for ethvert udtryk,

ii. forskellige udtryk har altid forskellige Gödel-tal, og

iii. der findes en effektiv procedure til bestemmelse af om et tal n er Gödel-tallet for et udtryk, og i så fald, en procedure til at genskabe udtrykket ud fra n.

Gödel selv baserer sin nummerering på aritmetikkens fundamentalsætning[2] og opnår derved den ønskede entydighed. Ethvert primitivt symbol s i det system, der betragtes, tildeles et naturligt tal $g(s)$. Lad $p_1, p_2, \ldots, p_n$ være de første n primtal $2, 3, 5, 7, 11, 13, \ldots, p_n$ givet i deres naturlige orden. Enhver streng $S_i = s_1 s_2 \ldots s_n$ for $1 \leq i \leq n$ tildeles da tallet

$$g(s_1 s_2 \ldots s_n) = p_1^{g(s_1)} p_2^{g(s_2)} \ldots p_n^{g(s_n)},$$

og enhver endelig sekvens $S_1, S_2, \ldots, S_n$ af strenge tildeles tallet

$$g(S_1, S_2, \ldots, S_n) = p_1^{g(S_1)} p_2^{g(S_2)} \ldots p_n^{g(S_n)}.$$

Gödel forklarer, at enhver sætning i et formelt sprog kan tildeles et sådant entydigt 'Gödel-tal' $\ulcorner\phi\urcorner$, og ligeledes kan ethvert bevis – der er trods alt kun tælleligt mange sætninger og beviser. Spørgsmålet om hvorvidt $q = p_1^{g(s_1)} p_2^{g(s_2)} \ldots p_n^{g(s_n)}$ er (den nummeriske indkodning af) et bevis for ϕ kan nu

[2] Aritmetikkens fundamentalsætning siger, at ethvert positivt tal kan repræsenteres på præcis én måde som et produkt af primtal.

udtrykkes som et repræsentabelt prædikat $\text{Prov}(q, \ulcorner\phi\urcorner)$, med den basale egenskab at

$$\text{hvis} \quad \vdash \phi \quad \text{så også} \quad \vdash \exists q(\text{Prov}(q, \ulcorner\phi\urcorner)). \tag{5.1}$$

Ved at benytte et diagonaliseringsargument var Gödel i stand til at angive et udsagn ϕ, som hævder sin egen ubeviselighed i systemet, det vil sige

$$\vdash \phi \equiv \neg\exists q(\text{Prov}(q, \ulcorner\phi\urcorner). \tag{5.2}$$

Hvis nu det formelle system er konsistent, må der gælde at ϕ *ikke* er et teorem, for i modsat fald, $\vdash \phi$, gælder ifølge 5.1 også $\vdash \exists q(\text{Prov}(q, \ulcorner\phi\urcorner))$ og ifølge 5.2 dermed $\vdash \neg\phi$. På den anden side er ϕ helt bestemt *sand*, idet sætningen jo netop erklærer denne ubeviselighed. [van Heijenoort, 1967, s. 438-440, 582-583, 592-595, 596-599, 601], [Harrison, 1996, s. 7-8], [Margaris, 1967, s. 185-190]

Ideen med Gödeltallene er kort fortalt, at oversætte sætninger *om* det formelle system til sætninger *i* det formelle system.

5.2.2 Den anden sætning

Gödels anden ufuldstændighedssætning fulgte allerede to måneder efter den første (det første kendskab til den er fra den 20. oktober 1930, hvor Gödel indleverede et resume af resultatet til universitetet i Wien)[3]. Sætningen siger i store træk følgende.

Sætning 5.2 (Gödels anden ufuldstændighedssætning)
En hvilken som helst realistisk matematisk teori er ude af stand til at bevise sin egen konsistens.

Det vil altså sige, at et konsistensbevis for et formelt system kræver *flere* matematiske resurcer end der er til rådighed i systemet. Med 'realistisk' menes et konsistent system, som indeholder en bestemt mængde af elementær talteori. Essensen af den anden sætning er, at ligemeget hvor stort vi gør vores system, ligemeget hvor meget vi fylder på, så vil det aldrig kunne bevise sin egen konsistens. Gödel formulerer selv dette i indledningen til sin artikel:

> *Even if we admit all the logical devices of Principia mathematica (hence in particular the extended functional calculus and the axiom of choice) in metamathematics, there does not exist a* consistency proof *for the system S (still less so if we restrict the means of proofs in any way). Hence a consistency proof for the system S can be carried out only by means of modes of inference that are not formalized in system S itself, ...* [van Heijenoort, 1967, s. 595-596]

(Gödels system S forklares i afsnit 5.2.3). Som ovenfor nævnt publicerede Gödel sine resultater i 1931, og hans artikel kom til at bære titlen *Über formal unentscheidbare Sätze der Principia mathematica und verwandter Systeme*[4]. Det

[3] von Neumann, som havde udvist stor interesse for Gödels resultat ved kongressen i Königsberg, viste uafhængigt samme resultat og formidlede dette per brev til Gödel den 20. november samme år. [Moore, 1970-80, s. 353]

[4] Vi har benyttet en engelsk oversættelse af artiklen fra *From Frege to Gödel* (side 595-616) af Jean van Heijenoort, hvorfor henvisningerne til Gödels arbejde vil se ud som følger: [van Heijenoort, 1967].

bør dog nævnes at Gödel ikke giver beviset for sin anden sætning i denne artikel, men kun skitserer det. Beviset for den anden sætning skulle følge i del II af artiklen, men Gödel fik aldrig skrevet denne. Hilbert og Bernays udførte dog i 1939 dette bevis i detaljer for to standardsystemer i talteori. [Harrison, 1996, s. 8], [O'Connor and Robertson, 2003], [van Heijenoort, 1967, s. 594]

Lad os lige opsummere resultaterne i Gödels ufuldstændighedssætninger. I sin slående enkelthed siger sætningerne, at der i ethvert aksiomatisk matematisk system, som er stærkt nok til at indeholde elementær talteori, vil findes udsagn hvorom gælder, at hverken de eller deres negation kan bevises i systemet. Derudover gælder at systemets konsistens ikke kan vises indenfor systemet.

5.2.3 Relevante definitioner og begreber

Vi vil ikke gå videre i dybden med Gödels beviser for ufuldstændighedssætningerne, men blot nævne nogle af de for vores gennemgang af matematikkens formalisering relevante definitioner og begreber, som Gödel benytter. Gödel lægger i sin artikel ud med at argumentere for, hvordan det at behandle den proposition, der siger om sig selv 'Jeg er ikke beviselig' i modsætning til propositionen 'Jeg er ikke sand', er på kanten af løgnerparadokset uden dog at falde i. Gödels argument støtter sig op af en diagonaliseringsprocedure i stil med Cantors (se afsnit 6.3.2) samt Richards paradoks, hvilket også senere blev fremhævet af især Jacques Herbrand[5] og Hermann Weyl. Gödel gør en del ud af at give en præcis beskrivelse af det system, hvori han arbejder. Variable skelnes ved deres type. Type 1 er de naturlige tal inklusiv 0, type 2 er klasser af naturlige tal, type 3 er klasser af klasser af naturlige tal og så videre. De logiske aksiomer er ækvivalente med dem fra *Principia mathematica* uden Russells ramificerede typeteori. De aritmetiske aksiomer er Peanos (også kendt som Peanos postulater), ifølge van Heijenoort formentlig en smule omskrevet. Vi angiver nedenfor de aksiomer, der anvendes i Gödels system. Først forklares lidt notation. Med $f(x_1)$ mener Gödel efterfølgeren til x_1, der er et naturligt tal. Med $\mathfrak{S}_b^v a$, hvor a er en formel, v er en variabel og b er et tegn af samme type som v, forstås den formel vi får fra a, når vi i a substituerer b for v. Prikken under S'et angiver at b er fri for v i a (dette forklares i kapitel 8).

I. 1. $\neg(f(x_1) = 0)$
   2. $(f(x_1) = f(y_1)) \to (x_1 = y_1)$
   3. $x_2(0) \wedge \forall x_1(x_2(x_1) \to x_2(f(x_1))) \to \forall x_1(x_2(x_1))$

II. 1. $(p \vee p) \to p$
   2. $p \to (p \vee p)$
   3. $(p \vee q) \to (q \vee p)$
   4. $(p \to q) \to ((r \vee p) \to (r \vee q))$

III. 1. $\forall v(a) \to \mathfrak{S}_c^v a$
   2. $\forall v(b \vee a) \to (b \vee \forall v(a))$

[5] Se for eksempel [van Heijenoort, 1967, s. 626-628].

IV. 1. $\exists u(\forall v(u(v) \equiv a))$

V. 1. $\forall x_1(x_2(x_1) \equiv y_2(x_1)) \rightarrow (x_2 = y_2)$

Aksiomerne I.1-3 stammer fra Peanos postulater, hvor aksiom I.3 er princippet for matematisk induktion. De resterende logiske aksiomer stammer fra *Principia mathematica*. Aksiom IV.1 er komprehensionsaksiomet, som her erstatter reducerbarhedsaksiomet (jævnfør afsnit 4.2). Aksiom V.1 er ekstentionalitetsaksiomet. Herefter introducerer Gödel den ovenfor beskrevne nummerering, der senere har fået navn efter ham selv. Selvom Gödel viste sine resultater for højereordens-logik pointerede han, at de også gjaldt i den aksiomatiske mængdelære. Tilføjelse af endeligt mange nye aksiomer ville ikke ændre på situationen, ej heller ville tilføjelsen af uendeligt mange nye aksiomer, så længe det resulterende system besidder en egenskab, som Gödel selv kalder ω-konsistens. En moderne definition af dette begreb bringes her:

Definition 5.2
En teori K siges at være ω-konsistent, hvis og kun hvis der for enhver formel $Q(x)$ i K med præcis én fri variabel v gælder, at hvis $\vdash_K \neg Q(\overline{n})$ for alle naturlige tal n, så er $\exists x(Q(x))$ ikke et teorem i K.

Med $\overline{n}$ menes det n'te numeral. Da vi har med en formel teori at gøre indføres denne betegnelse for at skelne disse fra de tilsvarende intuitive tal. Numeralerne $0, 0', (0')', ((0')')' \ldots$ angives ved $\overline{0}, \overline{1}, \overline{2}, \overline{3}, \ldots$. Det vises, at ethvert ω-konsistent system også er konsistent. Gödel viste altså, at enhver teori for mængdelæren, som er ω-konsistent er ufuldstændig. Beviset blev senere videreudviklet af J. Barkley Rosser, som ved hjælp af en mere kompliceret fremgangsmåde fik udskiftet kravet om ω-konsistens med kravet om konsistens i almindelig forstand. Resultatet af Gödels og Rossers arbejde kaldes i dag Gödel-Rossers sætning. Sætningen som essentielt siger, at for de betragtede systemer, herunder mængdelæren samt alle andre formelle udvidelser af talteori, gælder, at hvis systemet er konsistent er det ufuldstændigt. Gödel giver desuden en præcis definition af, hvad vi i dag forstår ved primitive rekursive funktioner. Rekursive funktioner havde allerede tidligere været benyttet i formaliseringsøjemed af for eksempel Julius Dedekind, Albert Skolem og Hilbert, men det var Gödel der gav den definition, som siden hen er blevet benyttet. [van Heijenoort, 1967, s. 592-601, 607], [Andrews, 2002, s. 81, 258-259], [Moore, 1970-80, s. 353], [Mendelson, 1997, s. 60, 205], [Margaris, 1967, s. 185-190]

5.2.4 Sætningernes betydning

Gödels sætninger satte adskillige årtiers forsøg på at etablere aksiomer, som hele matematikken kunne baseres på, i et nyt perspektiv, idet drømmen om at opstille et system, som indeholdt hele matematikken, blev ødelagt. Dette gik således også ud over de foregående omtalte arbejder af såvel Frege som Russell. Også Hilberts program blev sat i et nyt lys af disse sætninger (i højere grad af den anden sætning), selv om det tog Hilbert selv nogle år at indse dette. Sætningerne demonstrerede, at ethvert system nødvendigvis må være mere omfattende end først forudset af Hilbert. John von Neuman, som var en

af de første til at acceptere og indse relevansen af Gödels sætninger, forlæste snart over sætningerne i Princeton. Stephen Kleene fulgte disse forelæsninger og skulle efter sigende have været særdeles entusiastisk derved. Alonzo Church troede, fejlagtigt, at hans nye system, som involverede λ-kalkylen (se kapitel 7), kunne undsige sig ufuldstændighed. I Europa endte Paul Bernays, efter en længere korrespondance med Gödel, med at acceptere sætningerne. Zermelo derimod forblev skeptisk, såvel personligt som i korrespondance. O'Connor og Robertson skriver om Gödels resultater:

> *Gödel's results were a landmark in 20th-century mathematics, showing that mathematics is not a finished object, as had been believed. It also implies that a computer can never be programmed to answer all mathematical questions.* [O'Connor and Robertson, 2003]

Den sidste sætning i citatet hentyder igen til Gödels første sætning, idet denne jo siger, at der inden for matematikken findes sætninger hvorom gælder, at vi hverken kan vise dem selv eller deres negation. [Moore, 1970-80, s. 353], [O'Connor and Robertson, 2003]

Gödels sætninger, som præsenteret i dette kapitel, stammer fra [Harrison, 1996]. Gödel skriver i 1963 i en note til sin artikel følgende om beviserne for sine sætninger, i særdeleshed den anden sætning, som han jo kun skitserer beviset for.

> *In consequence of later advances, in particular of the fact that due to A. M. Turing's work a precise and unquestionably adequate definition of the generel notion of formal system can now be given, a completely generel version of Theorem VI og XI is now possible. That is, it can be proved rigorously that in every consistent formal system that contains a certain amount of finitary number theory there exist undecidable arithmetic propositions and that, moreover, the consistency of any such system cannot be proved in the system.* [van Heijenoort, 1967, s. 616]

'Sætning VI og XI' hentyder til Gödels første og anden ufuldstændighedssætning, som han nævner umiddelbart efter i citatet. Dette nu mulige bevis for sætningerne skyldes ifølge Gödel i særdeleshed Turings arbejde. Vi skal i næste kapitel (kapitel 6) se nærmere på dette og på det såkaldte *Entscheidungsproblem*, som Gödel ikke løser, men som hans sætninger på en vis måde implicerer løsningen af. [van Heijenoort, 1967, s. 616], [O'Connor and Robertson, 2003]

6 Turings definition af beregnelighed

Gödel nævner i det afsluttende citat i forrige kapitel (se side 45), at det er Turings arbejde, der gør det muligt at give en præcis og fyldestgørende definition af den generelle ide om et formelt system. Dette arbejde er netop Turings artikel, dateret 1936/1937. Det er også i dette arbejde, at Turing giver sin definition af beregnelighed gennem beskrivelsen af en beregningsmaskine, der i dag kaldes en Turingmaskine. Ved hjælp af denne definition og Turingmaskinen besvarer Turing Hilberts afgørlighedsspørgsmål. Men først Turings biografi.

6.1 Biografi

Alan Mathison Turing blev født i England i 1912. Han var søn af Julius Mathison og Ethel Sara Turing. Indtil 1919 boede Turings forældre i Indien, da faderen arbejdede for 'Indian Civil Service'. Turing voksede derfor op hos forskellige familier i London frem til dette tidspunkt. Som ung var Turing elev ved Sherborne

Figur 6.1 Alan Mathison Turing. Født 23. juni 1912 i London, England. Død 7. juni 1954 i Wilmslow, Cheshire, England. [O'Connor and Robertson, 2003]

School, men som mange andre store tænkere var han aldrig glad for at gå i skole – han ville meget hellere forfølge sine egne ideer. Med sin egen måde at gøre tingene på, klarede Turing sig dog godt i skolen, specielt i naturvidenskabelige fag og matematik. [van Rootselaar, 1970-80b, s. 497], [Hodges, 2003]

Året efter, i 1931, blev Turing indskrevet på Kings's College i Cambridge, hvor der var mere plads til hans selvstændige udfoldelse. Her studerede han blandt andet Whiteheads og Russells *Principia mathematica* og fulgte et kursus i topologi, hvor han for første gang hørte om Hilberts afgørlighedsproblem, det såkaldte *Entscheidungsproblem.* Turing besluttede sig allerede på dette tidspunkt for at forsøge at løse dette problem. I 1935 blev Turing udnævnt til 'King's College fellow' for sin afhandling *On Gaussian Error Function.* Turing arbejdede sammen med Church ved Princeton universitet i årene 1936-1938. I denne periode, nærmere bestemt i 1937, udkom hans største bidrag til den matematiske logik: *On Computable Numbers, With an Application to the* Entscheidungsproblem. Denne afhandling indeholdt et bevis for, at svaret til Hilberts spørgsmål om afgørlighed af matematikken var et nej. Det var i forbindelse med dette bevis, at Turing introducerede ideen om sin senere så berømte Turingmaskine. Turing havde faktisk skrevet afhandlingen året før, hvorfor artiklen normalt dateres til 1936. Tidligere samme år udkom en artikel af Church, som ligeledes viste, at der ikke findes nogen afgørlighedsmetode for aritmetikken, og at svaret til Hilberts afgørlighedsspørgsmål derfor er et nej. Turing udvidede efterfølgende sin artikel med et appendiks, hvori han viste at Churchs resultat var ækvivalent med hans eget. [Hodges, 2003], [Hodges, 1983, s. 25], [van Rootselaar, 1970-80b, s. 497]

I 1939 vendte Turing tilbage til King's College, men anden verdenskrig afbrød hans forskning, og han var i årene 1939-1948 ansat i kommunikationsafdelingen i det britiske udenrigsministerium, hvor han arbejdede med afkodningen af tyske meddelelser. Turing er i dag kendt for at have 'knækket' de koder, som tyskerne benyttede til deres berømte *Enigma*-maskiner. I disse år arbejdede Turing hårdt på at knække koder i samarbejde med et hold af unge begavede mennesker, her iblandt en pige ved navn Joan Clarke, som han friede til i 1942. Deres bryllup blev dog aldrig aktuelt. Turing aflyste det og erkendte i den forbindelse overfor sin forlovede, at han var homoseksuel. [Hodges, 2003], [van Rootselaar, 1970-80b, s. 498]

Turing studerede flittigt elektronik, med det mål for øje at bygge en elektronisk udgave af den universelle Turingmaskine. Dette arbejde resulterede mere eller mindre i, at han opfandt den digitale computer. Efter krigen, nærmere betegnet foråret 1945, blev han ansat ved *the National Physical Laboratory*, hvor han designede en automatisk computer. [Hodges, 2003], [O'Connor and Robertson, 2003]

Udover sit akademiske arbejde var sport en af Turings store lidenskaber, specielt deltog han tit i løbekonkurrencer (se figur 6.2). Han var kendt for at løbe til forskellige konferencer, og ankom gerne tidligere end passagerer med offentlige transportmidler. I 1948 blev han assisterende direktør for beregningslaboratoriet ved Manchesters universitet, hvor han ledte udviklingen af MADAM (Manchester automatic digital machine). [Hodges, 2003], [O'Connor and Robertson, 2003]

Turing blev i 1952 arresteret og dømt for at have haft et homoseksuelt forhold. I stedet for fængselsstraf modtog Turing frivilligt østrogenindsprøjtninger i et

Figur 6.2 Alan Turing var en dediceret og meget dygtig løber. I 1947 var han tæt på at blive udtaget til det britiske løbehold, der skulle deltage i de olympiske lege i 1948, men blev det ikke grundet en skade. Her ovenfor ses han ved et 3-mils løb i 1946. [Hodges, 2003]

år, disse skulle neutralisere hans afsporede seksualitet. Den 8. juni 1954 blev Turing fundet død, ved hans side lå et halvspist æble, som var forgiftet med cyanid – man mente, at der var tale om et selvmord. Turing betragtes i dag som én af grundlæggerne af datalogien og ophavsmand til sin samtids mest avancerede teknologi, selvom han aldrig opnåede anerkendelse for dette i sin levetid. [Hodges, 2003]

Nærværende kapitel omhandler primært en gennemgang af Turings 1936-arbejde *On Computable Numbers, With an Application to the* Entscheidungsproblem. I dette arbejde præsenterede han for første gang sin forestilling om Turing-maskinen.

6.2 Afgørlighedsproblemet

Afgørlighedsproblemet – det såkaldte *Entscheidungsproblem* – blev formuleret af Hilbert i 1928. Hilbert havde en anden tilgang til matematikken end Frege og Russell. Hilbert stillede dybtgående spørgsmål til matematiske systemer som for

eksempel Russells; spørgsmål af typen: *Kan man afgøre hvad der kan bevises, og hvad der ikke kan bevises, indenfor en sådan teori?* På den internationale kongres i 1928 stillede Hilbert tre hovedspørgsmål til matematikken – spørgsmål som lå til grund for hans program. Disse lød:

1. Er matematikken fuldstændig?
2. Er matematikken konsistent?
3. Kan matematikken afgøres?

Ved det sidstnævnte spørgsmål skal forstås, hvorvidt der findes en metode, som kan afgøre om enhver given påstand er sand eller falsk. Hilbert mente selv, at svaret til alle tre spørgsmål måtte være ja, men på daværende tidspunkt havde ingen hverken be- eller afkræftet nogen af påstandene. To år senere viste Gödel (jævnfør kapitel 5) at aritmetikken er ufuldstændig. Han viste ligeledes, at man indenfor aritmetikken ikke er i stand til at vise dens konsistens. Dette ødelagde for alvor Hilberts program som først fremlagt. Altså var der nu kun Hilberts tredje spørgsmål om afgørligheden af matematikken tilbage. Det var dette spørgsmål, Turing kastede sig over. [Hodges, 2003], [Hodges, 1983, s. 91], [Andrews, 2002, s. 302]

6.3 *On Computable Numbers*

I artiklen *On Computable Numbers, With an Application to the* Entscheidungsproblem[1] fra 1936 viste Turing ved hjælp af en beregningsmaskine, at svaret på Hilberts tredje spørgsmål er nej. Tidligere havde andre matematikere drømt om en maskine, der skulle kunne løse matematiske problemer, men ingen havde formået at konstruere en. Udgangspunktet for maskinen var en anden maskine som manipulerede symboler, nemlig skrivemaskinen. Fra den udsprang ideen til en maskine, som skulle kunne løse matematiske problemstillinger ved hjælp af en uhyre simpel mekanik. [Hodges, 1983, s. 96-97]

6.3.1 Turingmaskinen

Turing lægger i sin artikel ud med at give en definition af beregnelige tal.

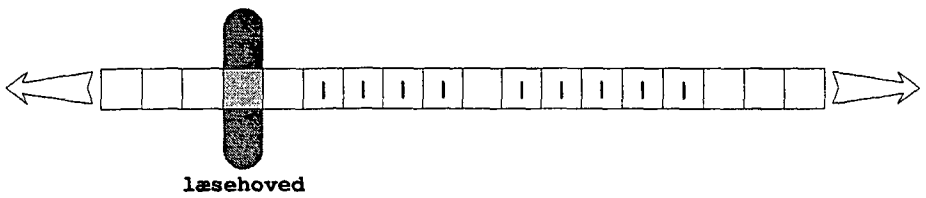
Definition 6.1
Et beregneligt tal er et reelt tal, hvis decimalfremstilling kan beregnes ved hjælp af en endelig metode.

Turing arbejder efter definitionen hen mod at definere sin beregningsmaskine. Til at begynde med definerer han en teoretisk maskine som kan være i endeligt mange forskellige tilstande

$$q_1, q_2, \ldots, q_n$$

[1] Vi har benyttet et genoptryk af artiklen fra *The Undecidable* (side 116-154) af Martin Davis, hvorfor henvisningerne til Turings arbejde vil se ud som følger: [Davis, 1965].

kaldet m-konfigurationer. Maskinen er udstyret med en uendelig lang strimmel, som går igennem maskinen. Strimlen er opdelt i felter, som kan indeholde ét symbol. Maskinen kan kun aflæse (scanne) ét felt ad gangen. Symbolet i det læste felt r kaldes $G(r)$. Figur 6.3 viser en Turingmaskine.



Figur 6.3 Turingmaskinen består af en strimmel og et læsehoved, der aflæser strimlens symboler. Pilene illustrerer, at strimlen fortsætter i begge retninger. [Hodges, 2003]

Maskinen kender kun det symbol, som netop er blevet læst, men ved hjælp af m-konfigurationerne kan den programmeres til at 'huske' tidligere symboler. Maskinens opførsel er bestemt udfra dens m-konfiguration q_n og det læste symbol $G(r)$. Maskinen virker helt automatisk udfra et givet regelsæt, som bestemmer følgende for enhver af maskinens konfigurationer:

1. Om den skal skrive et symbol i et tomt felt, lade det være uændret, eller slette feltets tegn og efterlade feltet tomt.
2. Om den skal blive i samme m-konfiguration eller skifte til en anden m-konfiguration.
3. Om den skal flytte sig ét felt til højre, ét felt til venstre, eller blive i samme felt.

Der kan opstilles en tabel, som indeholder alle disse oplysninger om maskinens opførsel. Tabellen definerer maskinen, forstået på den måde, at alle relevante oplysninger om maskinen vil være indeholdt i tabellen, uden at man behøver at bygge selve maskinen. En tabel *definerer* således en maskine. Der findes uendelig mange forskellige tabeller og derfor også uendelig mange forskellige maskiner. Disse maskiner kan have funktioner som at addere eller multiplicere tal, eller mere avancerede funktioner, som at afgøre om et tal er divisor i et andet tal og så videre. [Davis, 1965, s. 116-117], [Hodges, 1983, s. 98]

Da ethvert skridt i maskinen er automatisk kalder Turing maskinen for en a-maskine. Hvis en a-maskine skriver to slags symboler, hvor den første slags kun består af 0- og 1-taller, kaldes den for en beregningsmaskine. [Davis, 1965, s. 118]

En beregningsmaskine

Beregningsmaskinen kan eksemplificeres ved den simple maskine, der skriver talrækken

$$0101010101\ldots,$$

på en tom strimmel. Denne maskine er defineret ved tabel 6.1.

nuværende tilstand	læst symbol	skriv symbol	gå til	sluttilstand
q	tom	'0'	H	p
p	tom	'1'	H	q

Tabel 6.1 De to første søjler beskriver maskinens konfiguration. De tre sidste maskinens opførsel.

Når beregningsmaskinen skal finde ud af hvilken operation, den skal udføre, gennemsøger den sin tabel for en række, hvor *nuværende tilstand* og *læst symbol* stemmer overens med maskinens tilstand og det læste symbol $G(r)$. Hvis der findes en sådan række i tabellen udfører maskinen de operationer, som rækken foreskriver. Hvis der ikke findes en sådan række stopper maskinen. De to første pladser i rækkerne skal være unikke, så der ikke kan opstå tvivl om, hvilken operation maskinen skal udføre. Tabel 6.1 udtrykker, at hvis beregningmaskinen er i tilstanden q og læser symbolet 'tom', skal den skrive symbolet 0, gå til højre og skifte til tilstanden p. Hvis den er i tilstanden p og læser symbolet 'tom', skal den skrive symbolet 1, gå til højre og skifte til tilstanden q. Hvis maskinen startes med en tom strimmel i tilstanden q skriver den talrækken 0101010101... på strimlen. Hver række i ovenstående tabel kan opfattes som en 5-tupel. På den måde kan beregningmaskinerne og de øvrige Turingmaskiner programmeres ved hjælp af 5-tupler.

Den universelle maskine

Turing definerer også en anden maskine – en som kan udføre arbejdet for alle mulige Turingmaskiner. Turing kaldte denne maskine for *den universelle maskine*, og han fremstillede en præcis tabel for den. Vi vil ikke komme nærmere ind på denne maskine, da den er en hel diskussion for sig. Vi vil blot nævne, at den er i stand til at løse alle de problemer, som en almindelig Turingmaskine kan løse.

6.3.2 Cantors diagonalargument

Fra Turings konstruktion af en beregningsmaskine var der stadig langt til svaret på Hilberts tredje spørgsmål. Det skulle dog vise sig, at et diagonalargument udviklet af Cantor 50 år tidligere indeholdt nøglen til svaret.

Cantor havde benyttet en diagonaliseringsprocedure til at vise én-til-én-korrespondancen imellem de hele tal og de rationale tal. Denne procedure videreudviklede han til det såkaldte diagonalargument, som viste eksistensen af de irrationale tal. Cantor startede med at betragte alle tal imellem 0 og 1. Hvis man udelader tal, der forekommer flere gange, ser listen over samtlige rationale tal ud som følger:

$$\frac{1}{2}\ \frac{1}{3}\ \frac{2}{3}\ \frac{1}{4}\ \frac{3}{4}\ \frac{1}{5}\ \frac{2}{5}\ \frac{3}{5}\ \frac{4}{5}\ \frac{1}{6}\ \cdots$$

Betragter man nu alle endelige decimaltal som uendelige, hvor alle decimalerne efter et vist trin blot er 0'er, vil ovenstående liste i decimalfremstillingen se ud som følger:

$$
\begin{aligned}
&0,\underline{5}0000000000000000\ldots\\
&0,3\underline{3}333333333333333\ldots\\
&0,66\underline{6}66666666666666\ldots\\
&0,250\underline{0}0000000000000\ldots\\
&0,7500\underline{0}000000000000\ldots\\
&0,20000\underline{0}00000000000\ldots\\
&0,400000\underline{0}0000000000\ldots\\
&0,600000\underline{0}000000000\ldots\\
&0,800000000\underline{0}00000\ldots\\
&0,16666666\underline{6}6666666\ldots\\
&\vdots
\end{aligned}
$$

Ser man nu på det tal, der kommer ud af diagonalen (den er markeret ved understregning i ovenstående liste), nemlig tallet

$$0.5360000006\ldots$$

og derefter ændrer enhver decimal i dette tal, ved for eksempel at addere 1 til tallet på samtlige pladser og tage resultatet modulo ti[2]. Herved opnås tallet

$$0.6471111117\ldots$$

som ikke kan være rationalt, da det vil afvige fra det første rationale tal på første decimal, fra det andet rationale tal på anden decimal og så videre. Dermed findes det ikke på listen over rationale tal, altså må tallet være irrationalt. Cantor viste på denne måde ikke alene eksistensen af de irrationale tal, men tilmed, at der fandtes så mange af dem, at ingen liste kunne indeholde alle de reelle tal. Denne opdagelse var med til at afklare uendelighedsbegrebet. [Hodges, 1983, s. 101-102]

Turing diagonalargument

På samme måde kunne Turing argumentere for at de beregnelige tal gav anledning til uberegnelige tal. Turing skrev alle de tabeller, som definerede en Turingmaskine, der beregnede et tal, op på en liste. Denne liste gav anledning til en liste med de beregnelige tal. Fra listen påviste Turing ved hjælp af diagonalargumentet eksistensen af et uberegneligt tal. Denne observation gav Turing svaret på Hilberts tredje spørgsmål. Der findes ingen metode som kan løse alle matematiske problemer, da et uberegneligt tal er et eksempel på et uløseligt problem. Svarene på Hilberts tre spørgsmål viste sig altså alle at være nej, i modsætning til hvad Hilbert selv havde forudsagt. [Hodges, 1983, s. 102]

[2] Alternativt kan decimalfremstillingen af de rationale tal omskrives til binære tal, hvorefter man kun skal ændre 0 til 1 og 1 til 0.

En alternativ tilgang

En alternativ måde at vise ovenstående på kan være ved at benytte Cantors diagolanargument på følgende vis. Et reelt tal i intervallet mellem 0 og 1 kaldes beregneligt, hvis der eksisterer en Turingmaskine som, givet en tom strimmel, kan skrive dette tal (som et uendelig langt binært tal). Siden der kun er tællelig mange forskellige 5-tupler, er denne mængde af beregnelige tal ligeledes tællelig. Da der eksisterer overtælleligt mange reelle tal, må der være nogen af disse, som ikke er beregnelige. [Herken, 1988, s. 156]

6.4 Turingmaskinen som en effektiv procedure

Lad os her til sidst give et eksempel på, hvorfor Turings maskine er vigtig indenfor logikken og for formelle systemer i dag.

Turingmaskinen benyttes i forbindelse med aktiomatiske systemer. At et system er aksiomatisk vil sige, at der findes en effektiv procedure til at undersøge om en given formel er et aksiom. Selvom systemet er aksiomatisk, er der dog ingen garanti for, at det er afgørligt, det vil sige, om der findes en effektiv procedure til at bestemme, hvorvidt en vilkårlig given formel har et bevis. En teori, for hvilken der findes en sådan effektiv procedure, siges at være afgørlig, i modsat tilfælde siges teorien at være uafgørlig. De her omtalte 'effektive procedurer' er netop (terminerende) Turingmaskiner. [Mendelson, 1997, s. 34]

Lad os runde dette kapitel af hvor vi begyndte det, nemlig med Gödels 1963-note til sin artikel om ufuldstændighedssætningerne.

> ... *due to A. M. Turing's work a precise and unquestionably adequate definition of the generel notion of formal system can now be given,* ... [van Heijenoort, 1967, s. 616]

Gödel foretrak Turings definition af beregnlighed frem for Churchs definition, der gjorde brug af λ-kalkylen, da Gödel fandt Turingmaskinen mere intuitiv. De to definitioner er dog, som tidligere nævnt, ækvivalente. Vi skal i næste kapitel (kapitel 7) se nærmere på Churchs arbejde med λ-kalkylen og den simple typeteori.

7 Churchs typeteori

Alonzo Church er en af det tyvende århundredes store logikere. I 1930'erne opfandt han λ-kalkylen, som er et redskab til at repræsentere funktioner i formelle systemer. λ-kalkylen (den typede) er et vigtigt begreb i denne projektrapport, og Church har derfor fået sit eget lille kapitel her. Kapitlet indeholder korte beskrivelser af tre Church-artikler; *A Set of Postulates for the Foundation of Logic* fra 1932, som er den første artikel, hvori λ-symbolet anvendes til at repræsentere funktioner; *An Unsolvable Problem of Elementary Number Theory* fra 1936, som indeholder en række resultater vedrørende λ-kalkylen, der er interessante for det overordnede billede af matematikkens formalisering; *A Formulation of the Simple Theory of Types* fra 1940, der beskriver et formelt system, som er meget lig det system, der spiller en større rolle senere i denne rapport. Først gives dog en biografi for Church.

Figur 7.1 Alonzo Church. Født 14. juni 1903 i Washington, D.C. Død 11. august 1995 i Hudson, Ohio. [O'Connor and Robertson, 2003]

7.1 Biografi

Alonzo Church blev født i Washington D.C. Han var studerende ved universitetet i Princeton, hvor han opnåede sin første grad i 1924. I 1927 fik han sin doktorgrad (Ph.D.) under vejledning af Veblen. Churchs afhandling bar titlen *Alternatives to Zermelo's Assumption.* [O'Connor and Robertson, 2003]

Church tilbragte herefter et år ved universitet i Harvard og derefter et år i Europa. Først et halvt år i Göttingen og dernæst et halvt år i Amsterdam, hvor han arbejdede sammen med L. E. J. Brouwer. I 1929 vendte Church tilbage til USA for at blive ansat som professor ved sit gamle universitet i Princeton, en stilling han beholdt indtil 1967. I 1967 forlod Church Princeton til fordel for det varmere klima i Californien, hvor han fik tildelt en stilling som professor i matematik og filosofi ved UCLA. [O'Connor and Robertson, 2003], [Kfoury et al., 2003]

Figur 7.2 Alonzo Church i en lidt mere tænksom udgave. [Kfoury et al., 2003]

Church grundlagde *the Journal of Symbolic Logic* i 1936 og forblev redaktør for dette tidsskrift indtil 1979. I 1956 skrev han sin berømte lærebog *Introduction to Mathematical Logic*, som skulle være første bind i en serie af to. Andet bind udkom dog aldrig, på trods af at dets indholdsfortegnelse findes i bind 1. Church nåede i sin tid som professor at have 31 Ph.D.-studerende, her iblandt Turing, samt en række andre personer, som har tilknytning til denne rapport (Kleene, Rosser, Henkin, Davis og Andrews). Church døde i den høje alder af 92 år. Han blev begravet på kirkegården i Princeton. [O'Connor and Robertson, 2003], [Church, 1956, s. x], [Kfoury et al., 2003]

Kfoury, Xi og Pericas skriver følgende om Church:

> *His work has been greatly influential in the fields of mathematical logic, recursion theory, and theoretical computer science. At the time of his death, Church was widely regarded as the greatest living logician in the world.* [Kfoury et al., 2003]

7.2 λ-kalkyle og typeteori

Tilblivelsen af Churchs λ-kalkyle kan spores tilbage til de tre artikler, vi har valgt at inddrage i dette afsnit. λ-kalkylen er senere blevet videreudviklet og udgør i dag en omfattende teori. Kort fortalt er λ-kalkylen en teori for funktioner, i hvilken en funktion opfattes som en *regel*, der skaber en overgang fra argument til værdi gennem en formel. Særligt for λ-kalkylen er, at en funktion af flere variable opfattes som en slags sammensætning af funktioner af én variabel. En funktion f af to variable x og y virker først på den ene variabel x og derefter på den anden variabel y. Funktionsværdien af f virkende på x er da selv en funktion af y. Udtrykket fxy opfattes som $(fx)y$ hvor (fx) er en funktion af y. En funktion af flere variable er altså en funktion, der virker på sine variable en ad gangen. Ved hjælp af denne tilgang opnår man en teori, der ikke indeholder funktioner af flere variable. Funktionsbegrebet, der kommer til udtryk med λ-kalkylen, lægger mere vægt på det beregningsmæssige aspekt af funktionsbegrebet end den nutidige tilgang, hvor funktioner ofte opfattes som mængder af ordnede par. [Barendregt, 1981, s. 1]

Historien om hvorfor λ-kalkylen hedder, som den gør, er en sjov lille historie, hvorfor vi bringer den her. I *Principia mathematica* blev en funktion f med $f(x) = 2x + 1$ skrevet som $2\hat{x} + 1$. Church ønskede oprindeligt at anvende notationen $\hat{x}.2x+1$. Sættermaskinen kunne imidlertid ikke finde ud af at placere hatten over x'et, men satte den foran, så udtrykket blev til $\hat{}x.2x + 1$. En anden sættermaskine ændrede senere dette udtryk til $\lambda x.2x+1$, og sådan fik λ-kalkylen sit udseende og navn. [Barendregt, 1997, s. 182]

Churchs typeteori er en betegnelse for det system, han opstiller i sin artikel fra 1940. Dette system er en formulering af simpel typeteori, i hvilken λ-kalkylen optræder. Systemet er næsten identisk med det system, vi præsenterer i næste kapitel (kapitel 8) og som skal danne grundlaget for den senere diskussion af bevisføreren *Isabelle*.

7.2.1 *A Set of Postulates for the Foundation of Logic*

Church søgte i artiklen *A Set of Postulates for the Foundation of Logic* fra 1932 at opstille et system fri for de nu velkendte matematiske paradokser uden at gøre brug af Russells typeteori. Systemet, Church opstillede, blev senere vist inkonsistent af Kleene og Rosser, hvorfor Church opgav systemet. Han valgte dog at arbejde videre med et delsystem, som i dag kendes under navnet λ-kalkylen. I dette system benyttes λ-symbolet til at benævne funktioner således:

> *If M is any formula containing the variable x, then $\lambda x[M]$ is a symbol for the function whose values are those given by the formula.* [Church, 1932, s. 352]

λ-notation benyttes efterfølgende til at definere en række forkortelser, herunder de velkendte logiske symboler $\vee$, $\Rightarrow$, $\equiv$ og $\forall$[1]. Eksempelvis forkortes $\vee$ med

$$\lambda\mu\lambda\nu. \sim . \sim \mu \sim \nu,$$

som skal læses

$$\lambda\mu\lambda\nu[\neg[\neg\mu \wedge \neg\nu]].$$

Blandt slutningsreglerne for systemet er tre regler som Church kalder *konversioner*. Overordnet set beskriver disse, hvordan λ-symbolet opfører sig med henblik på substitutioner. Konversionerne er i dag en del af en række slutningsregler, der går under fællesbetegnelsen λ-konversion. Dette begreb vil vi vende tilbage til i kapitel 8. [Church, 1932, s. 355-356]

Efter påvisningen af inkonsistensen af hans systemet valgte Church at arbejde videre med den konsistente del, λ-kalkylen. Denne optræder derfor også i hans 1936-artikel.

7.2.2 *An Unsolvable Problem of Elementary Number Theory*

Artiklen fra 1936, *An Unsolvable Problem of Elementary Number Theory*[2], er en milepæl indenfor undersøgelsen af begrebet beregnelighed. Der kommer flere meget interessante resultater frem i artiklen. Church viser, at de funktioner, der kan defineres i λ-kalkylen, præcis er lig de rekursive funktioner, og at disse to begreber derfor må opfattes som et godt udgangspunkt for at definere beregnelighed. [Davis, 1965, s. 90]

Hovedresultatet i artiklen er dog Churchs påvisning af eksistensen af et 'uafgørligt problem' i elementær talteori. Resultat kendes i dag som Churchs sætning, der kan formuleres således:

Sætning 7.1 (Churchs sætning)
Der findes ingen afgørlighedsprocedure for den samlede prædikatlogik.

Resultatet er ækvivalent med Turings næsten samtidige resultat (1936-37, se kapitel 6), om at der findes et 'uberegneligt tal', det vil sige et tal, der ikke kan beregnes ved hjælp af en Turingmaskine. Turing viste efterfølgende, at λ-kalkylen og Turingmaskinen er ækvivalente i duelighed. Siden hen viste de sammen, at en række andre 'mekaniske processer for beregning' havde tilsvarende beregningsevner. Dette resulterede i Church-tesen, som også af og til kaldes Church-Turing-tesen. Tesen formoder, at en effektiv beregning er ækvivalent med en rekursiv funktion. [O'Connor and Robertson, 2003], [Kfoury et al., 2003]

[1] Church anvender en anden notation for disse symboler.

[2] Referencerne i dette afsnit vil have formen [Davis, 1965] fordi det er fra dette værk, vi har artiklen.

7.2.3 *A Formulation of the Simple Theory of Types*

Det system, som Church opstiller i sin 1940-artikel, *A Formulation of the Simple Theory of Types*, er centralt for vores projektrapport. Det skyldes, at vi senere ønsker at sammenligne et system, der optræder i bevisføreren *Isabelle* med et system, $\mathcal{Q}_0^\infty$, som er meget lig Churchs 1940-system. Systemet, der kaldes for simpel typeteori, er bygget op omkring en typet udgave af hans λ-kalkyle. Den simple typeteori kaldes også højereordens-logik, fordi den, i modsætning til Russells ramificerede typeteori, tillader kvantificering over propositioner af vilkårlig orden. Typesymbolerne i systemet defineres udfra de to grundtyper $\imath$ og o. Systemet vil ikke blive opstillet her. [Church, 1940, s. 56-57]

I Churchs system defineres lighed ved hjælp af negation og disjunktion samt ved hjælp af al-kvantoren. Altså er lighed ikke et primitiv begreb i dette system. Henkin viste i 1963 (se [Henkin, 1963]), at systemet i stedet kan opstilles med lighed som et primitiv begreb. Desuden påpegede Andrews en række forbedringer til Henkins arbejde (se [Andrews, 1963]), og hermed er vi nået frem til at give en fremstilling af den simple typeteori. Det system, der præsenteres i næste kapitel, er Andrews' system, som altså nedstammer fra Churchs system anno 1940.

8 Simpel typeteori

I dette kapitel gives en moderne fremstilling af den simple typeteori eksemplificeret ved præsentationen af et logisk system, der anvender typer. Kapitlet indledes med en præsentation af en overordnet strategi, der generelt benyttes, når man opstiller logiske systemer. Derefter vil vi opstille et konkret system, $\mathcal{Q}_0^\infty$, der skal danne grundlaget for diskussionen af bevisføreren *Isabelle* senere i projektrapporten.

8.1 Nogle fundamentale begreber

Motivationen for at opstille et logisk system er i denne rapport ønsket om at tilføre en videnskabelig disciplin (matematikken) et logisk grundlag. En helt grundlæggende ide er, at komplekse matematiske udsagn eller sætninger skal kunne reduceres til simplere udsagn, hvis gyldighed er enklere at fastslå. Alle sætninger skal kunne udledes ved hjælp af nogle få logiske *slutningsregler* på baggrund af en række *første principper*, der kaldes *aksiomer*. Denne tilgang til problemstillingen kaldes en *aksiomatisk tilgang*.

En anden grundlæggende ide er, at den disciplin, der skal aksiomatiseres, skal formuleres i et kunstigt sprog, kaldet et *objektsprog*, i hvilket der er defineret helt klare regler for sprogets *syntaks*, sådan at sprogets opbygning kan beskrives med få simple regler. Sprogets *semantik* skal ikke spille nogen rolle i første omgang, det vil sige, objektsproget skal være et *formelt* sprog. [Margaris, 1967, s. 14]

De ideer, der er præsenteret ovenfor kaldes samlet ideen om at *formalisere matematikken*. Et system, der er opstillet i overensstemmelse hermed, består af et *formelt sprog* samt en række *aksiomer* og *slutningsregler*. Aksiomerne kan opfattes som grundlæggende ideer, hvis konsekvenser man ønsker at undersøge ved hjælp af slutningsreglerne. En konsekvens, der er udledt fra systemets aksiomer, kaldes et *teorem*. Den samlede mængde af teoremer, der kan udledes på den måde, kaldes en *teori*. Aksiomerne er formuleret i objektsproget og er derfor udelukkende syntaktiske konstruktioner. Det samme gælder derfor for systemets teoremer, hvorfor teorien kaldes en *formel teori*. [Andrews, 2002, s. 21]

Den formelle teori tildeles efterfølgende mening ved at betragte den uformelle disciplin (matematikken) og lade denne bestemme den nye teoris betydning eller *semantik*. Semantikken fastlægges ved brug af *fortolkninger* og *modeller* – begreber, der vil blive forklaret senere.

Som tidligere nævnt består en del af formaliseringen af en disciplin som matematikken i at skabe et objektsprog, hvori den formelle teori kan udformes. Når man ønsker at opstille et kunstigt sprog, er man nødt til at anvende et andet

sprog til at forklare, hvordan man skal bære sig ad og til at diskutere objektsproget. Dette andet sprog kaldes et *metasprog*. De resultater, der formuleres i et metasprog (metasproget er i denne rapport hovedsageligt dansk), kaldes *metateoremer*. Vi skal senere formulere en række metateoremer, som spiller en stor rolle i den overordnede behandling af formelle teorier. [Mendelson, 1997, s. 36]

8.2 En formel teori skabes

Strategien for opstillingen af et logisk system og derigennem en formel teori vil blive gennemgået i større detalje i dette afsnit. Sideløbende med den generelle forklaring vil vi opstille et logisk system $\mathcal{Q}_0^\infty$, der anvender typeteori. Dette system er stærkt inspireret af det system, som Church opstillede i sin 1940-artikel. Systemet $\mathcal{Q}_0^\infty$ er et system, der opstilles i [Andrews, 2002], som eksemplificerer typeteorien.

8.2.1 Nogle generelle definitioner

Hvis man vil anvende et logisk system til at tilføre matematikken stringens, er det vigtigt at have præcise definitioner af elementerne i det logiske fundament. De fleste har en eller anden intuition om begrebet 'et sprog', men i denne sammenhæng er intuition ikke helt tilstrækkelig. Man vælger derfor helt præcist at definere, hvad der menes med et sprog.

Definition 8.1
Et formelt sprog $\mathcal{L}$ er defineret, når følgende betingelser er opfyldt:

1. *En mængde af* symboler *er valgt som symbolerne for $\mathcal{L}$.*
2. *Der findes en samling af regler kaldet* formationsregler *for sammensætning af symbolerne for $\mathcal{L}$.*

Blandt de symboler, der er tilladte i sproget, er en række symboler for variable og konstante størrelser. I et typeteoretisk sprog, $\mathcal{L}$, defineres en række typesymboler, som bruges til at skelne variable og konstanter med forskellige egenskaber fra hinanden.

Definition 8.2
En endelig følge af tilladte symboler, der er sammensat i overensstemmelse med formationsreglerne, kaldes en formel.

Begrebet *en formel* spiller en central rolle i den videre behandling af teorier.

Definition 8.3
Et logisk system S er defineret, når følgende betingelser er opfyldt:

1. *Der er defineret et formelt sprog $\mathcal{L}$.*

2. *Der findes en delmængde af mængden af formler. Formlerne i denne delmængde tjener som aksiomer for* $\mathcal{S}$.

3. *Der er fastlagt en endelig mængde af relationer* $R_1, R_2, \ldots, R_n$ *kaldet slutningsregler* (rules of inference). *For enhver slutningsregel* R_i *gælder, at der findes et positiv heltal* j *således, at det for enhver mængde af* j *formler og enhver formel* $\mathcal{B}$ *kan afgøres, om de* j *formler står i relationen* R_i *til* $\mathcal{B}$. *Hvis dette er tilfældet, siges* $\mathcal{B}$ *at være en direkte konsekvens af de* j *formler på baggrund af* R_i.[1]

Det logiske system definerer en struktur, der tillader at man udleder konsekvenser fra aksiomerne. I denne udledning indgår begrebet et *bevis*. [Mendelson, 1997, s. 34]

Definition 8.4
Et bevis *i* $\mathcal{S}$ *er en endelig følge* $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m$ *af formler, således at hvert* $\mathcal{B}_i$ *opfylder en af følgende betingelser:*

1. $\mathcal{B}_i$ *er et aksiom.*

2. $\mathcal{B}_i$ *er en direkte konsekvens af de foregående formler* $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_i - 1$ *samt slutningsreglerne.*

Denne definition af et bevis er i fin overensstemmelse med den mere intuitive forståelse af et bevis, som de fleste matematikere har. [Mendelson, 1997, s. 34]

Definition 8.5
Lad $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m$ *være et bevis i* $\mathcal{S}$. *Det sidste element* $\mathcal{B}_m$ *i følgen kaldes da et* teorem *i* $\mathcal{S}$.

Et teorem er altså *en formel* med et bevis i $\mathcal{S}$. At $\mathcal{B}$ er et teorem i $\mathcal{S}$ skrives $\vdash_{\mathcal{S}} \mathcal{B}$ eller blot $\vdash \mathcal{B}$, hvis dette ikke giver anledning til forvirring. [Mendelson, 1997, s. 34]

I næste afsnit vil vi opstille et konkret formelt system, $\mathcal{Q}_0$, som skal vise den proces, der gennemgås, når et formelt system opstilles. Vi vil senere udvide $\mathcal{Q}_0$ til et system kaldet $\mathcal{Q}_0^{\infty}$

8.3 Systemet $\mathcal{Q}_0$

Det første skridt, der skal tages, når $\mathcal{Q}_0$ skal opstilles, er at definere et formelt sprog $\mathcal{L}_{\mathcal{Q}_0}$.

[1] Som eksempel på en slutningsregel kan nævnes *modus ponens*.

8.3.1 Det formelle sprog $\mathcal{L}_{\mathcal{Q}_0}$

Det logiske system vil blive formuleret i et sprog, vi vil kalde $\mathcal{L}_{\mathcal{Q}_0}$. Sprogets symboler defineres her og forklares efterfølgende. Symbolerne $\alpha, \beta, \gamma, \ldots$ benyttes i hele afsnittet som syntaktiske typevariable.[2]

Typesymboler for $\mathcal{L}_{\mathcal{Q}_0}$ defineres ved:

1. $\imath$ er et typesymbol, der angiver typen af individer.
2. o er et typesymbol, der angiver typen af sandhedsværdier.
3. Hvis α og β er typesymboler, så er $(\alpha\beta)$ et typesymbol, der angiver typen af funktioner fra elementer af type α til elementer af typer β.[3]

De primitive symboler for $\mathcal{L}_{\mathcal{Q}_0}$ er

1. Symbolerne [ ] λ.
2. For enhver type α, en tællelig mængde af variable af typen α: $f_\alpha, g_\alpha, \ldots, y_\alpha, z_\alpha, f^1_\alpha, \ldots, z^1_\alpha, f^2_\alpha, \ldots$
3. Logiske konstanter for hver type α: $Q_{\alpha\alpha o} \quad \iota_{(\imath o)\imath}$.

Variable af typen $\imath$ kaldes individvariable og variable af typen o kaldes propositionsvariable. Typerne $\imath$ og o er de eneste *primitive* typer i systemet, det vil sige, at alle andre (sammensatte) typer, dannes udfra disse to typer. [Andrews, 2002, s. 210]

De logiske konstanter indføres med den hensigt, at $Q_{\alpha\alpha o}$ skal betegne lighed mellem elementer af typen α, og at $\iota_{(\imath o)\imath}$ skal betegne en udvælgelsesoperator for elementer af typen $\imath$, det vil sige for individer. [Andrews, 2002, s. 211]

I det følgende betegner $\mathbf{A}_\alpha$, $\mathbf{B}_\alpha$, $\mathbf{C}_\alpha$, ... syntaktiske variable, der spænder over formler af type α. $\mathbf{x}_\alpha$, $\mathbf{y}_\alpha$, $\mathbf{z}_\alpha$, ... er syntaktiske variable, der spænder over variable af type α. Formationsreglerne for $\mathcal{L}_{\mathcal{Q}_0}$ er følgende:

1. En variabel eller konstant af type α er en formel af type α.
2. $\mathbf{A}_{\beta\alpha}\mathbf{B}_\beta$ er en formel af type α.
3. $\lambda\mathbf{x}_\beta\mathbf{A}_\alpha$ er en formel af type $\beta\alpha$.

$\mathbf{A}_{\beta\alpha}\mathbf{B}_\beta$ kan opfattes som funktionen $\mathbf{A}_{\beta\alpha}$ evalueret i $\mathbf{B}_\beta$. Ligeledes kan $\lambda\mathbf{x}_\beta\mathbf{A}_\alpha$ opfattes som den funktion, der sender $\mathbf{x}_\beta$ ind i $\mathbf{A}_\alpha$ hvor $\mathbf{x}_\beta$ ofte forekommer i $\mathbf{A}_\alpha$.

[2] Det vil sige variable i metasproget, der dækker over typevariable i objektsproget.

[3] Parenteser udelades i det efterfølgende. Udtrykket x_{ooo} betyder herefter $x_{(o(oo))}$. Denne notation er omvendt i forhold til notationen i [Andrews, 2002].

For formler af typen o udelades typesymbolet i det efterfølgende og $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ ... vil blive benyttet som syntaktiske variable, der spænder over propositionsvariable. Her vil der blive indført en række forkortelser, som afspejler hvorledes vi senere skal fortolke symbolerne. Indtil videre er disse dog blot forkortelser.

$\mathbf{A}_\alpha = \mathbf{B}_\alpha$	står for	$Q_{\alpha\alpha o}\mathbf{A}_\alpha\mathbf{B}_\alpha$
$\mathbf{A} \equiv \mathbf{B}$	står for	$Q_{ooo}\mathbf{AB}$
T_o	står for	$Q_{ooo} = Q_{ooo}$
F_o	står for	$\lambda x_o T = \lambda x_o x_o$
$\Pi_{(\alpha o)o}$	står for	$Q_{(\alpha o)(\alpha o)o}[\lambda x_\alpha T_o]$
$\forall \mathbf{x}_\alpha \mathbf{A}$	står for	$\Pi_{(\alpha o)o}[\lambda \mathbf{x}_\alpha \mathbf{A}]$
$\wedge_{ooo}$	står for	$\lambda x_o \lambda y_o[\lambda g_{ooo}[g_{ooo}TT] = \lambda g_{ooo}[g_{ooo}x_o y_o]]$
$\mathbf{A} \wedge \mathbf{B}$	står for	$\wedge_{ooo}\mathbf{AB}$
$\Rightarrow_{ooo}$	står for	$\lambda x_o \lambda y_o[x_o = [x_o \wedge y_o]]$
$\mathbf{A} \Rightarrow \mathbf{B}$	står for	$\Rightarrow_{ooo}\mathbf{AB}$
$\neg_{oo}$	står for	$Q_{ooo}F$
$\vee_{ooo}$	står for	$\lambda x_o \lambda y_o[\neg[[\neg x_o] \wedge [\neg y_o]]]$
$\mathbf{A} \vee \mathbf{B}$	står for	$\vee_{ooo}\mathbf{AB}$
$\exists \mathbf{x}_\alpha \mathbf{A}$	står for	$\neg\forall \mathbf{x}_\alpha \neg\mathbf{A}$
$\mathbf{A}_\alpha \neq \mathbf{B}_\alpha$	står for	$\neg[\mathbf{A}_\alpha = \mathbf{B}_\alpha]$

Fra ovenstående skema er det klart at alle forkortelserne defineres udfra en grundlæggende ide om lighed, altså symbolet $Q_{\alpha\alpha o}$. En sådant grundlæggende begreb kaldes et *primitivt begreb*. I Churchs system fra 1940, var der valgt en række andre primitive begreber, igennem hvilke lighed var defineret. I dette system, er lighed imidlertid valgt, fordi det er et simpelt og intuitivt begreb. [Andrews, 2002, s. 208, 212]

Ovenstående forkortelser kræver en forklaring. Som tidligere nævnt repræsenterer symbolet $Q_{\alpha\alpha o}$ den binære relation, der beskriver lighed imellem elementer af typen α. For binære relationer benyttes ofte *infix-notation*, det vil sige symbolet, der betegner den binære relation skrives imellem de to størrelser (man skriver $[\mathbf{A}_\alpha = \mathbf{B}_\alpha]$ i stedet for $[[= \mathbf{A}_\alpha]\mathbf{B}_\alpha]$). Symbolet $\equiv$ betegner lighed imellem propositioner (udsagn). I forbindelse med symbolerne $=$ og $\equiv$ vil vi i denne rapport benytte infix-notation.

Det er ofte bekvemt at arbejde med et udtryk, der repræsenterer sandhed. Man definerer derfor 'det sande udsagn', og vælger i den forbindelse et simpelt udtryk af formen $[\mathbf{A} = \mathbf{A}]$ - nemlig udtrykket $[Q_{ooo} = Q_{ooo}]$. 'Det falske udsagn' defineres ved hjælp af funktionen $\lambda x_o x_o$, altså funktionen, der sender x_o ind i x_o. Falskhed er, at funktionen, der afbilder x_o i sig selv og funktionen, der afbilder x_o i T er ens. Dette kan også formuleres mere intuitivt som 'alle udsagn er sande'. Vi ønsker at kunne skelne imellem sande og falske udsagn, det vil sige, der må findes mindst ét falsk udsagn, og derfor må udsagnet 'alle udsagn er sande' være falsk. Det *formelle* begreb falskhed afspejler altså fint den opførsel, vi forbinder med et mere intuitivt falskhedsbegreb. [Andrews, 2002, s. 208]

Symbolet $\Pi_{(\alpha o)o}$ betegner den egenskab ved en mængde, at mængden indeholder alle elementer af typen α. Dette indses således: $\Pi_{(\alpha o)o}$ kan opfattes som en

funktion, der afbilder en anden funktion, $s_{\alpha o}$ i en sandhedsværdi. $\Pi_{(\alpha o)o}s_{\alpha o}$ betyder at $s_{\alpha o}$ opfylder $Q_{(\alpha o)(\alpha o)o}[\lambda x_\alpha T]s_{\alpha o}$, det vil sige, at $[\lambda x_\alpha T] = s_{\alpha o}$ er opfyldt. Funktionen $s_{\alpha o}$ fastlægger en mængde af elementer x_α, der opfylder $s_{\alpha o}x_\alpha = T$. Men ifølge definitionen af $\Pi_{(\alpha o)o}$ skal der gælde $[\lambda x_\alpha T] = s_{\alpha o}$, det vil sige, at $s_{\alpha o}$ afbilder alle elementer af type α i T eller at $s_{\alpha o}$ opfattet som mængde indeholder alle elementer af typen α. $\Pi_{(\alpha o)o}s_{\alpha o}$ udtrykker så, at $s_{\alpha o}$ opfattet som mængde indeholder alle elementer af typen α. Symbolet $\Pi_{(\alpha o)o}$ gør det nemmere at indføre et (for matematikere) mere velkendt begreb, al-kvantoren.

Al-kvantoren $\forall$ indføres nu. Udtrykket $\forall \mathbf{x}_\alpha \mathbf{A}$ skal stå for udsagnet $\Pi_{(\alpha o)o}[\lambda \mathbf{x}_\alpha \mathbf{A}]$ der jo betyder, at funktionen der afbilder $\mathbf{x}_\alpha$ i $\mathbf{A}$ – opfattet som mængde, indeholder alle elementer af typen α. Der gælder altså at alle elementer $\mathbf{x}_\alpha$ opfylder $\mathbf{A}$. Denne definition stemmer fint overens med den normale (ikke-formaliserede) fortolkning af symbolet $\forall$.

De velkendte symboler (fra propositionslogikken) $\wedge$, $\Rightarrow$, $\neg$ og $\vee$ indføres nu. Symbolet $\wedge_{ooo}$ kan opfattes som en funktion, der afbilder to propositionsvariable x_o, y_o i en sandhedsværdi. Kort fortalt betyder symbolet $\wedge_{ooo}$, at der for en vilkårlig sandhedsfunktion g_{ooo} gælder, at $g_{ooo}TT$ og $g_{ooo}x_oy_o$ er ens.

Udtrykket $\Rightarrow_{ooo}$ er en funktion fra to propositionsvariable x_o, y_o ind i en sandhedsværdi. Funktionen antager værdien T hvis udtrykkene x_o og $x_o \wedge y_o$ er ens, hvilket mere intuitivt svarer til, at y_o er givet, hvis x_o er. Hvis $x_o \equiv F$ gælder også $[x_o \wedge y_o] \equiv F$, det vil sige $[\Rightarrow_{ooo} x_o]y_o \equiv T$. Definitionen giver altså den ønskede opførsel.

Negationstegnet $\neg_{oo}$ er en funktion, der sammenligner en propositionsvariabel med det falske udsagn. Hvis variablen antager værdien falsk er sammenligningen sand, altså $\mathbf{A} \equiv F$ betyder $\neg\mathbf{A} \equiv T$ og omvendt hvis $\mathbf{A} \equiv T$ gælder $\neg\mathbf{A} \equiv F$.

Symbolet $\vee_{ooo}$ defineres på sædvanlig vis ved at kombinere $\wedge$ og $\neg$. Tilsvarende defineres eksistenskvantoren $\exists$ og symbolet for ulighed $\neq$ ved hjælp af de allerede introducerede symboler.

Den forklaring, der er givet for hvert symbol, må kun opfattes som en belejlig måde at tænke på ved behandling af symbolerne. Forkortelsernes formål er at give en intuition om symbolerne og deres indbyrdes forhold, men man skal hele tiden holde sig for øje, at symbolerne indtil videre kun er symboler. Matematikere, der ser udtrykket $\forall \mathbf{x}_\alpha \mathbf{A}$ er nok tilbøjelige til at fortolke udtrykket på sædvanlig vis, men strengt taget skal det opfattes som $Q_{(\alpha o)(\alpha o)o}[\lambda \mathbf{x}_\alpha[Q_{(ooo)(ooo)o}Q_{(ooo)}Q_{(ooo)}]][\lambda \mathbf{x}_\alpha \mathbf{A}]$, altså rent syntaktisk.

Det formelle sprog $\mathcal{L}_{\mathcal{Q}_0}$ synes meget tungt og klodset at anvende for (almindelige) mennesker. Eksemplet ovenfor viser, at simple udtryk bliver utroligt komplicerede, når der ikke anvendes forkortelser, og de kan derfor være svære at gennemskue ved første øjekast. Imidlertid er styrken ved sproget, at det kan håndteres 'mekanisk' uden brug af intuition for symbolerne. Derudover kan sprogets udtryk formuleres ved brug af meget få symboler.

8.3.2 Aksiomer og slutningsregler for $\mathcal{Q}_0$

Definitionen af det formelle sprog tillader formuleringen af systemets aksiomer.

Aksiomerne for $\mathcal{Q}_0$ er følgende:

1. $g_{oo}T_o \wedge g_{oo}F_o = \forall x_o[g_{oo}x_o]$
2. $x_\alpha = y_\alpha \Rightarrow h_{\alpha o}x_\alpha = h_{\alpha o}y_\alpha$
3. $f_{\beta\alpha} = g_{\beta\alpha} = \forall x_\beta[f_{\beta\alpha}x_\beta = g_{\beta\alpha}x_\beta]$
4. $[\lambda \mathbf{x}_\alpha \mathbf{B}_\beta]\mathbf{A}_\alpha = \mathbf{B}_\beta$, hvor $\mathbf{B}_\beta$ er en primitiv konstant eller variabel forskellig fra $\mathbf{x}_\alpha$.
5. $[\lambda \mathbf{x}_\alpha \mathbf{x}_\alpha]\mathbf{A}_\alpha = \mathbf{A}_\alpha$
6. $\lambda \mathbf{x}_\alpha[\mathbf{B}_{\gamma\beta}\mathbf{C}_\gamma]\mathbf{A}_\alpha = [[\lambda \mathbf{x}_\alpha \mathbf{B}_{\gamma\beta}]\mathbf{A}_\alpha][[\lambda \mathbf{x}_\alpha \mathbf{C}_\gamma]\mathbf{A}_\alpha]$
7. $\lambda \mathbf{x}_\alpha[\lambda \mathbf{y}_\gamma \mathbf{B}_\delta]\mathbf{A}_\alpha = \lambda \mathbf{y}_\gamma[[\lambda \mathbf{x}_\alpha \mathbf{B}_\delta]\mathbf{A}_\alpha]$, hvor $\mathbf{y}_\gamma$ er forskellig fra $\mathbf{x}_\alpha$ og fra alle variable i $\mathbf{A}_\alpha$.
8. $\lambda \mathbf{x}_\alpha[\lambda \mathbf{x}_\alpha \mathbf{B}_\delta]\mathbf{A}_\alpha = \lambda \mathbf{x}_\alpha \mathbf{B}_\delta$
9. $\iota_{(\iota o)\iota}[Q_{\iota\iota o}y_\iota] = y_\iota$

I ovenstående indgår der i linierne 2-8 syntaktiske variable og disse linier indeholder derfor ikke aksiomer, men aksiomsskemaer. Hvis man udskifter de syntaktiske variable i et aksiomsskema med objektvariable opnår man en instans af aksiomsskemaet (et aksiom). Som eksempel kan nævnes linie nummer 5. Hvis man udskifter $\mathbf{x}_\alpha$ med et af de symboler, der anvendes for variable af typen α, for eksempel x_α, og udskifter $\mathbf{A}_\alpha$ med en konkret formel af typen α, for eksempel y_α, opnår man et aksiom $[\lambda x_\alpha x_\alpha]y_\alpha = y_\alpha$. Selvom der egentlig er tale om aksiomsskemaer, vil vi kalde dem aksiomer. [Andrews, 2002, s. 213]

Aksiom 1 udtrykker, at der kun findes de to sandhedsværdier T og F som propositionsvariable kan antage. Aksiomet kaldes til tider *den ekskluderede midte.* Aksiom 2 udtrykker en grundlæggende egenskab ved lighed. Aksiom 3 udtrykker en egenskab vedrørende lighed mellem funktioner. Dette aksiom kaldes ekstensionalitetsaksiomet[4]. Aksiom 4-8 er aksiomer for λ-kalkyle, der fastlægger egenskaber for notationen. Aksiom 9 involverer den logiske konstant $\iota_{(\iota o)\iota}$, som ellers har fået lov til at dvæle indtil nu. Udtrykket $Q_{\iota\iota o}y_\iota$ er mængden af individer x_ι, der opfylder $Q_{\iota\iota o}y_\iota x_\iota$, det vil sige opfylder $y_\iota = x_\iota$. Mængden er altså en singletonmængde bestående af elementet y_ι. Symbolet $\iota_{(\iota o)\iota}$ kan da opfattes som den 'operator', der udvælger et element fra en singletonmængde. [Andrews, 2002, s. 214]

Slutningsregler for $\mathcal{Q}_0$

Det logiske system $\mathcal{Q}_0$ har kun én slutningsregel:

Regel R
Fra $\mathbf{C}$ og $\mathbf{A}_\alpha = \mathbf{B}_\alpha$ kan sluttes det resultat, der fremkommer ved at udskifte en forekomst af $\mathbf{A}_\alpha$ i $\mathbf{C}$ med en forekomst af $\mathbf{B}_\alpha$, forudsat at forekomsten af $\mathbf{A}_\alpha$ i $\mathbf{C}$ ikke er en variabel umiddelbart efter et λ.

[4] *Axiom of Extensionality.*

Definition 8.6
En forekomst af en variabel x_α er bunden *i en formel B_β hvis den er i en delformel af B_β med formen $\lambda x_\alpha C_\delta$.*

En bunden forekomst af en variabel er altså en forekomst af variablen, der er inden for *virkefeltet*[5] af et λ.

Definition 8.7
En forekomst af en variabel $\mathbf{x}_\alpha$ er fri, *hvis den ikke er bunden.*

En variabel $\mathbf{x}_\alpha$ siges at være fri (bunden) i en formel $\mathbf{B}_\beta$, hvis $\mathbf{x}_\alpha$ har en fri (bunden) forekomst i $\mathbf{B}_\beta$. Således kan en variabel både være bunden og fri i en og samme formel.

Definition 8.8
Hvis $x^1_{\alpha_1}, \ldots, x^n_{\alpha_n}$ er forskellige variable betegner $S^{x^1_{\alpha_1},\ldots,x^n_{\alpha_n}}_{A^1_{\alpha_1}\ldots A^n_{\alpha_n}} B_\beta$ resultatet af den simultane udskiftning af $A^i_{\alpha_i}$ for alle frie forekomster af $x^i_{\alpha_i}$ i B_β for $1 \leq i \leq n$.

Udskiftning af variable størrelser kaldes også substitutioner.

Definition 8.9
En variabel A_α er fri for *x_α i B_β hvis ingen frie forekomster af x_α i B_β er delformler af B_β af formen $\lambda y_\gamma C_\delta$ i hvilken y_γ er fri i A_α.*

Formålet med ovenstående definition er senere at kunne sikre, at der ikke ved substitutioner er frie variable der bliver bundne eller omvendt. [Andrews, 2002, s. 81-82]

8.3.3 Logik i $\mathcal{Q}_0$

Det logiske system $\mathcal{Q}_0$ er nu opstillet. Systemet består af det formelle sprog $\mathcal{L}_{\mathcal{Q}_0}$, en række aksiomer samt aksiomsskemaer, der potentielt indeholder uendeligt mange aksiomer samt slutningsreglen **Regel R**. I dette afsnit vil vi angive nogle få resultater, der kan udledes i $\mathcal{Q}_0$. Disse resultater figurerer i en senere diskussion af $\mathcal{Q}_0$. Først vil vi vise et meget simpelt resultat, der gør brug af $\mathcal{Q}_0$'s aksiomer og slutningsregel.

Sætning 8.1
$\vdash \mathbf{A}_\alpha = \mathbf{A}_\alpha$

Bevis

(1) $\vdash [\lambda \mathbf{x}_\alpha \mathbf{x}_\alpha]\mathbf{A}_\alpha = \mathbf{A}_\alpha$	aksiom 5
(2) $\vdash \mathbf{A}_\alpha = \mathbf{A}_\alpha$	**Regel R**

□

[5] Virkefelt er en oversættelse af ordet *scope*.

I ovenstående er $\mathbf{A}$ byttet ud med $[\lambda \mathbf{x}_\alpha \mathbf{x}_\alpha]\mathbf{A}$ i overenstemmelse med **Regel R**. [Andrews, 2002, s. 215]

Alle beviser i $\mathcal{Q}_0$ kan gennemføres som vist ovenfor. Mere komplicerede beviser har imidlertid en tendens til at blive lange og besværlige, når man kun benytter **Regel R**, og man viser derfor, at det fra slutningsreglen er muligt at udlede en række mere effektive slutningsregler kaldet *afledte slutningsregler*. Tilgangen hvor man vælger én simpel slutningsregel, der giver anledning til en række afledte slutningsregler giver et overskueligt system. Her følger et par eksempler på afledte slutningsregler, som vi senere skal se ligner regler, der anvendes i *Isabelle*.

For en vilkårlig formel $\mathbf{C}$ er følgende lovligt:

1. α-**konversion**. Udbytning af delformlen $[\lambda \mathbf{x}_\beta \mathbf{A}_\alpha]$ med $[\lambda \mathbf{z}_\beta \mathrm{S}^{\mathbf{x}_\beta}_{\mathbf{z}_\beta} \mathbf{A}_\alpha]$, så længe $\mathbf{z}_\beta$ ikke forekommer frit i $\mathbf{A}$ og $\mathbf{z}_\beta$ er fri for $\mathbf{x}_\beta$ i $\mathbf{A}$.
2. β-**reduktion**. Udbytning af delformlen $[\lambda x_\alpha \mathbf{B}_\beta]\mathbf{A}_\alpha$ med $\mathrm{S}^{x_\alpha}_{\mathbf{A}_\alpha} \mathbf{B}_\beta$, så længe $\mathbf{A}_\alpha$ er fri for x_α i $\mathbf{B}_\beta$.
3. β-**ekspansion**. Reglen er den inverse til β-**reduktion**.

De to sidste regler kaldes samlet for β-**konversion**. Alle reglerne hører under begrebet λ-**konversion**.[6] *Modus ponens* er et andet eksempel på en regel, der kan udledes fra **Regel R**. [Andrews, 2002, 219, 224]

Vi vil i næste afsnit kort beskrive systemet $\mathcal{Q}_0^\infty$, der er en udvidelse af $\mathcal{Q}_0$, og som gør det muligt at formalisere dele af matematikken.

8.4 Systemet $\mathcal{Q}_0^\infty$

Et af de mest fundamentale begreber indenfor matematikken er de naturlig tal, $\mathbf{N}$. Hvis man skal gøre sig noget håb om at opstille et system, der kan formalisere matematikken, må et af de første krav til systemet være, at de naturlige tal kan beskrives heri. Hvis man ønsker at anvende $\mathcal{Q}_0$ som udgangspunkt, er man nødt til at supplere de eksisterende aksiomer med et aksiom, der sikrer eksistensen af en mængde med uendeligt mange elementer. Formuleret i $\mathcal{L}_{\mathcal{Q}_0}$ kunne et *uendelighedsaksiom* se således ud:

$$\exists r_{\iota\iota o} \forall x_\iota \forall y_\iota \forall z_\iota [\exists w_\iota r_{\iota\iota o} x_\iota w_\iota \wedge \neg r_{\iota\iota o} x_\iota x_\iota \wedge [r_{\iota\iota o} x_\iota y_\iota \Rightarrow [r_{\iota\iota o} y_\iota z_\iota \Rightarrow r_{\iota\iota o} x_\iota z_\iota]]]$$

Systemet, der opstår når ovenstående uendelighedsaksiom føjes til aksiomerne i $\mathcal{Q}_0$ kaldes $\mathcal{Q}_0^\infty$. I dette system kan de naturlige tal defineres, som ækvivalensklasser af mængder af individer. Dette gøres ved hjælp af følgende forkortelser:

Definition 8.10
Typesymbolet σ er en forkortelse for $(\iota o)o$.

0_σ	*står for*	$Q_{(\iota o)(\iota o)o} \lambda x_\iota F_o$
$S_{\sigma\sigma}$	*står for*	$\lambda n_\sigma \lambda p_{\iota o} \exists x_\iota [p_{\iota o} x_\iota \wedge n_\sigma [\lambda t_\iota [t_\iota \neq x_\iota \wedge p_{\iota o} t_\iota]]]$
$\mathbf{N}_{\sigma o}$	*står for*	$\lambda n_\sigma \forall p_{\sigma o} [p_{\sigma o} 0_\sigma \wedge \forall x_\sigma [p_{\sigma o} x_\sigma \Rightarrow p_{\sigma o} [S_{\sigma\sigma} x_\sigma]]] \Rightarrow p_{\sigma o} n_\sigma$

[6] Disse regler er omtalt i forbindelse med Churchs 1936-artikel (se kapitel 7).

Symbolet 0_σ angiver samlingen af elementer med nul elementer, det vil sige samlingen, som kun indeholder den tomme mængde $\lambda x_\iota F_o$. $S_{\sigma\sigma}$ repræsenterer *efterfølgerfunktionen* og $\mathbf{N}_{\sigma o}$ repræsenterer de naturlige tal. [Andrews, 2002, s. 258-260]

Systemet $\mathcal{Q}_0^\infty$ er et system, i hvilket store dele af matematikken kan formaliseres. Som et meget simpelt eksempel kan nævnes Peanos postulater (jævnfør kapitel 5). Postulaterne er følgende:

1. Der findes en entitet kaldet 0, som er et naturligt tal.
2. Ethvert naturligt tal n har en efterfølger Sn, som også er et naturligt tal.
3. 0 er ikke efterfølgeren for noget naturligt tal.
4. Hvis n og m er naturlige tal med samme efterfølger, så er n og m det samme tal.
5. Princippet om *matematisk induktion.*

I $\mathcal{Q}_0^\infty$ ser Peanos postulater således ud:

1. $\vdash \mathbf{N}_{\sigma o} 0_\sigma$
2. $\vdash \forall x_\sigma [\mathbf{N}_{\sigma o} x_\sigma \Rightarrow \mathbf{N}_{\sigma o} S_{\sigma\sigma} x_\sigma]$
3. $\vdash \forall n_\sigma [S_{\sigma\sigma} n_\sigma \neq 0_\sigma]$
4. $\vdash \forall n_\sigma \forall m_\sigma [S_{\sigma\sigma} n_\sigma = S_{\sigma\sigma} m_\sigma \Rightarrow n_\sigma = m_\sigma]$
5. $\vdash \forall p_{\sigma o} [[p_{\sigma o} 0_\sigma \wedge \forall x_\sigma [p_{\sigma o} x_\sigma \Rightarrow p_{\sigma o} [S_{\sigma\sigma} x_\sigma]]] \Rightarrow \forall x_\sigma p_{\sigma o} x_\sigma]$

I $\mathcal{Q}_0^\infty$ er ovenstående ikke postulater, men teoremer, det vil sige beviselige udtryk. Dette skyldes, at uendelighedsaksiomet er føjet til systemet. Eksemplet skulle gerne illustrere hvor langt man kan komme med et enkelt velvalgt aksiom. [Andrews, 2002, s. 258-266]

Systemet $\mathcal{Q}_0^\infty$ er interessant både fordi det benyttes senere af bevisføreren *Isabelle*, men også fordi man kan vise nogle resultater omkring systemet, som i stort omfang kan generaliseres til at gælde for alle formelle systemer, i hvilke matematikken kan formaliseres.

I det følgende afsnit vil vi beskæftige os med den semantiske side af systemerne $\mathcal{Q}_0$ og $\mathcal{Q}_0^\infty$. Ideen om at opstille et formelt system ville være nytteløs, hvis denne side af teorien blev overladt til tilfældigheder. Derfor skal denne side af sagen behandles lige så omhyggeligt som opstillingen af de logiske systemer.

8.5 Semantik for $\mathcal{Q}_0$ og $\mathcal{Q}_0^\infty$

I dette afsnit vil vi diskutere semantikken for de formelle teorier, det vil sige teoriernes betydning. Det man håber på, når man har opstillet en formel teori er, at man er i stand til at tildele symbolerne og dermed formlerne i teorien

en betydning på en sådan måde, at den formelle teori afspejler den uformelle teori. Vi håber altså på, at vi kan genkende den uformelle matematik i den formaliserede matematik.

I opstillingen af det logiske system blev begrebet *en formel* gjort til et helt centralt begreb i forbindelse med behandlingen af objektsprogets syntaks. Den samme centrering omkring formler vil ske i behandlingen af objektsprogets semantik. På den måde kommer begrebet en formel til at skabe koblingen mellem sprogets form og betydning. Undersøgelsen af koblingen resulterer i definitioner af en række overordnede egenskaber ved logiske systemer, der er helt grundlæggende for en mere generel undersøgelse og vurdering af disse.

Givet et logisk system $\mathcal{S}$ med tilhørende sprog $\mathcal{L}_\mathcal{S}$, er den overordnede strategi for fastlæggelsen af semantikken for $\mathcal{L}_\mathcal{S}$ at lade hvert symbol for $\mathcal{L}_\mathcal{S}$ tilsvare en entitet kendt fra den uformelle teori, der ønskes formaliseret. Parringen af symboler og entiteter sker ved hjælp af afbildninger, der er præcist definerede. På den måde tildeles alle formler i $\mathcal{L}_\mathcal{S}$ en betydning. Denne tildeling af betydning involverer begrebet en *fortolkning*, som vil blive defineret nedenfor. Når en teori betragtes gennem en fortolkning repræsenterer teoriens teoremer altså formelle udsagn, der kan sammenlignes med udsagn i den uformelle teori.

Det er klart at alle formlernes og dermed teoremernes betydning er bestemt igennem de nævnte afbildninger i en fortolkning, og man kan generelt ikke være sikker på, at fortolkninger udviser den ønskede opførsel. Man vælger derfor at klassificere særlige fortolkninger som *modeller*. Begrebet en *model* defineres nedenfor.

For at undgå, at den efterfølgende behandling af fortolkninger og modeller bliver for generel, vil vi lade den tage udgangspunkt i de opstillede systemer $\mathcal{Q}_0$ og $\mathcal{Q}_0^\infty$.

8.5.1 Fortolkninger og modeller

Den diskussion af fortolkninger og modeller, som følger her, er inspireret af diskussionen i [Andrews, 2002]. Den fremstilling af modelbegrebet, vi har valgt at bringe, afviger derfor fra andre fremstillinger, eksempelvis fremstillingen i [Mendelson, 1997].

Diskussionen kræver en række indledende definitioner. I den indledende diskussion er det ikke så afgørende om det system, der betragtes er $\mathcal{Q}_0$ eller $\mathcal{Q}_0^\infty$, og vi vil derfor tage udgangspunkt i $\mathcal{Q}_0$.

Definition 8.11
For hvert typesymbol α i $\mathcal{L}_{\mathcal{Q}_0}$ defineres et domæne *for α som en ikke-tom mængde $\mathcal{D}_\alpha$ af elementer.*

For typesymbolet o er det tilhørende domæne, $\mathcal{D}_o$, defineret ved $\mathcal{D}_o = \{T, F\}$. Elementerne i $\mathcal{D}_o$ kaldes *sandhedsværdier*. Elementerne i $\mathcal{D}_\iota$ kaldes *individer*. Alle andre typesymboler α er som tidligere nævnt sammensætninger af symbolerne o og ι, når vi betragter $\mathcal{Q}_0$.

De individer og sandhedsværdier, der nævnes ovenfor, skal skelnes fra de individer og sandhedsværdier, som blev omtalt, da vi definerede typesymbolerne o

og $\imath$. Grunden til sammenfaldet imellem navnene vil blive gjort klart nedenfor, når vi definerer tildelinger.

Symbolet $\mathcal{D}_{\alpha\beta}$ betegner en samling af funktioner, der afbilder $\mathcal{D}_\alpha$ ind i $\mathcal{D}_\beta$.

Definition 8.12
En ramme *er en samling $\{\mathcal{D}_\alpha\}_\alpha$ af domæner $\mathcal{D}_\alpha$, et domæne for hvert typesymbol i $\mathcal{L}_{\mathcal{Q}_0}$.*

Ideen er nu, at der skal etableres en sammenhæng mellem symbolerne i den formelle teori og elementerne i rammens domæner. På den måde fastlægger rammen de mulige betydninger som teorien kan tildeles. [Andrews, 2002, s. 238]

Definition 8.13
En fortolkning, *$\mathcal{M} = (\{\mathcal{D}_\alpha\}_\alpha, \mathcal{J})$ for $\mathcal{Q}_0$ er et ordnet par, hvor $\{\mathcal{D}_\alpha\}_\alpha$ er en ramme og $\mathcal{J}$ er en funktion, der afbilder hver konstant af typen α i et element i $\mathcal{D}_\alpha$.*

Hvis a er en konstant kaldes dets billede $\mathcal{J}a$ for *betydningen* af a. Fra ovenstående definition er det klart, at betydningen af konstanterne i $\mathcal{Q}_0$ afhænger af den fortolkning, der betragtes.

Definition 8.14
En tildeling *ind i en ramme $\{\mathcal{D}_\alpha\}_\alpha$ er en funktion φ fra mængden af variable for $\mathcal{Q}_0$ ind i rammens domæner, sådan at der for hvert $\mathbf{x}_\alpha$ gælder, at $\varphi\mathbf{x}_\alpha \in \mathcal{D}_\alpha$.*

En tildeling ind i en ramme tager altså hver variabel i det formelle system, og tildeler denne variabel en værdi fra det tilhørende domæne i rammen. Variable af typen $\imath$ tildeles altså værdier fra $\mathcal{D}_\imath$, det vil sige $\varphi\mathbf{x}_\imath \in \mathcal{D}_\imath$. Enhver variabel af typen $\imath$ tildeles altså et individ som værdi. Dette retfærdiggør den tvetydighed, der er forbundet med ordet individ. Tilsvarende gælder for sandhedsværdierne.

Givet en tildeling φ, en variabel $\mathbf{x}_\alpha$ og et $z \in \mathcal{D}_\alpha$, defineres da en anden tildeling $(\varphi : \mathbf{x}_{\alpha/z})$ som den tildeling ψ for hvilken $\psi\mathbf{x}_\alpha = z$ og $\psi\mathbf{y}_\beta = \varphi\mathbf{y}_\beta$ for $\mathbf{y}_\beta \neq \mathbf{x}_\alpha$.

Definition 8.15
En fortolkning, $\mathcal{M} = (\{\mathcal{D}_\alpha\}_\alpha, \mathcal{J})$, kaldes en generel model *for $\mathcal{Q}_0$ hvis der findes en binær funktion $\mathcal{V}^{\mathcal{M}}$, således at der for enhver tildeling φ og formel $\boldsymbol{A}_\alpha$ gælder, at $\mathcal{V}^{\mathcal{M}}_\varphi \boldsymbol{A}_\alpha \in \mathcal{D}_\alpha$ og at følgende betingelser er opfyldt:*

1. $\mathcal{V}^{\mathcal{M}}_\varphi \mathbf{x}_\alpha = \varphi\mathbf{x}_\alpha$.
2. $\mathcal{V}^{\mathcal{M}}_\varphi A_\alpha = \mathcal{J}A_\alpha$, *hvis A_α er en primitiv konstant.*
3. $\mathcal{V}^{\mathcal{M}}_\varphi[\boldsymbol{A}_{\alpha\beta}\boldsymbol{B}_\alpha] = (\mathcal{V}^{\mathcal{M}}_\varphi \boldsymbol{A}_{\alpha\beta})(\mathcal{V}^{\mathcal{M}}_\varphi \boldsymbol{B}_\alpha)$.
4. $\mathcal{V}^{\mathcal{M}}_\varphi[\lambda\mathbf{x}_\alpha\boldsymbol{B}_\beta]$ = *den funktion $f \in \mathcal{D}_{\alpha\beta}$, for hvilken værdien af fz for hvert $z \in \mathcal{D}_\alpha$ er $\mathcal{V}^{\mathcal{M}}_{(\varphi:\mathbf{x}_\alpha/z)}\boldsymbol{B}_\beta$.*

$\mathcal{V}_\varphi^{\mathcal{M}}\mathbf{A}_\alpha$ kaldes *værdien* af $\mathbf{A}_\alpha$ i $\mathcal{M}$ med hensyn til φ. Hvis $\mathbf{A}_\alpha$ er en konstant er $\mathcal{V}_\varphi^{\mathcal{M}}\mathbf{A}_\alpha = \mathcal{J}\mathbf{A}_\alpha$, altså uafhængig af φ. For en konstant gælder derfor, at *værdien* af konstanten og *betydningen* af konstanten er den samme. [Andrews, 2002, s. 238-239]

Betingelse 3 er lidt løst formuleret et krav om, at der skal være overensstemmelse mellem funktionsværdierne i det formelle system og de tilhørende funktionsværdier i rammen. Om en fortolkning er en generel model eller ej afhænger af fortolkningens ramme. De tre første betingelser er ikke så svære at opfylde, da rammens domæner er ikke-tomme. Den sidste betingelse kræver imidlertid eksistensen af en vis mængde af funktioner. De helt præcise forhold, der gør sig gældende i denne problemstilling, vil vi ikke komme nærmere ind på her. Vi vil i stedet definere et andet modelbegreb som stiller strengere krav.

Definition 8.16
En fortolkning $\mathcal{M} = (\{\mathcal{D}_\alpha\}_\alpha, \mathcal{J})$ kaldes en standardmodel *for $\mathcal{Q}_0$, hvis der for alle α, β gælder, at $\mathcal{D}_{\alpha\beta}$ er mængden af alle funktioner fra $\mathcal{D}_\alpha$ ind i $\mathcal{D}_\beta$.*

I en standardmodel kræves altså at alle funktioner fra $\mathcal{D}_\alpha$ ind i $\mathcal{D}_\beta$ eksisterer i rammen, og der gælder derfor at enhver standardmodel for $\mathcal{Q}_0$ også er en generel model for $\mathcal{Q}_0$. [Andrews, 2002, s. 239]

I en standardmodel er det klart, at domænet $\mathcal{D}_\iota$ entydigt bestemmer rammen, fordi $\mathcal{D}_o = \{T, F\}$ allerede er fastlagt, og alle funktionsdomænerne er fastlagt ved kravet fra definitionen af en standardmodel. [Andrews, 1972, s. 390]

Hvis en fortolkning er en generel model, er $\mathcal{V}^{\mathcal{M}}$ entydigt bestemt, det vil sige, at i en generel model kan systemets konstanter kun tildeles én betydning. Dette gælder således også for enhver standardmodel. [Andrews, 2002, s. 238]

Denne fremstilling af modelbegrebet afviger som tidligere nævnt fra andre fremstillinger hvor en fortolkning $\mathcal{M}$ kaldes en model for en mængde af formler Γ hvis enhver formel i Γ er sand for $\mathcal{M}$. [Mendelson, 1997, s. 60]

Inden vi fortsætter behandlingen af modeller, er det hensigtsmæssigt at kigge lidt nærmere på nogle af symbolerne i $\mathcal{Q}_0$.

Identitetsrelationen for $\mathcal{Q}_0$

Lad $\{\mathcal{D}_\alpha\}_\alpha$ være en ramme, og lad $x \in \mathcal{D}_\alpha$. Singletonmængden $\{x\}$ bestående af elementet x er den funktion $g \in \mathcal{D}_{\alpha o}$, som opfylder, at $gx = T$ og $gy = F$ for alle elementer y, der er forskellige fra x.

Definition 8.17
Identitetsrelationen *på $\mathcal{D}_\alpha$ er en funktion h, som for hvert $x \in \mathcal{D}_\alpha$ opfylder, at $hx = \{x\}$. Det vil altså sige, at for x, $y \in \mathcal{D}_\alpha$ gælder, at $(hx)y = T$ hvis $x = y$.*

I en fortolkning $(\{\mathcal{D}_\alpha\}_\alpha, \mathcal{J})$ for $\mathcal{Q}_0$ afbilder funktionen $\mathcal{J}$ konstanten $Q_{\alpha\alpha o}$ i identitetsrelationen på $\mathcal{D}_\alpha$, og $\mathcal{J}$ afbilder $\iota_{(\iota o)\iota}$ i en funktion fra $\mathcal{D}_{\iota o}$ ind i $\mathcal{D}_\iota$. Sidstnævnte funktion $\mathcal{J}\iota_{(\iota o)\iota}$ afbilder singletonmængder i $\mathcal{D}_{\iota o}$ i deres unikke element i $\mathcal{D}_\iota$.

Ovenstående kan også formuleres således. *Betydningen* af det konstante logiske symbol $Q_{\alpha\alpha o}$ i en fortolkning for $\mathcal{Q}_0$ er identitetsrelationen på $\mathcal{D}_\alpha$. Dette retfærdiggør i en vis forstand den tidligere opfordring til at opfatte $Q_{\alpha\alpha o}$ som en identitetsrelation. [Andrews, 2002, s. 238]

8.5.2 Gyldighed

Formålet med den videre behandling af modeller er at undersøge dem med særligt henblik på vores to systemer $\mathcal{Q}_0$ og $\mathcal{Q}_0^\infty$. Denne undersøgelse skulle gerne bidrage til vores billede af de formelle systemers muligheder og afgrænsninger. Først er vi nødt til at bringe et par definitioner.

Definition 8.18
Lad $\mathbf{A}$ *være en formel af type* o, $\mathcal{M}$ *en generel model og* φ *en tildeling ind i* $\mathcal{M}$.

1. $\mathbf{A}$ *er* gyldig *i* $\mathcal{M}$, *hvis alle tildelinger ind i* $\mathcal{M}$ *tilfredsstiller* $\mathbf{A}$ *i* $\mathcal{M}$, *det vil sige, hvis der for hver tildeling* φ *ind i* $\mathcal{M}$ *gælder, at* $\mathcal{V}_\varphi^{\mathcal{M}}\mathbf{A} = T$.
2. $\mathbf{A}$ *er* gyldig i generel forstand, *hvis* $\mathbf{A}$ *er gyldig i enhver generel model for* $\mathcal{Q}_0$.
3. $\mathbf{A}$ *er* gyldig i standard forstand, *hvis* $\mathbf{A}$ *er gyldig i enhver standardmodel for* $\mathcal{Q}_0$.
4. *En* model *for en mængde af formler* $\mathcal{G}$ *af typen* o *er en model for* $\mathcal{Q}_0$ *(generel eller standard), i hvilken alle formler i* $\mathcal{G}$ *er gyldige.*

Det er værd at bemærke, at gyldighed altid skal ses relativt til den betragtede model. At en formel er gyldig i generel forstand betyder ifølge ovenstående, at der for en vilkårlig generel model $\mathcal{M}$ og en vilkårlig tildeling φ gælder $\mathcal{V}_\varphi^{\mathcal{M}}\mathbf{A} = T$. En formel, der er gyldig i generel forstand, er gyldig i alle generelle modeller. Så er formlen også gyldig i alle standardmodeller og dermed gyldig i standard forstand. Altså medfører generel gyldighed, gyldighed i standard forstand. [Andrews, 2002, s. 239]

Gyldighed er så vigtigt et begreb, at det har fået sin egen notation:

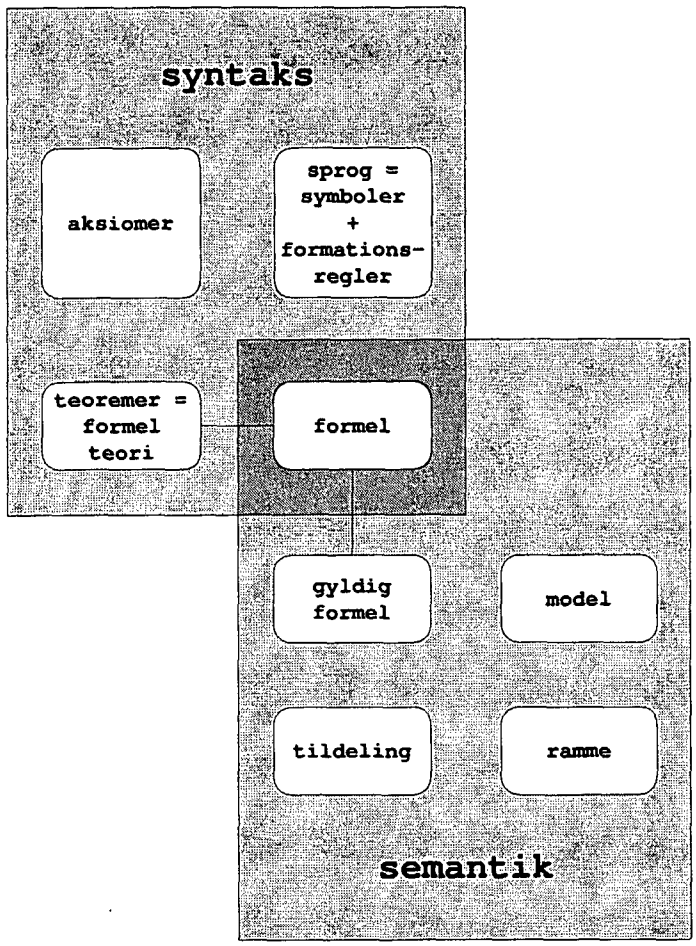
$$\begin{aligned} \mathcal{M} \models \mathbf{A} \quad &\text{står for} \quad \mathbf{A} \text{ er gyldig i } \mathcal{M}. \\ \models \mathbf{A} \quad &\text{står for} \quad \mathbf{A} \text{ er gyldig i generel forstand.} \end{aligned}$$

Et grundlæggende ønske, når man har opstillet en formel teori, er som tidligere beskrevet, at den formelle teori afspejler den uformelle teori. I forhold til ovenstående definitioner kan vi definere dette som ønsket om, at alle teoremerne for $\mathcal{Q}_0$ skal være gyldige formler og omvendt. Vi ønsker os altså, at en model for $\mathcal{Q}_0$ skal være en model for $\mathcal{Q}_0$'s teoremer. Ved hjælp af den nye notation kan dette formuleres således:

$$\vdash \varphi \quad \text{hvis og kun hvis} \quad \models \varphi.$$

[Andrews, 2002, s. 239]

Begrebet *en formel* blev brugt til at etablere en sammenhæng mellem syntaks og semantik for et formelt system og dets modeller. Sammenhængen, samt et overblik over hvor de andre involverede begreber hører til gives på figur 8.1.



Figur 8.1 Billedliggørelse af matematikkens formalisering.

I næste afsnit vil vi undersøge, i hvor høj grad ovenstående ønske går i opfyldelse for vores formelle systemer.

8.6 Metateoremer for $\mathcal{Q}_0$ og $\mathcal{Q}_0^\infty$

Når man har opstillet en formel teori, kan det være interessant at stille følgende spørgsmål:

1. Er det muligt at udlede P i systemet og samtidigt udlede $\neg P$, altså $\vdash P$ og $\vdash \neg P$?
2. Findes der gyldige formler i modeller for teorien, der ikke kan udledes som teoremer i det formelle system?
3. Er alle teoremerne i det formelle system gyldige formler i en model?

4. Er det muligt at afgøre om en formel er et teorem eller ej?

Studiet af ovenstående spørgsmål er fundamentale for studiet af logiske systemers opførsel. Spørgsmålene er knyttede til begreberne *konsistens, fuldstændighed, sundhed* og *afgørlighed.* Nogle af disse spørgsmål er vi allerede stødt på adskillige gange i rapporten.

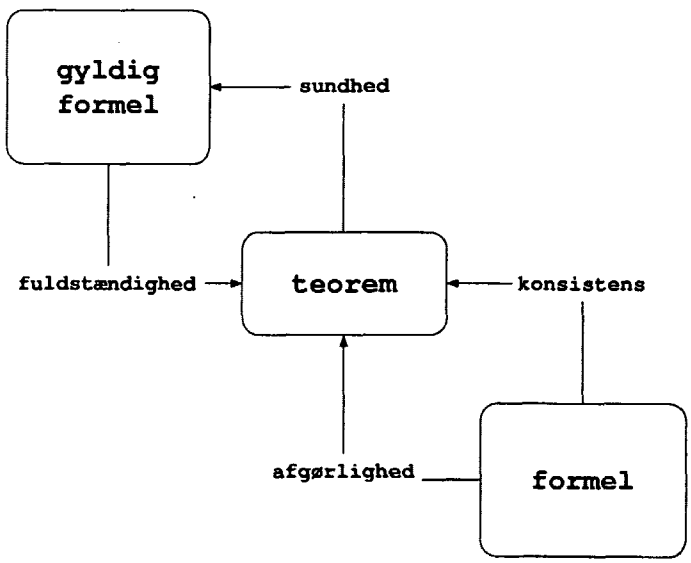
Vi ønsker os naturligvis ikke et system, hvor det er muligt både at udlede P og $\neg P$. I et sådant system vil det være muligt at udlede alle formler som teoremer. Et system hvor det, at P er et teorem, medfører, at $\neg P$ ikke er et teorem, kaldes et *konsistent* system.

Til tider vil det være muligt at afgøre, om en formel er gyldig eller ej uden at gøre brug af et systems logik. I den situation er det interessant at undersøge, om den gyldige formel så er et teorem, det vil sige, om den er indeholdt i teorien. Et system for hvilket en vilkårlig gyldig formel er et teorem kaldes *fuldstændigt.*

Omvendt kan man interesse sig for om alle teoremerne i systemet er gyldige i modeller for systemet. Et system, hvor alle teoremerne er gyldige, kaldes for et *sundt* system.

Det er naturligvis interessant at vide hvilke formler, der er teoremer i systemet. Et system, i hvilket man kan afgøre om en vilkårlig formel er et teorem eller ej, kaldes et *afgørligt* system.

Begreberne *konsistens* og *afgørlighed* er udelukkende knyttet til den syntaktiske del af en formel teori. *Fuldstændighed* og *sundhed*, derimod, fortæller noget om forholdet imellem det formelle system og dets modeller, det vil sige imellem syntaks og semantik. Dette er illustreret på figur 8.2.



Figur 8.2 Et forsøg på at vise sammenhængen mellem metabegreberne.

Vores ønske fra slutningen af sidste afsnit, $\vdash \varphi$ hvis og kun hvis $\models \varphi$, er altså ensbetydende med, at vi ønsker os et system, der er fuldstændigt og sundt.

8.6.1 Resultater for $\mathcal{Q}_0$

Begreberne *konsistens, sundhed, fuldstændighed* og *afgørlighed* fortæller meget om logiske systemers muligheder og afgrænsninger, og de er derfor selvstændige studier værdige. Sådanne generelle studier er dog udenfor denne rapports fokus, og vi vil derfor nøjes med at gengive og diskutere nogle resultater, der vises i [Andrews, 2002].

For det første vises, at $\mathcal{Q}_0$ er *konsistent.* Inkonsistente systemer er ikke interessante for vores problemstilling, og dette resultat må derfor opfattes som særdeles vigtigt. Det vises også, at $\mathcal{Q}_0$ er *sundt*, det vil sige, at alle teoremer er gyldige i generel forstand og dermed også i standard forstand. [Andrews, 2002, s. 240, 243]

Fuldstændighedsspørgsmålet kræver lidt mere end bare en opremsning. Det vises nemlig, at enhver formel af type o er et teorem, hvis og kun hvis det er gyldigt i generel forstand, altså

$$\vdash \varphi \quad \text{hvis og kun hvis} \quad \models \varphi.$$

I forhold til *generelle modeller* er $\mathcal{Q}_0$ altså *fuldstændigt.* Sådan forholder det sig dog ikke i forhold til standardmodeller.

Sundhedsresultatet giver, at alle teoremerne er gyldige i standard forstand. Imidlertid kan vi ikke slutte den anden vej. Andrews forklarer, at der i forhold til standardmodeller altid findes gyldige formler, som ikke er teoremer – eksempelvis udvalgsaksiomet. I forhold til *standardmodeller* er $\mathcal{Q}_0$ altså *ikke fuldstændigt.* [Andrews, 2002, s. 254-255]

Det vises tilsidst, at $\mathcal{Q}_0$ er uafgørligt. Dette resultat er ikke overraskende set i lyset af Churchs sætning (se kapitel 7). [Andrews, 2002, s. 301]

8.6.2 Resultater for $\mathcal{Q}_0^\infty$

Systemet $\mathcal{Q}_0^\infty$ må opfattes som et godt sted at starte, hvis man ønsker at udtale sig om systemer, i hvilke større dele af matematikken kan formaliseres. De resultater angående $\mathcal{Q}_0^\infty$, der bringes i dette afsnit, gælder (med passende modifikationer) for ethvert logisk system hvori matematikken kan formaliseres. [Andrews, 2002]

Resultaterne, Andrews viser for $\mathcal{Q}_0^\infty$, er følgende:

1. *Konsistensen* af $\mathcal{Q}_0^\infty$ kan ikke vises af metoder, der kan formaliseres i $\mathcal{Q}_0^\infty$.
2. $\mathcal{Q}_0^\infty$ er *ufuldstændigt*, og der findes ingen udvidelser af $\mathcal{Q}_0^\infty$, der kan defineres præcist[7], som er fuldstændige.
3. $\mathcal{Q}_0^\infty$ er *uafgørligt.*

[Andrews, 2002, s. 301-302]

Systemet $\mathcal{Q}_0^\infty$ vil vi vende tilbage til i anden del af rapporten.

[7] Hermed menes, at der ingen rekursivt aksiomatiserbare udvidelser findes. Dette vil vi ikke komme nærmere ind på.

9 Opsamling I

Vi bringer her en opsamling på rapportens del I og i forbindelse med denne en kort diskussion af typeteoriens fremkomst og udvikling.

Frege revolutionerede som nævnt Aristoteles' godt totusinde år gamle logik. Han var en af de første som indførte den sandhedsbaserede propositionslogik, i hvilken prædikat og subjekt opfattes som funktion og argument, og han opstillede uafhængigt af andre kvantificeringsteorien. Derudover er Frege en af de første, der opstiller et formelt system. Hans *Begriffsschrift* fra 1879 var et formelsprog, hvori det er muligt at udlede resultater på ren 'mekanisk' vis ud fra en række aksiomer ved hjælp af nogle få slutningsregler. Som påpeget af Russell var Freges system dog inkonsistent. Ikke desto mindre regnes Frege i dag for at være en af den moderne logiks grundlæggere.

Russell viste, at Freges system var inkonsistent ved at påvise eksistensen af sit paradoks i systemet. For at undgå dette og lignende matematiske paradokser introducerede Russell i 1908 typeteorien. Denne udgave af typeteorien, kaldet ramificeret typeteori, blev benyttet som en del af grundlaget for Whiteheads og Russells store værk *Principia mathematica*, i hvilket forfatterne søgte at udlede en stor del af matematikken. Den ramificerede typeteori er kendetegnet ved princippet om, at onde cirkler (*vicious circles*) skal undgås. Russells typeteori gør det muligt både at undgå de tidligere beskrevne matematisk-logiske paradokser samt semantiske paradokser. Teorien blev dog aldrig rigtigt accepteret på grund af de vanskeligheder, der var forbundet med den.

I 1931 viste Gödel sine to ufuldstændighedssætninger. Med disse blev det klart, at man ikke kan opstille et system, i hvilket hele matematikken kan formaliseres, og at man inden for et system ikke er i stand til at bevise systemets konsistens. Gödel tog i sin fremstilling af sine resultater netop udgangspunkt i et system i stil med systemet fra *Principia mathematica*. Med de to resultater besvarede Gödel de to første af Hilberts tre spørgsmål fra 1928, nemlig spørgsmålene om hvorvidt matematikken var (1) fuldstændig og (2) konsistent. Svarene på begge spørgsmål var negative, hvilket ødelagde Hilberts program som oprindeligt fremsat. Gödels resultater var nogle af de mest overraskende og bemærkelsesværdige i moderne matematik.

Hilberts tredje spørgsmål, hvorvidt matematikken er afgørlig – det såkaldte *Entscheidungsproblem* – var dog stadig ubesvaret. Men det tog ikke mere end fem år (1936) førend dette spørgsmål også fik et negativt svar. Church og Turing viste omtrent samtidigt og uafhængigt af hinanden, at dette var tilfældet. Deres metoder var dog vidt forskellige. Church anvendte sin λ-kalkyle, og Turing sin beregningsmaskine (Turingmaskinen) til at give en definition af beregnelighed. Church viste tilmed at hans λ-definerbare funktioner var lig de rekursive funktioner. Turing viste i et appendiks til sin artikel, som blev udgivet i 1937, at

Turingmaskinen var ækvivalent med λ-kalkylen (og dermed også med de rekursive funktioner).

I 1940 introducerede Church et system der var baseret på λ-kalkyle og simpel typeteori. Dette system minder meget om systemet $\mathcal{Q}_0^\infty$ der er præsenteret i kapitel 8. En af forskellene på den simple og den ramificerede typeteori kan illustreres ved følgende eksempel:

Definitionen af falskhed i $\mathcal{Q}_0^\infty$ ser ud som følger:

$$\lambda x_o.T = \lambda x_o.x_o$$

hvilket uden forkortelser skrives

$$Q_{ooo}[\lambda x_o.T_o][\lambda x_o.x_o]$$

Med forkortelsen Π_{ooo} kan dette skrives

$$\Pi_{ooo}\lambda x_o.x_o$$

som kan forkortes yderligere til

$$\forall x_o.x_o$$

Ovenstående læses: *Alle propositioner er sande.* Dette er et ulovligt udtryk i Russells system, da det udtaler sig om 'alle propositioner'.

Lad os slutte denne del af rapporten det samme sted, som vi begyndte den, nemlig ved Liebniz og hans drøm. Frege opfattede sit *Begriffsschrift* som indeholdende det universalsprog, Leibniz havde eftersøgt. Leibniz selv ville nok have været skuffet, idet han ikke kun forestillede sig, at universalsproget skulle kunne anvendes til logisk deduktion, men at det også samtidig ville indeholde alle naturvidenskabelige sandheder såvel som filosofiske.

Udforskningerne førte til Gödels højst usædvanlige og overraskende resultater, som implicerede et negativt svar på afgørlighedsspørgsmålet, der kan betragtes som en del af Leibniz' drøm. Dette var dårlige nyheder for drømmen. Hvis svaret på afgørlighedsproblemet havde været et ja, havde det være muligt at realisere Leibniz' drøm, i hvert fald indenfor matematikken. Svaret var imidlertid et nej.

Lad os som en sidebemærkning nævne noget, som nok ville have glædet Leibniz. Leibniz byggede omkring 1673 en mekanisk regnemaskine, som overgik den Blaise Pascal tidligere havde bygget i 1623, idet Leibniz' maskine kunne udføre alle fire basale regneoperationer mens Pascals kun kunne udføre addition og subtraktion. Leibniz havde dog langt større planer for hvad maskinerne skulle kunne, idet han sammenlignede logisk ræsonnering med en mekanisme, som burde kunne udføres af en maskine. I processen med at besvare afgørlighedsproblemet, viste Turing, som tidligere nævnt, at det i princippet var muligt at bygge en enkelt 'universalmaskine' kan udføre en hvilken som helst ladesiggørlig udregning. Så selv om Leibniz måtte give afkald på én drøm, fik han på sin vis en anden opfyldt. [Davis, 2000, s. 8, 58]

I det næste skal vi se på typeteoriens anvendelse i den moderne bevisfører *Isabelle* og beviset for korrektheden af den datalogiske sorteringsalgoritme Quicksort i denne.

Del II

Datalogiens anvendelse af matematikkens formalisering og typeteorien

10 Om bevisførere

I dette kapitel vil vi beskrive hvad en bevisfører er, og hvad den består af. Vi vil også opremse femten matematiske 'state of the art'-bevisførere og give en kort beskrivelse af disse. Vi vil undervejs drage paralleller til bevisføreren *Isabelle*, da det er den, vi senere skal beskæftige os med.

10.1 De tre hovedtyper af bevisførere

En bevisfører er et stykke programmel. Ved hjælp af en inkorporeret logik er den i stand til at udlede resultater, verificere computerprogrammer eller bevise, for eksempel, matematiske sætninger på mere eller mindre 'automatisk' vis. For en mere uddybende diskussion af dette se næste kapitel (kapitel 11). Helt overordnet kan bevisførere inddeles i tre kategorier: Beviskontrollører, bevisgeneratorer og bevisassistenter (en konkret maskine kan dog godt tilhøre flere kategorier). Vi præsenterer samtlige kategorier nedenfor.

10.1.1 Beviskontrollører

Beviskontrollørerne modtager et formelt bevis som inddata. Hvert skridt i beviset kontrolleres ved direkte anvendelse af den inkorporerede logiks regler. Maskinen assisteres ikke under sin verifikation af beviset. Brugeren forventes at give eventuel 'assistance' som inddata. [Bramley-Moore, 2003, s. 3-4]

10.1.2 Bevisgeneratorer

Bevisgeneratorerne benytter heuristikker til at guide søgningen efter et bevis for formler, der gives som inddata. Bevisføreren *Otter* er et eksempel på en sådan generator. Den læser inddataen og bestemmer sig derefter for slutningsregler og strategier. I tilfælde af opstillingen af et bevis fejler, hvilket hænder, har brugeren også mulighed for at fastsætte begyndelsesbetingelser og strategier for beviset. [Bramley-Moore, 2003, s. 4]

Bevisførere gør ofte brug af tilhørende matematiske biblioteker. Et sådant bibliotek består af de teoremer, som bevisføreren tidligere har bevist. Dette muliggør bevisførelse for mere komplekse teoremer. Dette vil vi vende tilbage til senere i kapitlet.

10.1.3 Bevisassistenter

Bevisassistenterne guides af mennesker igennem deres generering af beviser. Bevisassistentens opgave er at foretage de mest trivielle og trættende skridt i et formelt bevis. Brugernes opgave bliver da at foreslå strategier til bevisassistenten. Delvis automatisering benyttes ofte i den bevisskabende proces for at skjule de mest simple udregninger fra beviserne. [Bramley-Moore, 2003, s. 5]

Bevisassistenterne opstod fra et ønske om at foretage lange og komplekse formelle beviser uden at skulle producere disse manuelt. Robin Milner gjorde brug af et meta-sprog (her et formelt symbolsprog som benyttes til at beskrive og ræsonnere i et andet sprog) i bevisføreren *Edinburgh LCF*, hvilket muliggjorde det for programmører at repræsentere bevistaktikker for forskellige logikker. Ideen om et metasprog (meta language – ML) spredte sig til andre bevisassistenter, som for eksempel *Isabelle*, *NuRPL* og *HOL*. [Bramley-Moore, 2003, s. 5]

10.1.4 Bevisførernes systemer

Bevisførernes systemer kan variere fra bevisfører til bevisfører. Der er hovedsageligt tale om nogle få forskellige typer af systemer, som førerne arbejder med. Vi vil kort beskrive to af disse nedenfor – systemer i Hilbert-stil og systemer, der anvender naturlig deduktion – da disse er relevante i *Isabelle*-sammenhæng. De resterende vil vi blot nævne ved navn.

Et system i *Hilbert-stil* (nogen gange også kaldet Frege-stil) er et system, der anvender den aksiomatiske metode. Her udpeges en række 'fornuftige' udsagn til at være aksiomer, hvorefter konsekvenser (teoremer) af disse udledes ved hjælp af slutningsreglerne i systemet. Ideen i Hilbert-stil er at grundlaget for et bevis skal være så simpelt som muligt. Derfor skal alle teoremer kunne føres tilbage til aksiomerne. Systemet $\mathcal{Q}_0^\infty$, der tidligere er blevet opstillet i kapitel 8, er et eksempel på et system i Hilbert-stil. [Andrews, 2002, s. 21-22]

Den form for logisk deduktion, der blev anvendt af Frege, Russell og Hilbert, ligger langt fra den type deduktion, som normalt anvendes i matematiske beviser. Dette skyldes, at Hilbert-stilen i praksis ikke er særligt tilfredsstillende, når der skal opstilles beviser. Beviser i Hilbert-stil-systemer bliver nemlig ofte lange og uoverskuelige.

For at lette bevisarbejdet i formelle systemer udviklede Gerhard Gentzen i 1935 *naturlig deduktion*. I et system, der anvender naturlig deduktion er det aksiomatiske bevisgrundlag fra Hilbert-stilen udskiftet med en række *antagelser* (også kaldet *hypoteser*). Ligesom i Hilbert-stilen benyttes desuden en række slutningsregler. Generelt for systemer i naturlig deduktion gælder, at de indeholder både introduktions- og eliminationsregler[1]. Denne tilgang gør det muligt at arbejde i et formelt system, og samtidigt give naturlige beviser, det vil sige at anvende argumenter, som ligner de argumenter, der anvendes i matematisk praksis. [Andrews, 2002, s. 151-152]

De resterende tilgange/strategier er *sekventkalkyle*, *resolution* og *tableau-beviser*.

[1] For en uddybelse af dette kan [Nerode and Shore, 1997] konsulteres.

10.2 De femten bevisførere

Freek Wiedijk har i sin artikel *Comparing Mathematical Provers* samt i et tilhørende arbejdspapir[2] beskrevet femten matematiske bevisførere[3] og eksemplificeret deres tilgang til problemstillinger ved at beskrive og undersøge deres beviser for at $\sqrt{2}$ er et irrationalt tal. Wiedijk omtaler selv sin undersøgelse af de femten bevisførere som en form for 'forbrugerundersøgelse'. I og med at det er krævende nok, at sætte sig ind i blot én bevisfører, må Wiedijks undersøgelse siges at være noget speciel, da den jo kræver indgående kendskab til hele femten bevisførere. [Wiedijk, 2003a, s. 189]

I artiklen bringes indledningsvis en tabel, som giver et indblik i de forskellige bevisførere. Vi gengiver et uddrag af tabellen (se tabel 10.1), og forklarer den i det følgende.

de Bruijns kriterie

Kravet om en lille beviskerne er kendt som *de Bruijns kriterie*. Dette siger, at korrektheden af matematikken i et system skal kunne garanteres af en *lille beviskerne*, hvorigennem matematikken skal filtreres. Det at beviskernen er lille bidrager til troværdigheden af systemet, idet systemet gøres mere overskuelig. *HOL Light*, en variant af *HOL*-systemet, har en ekstremt lille kerne, som udelukkende består af 285 liniers ML-kode. [Wiedijk, 2003a, s. 197-198]

Poincarés princip og automatisering

Et vigtigt aspekt i et matematisk system er automatiseringen af trivielle opgaver. Specielt bør en bruger ikke skulle guide systemet igennem udregninger i detaljer. Et system, som kan bevise korrektheden af beregninger automatisk, siges at opfylde Poincarés princip. [Wiedijk, 2003a, s. 198-199]

Et andet vigtigt aspekt af en bevisfører er, om den har en såkaldt 'åben' arkitektur. Det vil sige, om en bruger er i stand til at skrive programmer som løser bevisproblemer automatisk. En del bevisførere tillader noget automatisering som sådan. I tredje søjle af tabel 10.1 hentyder Wiedijk til, hvorvidt programmerbarheden af en sådan automatisering er på niveau med implementeringen af systemet. [Wiedijk, 2003a, s. 198-199]

Fjerde søjle i tabellen angiver om bevisføreren har en kraftfuld indbygget automatisering. Eksempler på en sådan automatisering er for eksempel afgørlighedsprocedurer for algebraiske problemer, procedurer for bevissøgning og automatisering af induktion. [Wiedijk, 2003a, s. 198-199]

[2] *The Fifteen Provers of the World.*

[3] For en uddybende begrundelse for, hvorfor det netop er disse femten bevisførere der er valgt se [Wiedijk, 2003a].

Læsbare bevis-inddata-filer

Denne femte søjle i tabel 10.1 angiver om bevis-inddata-filerne er læsbare. Med 'læsbare' menes 'forståelige', det vil sige, hvor nemt det er for en bruger at læse, tyde og forstå bevis-inddata-filerne.

bevisfører	de Bruijns kriterie	Poincarés princip	brugerautomatisering	kraftfuld indbygget automatisering	læsbare bevis-inddata-filer	logisk ramme	typet	baseret på højereordens-logik	baseret på ZFC mængdelære	stort matematisk standardbibliotek	udsagn om $\mathbf{R}$	udsagn om $\sqrt{\ }$
HOL	+	+	+	+	−	−	+	+	−	+	+	+
Mizar	−	−	−	−	+	−	+	−	+	+	+	+
PVS	−	+	+	+	−	−	+	+	−	+	+	+
Coq	+	+	+	−	−	−	+	+	−	+	+	+
Otter/Ivy	+	+	−	+	−	−	−	−	−	−	−	−
Isabelle/Isar	+	+	+	+	+	+	+	+	+	+	+	+
Alfa/Agda	+	−	−	−	−	−	+	+	−	−	−	−
ACL2	−	+	−	+	+	−	−	−	−	−	+	−
PhoX	+	+	−	−	−	−	+	+	−	−	−	−
IMPS	−	+	−	+	−	−	+	+	−	+	+	+
Metamath	+	−	−	−	−	+	−	−	+	−	+	+
Theorema	−	+	−	+	+	−	−	+	−	−	+	+
Lego	+	+	−	−	−	−	+	+	−	−	−	−
NuRPL	−	+	+	−	−	−	+	+	−	+	−	−
Ωmega	+	+	+	+	−	−	+	+	−	−	+	+

Tabel 10.1 Tabel over indholdet af Wiedijks femten udvalgte matematiske bevisførere. Inddelingen af søjlerne svarer til gennemgangen af tabellen. [Wiedijk, 2003b, s. 2]

Den logiske ramme

Systemerne varierer i underlæggende logik og typesystem. Wiedijk bringer følgende tabel (se tabel 10.2) til anskueliggørelse af de forskellige logikker i de femten systemer.

bevisfører	primitiv rekursiv aritmetik	førsteordens-logik	højereordens-logik	førsteordens-mængdelære	højereordens-typeteori	klassisk logik	konstruktiv logik	kvantum-logik	fikseret logik	logisk ramme
HOL	−	−	+	−	−	+	−	−	+	−
Mizar	−	−	−	+	−	+	−	−	+	−
PVS	−	−	+	−	−	+	−	−	+	−
Coq	−	−	−	−	+	−	+	−	+	−
Otter/Ivy	−	+	−	−	−	+	−	−	+	−
Isabelle/Isar	−	−	+	+	−	+	−	−	−	+
Alfa/Agda	−	−	+	−	−	−	+	−	+	−
ACL2	+	−	−	−	−	+	−	−	+	−
PhoX	−	−	+	−	−	+	−	−	+	−
IMPS	−	−	+	−	−	+	−	−	+	−
Metamath	−	−	−	+	−	+	−	+	−	+
Theorema	−	−	+	−	−	+	−	−	+	−
Lego	−	−	+	−	−	−	+	−	+	−
NuRPL	−	−	−	−	+	−	+	−	+	−
Ωmega	−	−	+	−	−	+	−	−	+	−

Tabel 10.2 Tabel over de forskellige logikker i Wiedijks femten udvalgte bevisførere. Bemærk, at den sidste søjle i ovenstående tabel også forekommer i tabel 10.1. [Wiedijk, 2003a, s. 196]

En logisk ramme understøtter ikke en enkelt given logik. I stedet kan brugeren definere sine egne logikker – nogen er dog foruddefineret. Bemærk at der i tilfældet af en logisk ramme i tabel 10.2, det vil sige for *Isabelle* og *Metamath*, kun er indikeret de mest almindelige benyttede logikker i systemet i de foregående søjler af tabellen.

Bevisførernes typeteori

Om en bevisfører er typet eller ej, hentyder til hvorvidt den kan benytte sig af typeteori. Wiedijk bringer følgende tabel (se tabel 10.3) over typesystemerne i de 15 bevisførere.

bevisfører	utypede	afgørlige ikke-afhængige typer	afgørlige afhængige typer	uafgørlige afhængige typer
HOL	−	+	−	−
Mizar	−	−	+	−
PVS	−	−	−	+
Coq	−	−	+	−
Otter/Ivy	+	−	−	−
Isabelle/Isar	−	+	−	−
Alfa/Agda	−	−	+	−
ACL2	+	−	−	−
PhoX	−	+	−	−
IMPS	−	+	−	−
Metamath	+	−	−	−
Theorema	+	−	−	−
Lego	−	−	+	−
NuRPL	−	−	−	+
Ωmega	−	+	−	−

Tabel 10.3 De typede systemer i Wiedijks udvalgte bevisførere. Bemærk at første søjle modsvarer typeteori-søjlen i tabel 10.1. [Wiedijk, 2003a, s. 197]

Med afgørlighed menes, at man for et givet term kan bestemme, hvilken type det har.

At en type er afhængig, kan betyde at en type kan afhænge af en term såvel som af en anden type. Vi vil ikke komme nærmere ind på dette, da *Isabelle* kun benytter afgørlige ikke-afhængige typer i metalogikken og objektlogikken *HOL*.

Logiske fundamenter for bevisførere

Bevisførerne kan være baseret på forskellige logikker, for eksempel højereordenslogik (HOL) eller ZFC-mængdelære, og derved tage udgangspunkt i forskellige aksiomer og slutningsregler. Dette vil blive uddybet i senere afsnit.

Matematisk bibliotek

I praksis er et stort matematisk bibliotek, ifølge Wiedijk, mere vigtigt end et brugervenligt system. *Mizar*-systemet har langt det største bibliotek. Det indeholder beviser for over 32.000 lemmaer og fylder mere end 50 megabytes eller 1,4 millioner linier. [Wiedijk, 2003a, s. 196]

Ikke alle systemer indeholder teoremer om reelle tal og kvadratrødder i det matematiske bibliotek. I tabel 10.1 er opremset de systemer, som gør. Har man således to teoremer indeholdt i biblioteket; et som siger at kvadratroden af et primtal er irrationalt og; et andet som siger at 2 er et primtal, følger direkte at $\sqrt{2}$ er irrational.

Vi skal i næste kapitel se nærmere på, hvilke resultater bevisførerne kan producere.

11 Bevisførernes domæne

Som nævnt i forrige kapitel strækker bevisførernes domæne sig fra matematiske beviser over verifikation af computerhardware til beviser for korrekthed af algoritmer. Vi vil i dette kapitel gå lidt mere i dybden med disse områder og samtidig give en generel gennemgang af sorteringsalgoritmen Quicksort, hvis korrekthed vi senere skal se på. Vi vil undervejs i kapitlet tillade os at drage paralleller til bevisføreren *Isabelle*, hvor vi finder det relevant, da det jo er denne vi senere skal se på.

11.1 Matematiske beviser

Et af de mest oplagte områder for bevisførerne er beviserne for matematiske resultater og sætninger. Der kan være tale om mindre matematiske resultater, så som at vise at et bestemt tal er irrationalt, for eksempel $\sqrt{2}$ ligesom i Wiedijks *Fifteen Provers*. Men der kan også være tale om mere komplekse opgaver, eksempelvis større matematiske teorier.

Som vi så i forrige kapitel (kapitel 10) kan bevisførere kontrollere allerede eksisterende beviser såvel som generere beviser med mere eller mindre vejledning fra en bruger. Specielt i forbindelse med beviser for større og mere komplekse teorier, kan der være behov for en vejledning fra brugerens side.

11.2 Beviser for algoritmers korrekthed

Et andet område for bevisførerne, er beviserne for korrektheden af algoritmer. Et eksempel, som er relevant for vores projekt, er sorteringsalgoritmer.

For at bevise korrektheden af en sorteringsalgoritme kan man bevise følgende:

- At algoritmen terminerer, det vil sige kører færdig i endelig tid.
- At der ikke forsvinder elementer undervejs i sorteringsproceduren.
- At algoritmen sorterer de pågældende elementer i overenstemmelse med en beskreven orden.

Kan man vise disse tre ting, har man bevist, at algoritmen er korrekt.

I *Isabelle* vises for eksempel korrektheden af de tre sorteringsalgoritmer Mergesort, Insertion-sort og Quicksort. Vi skal senere i projektet se, hvorledes dette gøres for Quicksort.

11.3 Yderligere områder

Et andet nærliggende område for bevisførerne er køretider for algoritmer. En bevisfører kan således vise, hvad køretiden er for en konkret algoritme, såvel det værste tilfælde som det forventede. Også her kan der være tale om at bevisføreren kontroller beviset for en køretid, eller at den selv genererer et bevis for køretiden for en konkret algoritme.

Bevisførere kan også benyttes til at verificere computersoftware eller -hardware. Eksempler på dette kan være verificering af en compiler eller verificering af assembler-programmer i chips som anvendes i computerhardware. Nipkow og Paulson har ved hjælp af *Isabelle* blandt andet verificeret dele af Java-programmer.

Bevisførere kan også anvendes indenfor kryptografi, idet de kan benyttes til at verificere kryptografiprotokoller.

11.4 Gennemgang af Quicksort

Vi ønsker senere at se på *Isabelles* bevis for korrektheden af sorteringsalgoritmen Quicksort, så vi bringer i det følgende en kort gennemgang af pseudokode for Quicksort, samt en gennemgang af algoritmens overordnede principper.

11.4.1 Del-og-hersk

Quicksort er baseret på det generelle paradigme til algoritmedesign kaldet *del-og-hersk*. Del-og-hersk kan beskrives i tre skridt:

1. *Del*: Hvis størrelsen af inddata S er under en bestemt grænse (for eksempel et eller to elementer), løses problemet direkte ved at benytte en ligefrem metode, og den opnåede løsning returneres. I modsat tilfælde, opdel inddata S i to disjunkte delmængder S_1 og S_2.
2. *Løs*: Løs rekursivt de delproblemer, der svarer til S_1 og S_2.
3. *Hersk*: Kombiner løsningerne for S_1 og S_2 til en løsning for det oprindelige problem S.

[Goodrich and Tamassia, 2002, s. 219], [Helsgaun, 2003]

11.4.2 Sorteringsalgoritmen Quicksort

Quicksort-algoritmen er en sorteringsalgoritme, der er baseret på del-og-hersk paradigmet. Quicksort-algoritmen virkende på en inddata-sekvens S indeholdende n elementer består af følgende tre trin:

1. *Del*: Vælg et tilfældigt element x (kaldet *pivot*-elementet) og opdel S i

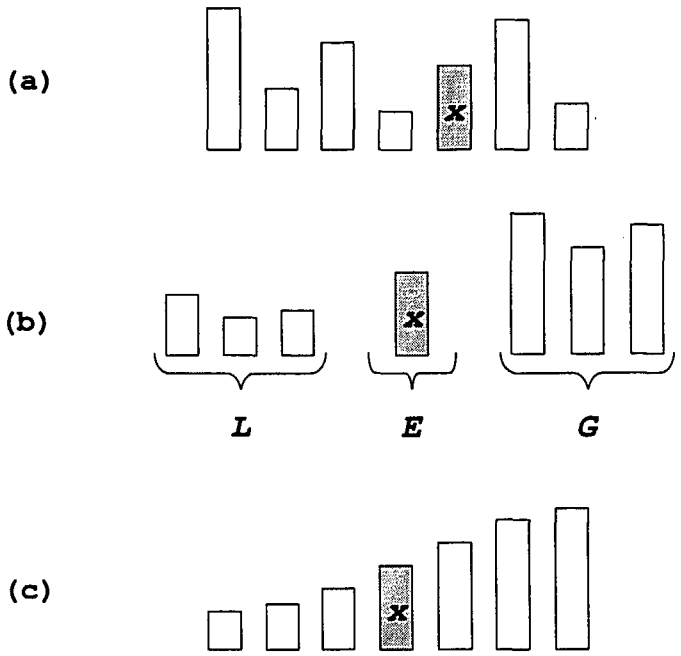
L: Elementer mindre end x.

E: Elementer lig med x.

G: Elementer større end x.

2. *Løs*: Sorter L og G.
3. *Hersk*: Sammenføj L, E og G.

En illustration af disse tre trin kan ses i figur 11.1.



Figur 11.1 En billedliggørelse af *del*-, *hersk*- og *løs*-trinnet i Quicksort-algoritmen. (a) Et pivot-element x udvælges. (b) Elementerne opdeles i de tre sekvenser L, E og G. (c) Efter sortering og sammenføjning. [Helsgaun, 2003]

Ligesom mange andre sorteringsalgoritmer benytter Quicksort en komparator, her kaldet C, som gør det muligt at sammenligne elementer indbyrdes i forhold til en eller anden given ordning[1]. Pseudo-koden for Quicksort kan se ud som i algoritme 11.1.

Inddata-sekvensen S opdeles på følgende vis:

- Ét efter ét fjernes ethvert element y fra S.
- Afhængigt af sammenligningen med pivot-elementet x overføres y til L, E eller G.

Pseudo-koden for denne opdeling findes i algoritme 11.2.

[1]For en mere uddybende beskrivelse se [Goodrich and Tamassia, 2002].

```
Input sequence S with n elements, comparator C
Output sequence S sorted according to C

if S.size()>1 then
  p ← pivot of S
  (L,E,G) ← partition(S,C,p)
  quickSort(L,C)
  quickSort(G,C)
  S ← join(L,E,G)
```

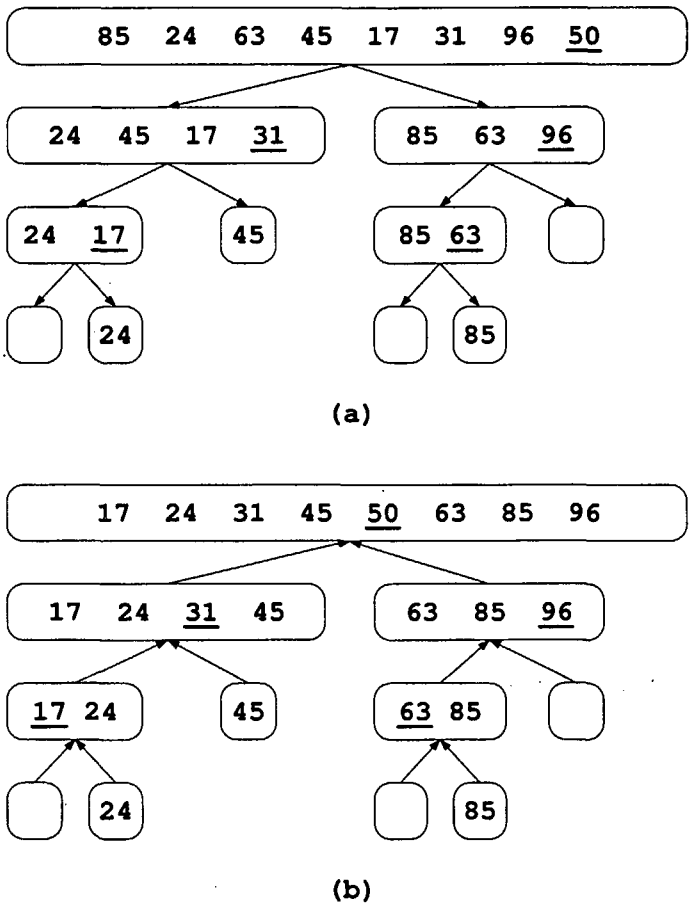
Algoritme 11.1 Pseudo-kode for *quickSort*(S,C). `size()` returnerer antallet af elementer i sekvensen S. `pivot of` S giver positionen for et pivotelement x, og kan antage værdier fra 1 til n. `join()` sammenføjer sekvenserne i den angivne rækkefølge og returnerer en ny sekvens. [Helsgaun, 2003]

```
Input sequence S, comparator C, position p of pivot
Output subsequences L,E,G of the elements of S less
than, equal to, or greater than the pivot, resp.

L,E,G ← empty sequences
x ← S.remove(p)
while ¬S.isEmpty() do
  y ← S.remove(S.first())
  if compare(y,x)<0 then
    L.insertLast(y)
  else if compare(y,x)=0 then
    E.insertLast(y)
  else{compare(y,x)>0}
    G.insertLast(y)
return L,E,G
```

Algoritme 11.2 Pseudo-kode for *partition*(S,C,p). Del af *quickSort*-algoritmen. `remove()` fjerner elementet på den angivne position og returnerer dette. `isEmpty()` undersøger om en sekvens er tom og returnerer `true`, hvis den er, og `false`, hvis den ikke er. `first()` returnerer det første element i en sekvens. `compare()` sammenligner to elementer og returnerer -1, 0 eller 1, hvis det først angivne element er mindre end, lig med eller større end det sidst angivne i forhold til komparatoren. `insertLast()` indsætter et element sidst i en sekvens. [Helsgaun, 2003]

Udførelsen af Quicksort kan anskueliggøres ved hjælp af et binært træ (se figur 11.2). Hver knude repræsenterer et rekursivt kald af *quickSort* og indeholder (1) en usorteret sekvens før sin opdeling og (2) en sorteret sekvens efter udførelsen. Roden er det første kald, og bladene er delsekvenser af længde 0 eller 1.



Figur 11.2 En illustration af Quicksort ved et binært træ. Quicksort er udført på otte elementer. (a) Inddata-sekvenserne udført ved hver knude af træet. (b) Uddata-sekvenserne genereret ved hver knude af træet. Pivot-elementet benyttet ved hvert niveau af rekursionen er fremhævet ved understregning. [Goodrich and Tamassia, 2002]

Lad os for god ordens skyld lige omtale køretiden for Quicksort. I værste tilfælde er køretiden $O(n^2)$, den forventede køretid for Quicksort er dog mere medgørlig, idet denne er $O(n \log n)$. [Helsgaun, 2003], [Goodrich and Tamassia, 2002, s. 237-238]

12 Om *Isabelle*

I dette kapitel vil vi forsøge at give et overblik over *Isabelle*. Programmet *Isabelle* er en generisk bevisassistent, der er udarbejdet af Lawrence C. Paulson fra Cambridge University og Tobias Nipkow fra Technische Universität München. At *Isabelle* er *generisk* vil sige, at *Isabelle* kan benytte forskellige logikker som grundlag for beviser. *Isabelle* er nemlig konstrueret således, at brugeren kan specificere hvilken logik, han/hun ønsker skal anvendes. For at lette arbejdet med at opskrive og bevise teoremer kan *Isabelle* anvendes gennem en brugergrænseflade kaldet *Proof General*. [Paulson and Nipkow, 2003]

Figur 12.1 *Proof General.* Emblemerne på generalens bryst angiver nogle af de bevisførere som gør brug af grænsefladen.

12.1 Logik i *Isabelle*

Isabelles logik består af en *metalogik*, der er implementeret direkte i programmet, samt en række *objektlogikker*, der formaliseres i metalogikken. Objektlogikkernes formål er at levere et logisk grundlag (tidligere omtalt som en beviskerne) for *Isabelles* beviser. Denne struktur tillader brug af forskellige objektlogikker, så længe disse kan formaliseres i metalogikken. [Paulson, 1989, s. 3]

12.1.1 *Isabelles* metalogik

Isabelles metalogik $\mathcal{M}$ er en intuitionistisk højereordenslogik, der er baseret på et fragment af den simple typeteori, som beskrevet i kapitel 8. Typerne for $\mathcal{M}$ afhænger af den logik, der skal repræsenteres formelt, dog findes typen *prop* (proposition) altid. De logiske konstanter for $\mathcal{M}$ er følgende (σ angiver en vilkårlig type):

$$\begin{aligned} \Rightarrow &: prop \to (prop \to prop). \\ \bigwedge\nolimits_\sigma &: (\sigma \to prop) \to prop. \\ \equiv_\sigma &: \sigma \to (\sigma \to prop). \end{aligned}$$

For metalogikken spiller $\bigwedge$ rollen som en $\forall$-kvantor. Metalogikken gør brug af *naturlig deduktion* (jævnfør kapitel 10). Implikationsreglerne for $\mathcal{M}$ er følgende:

$$\frac{\begin{array}{c}[\phi] \\ \psi\end{array}}{\phi \Rightarrow \psi} \quad \Rightarrow\text{-introduktion}$$

og

$$\frac{\phi \Rightarrow \psi \quad \phi}{\psi} \quad \Rightarrow\text{-elimination.}$$

Notationen $[\phi]$ betyder, at ϕ er en antagelse. $\Rightarrow$-introduktion etablerer altså en formel på baggrund af en antagelse ved at afgive[1] antagelsen. $\Rightarrow$-elimination er blot *modus ponens* i metalogikken.

Reglerne for universel kvantificering er følgende:

$$\frac{\phi}{\bigwedge x.\phi} \quad \bigwedge\text{-introduktion}$$

og

$$\frac{\bigwedge x.\phi}{\phi[b/x]} \quad \bigwedge\text{-elimination.}$$

Ovenstående regler kaldes også generalisering og specialisering. Symbolet $\bigwedge$ anvendes i stedet for $\forall$, fordi man ønsker at skelne imellem symbolerne i metalogikken og symbolerne i objektlogikken. Generaliseringsreglen forudsætter, at x ikke er fri i antagelserne.

Reglerne for lighed er refleksivitet, symmetri og transitivitet:

$$a \equiv a \qquad \frac{a \equiv b}{b \equiv a} \qquad \frac{a \equiv b \quad b \equiv c}{a \equiv c}.$$

Ovenstående regler benyttes til at udtrykke forskellige forhold i objektlogikken gennem metalogikken. Implikationsreglerne bruges til at etablere teoremer i objektlogikken, kvantificeringsreglerne i metalogikken udtrykker aksiomsskemaer

[1] På engelsk: *Discharge.*

i objektlogikken, og lighed udtrykker definitioner i objektlogikken. [Paulson, 1989, s. 3-6]

Udover ovenstående regler gør *Isabelles* metalogik brug af α-konversion, β-konversion og ekstensionalitet:

$$\lambda x.a \equiv \lambda y.a[y/x] \qquad [\lambda x.a]b \equiv a[b/x] \qquad \frac{f[x] \equiv g[x]}{f \equiv g}.$$

Isabelles metalogik er placeret i *ProtoPure* og *CPure* (det sidste bibliotek er specifikt for *HOL*).

12.1.2 *Isabelles* objektlogikker

Metalogikken giver som tidligere beskrevet mulighed for at anvende forskellige objektlogikker. Disse objektlogikker omfatter to højereordens-logikker, adskillige førsteordens-logikker, herunder *ZF*, samt en række andre logikker. I denne projektrapport har vi valgt, at beskæftige os med den ene af de to højereordens-logikker – en logik kaldet *HOL*. Denne logik skal vi udforske i det følgende.

Betegnelsen *HOL* bliver brugt på en tvetydig måde. For det første er *HOL* en samling af teorier, der ligger i en pakke. Det er denne pakke, vi benytter. For det andet bruges *HOL* også til at benævne grundsystemet i pakken – det system, som alle de resterende systemer i pakken bygger på. I afsnit 12.2 vil vi forklare dette nærmere samt præsentere de teorier, som benyttes i beviset for Quicksort. Først vil vi dog fortælle, hvorfor *Isabelles* logo ser ud, som det gør.

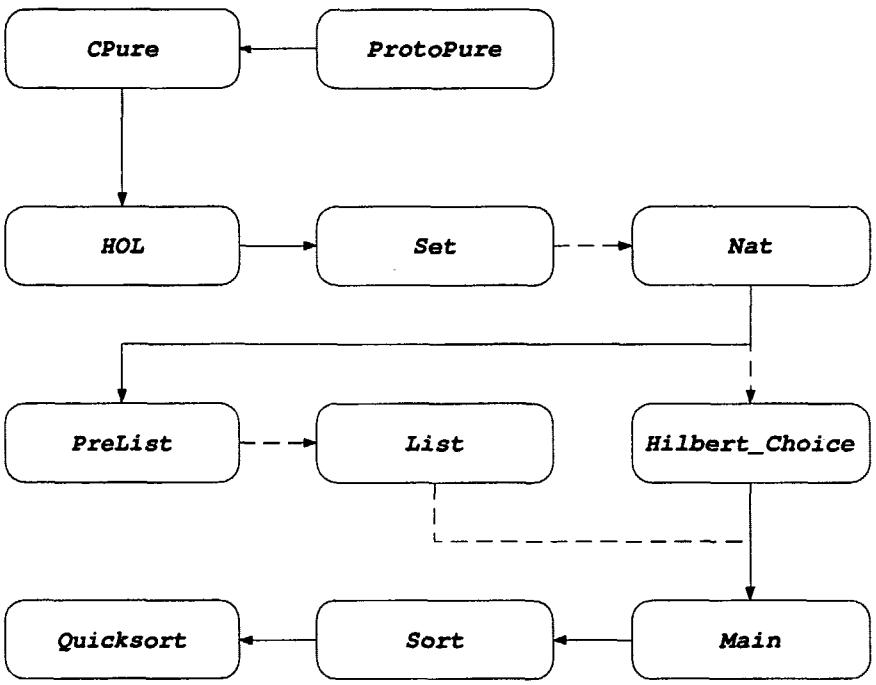
12.1.3 *Isabelles* logo

Isabelles logo, som er vist på forsiden af rapporten, udtrykker nogle af ideerne bag *Isabelle*. Logoet udtrykker, at *Isabelle* er opbygget af små velforståede 'byggeklodser', som er arrangeret i grupper og lag. Tilsammen udgør byggeklodserne *Isabelles* metalogik. De blå byggeklodser (λ, $=$, β) symboliserer en typet λ-kalkyle med β-konversion, de røde (α, $\rightarrow$) symboliserer type-systemet med typevariable (α) samt funktionsrum $\rightarrow$, og de gule byggeklodser ($\forall$, $\vdash$) symboliserer de egentlige logiske dele, det vil sige metaimplikation og universel kvantificering på metaniveau.

12.2 Baggrundsteorien for Quicksort

Isabelle indeholder et bibliotek af teorier, som angives i et træ, hvor hver knude repæsenterer en teori. Enhver knude i træet afhænger af de teorier som findes på stien fra den pågældende knude og op til træets rod. På *Isabelles* hjemmeside findes en oversigt over teoriafhængigheden af alle teorierne i *HOL*, samt en række eksempler, der gør brug af teorierne.

I beviset for korrektheden af algoritmen Quicksort trækkes på forskellige teorier i *Isabelle*. Vi har i figur 12.2 gengivet de væsentligste af disse samt sammenhængen imellem dem. De teorier, der er vist på figuren er valgt, fordi de indeholder vigtige definitioner eller aksiomer, som vi ønsker at præsentere.



Figur 12.2 Et udsnit af *Isabelles* træ over teoriafhængigheder. *ProtoPure* udgør roden af træet, *Quicksort* et blad og alt det mellemliggende er knuder i træet. En fuld optrukket linie imellem to knuder indikerer, at disse kommer i umiddelbar forlængelse af hinanden i træet. En punkteret linie indikerer, at der er klippet knuder (teorier) ud af *Isabelles* oprindelige træ.

Som tidligere nævnt er *HOL* grundsystemet for teorien. I *HOL* fastlægges de mest overordnede egenskaber ved teorien gennem en række aksiomer, som vi kigger på i næste afsnit. Efter *HOL* følger *Set*, som indeholder aksiomer og definitioner for typet mængdelære. Derefter følger *Nat*, som indeholder en teori for de naturlige tal. *Hilbert_Choice* indeholder blandt andet udvalgsaksiomet. De resterende lag i teorien indeholder kun definitioner, ingen aksiomer. *Prelist* og *List* indeholder definitioner, der er nødvendige for behandlingen af lister. Teorien *Main* er en stor teori, i hvilken alle forgreninger fra *HOL* ender. *Sort* og *Quicksort* indeholder så de definitioner, der tillader, at vil kan behandle sortering af lister herunder sortering med Quicksort.

I næste afsnit præsenteres aksiomerne for *HOL*, *Set*, *Nat* og *Hilbert_Choice*.

12.2.1 Aksiomer i *HOL*

Nedenstående er aksiomerne for *HOL*:

```
axioms
 eq_reflection: "(x=y) ⟹ (x≡y)"

 refl:          "t = (t::'a)"
 subst:         "⟦ s = t; P(s) ⟧ ⟹ P(t::'a)"

 ext:           "(⋀x::'a. (f x ::'b) = g x) ⟹
                 (λx. f x) = (λx. g x)"

 the_eq_trivial: "(THE x. x = a) = (a::'a)"

 impI:          "(P ⟹ Q) ⟹ P⟶Q"
 mp:            "⟦ P⟶Q;  P ⟧ ⟹ Q"

axioms
 iff:           "(P⟶Q) ⟶ (Q⟶P) ⟶ (P=Q)"
 True_or_False:  "(P=True) ∨ (P=False)"
```

Aksiomerne gennemgår vi overordnet her. Senere hen vil vi forsøge at relatere aksiomerne til typeteorien i kapitel 8. Alle de aksiomer, som indeholder symbolet $\Rightarrow$ er slutningsregler for objektlogikken, der er formaliseret i metalogikken. I aksiomerne `subst` og `mp` forekommer symbolet $[\![\ ]\!]$. Dette angiver en antagelse i objektlogikken. De formaliserede slutningsregler er følgende:

`eq_reflection`	er en formalisering af	$\dfrac{x = y}{x \equiv y}$
`subst`	er en formalisering af	$\dfrac{s = t \quad P(s)}{P(t)}$
`ImpI`	er en formalisering af	$\dfrac{\begin{array}{c}[P] \\ Q\end{array}}{P \to Q}$
`mp`	er en formalisering af	$\dfrac{P \to Q \quad P}{Q}$.

Ekstensionalitetsaksiomet kaldet `ext` kræver en forklaring. Symbolet $\bigwedge$ bliver som tidligere nævnt brugt til at formalisere aksiomsskemaer, på den måde, at x skal opfattes som en syntaktisk variabel. Aksiomet bliver da en formalisering af

$$\frac{f(x) = g(x)}{f = g}.$$

Detaljerne omkring dette aksiom vil vi udelade.

Aksiomet `refl` udtrykker refleksivitet af lighed for vilkårlige typer. I aksiomet `the_eq_trivial` optræder symbolet `THE`, som er en udvælgelsesoperator. Aksiomet fortæller, at når operatoren udvælger et `x` ved brug af udtrykket `x=a`, så vælger den `a`.

De to sidste aksiomer er ikke svære at forstå, det første udtrykker hvornår to udsagn er ækvivalente og det andet er det velkendte aksiom om den ekskluderede midte[2], fra hvilket vi opnår klassisk logik.

12.2.2 Aksiomer i *Set*

Her følger aksiomerne for *Set*:

axioms

```
mem_Collect_eq [iff]:  "(a∈{x. P(x)}) = P(a)"
Collect_mem_eq [simp]: "{x. x∈A} = A"
```

Det første aksiom udtrykker at udsagnene `a∈ {x:P(x)}` og `P(a)` er ækvivalente, det vil sige, at `a` tilhører mængden `{x:P(x)}` hvis og kun hvis, der gælder `P(a)`.

Det andet aksiom siger, at mængden af de elementer, der tilhører `A`, er lig `A`.

12.2.3 Aksiomer i *Nat*

Aksiomerne for *Nat* er:

axioms

```
inj_Suc_Rep:          "inj Suc_Rep"
Suc_Rep_not_Zero_Rep: "Suc_Rep x ≠ Zero_Rep"
```

Det første aksiom fortæller, at funktionen `Suc_Rep` er injektiv. `Suc_Rep` benyttes i *Nat* til at definere efterfølgerfunktionen `Suc` for de naturlige tal. De tekniske detaljer i den forbindelse vil vi ikke komme ind på her.

`Zero_Rep` benyttes til at definere det naturlige tal nul. Hvis man opfatter `Zero_Rep` og `Suc_Rep` som repræsentanter for det naturlige tal nul og efterfølgerfunktionen, udtrykker aksiomet, at efterfølgeren til et naturligt tal er forskelligt fra nul.

De to aksiomer sikrer tilsammen uendeligheden af de naturlige tal.

12.2.4 Aksiomer i *Hilbert_Choice*

Det sidste aksiom, der skal præsenteres, er følgende:

[2] På engelsk: *Excluded middle.*

axioms

```
someI: "P (x::'a) ⟹ P (ε x. P x)"
```

Dette aksiom er udvalgsaksiomet formuleret ved hjælp af Hilberts ε-operator. Udvalgsaksiomet er ikke en del af $\mathcal{Q}_0^\infty$ og vi har derfor ikke tidligere beskæftiget os med Hilberts ε-operator. Af samme grund vil vi ikke gå i detaljer med dette aksiom.

I stedet vil vi gå videre til næste kapitel. I dette kapitel findes en præsentation af *Isabelles* bevis for korrektheden af Quicksort samt en beskrivelse af, hvordan *Isabelle* benyttes.

13 Præsentation af bevis

I dette kapitel præsenteres og kommenteres *Isabelles* bevis for korrektheden af sorteringsalgoritmen Quicksort. Beviset er en lettere modificeret udgave af den version, som er tilgængelig på *Isabelles* hjemmeside. Først en gennemgang af, hvordan *Isabelle* benyttes i praksis.

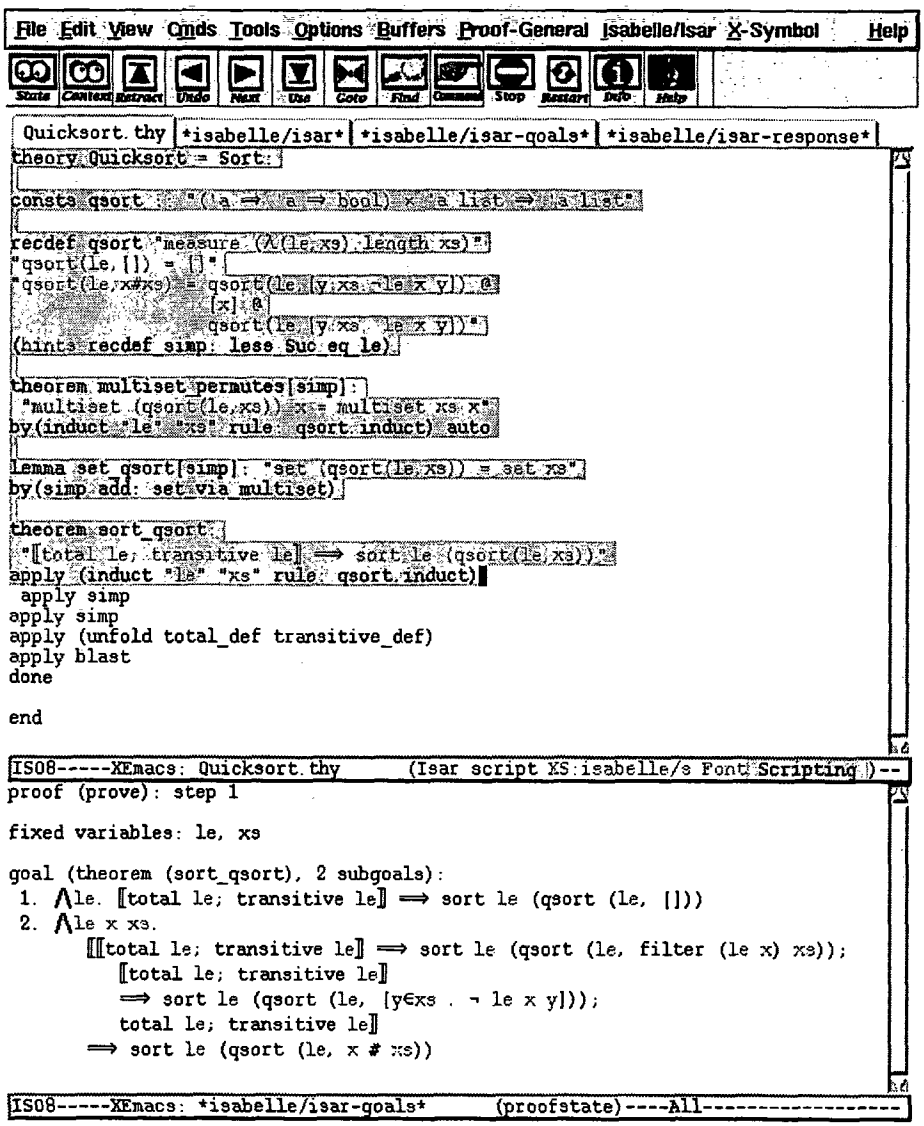
13.1 Brug af *Isabelle*

Ved bevisførelse i *Isabelle* kan man arbejde i den brugervenlige grænseflade *Proof General* [1].

13.1.1 Bevisførelse ved hjælp af *Proof General*

I *Proof General* deles vinduet typisk op i to dele, hvor koden for beviset skrives i det ene, og hvor man i det andet kan følge med i *Isabelles* respons. Bevisføreren aktiveres ved et enkelt klik på en knap, enten for at køre beviset i sin helhed, eller for at køre enkelte dele af beviset. Den kode som *Isabelle* læser i hvert skridt markeres ved highlighting, hvorefter det ikke længere er muligt at ændre i koden. Man følger med i systemets respons i et separat vindue nedenfor. Systemet oplyser i 'goals' hvilke variable, det har genkendt, hvilket teorem eller lemma, der skal bevises, samt hvilke 'subgoals', der kræves for at dette kan løses. I 'trace' kan man se hvilke regler, systemet har anvendt i sidste skridt, og i 'response' finder man eventuelle fejlmeldinger. Herudover kan systemet finde de definerede variable og metoder frem i 'find', hvorved man kan se hvilke oplysninger systemet indeholder om disse. Et screenshot af *Isabelle* kan ses i figur 13.1.

[1] *Proof General* anvendes som tidligere nævnt også af andre bevisførere.

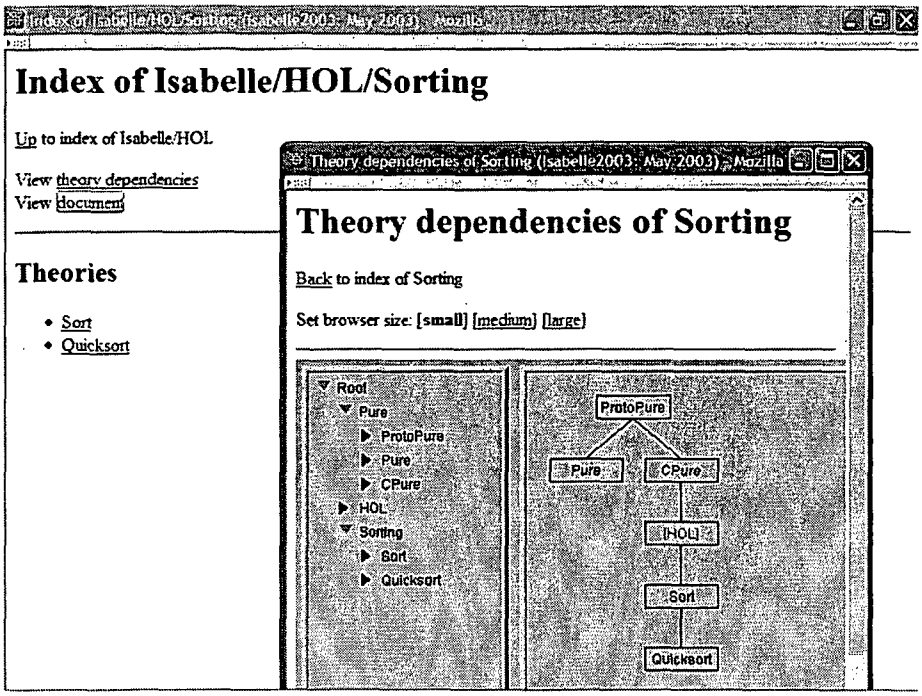


Figur 13.1 Screenshot af brugergrænsefladen for *Isabelle* i *Proof General*. Øverst er koden for `Quicksort`, hvor systemet har verificeret den markerede del. Nederst er systemets 'goals' for et af skridtene i beviset for det sidste teorem.

13.1.2 Teoriafhængigheder i *Isabelle*

Efter at have kørt sit bevis (sin teori) i *Proof General*, indgår teorien i det træ af teoriafhængigheder, som allerede eksisterer i *Isabelle*. Dele af dette træ kan ses i figur 12.2 i kapitel 12. Her fremgår det blandt andet hvilken teori (eller teorier), det kørte bevis afhænger af.

Tages for eksempel beviset for Quicksort, starter beviset med **`theory`** `Qsort`



Figur 13.2 Et screenshot af vores teoriers bibliotek. Det lille vindue viser træet over teoriafhængigheder.

= `Sort`. Her er det `Sort` som `Quicksort` afhænger af, og ved et klik på `Sort` føres man over i selve teorien for `Sort`, hvor der øverst står **`theory`** `Sort = Main` (det er teorien `Main`, som `Sort` afhænger af). På denne måde kan man som bruger bygge sit eget træ op i *Isabelle*. Dette indeholder alle de beviser eller teorier, man selv definerer, samt deres relationer til de allerede definerede teorier i *Isabelles* bibliotek.

Biblioteket for vores eksempel, `Quicksort`, og den teori, den afhænger af, det vil sige `Sort`, kan ses på figur 13.2. Man kan vælge at udvide sit bibliotek, ved for eksempel at tilføje andre sorteringsalgoritmer som Mergesort eller Insertionsort, som også kan afhænge af `Sort`.

13.2 Bevis for Quicksort

Beviset for Quicksort præsenteres her i *Isar*-sproget[2]. *Isar* er et bevissprog som anvendes af *Isabelle* og andre bevisførere, hvori beviser struktureres i et elegant formelt sprog, og hvor beviserne kan læses af såvel maskiner som mennesker. En af *Isar*-sprogets store fordele er, at det er generisk, hvorfor kun enkelte af konstruktionerne i sproget gælder specifikt for *HOL*, mens de fleste kan anvendes for enhver logik defineret i *Isabelle*, eksempelvis *ZF*. Beviset for Quicksort bygger på en række definitioner og lemmaer, som findes i `Sort`. Koden, som her præsenteres, er en lettere modificeret udgave af teorien `Sorting` og beviset for `Qsort` fra *Isabelles* eget bibliotek.

Nedenstående forklaringer af koden er baseret på [Nipkow et al., 2002].

13.2.1 `Sort`

Her præsenteres indholdet af `Sort`. Strategien, vi vil anvende, er at vise et par linier ad gangen og derefter forklare dem. I præsentationen vil vi lægge ud med at give meget grundige forklaringer. Senere vil linier, som bygger på notation eller ideer, der allerede er forklaret, blive gennemgået mere overfladisk.

Først defineres det teoretiske grundlag for `Sort`.

```
theory Sort = Main:
```

Kommandoen **theory** fortæller *Isabelle*, at der nu defineres hvilket system, der skal tjene som grundlag for det efterfølgende. Linien udtrykker, at `Sort` bygger på den teori, der er opstillet i `Main`.

```
consts
  sort :: "('a ⇒ 'a ⇒ bool) ⇒ 'a list ⇒ bool"
  multiset :: "'a list ⇒ 'a ⇒ nat"
```

const angiver, at der introduceres en konstant (som i dette tilfælde er en funktion, hvilket er muligt i højereordens-logik). De to sidste linier er så typeerklæringerne for de to konstanter `sort` og `multiset`. Vi starter med først at beskrive nogle af symbolerne. `::` betyder, at der gives en type. Højresiden af `::` er sat i citationstegn, hvilket betyder, at udtrykket er i *HOL*-specifik syntaks. Symbolet `'a` angiver en typevariabel, altså en vilkårlig type. Udtrykket `('a ⇒ 'a ⇒ bool)` angiver en type, som hører til en funktion, der sender to værdier af typen `a'` ind i en `bool`. Udtrykket i parentesen er typen for en komparator. Udtrykket `'a list` angiver typen for en liste med elementer af typen `'a`. Samlet er linien altså en typeerklæring for en funktion, `sort`, som sender en komparator og en liste ind i en sandhedsværdi. På tilsvarende vis er

[2] *Isar* står for: Intelligible semi-automated reasoning.

`multiset` en funktion, der sender en liste med elementer af typen `a'` ind i en funktion, der afbilder elementer af typen `a'` i et naturligt tal (`nat`).

Allerede på nuværende tidspunkt kan der altså ikke være tvivl om, at *Isabelle* anvender typeteori. Hvordan typeteorien indgår vil vi først diskutere i rapportens undersøgelse (kapitel 14).

primrec

```
  "sort le [] = True"
  "sort le (x#xs) = ((∀y∈set xs. le x y) ∧ sort le xs)"
```

Funktionen `sort` fortæller om en liste er sorteret i overensstemmelse med en given komparator. Kommandoen **`primrec`** angiver, at der gives en definition af en primitiv rekursiv funktion, i hvilken terminering er garanteret. Som vi så ovenfor, kunne `sort` opfattes som en funktion med to argumenter; en komparator og en liste. Linien efter **`primrec`** definerer altså, at `sort` virkende på komparatoren `le` og listen `[]` (den tomme liste) giver `True`. Sidste linie kræver lidt mere forklaring. Udtrykket `(x#xs)` angiver dekomposition af en liste, sådan at listen opdeles i et hoved bestående af elementet `x` og en hale bestående af listen `xs`. Udtrykket `∀y∈set xs.` skal læses: For alle `y` tilhørende mængden `set xs` gælder ... Udtrykket `le x y` betyder at `x` og `y` står i relation til hinanden, som defineret ved komparatoren (eksempelvis $x \leq y$). Samlet betyder sidste linie, at `sort` virkende på komparatoren `le` og listen `(x#xs)` returnerer sandhedsværdien for udtrykket 'alle elementer i listen `xs` står i relation til `x` som defineret af komparatoren og (∧) `sort le xs`'. Hvis vi vælger $\leq$ som komparator er `sort` altså funktionen, der returnerer `True`, hvis der for hvert element i listen gælder, at elementet er mindre end eller lig med alle efterfølgende elementer.

primrec

```
  "multiset [] y = 0"
  "multiset (x#xs) y = (if x=y then Suc(multiset xs y)
else multiset xs y)"
```

`multiset` er en funktionen, der tæller antallet af forekomster af et givent element i en liste. Funktionen `multiset` virkende på den tomme liste og en variabel `y` af type `'a` returnerer 0 (nul). Sidste linie i definitionen af `multiset` fortæller overordnet set, at `multiset` sammenligner elementet `y` med det første element i listen `(x#xs)` og 'noterer' om elementet er lig `y`. Derefter smides det første element, `x`, væk og samme operation foretages på den nye delliste, `xs`. Kort fortalt tæller `multiset`, hvor mange gange elementet `y` forekommer i listen `(x#xs)` og returnerer tilsidst dette antal. Dette antal, for eksempel tallet k, bliver returneret som

$$\overbrace{\text{Suc}(\text{Suc}(\ldots(\text{Suc}(}^{k}\texttt{multiset [])).$$

Dette er jo lig k (Suc angiver efterfølgerfunktionen).

```
constdefs
  total :: "('a ⇒ 'a ⇒ bool) ⇒ bool"
   "total r ≡ ∀ x y. r x y ∨ r y x"

  transitive :: "('a ⇒ 'a ⇒ bool) ⇒ bool"
   "transitive r ≡ ∀ x y z. r x y ∧ r y z ⟶ r x z"
```

Nøgleordet **constdefs** er en kombination af **consts** og **defs**. **defs** angiver, at der nu gives en definition. Hele udtrykket betyder, at der gives en definition af en konstant. Først defineres konstanten `total`, som ifølge dens typedefinition afbilder en komparator i en sandhedsværdi. `total` afbilder en komparator `r` i sandhedsværdien for udtrykket 'for alle `x,y` gælder `r x y` eller `r y x`'. For eksemplet med $\leq$ kan dette formuleres

$$\forall x, y : x \leq y \text{ eller } y \leq x.$$

`total` undersøger altså om en komparator er defineret overalt (om den er total).

Tilsvarende defineres konstanten `transitive`, som undersøger om en komparator er transitiv. Funktionen returnerer `True` for en komparator `r`, hvis `r` er transitiv, altså hvis `r x y` og `r y z` medfører `r x z` for alle `x`, `y`, `z`.

```
lemma multiset_append[simp]:
 "multiset (xs@ys) x = multiset xs x + multiset ys x"
by (induct "xs") auto
```

Første linie angiver med kommandoen **lemma**, at der nu introduceres et lemma, som skal hedde `multiset_append`, og at lemmaet efterfølgende kan anvendes som simplificeringsregel. Linie to udtrykker, at værdien `multiset` virkende på en sammenføjning (angivet ved `@`) af to lister, `xs` og `ys` er lig summen af `multiset xs` og `multiset ys`. I sidste linie figurerer kommandoen **by**, som fortæller *Isabelle*, at lemmaet skal bevises på en særlig måde – nemlig ved induktion over `xs` efterfulgt af noget automatik.

```
lemma multiset_compl_add[simp]:
 "multiset [x∈xs.¬p x] y + multiset [x∈xs. p x] y =
multiset xs y"
by (induct "xs") auto
```

Dette lemma kaldes `multiset_compl_add` og lemmaet kan igen anvendes som simplificeringsregel. I lemmaet deles listen `xs` op i to dele: En del, som opfylder `¬p x` og en del, som opfylder `p x`. Lemmaet siger, at for et vilkårligt element `y` er `multiset` virkende på listen `xs` lig summen af `multiset` virkende på de to dele.

```
theorem set_via_multiset:
 "set xs = {x. multiset xs x ≠ 0}"
by (induct "xs") auto
```

Teoremet udtrykker, at listen `xs` opfattet som mængde er lig den mængde, som for alle `xs` opfylder, at `multiset x` er forskellig fra nul, det vil sige, at `x` forekommer i listen et antal gange forskelligt fra nul. Teoremet er i modsætning til de tidligere lemmaer ikke en simplificeringsregel. Om man bruger teorem eller lemma betyder ikke så meget her. Forskellen er af en mere teknisk karakter, som vi ikke har sat os nærmere ind i.

```
lemma sort_append[simp]:
 "sort le (xs@ys) = (sort le xs ∧ sort le ys ∧
                     (∀x ∈ set xs. ∀y ∈ set ys. le x y))"
by (induct "xs") auto

end
```

Det sidste lemma i `Sort` udtrykker, at sammenføjningen af en liste er sorteret, hvis og kun hvis den første del er sorteret, den anden del er sorteret, og der for alle elementer i første del og alle elementer i anden del gælder, at elementet i første del står i relation til elementet i anden del som defineret af den givne komparator. Kommandoen **end** angiver tilsidst, at vi nu er færdige med udviklingen af teorien `Sort`. Vi vil derfor kaste os over `Quicksort`.

13.2.2 Quicksort

Beviset for korrektheden af en sorteringalgoritme består som tidligere nævnt af tre dele: (1) Bevis, at algoritmen terminerer, (2) Bevis, at den ikke smider elementer væk under sorteringen (3) Bevis, at algoritmen rent faktisk sorterer.

Her kommer beviset så.

```
theory Quicksort = Sort:
```

`Quicksort` bygger altså på teorien `Sort`, som selv er afhængig af teorien `Main` (se figur 12.2 over teoriafhængigheder og nærmere beskrivelse i kapitel 12).

```
consts qsort :: "('a ⇒ 'a ⇒ bool) × 'a list  ⇒ 'a list"
recdef qsort "measure (λ(le,xs).length xs)"
"qsort(le,[])    = []"
"qsort(le,x#xs) = qsort(le,[y:xs.¬le x y]) @
                  [x] @
                  qsort(le,[y:xs. le x y])"
(hints recdef_simp: less_Suc_eq_le)
```

I første linie fastlægges typen for funktionen `qsort`. I typespecificeringen indgår symbolet `×`, som angiver et kartesisk produkt. Det kartesiske produkt skyldes en teknikalitet i **recdef**, som vi ikke vil komme nærmere ind på her. I beviset er det tilstrækkeligt, at opfatte udtrykket `('a ⇒ 'a ⇒ bool) × 'a list`, som et ordnet par, hvor førstekomposanten er en komparator, og andenkomposanten er en liste af elementer med typen `'a`. Funktionen `qsort` afbilder da et sådant ordnet par i en ny liste med elementer af samme type.

recdef `qsort` definerer en rekursiv funktion kaldet `qsort`, der er total, det vil sige, at den er defineret overalt. Definitionen efterfølges af en `measure`-funktion `λ(le,xs).length xs`, som tager det ordnede par `(le,xs)`, hvor `le` er en komparator, og `xs` er en liste, og sender andenkomposanten over i længden[3] af andenkomposanten; `length.xs`. λ angiver her en λ-abstraktion. *Isabelle* anvender `measure`-funktionen til at vise at `qsort` terminerer, dette uddybes om et øjeblik.

De to næste linier er en rekursiv definition af `qsort`. Den første linie siger at `qsort` virkende på den tomme liste, `[]`, er lig den tomme liste.

Anden linie af den rekursive definition fortæller, at `qsort` virkende på listen `(x#xs)` er lig sammenføjningen af `qsort` virkende på henholdsvis den liste af elementer `y` for hvilke `¬le x y`, listen bestående af `[x]` og listen bestående af elementerne `y` således at `le x y`, hvor `le` er komparatoren, der er givet ved førstekomposanten af det ordnede par. Hvis vi igen anvender eksemplet hvor `le` er $\leq$, gør `qsort` følgende: Listen `(x#xs)` opdeles ved hjælp af pivot-elementet `x` (listens første element) i tre dele. En del hvor elementerne `y` i `xs` opfylder `x` $\leq$ `y`, en del som kun består af pivot elementet, og en del, som består af de elementer `y` i `xs`, som opfylder `¬(x` $\leq$ `y)`, det vil sige `x` $>$ `y`. Disse tre lister 'sorteres' derefter.

Ovenstående definition stemmer fint overens med den tidligere gennemgang af Quicksort i kapitel 11. Bemærk dog, at den her præsenterede formulering af Quicksort er langt mere elegant og komprimeret end den i kapitel 11 præsenterede imperative formulering. Dette skyldes, at der i *HOL* anvendes funktionsprogrammering.

Når en funktion er defineret via **recdef** forsøger *Isabelle* automatisk at bevise at den terminerer ved hjælp af `measure`-funktionen. Der føres altså kontrol med om algoritmen `qsort` terminerer. Dette gøres ved at undersøge om 'målet' af argumentet på venstresiden af = er større end 'målet' for hvert enkelt argument på højresiden. Da listen består af endeligt mange elementer sikrer rekursiviteten at algoritmen (funktionen) terminerer. *Isabelle* kan automatisk bevise

[3] `length.xs` er defineret i teorien *List*. `length.xs` angiver antallet af elementer i listen `xs`.

terminering for mange funktioner, men det går ikke så nemt for Quicksort. For at hjælpe *Isabelle* gives derfor et vink, **hints**, i form af simplificeringsreglen `less_Suc_eq_le`. Denne siger at

$$(m < \text{Suc } n) = (m \leq n),$$

hvor Suc n angiver efterfølgeren til n. `recdef_simp` angiver at `less_Suc_eq_le` tilføjes som en midlertidig simplificeringsregel.

Nu er Quicksort-algoritmen, kaldet `qsort`, defineret, og termineringen af algoritmen er sikret. Næste skridt i beviset for algoritmens korrekthed er at vise, at der ikke mistes elementer undervejs i sorteringsproceduren.

```
theorem multiset_qsort[simp]:
 "multiset (qsort(le,xs)) x = multiset xs x"
by(induct "le" "xs" rule: qsort.induct) auto

lemma set_qsort[simp]: "set (qsort(le,xs)) = set xs"
by(simp add: set_via_multiset)
```

Her opstilles et teorem kaldet `multiset_qsort`, som skal anvendes som simplificeringsregel. `qsort(le,xs)` er en liste, så teoremet udtrykker, at der for en vilkårlig variabel `x` gælder, at antallet af forekomster af `x` i en sorteret liste, `qsort(le,xs)`, er lig antallet af forekomster i den tilsvarende usorterede liste `xs`.

Lemmaet under teoremet kræver en forklaring. `set` er en funktion fra teorien *List*, der afbilder en liste i en mængde. Lemmaet under teoremet fortæller altså, at den mængde, der fremkommer, når en sorteret liste afbildes i en mængde ved funktionen `set`, er lig den tilhørende usorterede listes billede under `set`. Lemmaet kan bruges som en simplificeringsregel.

Således har vi nu fået sikret at algoritmen ikke mister elementer undervejs i sorteringsproceduren. Det sidste skridt i beviset for Quicksorts korrekthed er at bevise at algoritmen rent faktisk returnerer en sorteret liste. For at vise dette, skal vi igen have `Sort`-biblioteket i sving.

```
theorem sort_qsort:
 "⟦total le; transitive le⟧ ⟹ sort le (qsort(le,xs))"
apply (induct "le" "xs" rule: qsort.induct)
 apply simp
apply simp
apply (unfold total_def transitive_def)
apply blast
done
end
```

Det sidste teorem er hovedresultatet i beviset. Givet en liste `xs` og en komparator `le`, så gælder, at hvis komparatoren er total og transitiv (se `sort`) kan

man etablere `sort le qsort(le,xs)`. Dette sidste resultat vil sige at listen `qsort(le,xs)` er sorteret (se `sort`). I det første **apply** fortæller *Isabelle*, at teoremet skal bevises ved induktion. Herefter tilføres to ikke-specificerede simplificeringer, der virker på forskellige delmål. *Isabelle* vælger selv de passende regler, der skal anvendes. Metoden `unfold` henter to definitioner, som skal bruges i beviset; nemlig dem om totalitet og transitivitet fra `Sort`. Til sidst anvendes bevismetoden `blast`, som er et slagkraftigt værktøj i ikke-trivielle beviser. Vi vil ikke komme nærmere ind på, hvordan de to sidstnævnte metoder, `unfold` og `blast`, virker.

De tre tidligere nævnte betingelser er nu bevist, og dermed er korrektheden af sorteringsalgoritmen Quicksort ligeledes bevist.

I næste kapitel vil vi forsøge at analysere de aspekter af beviserne, som relaterer til de begreber, vi er stødt på i den historiske gennemgang af matematikkens formalisering.

14 Undersøgelse af aksiomer og bevis

I dette kapitel vil vi undersøge beviset for Quicksort, samt de aksiomer og definitioner, som beviset bygger på. Formålet med undersøgelsen er at skabe en forbindelse mellem rapportens to dele, og vise hvorledes *Isabelle* anvender de begreber og teorier, vi har behandlet tidligere i forbindelse med den historiske gennemgang af matematikkens formalisering. Vi vil forsøge at relatere beviset samt aksiomerne og definitionerne til de begreber, vi har præsenteret i rapportens del I. I den forbindelse vil der indgå generelle betragtninger omkring formelle systemer, typeteori, λ-kalkyle og så videre samt betragtninger, som er mere direkte rettet mod systemet $\mathcal{Q}_0^\infty$.

Allerede i gennemgangen af beviset for Quicksort blev det klart, at *Isabelle* gør brug af typer, så denne del af undersøgelsen er nemt overstået. Undersøgelsens omdrejningspunkt vil derfor være spørgsmålet om, *hvordan* typeteorien optræder i *Isabelle*.

Vi vil forsøge at dele undersøgelsen i to dele, dels en undersøgelse af de aksiomer og definitioner, der befinder sig i de teorier, der ligger til grund for Quicksort-beviset, med særligt fokus på *HOL*, og dels en undersøgelse af beviset for Quicksort. Undersøgelsen af definitionerne i *HOL* kan give os en forståelse af de helt grundlæggende ting, som *Isabelle* anvender i beviserne. Derudover kan vi få et lille glimt af metalogikken, som ellers er vanskelig at undersøge.

14.1 Definitioner og aksiomer

Som nævnt ovenfor ønsker vi i undersøgelsen at vise de ligheder, der er imellem *Isabelles* objektlogik og det system, vi opstillede i kapitel 8. En sammenligning af systemerne er imidlertid vanskelig at foretage fordi systemerne er opstillet på forskellig vis.

14.1.1 Hilbert-stil vs. naturlig deduktion

Systemet $\mathcal{Q}_0^\infty$ er som tidligere nævnt opstillet i Hilbert-stil. Ideen bag et sådant system er, at alle teoremer i systemet skal udledes fra aksiomerne. I et system i Hilbert-stil starter alle beviser altså fra bunden, nemlig ved aksiomerne. Den eneste slutningsregel for $\mathcal{Q}_0^\infty$ er **Regel R** (se kapitel 8). Denne slutningsregel er valgt fordi den er simpel, og fordi mere avancerede slutningsregler kan udledes fra den.

Isabelles objektlogik (og metalogik) anvender naturlig deduktion. I sådanne systemer etableres teoremer ikke på samme måde på baggrund af aksiomer, i stedet bevises teoremer på baggrund af antagelser. På den måde undgår man at skulle starte samme sted hver gang, man skal opstille et bevis. Slutningsreglerne for et system, der anvender naturlig deduktion er valgt for at muliggøre naturlig og effektiv ræsonnering, og der anvendes derfor typisk flere regler end i et system i Hilbert-stil.

En direkte sammenligning af *Isabelles* logikker og $\mathcal{Q}_0^\infty$ er altså vanskelig fordi systemerne er opstillet med forskellige formål. Vi vil alligevel forsøge at vise, at der *er* ligheder imellem *Isabelles* logikker og systemet $\mathcal{Q}_0^\infty$.

14.1.2 Definitioner i *HOL*

De aksiomer og definitioner, der findes i *HOL* er de mest grundlæggende og derfor dem, som vi bedst kan sammenligne med $\mathcal{Q}_0^\infty$. For at kunne diskutere aksiomerne er vi nødt til at kigge lidt nærmere på definitionerne i *HOL*.

I starten af *HOL* defineres i et afsnit kaldet 'Primitive Logic' en række grundlæggende symboler. Af disse har vi udvalgt de, der er interessante i forhold til sammenligningen af *HOL* og $\mathcal{Q}_0^\infty$.

Typeerklæringer og definitioner er gengivet i skemaet nedenfor.

consts

```
  True        :: bool
  All         :: "('a ⇒ bool) ⇒ bool"
  Ex          :: "('a ⇒ bool) ⇒ bool"
  False       :: bool
  Not         :: "bool ⇒ bool"
  ∧           :: "[bool, bool] ⇒ bool"
  ∨           :: "[bool, bool] ⇒ bool"
```

defs

```
  True_def:   "True      ≡ ((λx::bool. x) = (λx. x))"
  All_def:    "All(P)    ≡ (P = (λx. True))"
  Ex_def:     "Ex(P)     ≡ ∀Q. (∀x. P x ⟶ Q) ⟶ Q"
  False_def:  "False     ≡ (∀P. P)"
  not_def:    "¬P        ≡ P ⟶ False"
  and_def:    "P ∧ Q     ≡ ∀R. (P⟶Q⟶R) ⟶ R"
  or_def:     "P ∨ Q     ≡ ∀R. (P⟶R) ⟶ (Q⟶R) ⟶ R"
```

`True` og `False` er definitioner af 'det sande' henholdsvis 'det falske' udsagn. `All` og `Ex` betegner al-kvantoren og eksistens-kvantoren og `Not` er negation (her bruges forkortelsen ¬). ∧ og ∨ har deres sædvanlige betydning. Vi vil forsøge at opskrive disse definitioner med den notation, som blev introduceret i kapitel 8 og derefter diskutere definitionerne. Med den nævnte notation kan de to første

konstanter i det ovenstående skrives:

$$\begin{array}{rll} \texttt{True}_o & \text{kan skrives} & \lambda x_o.x_o = \lambda x_o.x_o \\ \texttt{All}_{(\alpha o)o}\texttt{P}_{\alpha o} & \text{kan skrives} & \lambda \mathbf{x}_\alpha.P_{\alpha o}x_\alpha = \lambda \mathbf{x}_\alpha True_o \end{array}$$

hvor vi har benyttet, at $P_{\alpha o} = \lambda x_\alpha.P_{\alpha o}x_\alpha$, som er en lovlig omskrivning i kraft af metalogikkens ekstensionalitet (se kapitel 12). Desuden har vi brugt typesymbolet o i stedet for *Isabelles* `bool` og α i stedet for `'a`.

Definitionen af `True` benytter ligesom definitionen af 'det sande udsagn' i $\mathcal{Q}_0^\infty$ et simpelt udtryk af formen $\mathbf{A} = \mathbf{A}$. Bemærk dog i denne sammenhæng, at ligheden her er mellem to størrelser med typen *oo*, mens ligheden i definitionen i $\mathcal{Q}_0^\infty$ var mellem størrelser af typen *ooo*.

Definitionen af al-kvantoren ligner umiddelbart ikke den definition, der bliver benyttet i $\mathcal{Q}_0^\infty$. Imidlertid kan sidstnævnte let omskrives til et udtryk, der næsten er identisk med *HOL*'s al-kvantor:

$$\begin{array}{rrl} \forall \mathbf{x}_\alpha \mathbf{P} & \text{står for} & \Pi_{(\alpha o)o}[\lambda \mathbf{x}_\alpha \mathbf{P}] \\ & \text{som omskrives til} & Q_{(\alpha o)(\alpha o)o}[\lambda \mathbf{x}_\alpha T][\lambda \mathbf{x}_\alpha \mathbf{P}] \\ & \text{som forkortes} & \lambda \mathbf{x}_\alpha T = \lambda \mathbf{x}_\alpha \mathbf{P}. \end{array}$$

Eksistens-kvantoren defineres ligesom i $\mathcal{Q}_0^\infty$ ved hjælp af al-kvantoren. Det falske udsagn er mere interessant. Falskhed er defineret ved udsagnet $\forall P.P$, der betyder 'alle udsagn er sande' fuldstændigt ligesom i $\mathcal{Q}_0^\infty$. Denne definition illustrerer klart at den typeteori, der anvendes i *HOL*, er den simple typeteori (vi husker fra kapitel 4, at $\forall P.P$ ikke er et tilladt udtryk i den ramificerede typeteori).

I modsætning til systemet $\mathcal{Q}_0^\infty$ benyttes $\longrightarrow$ som et primitivt begreb i *HOL*, og vi kan derfor umiddelbart ikke sammenligne de tre sidste konstanter, dog minder definitionen af negation, `¬ P ≡ P ⟶ False`, om definitionen fra $\mathcal{Q}_0^\infty$, hvor `¬ P` står for $Q_{ooo}F\,P$, som forkortes $F = P$.

Inden vi går videre til at betragte aksiomerne for *HOL*, er der yderligere en konstant vi ønsker at præsentere konstanten `The`.

const

```
The          :: "('a ⇒ bool) ⇒ 'a"
```

Konstanten danner grundlaget for at indføre en ny forkortelse, nemlig `THE x. P ≡ The λx. P`, som optræder i diskussionen af aksiomerne for *HOL*.

14.1.3 Aksiomer for *HOL*

Som vi tidligere har bemærket er en del af aksiomerne i *HOL* slutningsregler, der er formaliseret i *Isabelles* metalogik. Af disse er det kun aksiomet `ext` (ekstensionalitetsaksiomet), som er interessant i forhold til typeteorien og $\mathcal{Q}_0^\infty$. I ekstensionalitetsaksiomet kan vi se, at λ-kalkylen anvendes i metalogikken til

at behandle funktioner i objektlogikken. Aksiomet er en modificeret udgave af aksiom 3. fra $\mathcal{Q}_0^\infty$ (se kapitel 8).

Konstanten `THE`, som blev defineret i sidste afsnit optræder i aksiomet, der kaldes `the_eq_trivial`. Hvis vi opskriver aksiomet ved hjælp af konstanten `The` i stedet for `THE`, og anvender notationen fra kapitel 8 kommer det til at se således ud:

$$\texttt{the\_eq\_trivial:}\quad The_{(\alpha o)\alpha}[\lambda x_\alpha.[x_\alpha = a_\alpha]] = a_\alpha.$$

Notationen $\lambda x_\alpha.[x_\alpha = a_\alpha]$ kan i $\mathcal{Q}_0^\infty$ reduceres til $Q_{\alpha\alpha o}a_\alpha$, så aksiomet kan skrives som:

$$\texttt{the\_eq\_trivial:}\quad The_{(\alpha o)\alpha}[Q_{\alpha\alpha o}a_\alpha] = a_\alpha.$$

Dette aksiom svarer til *Deskriptionsaksiomet*[1]

$$\iota_{(\iota o)\iota}[Q_{\iota\iota o}y_\iota] = y_\iota,$$

som er det sidste aksiom for $\mathcal{Q}_0^\infty$. Vi har ikke gjort så meget ud af operatoren ι, og derfor vil vi her nøjes med at bemærke, at *HOL* muligvis har overtaget dette aksiom fra $\mathcal{Q}_0^\infty$.

Aksiomet `refl` svarer til det simple resultat $\mathbf{A}_\alpha = \mathbf{A}_\alpha$, som blev udledt i $\mathcal{Q}_0^\infty$ på side 68. Dette aksiom er ikke særligt overraskende eller svært at forstå, og vi vil derfor gå videre.

14.1.4 De resterende aksiomer

De resterende aksiomer spiller ikke nogen stor rolle i vores undersøgelse. Vi vil kort gennemgå dem her, og forklare, hvorfor de ikke er så interessante for os.

Set

De to aksiomer i *Set*, `mem_Collect_eq` og `Collect_mem_eq`, introducerer syntaks for mængdelære. Vi har ikke tidligere i rapporten beskæftiget os med mængdelære, og aksiomerne indeholder ingen eksplicitte referencer til λ-kalkyle eller typeteori, så vi vil ikke undersøge dem nærmere.

Nat

Aksiomerne i *Nat* udgør tilsammen et uendelighedsaksiom. Definitionen af de naturlige tal foregår ikke på samme måde i *HOL* som i $\mathcal{Q}_0^\infty$. Godt nok defineres et naturligt tal i *HOL* som en mængde af individer ligesom i $\mathcal{Q}_0^\infty$, men den tekniske håndtering af naturlige tal er noget anderledes, og vi har derfor valgt ikke at forfølge disse aksiomer nærmere. Dog skal det nævnes, at aksiomerne er udformet ligesom Peanos tredje og fjerde postulat, som vi har valgt at bringe her for sammenligningens skyld.

3. 0 er ikke efterfølgeren for noget naturligt tal.
4. Hvis n og m er naturlige tal med samme efterfølger, så er n og m det samme tal.

[1] *Axiom of Descriptions.*

Hilbert_Choice

Som tidligere nævnt har vi ikke noget grundlag for at behandle udvalgsaksiomet nærmere, fordi det ikke figurerer i $\mathcal{Q}_0^\infty$. Vi vil derfor blot nævne, at udvalgsaksiomet for *HOL* faktisk minder ret meget om udvalgsaksiomet for Churchs 1940-system. Dette er formuleret således:

$$f_{\alpha o}x_\alpha \Rightarrow f_{\alpha o}(\iota_{(\alpha o)\alpha}f_{\alpha o}).$$

Dette resultat vil vi kort vende tilbage til i rapportens diskussion. Nu vil vi gå videre til undersøgelsen af beviset for Quicksort.

14.2 Undersøgelse af `Sort` og `Quicksort`

Det system, vi præsenterede i kapitel 8, er opstillet og forklaret overordnet, men ikke udviklet i stort omfang i denne rapport. *Isabelle* indeholder en del begreber, som er for avancerede til at vi umiddelbart kan sammenligne dem med noget, der optræder i $\mathcal{Q}_0^\infty$. Som eksempel kan nævnes lister af naturlige tal, der konstrueres i *HOL*, som noget der hedder en *datatype*. Dette begreb har ikke været muligt for os at undersøge i løbet af projektperioden, og vi kan derfor ikke undersøge lister til bunds. Desværre indgår lister næsten i alle linierne i beviset for Quicksort, blandt andet i selve definitionen af konstanten `qsort`, så det er ikke muligt for os at oversætte beviset til notationen fra kapitel 8. Beviset er derfor meget vanskeligt at sammenligne med noget, der er i $\mathcal{Q}_0^\infty$, og vi må nøjes med mere generelt at pege på de steder i beviset hvor typeteorien optræder.

Teorien `Sort` og beviset for Quicksort (`Quicksort`) er ligesom aksiomerne fyldt med typeerklæringer. Hvis man betragter `Sort` – eksempelvis definitionen af konstanten `transitive` – er det klart, at typeteorien er vigtig i beviserne. Definitionen af `transitive` *kan* omskrives til den notation, der blev brugt i forbindelse med $\mathcal{Q}_0^\infty$ således:

$$transitive_{(\alpha\alpha o)o}r_{\alpha\alpha o} \equiv [\forall x_\alpha \forall y_\alpha \forall z_\alpha [r_{\alpha\alpha o}x_\alpha y_\alpha \wedge r_{\alpha\alpha o}y_\alpha z_\alpha] \Rightarrow r_{\alpha\alpha o}x_\alpha z_\alpha].$$

Den typeteori, der forekommer i *Isabelle* er altså ikke kun begravet dybt i *Isabelles* beviskerne eller i metalogikken, men også fremtrædende i selve beviserne. λ-kalkylen optræder også synligt i beviset for Quicksort, nemlig i forbindelse med definitionen af `measure`-funktionen.

Beviset for Quicksort indeholder en del begreber, som defineres andetsteds. Især hvis man betragter **apply**-kommandoerne i Quicksort-beviset får man en fornemmelse af, at beviset indeholder en del mere, end man umiddelbart kan se. Vores gennemgang af beviset må derfor opfattes som en overfladisk behandling.

14.3 Yderligere kommentarer

Tilstedeværelsen af $\mathcal{Q}_0^\infty$ i *HOL* og i *Isabelles* metalogik er klar hvis man undersøger dokumentationen for *Isabelle* og *HOL*. Vi har imidlertid syntes, at det var

mere interessant at lave vores egen undersøgelse af *Isabelle* og forsøge at genfinde $\mathcal{Q}_0^\infty$. Metalogikken har dog være svær at undersøge på denne måde, fordi vi ikke har adgang til den. Vi har derfor overfladisk læst en del af dokumentationen for *Isabelles* metalogik, og her er det værd at bemærke, at λ-kalkylen optræder hyppigt. Som vi tidligere har beskrevet gør metalogikken brug af de regler for λ-konversion, der blev nævnt i præsentationen af $\mathcal{Q}_0^\infty$. Denne observation vil vi vende tilbage til i rapportens diskussion.

Diskussionen befinder sig i kapitel 16. Først vil vi imidlertid forsøge at samle lidt op på de observationer, vi har gjort i rapportens del II.

15 Opsamling II

Vi bringer her en opsamling på rapportens del II.

Bevisførere kan deles i tre hovedgrupper: (1) beviskontrollører, (2) bevisgeneratorer og (3) bevisassistenter. *Isabelle* er et eksempel på en bevisassistent. *Isabelle* gør brug af en metalogik, som er en intuitionistisk højereordens-logik. Metalogikken gør det muligt at anvende forskellige objektlogikker, som formaliseres i metalogikken. Vi har valgt at beskæftige os med objektlogikken *HOL*, som også er en højereordens-logik. *Isabelle* kan også anvende eksempelvis *ZF*.

HOL gør brug af naturlig deduktion, som gør det muligt at arbejde i et formelt system og samtidig give naturlige beviser, som minder om dem, der anvendes i praksis af matematikere. *Isabelle* opfylder de Bruijns kriterie om, at en bevisfører skal have en lille overskuelig beviskerne. Samtidig opfyldes Poincarés princip, hvilket vil sige at *Isabelles* system kan bevise korrektheden af beregninger automatisk. Endelig har *Isabelle* et stort matematisk bibliotek, hvilket ofte letter arbejdet i forbindelse med mere komplicerede beviser.

Bevisførere kan benyttes til at bevise korrektheden af algoritmer, eksempelvis sorteringsalgoritmen Quicksort, som vi har set på i dette projekt. Beviset består i at vise (1) at algoritmen terminerer, (2) at algoritmen sorterer og (3) at algoritmen ikke mister elementer undervejs.

Isabelles metalogik er placeret i bibliotekerne *ProtoPure* og *CPure*. Oven på disse biblioteker ligger biblioteket *HOL*, som udgør roden i et større træ af teoriafhængigheder. Alle teorierne bygger på dette rodbibliotek. For at bevise korrektheden af Quicksort gør man brug af en teori kaldet *Sort*, som indeholder en række definitioner. Selve beviset bygger oven på en mængde aksiomer fra *HOL*, *Set*, *Nat* og *Hilbert_Choice*. Aksiomerne i *HOL* indeholder blandt andet slutningsreglerne for objektlogikken.

Typebegrebet er en meget synlig del af *Isabelle*. Typeerklæringer forekommer i metalogikken, i definitionerne og aksiomerne for *HOL*, samt i selve Quicksort-beviset. λ-kalkylen er ligeledes synlig både i metalogikken og i objektlogikken. I rapportens diskussion, som findes i næste kapitel, vil vi forsøge at diskutere disse forhold nærmere.

16 Diskussion

Dette kapitel indeholder projektrapportens diskussion. I diskussionen vil vi forsøge at holde rapportens to dele op mod hinanden. Den første del af diskussionen vil hovedsageligt indeholde overordnede betragtninger omkring sammenhængen imellem matematikkens formalisering og bevisførere. Anden del af diskussionen er mere direkte rettet mod *Isabelles* logik *HOL* og Andrews' system $\mathcal{Q}_0^\infty$. Til sidst i diskussionen vil vi beskrive projektets perspektiver.

16.1 Matematikkens formalisering

I denne første del vil vi diskutere den overordnede betydning de forskellige arbejder, vi har gennemgået, har haft for moderne bevisførere. Især vil vi forsøge at beskrive hvordan de ideer, som ydede indflydelse på matematikkens formalisering har ydet indflydelse på bevisførerne.

16.1.1 Frege

Freges betydning for bevisførere er svær at vurdere. Hans værk fra 1879, *Begriffsschrift*, er et banebrydende arbejde indenfor matematisk logik, som har haft stor indflydelse på den efterfølgende udvikling indenfor logik. Ideen om at opstille et formelt system, der gør brug af præcist definerede slutningsregler, er gennemgående for de arbejder, vi har behandlet i rapporten. Denne ide må tilskrives Frege. Derudover er Frege også ophavsmanden til den sandhedsfunktionelle propositionslogik, i hvilken prædikat og subjekt opfattes som funktion og argument. Opfindelsen af kvantorerne er ligeledes Freges fortjeneste.

Freges betydning for bevisførere skal derfor ses i lyset af hans store indflydelse på matematisk logik. Frege har ingen direkte indflydelse haft på bevisførerne, men han har haft en indirekte indflydelse, nemlig i kraft af hans udvikling af ideen om et formelt system, der gør brug af specifikke slutningsregler. Sådanne systemer, det vil sige systemer, der udleder resultater på 'mekanisk' vis, er grundlæggende for en hel række af bevisførere, heriblandt *Isabelle*.

16.1.2 Russell

Russells betydning for bevisførere skal findes dels i hans betydning for oprindelsen af typeteorien og dels i hans arbejde med at formalisere store dele af matematikken. Russell opdagede det paradoks i Freges system, som nu kaldes Russells paradoks, og hans forsøg på at udvikle et system fri for paradokser førte

til opfindelsen af den ramificerede typeteori. Typeteorien har, siden Russell opstillede og forklarede den første gang i 1908, gennemgået en større udvikling og er blevet til mange forskellige former for typeteori. Russells bidrag til disse teorier er hans grundlæggende ide om, at adskille objekter med forskellige egenskaber ved at tildele dem en type.

Whiteheads og Russells *Principia mathematica* må betegnes som et af de første forsøg på at udvikle mere avanceret matematik i et formelt system, for eksempel udvikler Whitehead og Russell dele af den matematiske analyse i *Principia mathematica*. Sådanne forsøg på at give formelle beviser for mere avanceret matematik overlades i dag i stor stil til bevisførere.

Typeteori anvendes i forskellige udgaver i adskillige bevisførere i dag. I *Isabelle* anvendes den simple typeteori, der som bekendt blev foreslået som en forbedring af den ramificerede typeteori allerede i starten af 1920'erne. Russells bidrag til de bevisførere, hvis logikker bygger på typeteori, må siges at være selve ideen bag typebegrebet.

16.1.3 Gödel

Gödels rolle i forhold til bevisførere er af en anden karakter end Russells. Gödels ufuldstændighedssætninger har tjent til at sætte formelle systemer, og derigennem bevisførere, i perspektiv. Sætningerne giver en ide om, hvad man kan og i særdeleshed, hvad man ikke kan i et formelt system. Resultaterne om at der i ethvert system af en vis styrke eksisterer sande udsagn, der ikke kan vises i systemet, og at sådanne systemer ikke er i stand til at vise deres egen konsistens, er helt grundlæggende for den forståelse af matematikkens formalisering, man har i dag. Resultaterne fra ufuldstændighedssætningerne var også med til at øge opmærksomheden på begreber, som er blevet fundamentale indenfor studiet af formelle systemer. Vi tænker her på fuldstændighed, konsistens, afgørlighed og sundhed.

16.1.4 Turing

Turings indflydelse på bevisførere er lidt af samme karakter som Gödels. Turings definition af beregnelighed, som byggede på Turingmaskinen, gjorde det muligt for ham at besvare Hilberts tredje spørgsmål – det såkaldte *Entscheidungsproblem* – med et nej. Turing viste, at der findes uberegnelige tal og dermed uafgørlige spørgsmål indenfor matematikken. Ligesom Gödels ufuldstændighedssætninger giver Turings resultat en ide om de begrænsninger, der gør sig gældende for de formelle systemer.

Turings besvarelse af afgørlighedsspørgsmålet var ikke den første (Church publicerede sin først), men det vigtige ved Turings resultat var at hans definition af Turingmaskinen og dermed af beregnelighed er nemmere, rent intuitivt, at sætte i forbindelse med computere end Churchs λ-kalkyle er. Turing viste iøvrigt, at hans definition af beregnelighed er ækvivalent med Churchs definition, og på den måde, at Turingmaskinen er ækvivalent med de rekursive funktioner.

16.1.5 Church

Churchs viste som den første eksistensen af et uafgørligt problem i matematikken, og på den måde har Church ligesom Gödel og Turing været med til at belyse nogle overordnede forhold, der gør sig gældende for formelle systemer. For denne rapport er det mest interessante i denne sammenhæng hans introduktion af λ-kalkylen. Church viste iøvrigt, at der kan sættes lighedstegn imellem de funktioner, der kan defineres i λ-kalkylen og de rekursive funktioner. Sidstnævnte er som vi tidligere har set et vigtigt begreb i forhold til matematik logik og bevisførerne (men også i forhold til teoretisk datalogi).

Churchs vigtigste bedrift i forhold til bevisførerne er imidlertid hans opstilling af 1940-systemet. Dette system er næsten identisk med Andrews system $\mathcal{Q}_0^\infty$. Den store lighed imellem de to systemer betyder, at Church er vores vigtigste grundlag for at spore *Isabelles* logikker tilbage til de personer, vi har mødt i rapportens del I.

Ligheder imellem $\mathcal{Q}_0^\infty$ og *Isabelles* logikker vil vi tolke som en kobling imellem sidstnævnte logikker og Churchs 1940-system. Disse ligheder vil vi med udgangspunkt i *HOL* diskutere i næste afsnit.

16.2 Ligheder imellem *HOL* og $\mathcal{Q}_0^\infty$

Som tidligere beskrevet er der forskelle mellem *HOL* og $\mathcal{Q}_0^\infty$, idet *HOL* anvender naturlig deduktion, mens $\mathcal{Q}_0^\infty$ er et system i Hilbert-stil. Slutningsreglerne er af den grund – som tidligere beskrevet – vanskelige at sammenligne. Vi vil derfor her forsøge at påpege de ligheder imellem systemerne, der kommer til udtryk i definitionerne og aksiomerne for *HOL*.

På et overordnet plan kan det observeres, at begge systemer bygger på en typet λ-kalkyle, ligesom Churchs system.

Definitionerne af 'det sande' og 'det falske' udsagn er næsten identiske for *HOL* og $\mathcal{Q}_0^\infty$. For det 'sande udsagn' benyttes i begge systemer et udtryk af formen $\mathbf{A} = \mathbf{A}$, mens 'det falske' udsagn i begge systemer repræsenteres af formler, der udtrykker, at alle propositioner er sande. Udtrykket, der anvendes i $\mathcal{Q}_0^\infty$ kan ved få omskrivninger vises at være det samme udtryk, som benyttes i *HOL*.

Definitionen af al-kvantoren er ligeledes ens i de to systemer. Igen kan man ved få omskrivninger vise, at de udtryk, der repræsenterer $\forall \mathbf{x}_\alpha \mathbf{A}$ er ens. Begge definitioner anvender λ-abstraktion til at definere al-kvantoren. Definitionerne af eksistens-kvantoren er imidlertid forskellige. Dette skyldes, at $\longrightarrow$ benyttes i *HOL* som et primitivt begreb, og eksistens-kvantoren kan derfor defineres ud fra den. Der er altså den forskel på *HOL* og $\mathcal{Q}_0^\infty$ at i sidstnævnte er lighed et primitivt begreb mens både lighed og $\longrightarrow$ er primitive begreber i førstnævnte. At $\longrightarrow$ benyttes som et primitivt begreb skyldes muligvis, at man ønsker en vis fleksibilitet i systemet i forhold til formalisering af intuitionistisk logik. Dette vil vi dog ikke forfølge nærmere. Symbolerne $\wedge$, $\vee$ og $\neg$ defineres ved hjælp af symbolet $\longrightarrow$.

Som vi viste i undersøgelsen er der også overensstemmelse mellem konstanten `The` fra *HOL* og udvælgelsesoperatoren $\iota_{(\iota o)_\iota}$ i den forstand, at deskriptionsak-

siomet er ens i de to systemer. Ekstensionalitetsaksiomet for *HOL* kan ligeledes genfindes i $\mathcal{Q}_0^\infty$ i en lidt modificeret udgave.

Med baggrund i ovenstående sammenligning mener vi at kunne slutte, at systemet *HOL* er en udgave af $\mathcal{Q}_0^\infty$, der er tilpasset, på en sådan måde, at det er nemmere at arbejde med i en bevisfører.

16.2.1 *Isabelles* metalogik

Isabelles metalogik har vi ikke haft den store mulighed for at undersøge, fordi der ikke er adgang til den på samme måde som til *HOL*. Fra dokumentationen af *Isabelle* og rapportens beskrivelse af metalogikken er det imidlertid klart, at denne, ligesom *HOL*, bygger på en typet λ-kalkyle.

16.3 Sporing af ideer

Det første og det sidste spørgsmål i projektrapportens problemformulering (se kapitel 1) mener vi, at have diskuteret i afsnittene 16.1 og 16.2. I dette afsnit vil vi diskutere det midterste spørgsmål med udgangspunkt i de to andre.

Vi har tidligere i diskussionen fremført, at vi på grund af de store ligheder imellem Churchs 1940-system og $\mathcal{Q}_0^\infty$ vil tolke ligheder imellem $\mathcal{Q}_0^\infty$ og *HOL* som en direkte kobling imellem *HOL* og Churchs system. På denne baggrund kan en stor del af *HOL* spores tilbage til Church.

Når det gælder den konkrete formulering af aksiomerne for *HOL* er der imidlertid ingen mulighed for, at spore aksiomerne længere tilbage end til Andrews' system $\mathcal{Q}_0^\infty$. Som vi har beskrevet, er der i høj grad overensstemmelse imellem de to systemers deskriptionsaksiomer. Men hvad dette aksiom angår afviger Andrews' system imidlertid fra Churchs 1940-system. Grunden til at de konkrete formuleringer ikke stemmer overens i systemerne er, at disse ikke bygger på de samme primitive begreber. Dette må siges også at gælde sporingen tilbage til de personer, vi har mødt i forbindelse med matematikkens formalisering.

I denne forbindelse vil vi lige nævne to ting, som i en vis forstand kan spores tilbage. Det første er udvalgsaksiomet for *HOL*, som ligner det tilsvarende aksiom i Churchs 1940-system. Om Church har lånt denne formulering andetsteds fra, mener vi ikke at have beskæftiget os nok med aksiomet til at kunne afgøre. Formuleringen stammer muligvis fra Hilbert. Den anden ting angår aksiomerne for `Nat`, som er formuleret, så de ligner Peanos postulater. Denne observation er dog svær at relatere til de personer, vi har mødt i rapportens del I, hvorfor vi ikke vil forfølge den længere.

Sporingen af *ideerne bag* aksiomerne for *HOL* er vanskelig, fordi aksiomerne i de forskellige systemer, vi har kigget på, skal udgøre et aksiomatisk grundlag for matematikken. Derfor er ideerne bag aksiomerne i stort omfang ens og ligeledes mulige at genfinde i alle systemerne. Som eksempel kan nævnes aksiomet `refl`, der udtrykker $t_\alpha = t_\alpha$. Denne *ide* kan genfindes overalt. Det samme gør sig gældende for eksempelvis aksiomet om den ekskluderede midte.

Vi vil nu afrunde diskussionen med et afsnit, der er af en lidt mere perspektiverende karakter.

16.4 Perspektiver

Da vi ikke tidligere har beskæftiget os med matematisk logik eller typeteori, har en stor del af vores arbejde med nærværende projekt bestået i at sætte os ind i og forstå typeteori på et niveau, så det var muligt for os at undersøge *Isabelles* logikker. *Isabelles* logik *HOL* består af såvel logik som funktionsprogrammering. Denne form for programmering har vi også først stiftet bekendtskab med i kraft af dette projekt.

Motiveret af en historisk interesse og en forestilling om at en indsigt i matematikkens formalisering kunne bidrage til en bedre forståelse af typeteorien, gik vi i gang. Vi var klar over, at det var nødvendigt også at arbejde med en mere moderne fremstilling af typeteorien, men havde en forventning om, at dette var nemmere med en vis portion historisk viden i kufferten. Efter at have skrevet rapporten færdig er vores opfattelse den samme.

Da vi i en vis forstand er startet på bar bund, har de bagvedliggende ideer for formelle systemer og typeteori gjort den moderne fremstilling af den simple typeteori nemmere for os. Samtidigt har den historiske indsigt forstærket vores opmærksomhed på typeteoriens relevans. Her tænker vi specielt i forhold til paradokserne og Russells løsning af de problemer, der var forbundet hermed. Behandlingen af begreber, som for eksempel ufuldstændighed og uafgørlighed, har hjulpet os til at forstå de begrænsninger som formelle systemer, og dermed bevisførere er underlagt, selvom begreberne ikke optræder direkte i forbindelse med beviset for Quicksort. Endelig har den matematikhistoriske tilgang gjort det muligt for os at tage udgangspunkt i noget, som vi på forhånd havde et forhold og vist kendskab til; matematikken. På den måde har projektet også været med til at udvikle vores forståelse af dette fag.

Oplagte spørgsmål af perspektiverende art, som man kan stille sig selv, når man er færdig med et projekt er, dels om der er noget, man kunne have grebet anderledes an og dels, hvad man ville have gået videre med, hvis man havde haft mere tid.

Havde vi haft vores nuværende kendskab til typeteori som udgangspunkt, er det klart, at vi kunne have dykket dybere ned adskillige steder i rapporten. Et oplagt område man kunne have behandlet mere indgående er det *datatype*-begreb, som optræder i *Isabelle*. Dette ville have gjort det muligt for os at få en bedre forståelse af lister. En sådan undersøgelse ville muligvis have gjort os i stand til at oversætte flere ting i beviset for Quicksort til noget, vi kunne have relateret til systemet $\mathcal{Q}_0^\infty$.

Vi har i dette projekt, som ovenfor beskrevet, valgt at lægge vægt på det historiske. Det har betydet en nedtoning af andre områder, for eksempel det filosofiske. Man kunne i et projekt som dette sagtens have gjort mere ud af at diskutere forholdet imellem formaliseret matematik og uformel matematik, og på den måde have draget filosofiske betragtninger ind i projektet. Man kunne eksempelvis have undersøgt Gödel, der var en dedikeret platonist, nærmere. Vi har dog fravalgt denne filosofiske diskussion til fordel for den historiske tilgang.

En anden mulighed kunne have været udelukkende at behandle den oplagt datalogiske del, det vil sige bevisførerne. Ved denne tilgang mener vi dog, at vigtige aspekter ved udviklingen af og tankerne bag bevisførerne, ville have gået tabt.

17 Konklusion

I rapporten har vi givet en gennemgang af matematikkens formalisering, herunder typeteoriens fremkomst og udvikling. Specielt har vi behandlet relevante arbejder af Frege, Russell, Gödel, Turing og Church. Vi har kort beskrevet den simple typeteori samt præsenteret det formelle system $\mathcal{Q}_0^\infty$. Til sidst har vi præsenteret beviset for korrektheden af sorteringsalgoritmen Quicksort i *Isabelles* logik *HOL* og diskuteret dette i forhold til ovenstående. I denne rapport konkluderer vi følgende.

Isabelles logik *HOL* bygger ligesom systemet $\mathcal{Q}_0^\infty$ på den simple typeteori. Derudover er en væsentlig del af de grundlæggende definitioner, der optræder i de to systemer, ens. De forskelle, der er på systemerne, skyldes hovedsageligt, at *HOL* anvender naturlig deduktion, mens $\mathcal{Q}_0^\infty$ er i Hilbert-stil. *HOL* må siges at være baseret på $\mathcal{Q}_0^\infty$.

De behandlede personer i gennemgangen af matematikkens formalisering opstiller alle systemer i Hilbert-stil. Da *HOL*, som ovenfor nævnt, anvender naturlig deduktion (ikke-Hilbert-stil), er det vanskeligt at spore aksiomerne og slutningsreglerne i *HOL* tilbage til disse. Ideerne bag aksiomerne er dog i en vis forstand, at aksiomerne tilsammen skal udgøre et fundament for matematikkens formalisering. Af denne grund kan ideerne på sin vis spores tilbage til mange af de behandlede systemer.

Matematikkens formalisering har bidraget til bevisførerne ved at tilføre disse en række overordnede ideer. Ideen bag de formelle systemer, der er helt grundlæggende for bevisførerne, tilskrives Frege. Flere basale ideer, der ofte optræder i sådanne systemer, herunder at prædikat og subjekt opfattes som funktion og argument samt brugen af kvantorer, kan også spores tilbage til Frege. Den grundlæggende ide bag typeteorien, som anvendes i adskillige bevisførere, stammer fra Russell. Gödel, Church og Turing har haft en betydning for bevisførerne, der er af en overordnet karakter. Gödels ufuldsstændighedssætninger samt Churchs og Turings svar på Hilberts afgørlighedsspørgsmål har haft betydning for forståelsen af bevisførere, da de udtaler sig generelt om formelle systemer, og dermed beskriver bevisførernes begrænsninger. Derudover er et af Churchs væsentligste bidrag hans udvikling af den simple typeteori med λ-kalkyle, som anvendes i flere af bevisførerne, heriblandt *Isabelle*.

18 Epilog

Vi har i denne projektrapport forsøgt at skitsere noget af vejen fra Leibniz' drøm og frem til moderne bevisførere som *Isabelle*. Det at bevise en algoritmes korrekthed formelt ved hjælp af bevisførere, må i et vist omfang stadig opfattes som en teoretisk leg – i den sidste ende handler det jo om, hvorvidt algoritmen kan implementeres, så den fungerer korrekt. Don Knuth, algoritmernes 'grand old man', har i 1977 i en korrespondance med kolleger skrevet følgende[1]:

> *Beware of bugs in the above code; I have only proved it correct, not tried it.*

Knuths her omtalte bevis er et uformelt bevis. Uformelle beviser er som regel langt fra de formelle, idet der i de uformelle ikke direkte redegøres for, hvilke aksiomer og slutningsregler, der præcis anvendes. I et uformelt bevis overlades også skridt og detaljer af mere triviel karakter til læseren, for eksempel induktion og omskrivninger. Derfor kan der optræde fejl i de uformelle beviser[2], hvilket kan føre til fejl i en given algoritme. Afprøvning i forbindelse med implementering vil af denne grund ofte være en god ide. Praksis er da også, at resultater kontrolleres af kolleger og andre med den nødvendige faglige kompetence. Men her kommer netop de formelle beviser og vor dages bevisførere ind i billedet. Som set i denne projektrapport går de formelle beviser i modsætning til de uformelle helt ned i detaljer. Med formelle beviser som standard for konkrete implementeringer kan man på længere sigt – måske – helt undvære afprøvning.

[1] Citatet er taget fra Knuths egen hjemmeside, mere præcist `http://www-cs-faculty.stanford.edu/~knuth/faq.html`.

[2] Undersøgelser i forbindelse med formalisering af uformelle beviser bekræfter også dette. Hovedresultaterne i de uformelle beviser er dog som regel korrekte alligevel.

A Pretty-print-kode fra *Isabelle*

Dette appendiks indeholder beviset for korrektheden af Quicksort samt det bibliotek, Sort, beviset bygger på i *Isabelle*.

A.1 Sort

```
theory Sort = Main:

consts
  sort :: "('a ⇒ 'a ⇒ bool) ⇒ 'a list ⇒ bool"
  multiset :: "'a list ⇒ 'a ⇒ nat"

primrec
  "sort le [] = True"
  "sort le (x#xs) = ((∀y∈set xs. le x y) ∧ sort le xs)"

primrec
  "multiset [] y = 0"
  "multiset (x#xs) y = (if x=y then Suc(multiset xs y)
else multiset xs y)"

constdefs
  total :: "('a ⇒ 'a ⇒ bool) ⇒ bool"
   "total r ≡ ∀ x y. r x y ∨ r y x"

  transitive :: "('a ⇒ 'a ⇒ bool) ⇒ bool"
   "transitive r ≡ ∀ x y z. r x y ∧ r y z ⟶ r x z"

lemma multiset_append[simp]:
 "multiset (xs@ys) x = multiset xs x + multiset ys x"
by (induct "xs") auto

lemma multiset_compl_add[simp]:
 "multiset [x∈xs.¬p x] y + multiset [x∈xs. p x] y =
multiset xs y"
by (induct "xs") auto

theorem set_via_multiset: "set xs = {x. multiset xs x ≠
0}"
by (induct "xs") auto
```

```
lemma sort_append[simp]:
 "sort le (xs@ys) = (sort le xs ∧ sort le ys ∧
                     (∀x ∈ set xs. ∀y ∈ set ys. le x
y))"
by (induct "xs") auto

end
```

A.2 Quicksort

```
theory Quicksort = Sort:

consts qsort :: "('a ⇒ 'a ⇒ bool) × 'a list ⇒ 'a list"

recdef qsort "measure (λ(le,xs).length xs)"
"qsort(le,[])   = []"
"qsort(le,x#xs) = qsort(le,[y:xs.¬le x y]) @
                  [x] @
                  qsort(le,[y:xs. le x y])"
(hints recdef_simp: less_Suc_eq_le)

theorem multiset_permutes[simp]:
 "multiset (qsort(le,xs)) x = multiset xs x"
by(induct "le" "xs" rule: qsort.induct) auto

lemma set_qsort[simp]: "set (qsort(le,xs)) = set xs"
by(simp add: set_via_multiset)

theorem sort_qsort:
 "⟦total le; transitive le⟧ ⟹ sort le (qsort(le,xs))"
apply (induct "le" "xs" rule: qsort.induct)
 apply simp
apply simp
apply (unfold total_def transitive_def)
apply blast
done

end
```

B ASCII-kode fra *Isabelle*

ASCII-koden for beviset for korrektheden af Quicksort i *Isabelle* bringes i dette appendiks.

B.1 Sort

```
theory Sort = Main:

consts
  sort :: "('a => 'a => bool) => 'a list => bool"
  multiset :: "'a list => 'a => nat"

primrec
  "sort le [] = True"
  "sort le (x#xs) = ((!y:set xs. le x y) & sort le xs)"

primrec
  "multiset [] y = 0"
  "multiset (x#xs) y = (if x=y then Suc(multiset xs y) else multiset xs y)"

constdefs
  total :: "('a => 'a => bool) => bool"
   "total r == !x y. r x y | r y x"

  transitive :: "('a => 'a => bool) => bool"
   "transitive r == ! x y z. r x y & r y z --> r x z"

lemma multiset_append[simp]:
 "multiset (xs@ys) x = multiset xs x + multiset ys x"
by (induct "xs") auto

lemma multiset_compl_add[simp]:
 "multiset [x:xs.~p x] y + multiset [x:xs. p x] y = multiset xs y";
by (induct "xs") auto

theorem set_via_multiset: "set xs = {x. multiset xs x ~= 0}"; by
(induct "xs") auto

lemma sort_append[simp]:
 "sort le (xs@ys) = (sort le xs & sort le ys &
                     (!x:set xs. !y:set ys. le x y))"
by (induct "xs") auto

end
```

B.2 Quicksort

```
theory Quicksort = Sort:

consts qsort :: "('a => 'a => bool) * 'a list => 'a list"

recdef qsort "measure (%(le,xs).length xs)"
"qsort(le,[])   = []"
"qsort(le,x#xs) = qsort(le,[y:xs.~le x y]) @
                  [x] @
                  qsort(le,[y:xs. le x y])"
(hints recdef_simp: less_Suc_eq_le)

theorem multiset_permutes[simp]:
 "multiset (qsort(le,xs)) x = multiset xs x"
by(induct "le" "xs" rule: qsort.induct) auto

lemma set_qsort[simp]: "set (qsort(le,xs)) = set xs"
by(simp add: set_via_multiset)

theorem sort_qsort:
 "[|total le; transitive le|] ==> sort le (qsort(le,xs))"
apply (induct "le" "xs" rule: qsort.induct)
 apply simp
apply simp
apply (unfold total_def transitive_def)
apply blast
done

end
```

C Aksiomer

Beviset for korrektheden af Quicksort bygger på aksiomerne fra *HOL*, *Set*, *Nat* og *Hilbert_ Choice*. Disse bringes her som udklip fra *Isabelles* hjemmeside. (Aksiomerne er i ASCII-kode).

C.1 Aksiomer i *HOL*

```
axioms
  eq_reflection: "(x=y) ==> (x==y)"

  refl:          "t = (t::'a)"
  subst:         "[| s = t; P(s) |] ==> P(t::'a)"

  ext:           "(!!x::'a. (f x ::'b) = g x) ==>
                  (%x. f x) = (%x. g x)"
    -- {* Extensionality is built into the meta-logic,
          and this rule expresses *}
    -- {* a related property.  It is an eta-expanded
          version of the traditional *}
    -- {* rule, and similar to the ABS rule of HOL *}

  the_eq_trivial: "(THE x. x = a) = (a::'a)"

  impI:          "(P ==> Q) ==> P-->Q"
  mp:            "[| P-->Q;  P |] ==> Q"

axioms
  iff:           "(P-->Q) --> (Q-->P) --> (P=Q)"
  True_or_False:  "(P=True) | (P=False)"
```

C.2 Aksiomer i *Set*

```
axioms
  mem_Collect_eq [iff]: "(a : {x. P(x)}) = P(a)"
  Collect_mem_eq [simp]: "{x. x:A} = A"
```

C.3 Aksiomer i *Nat*

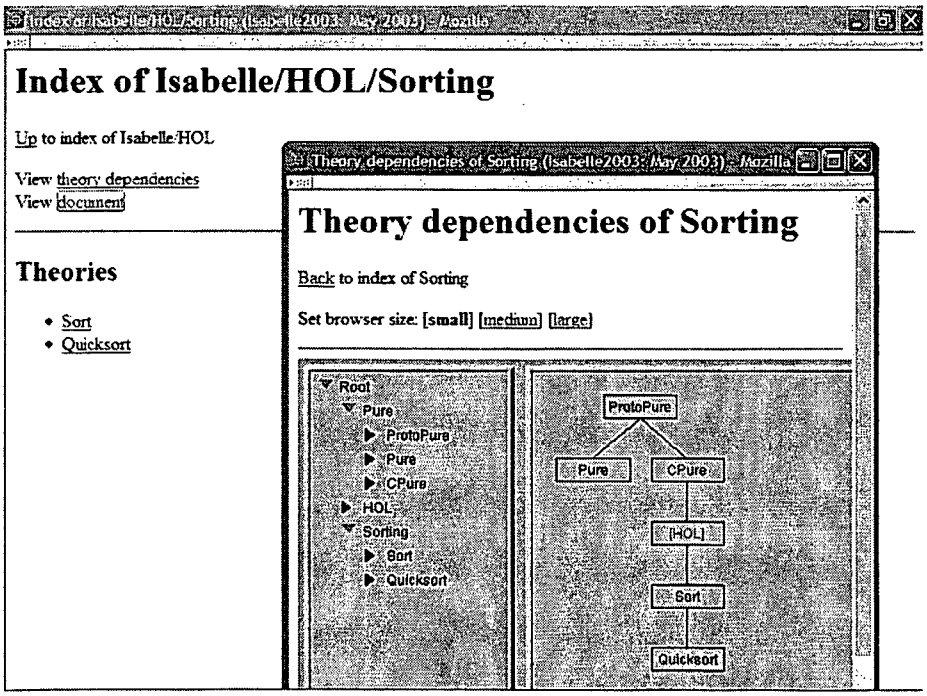
```
axioms
  -- {* the axiom of infinity in 2 parts *}
  inj_Suc_Rep:          "inj Suc_Rep"
  Suc_Rep_not_Zero_Rep: "Suc_Rep x ~= Zero_Rep"
```

C.4 Aksiomer i *Hilbert_ Choice*

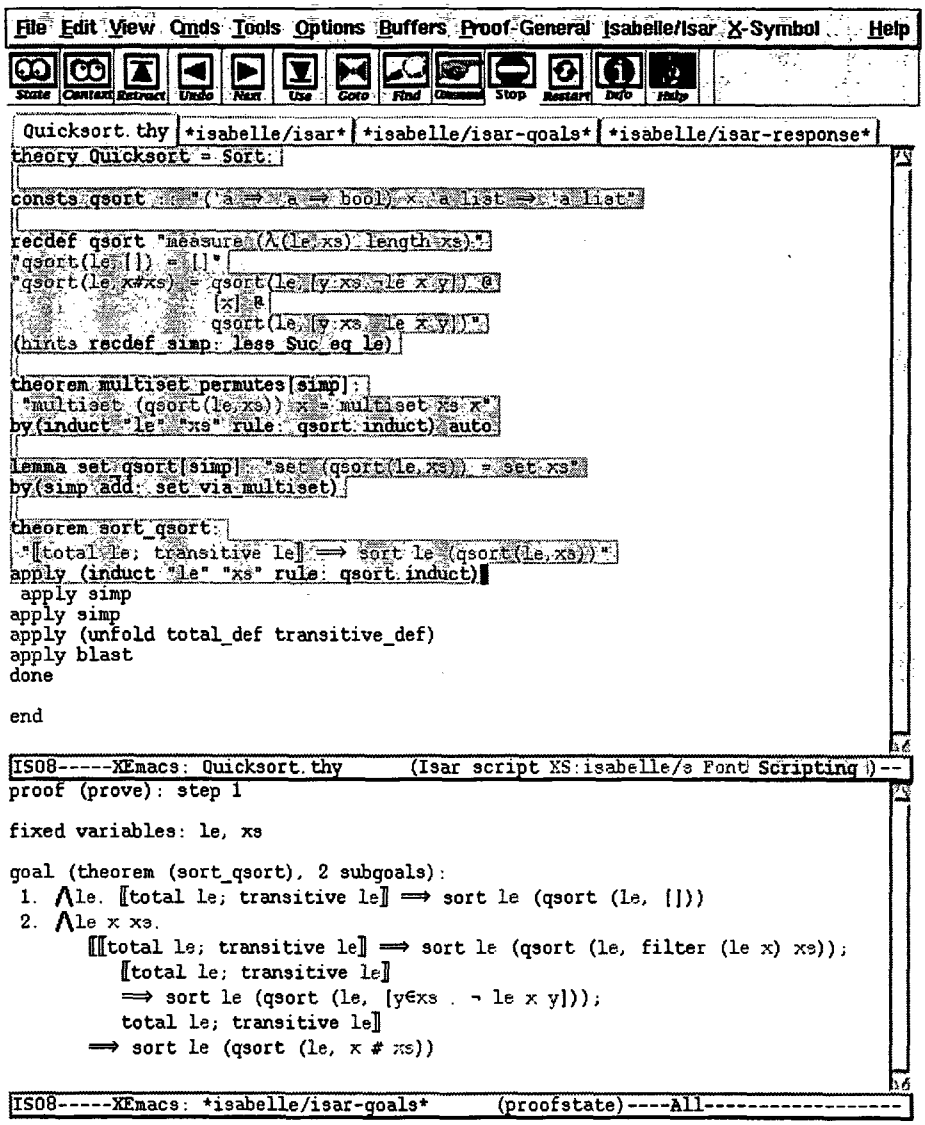
```
axioms
  someI: "P (x::'a) ==> P (SOME x. P x)"
```

D Screenshots fra *Isabelle*

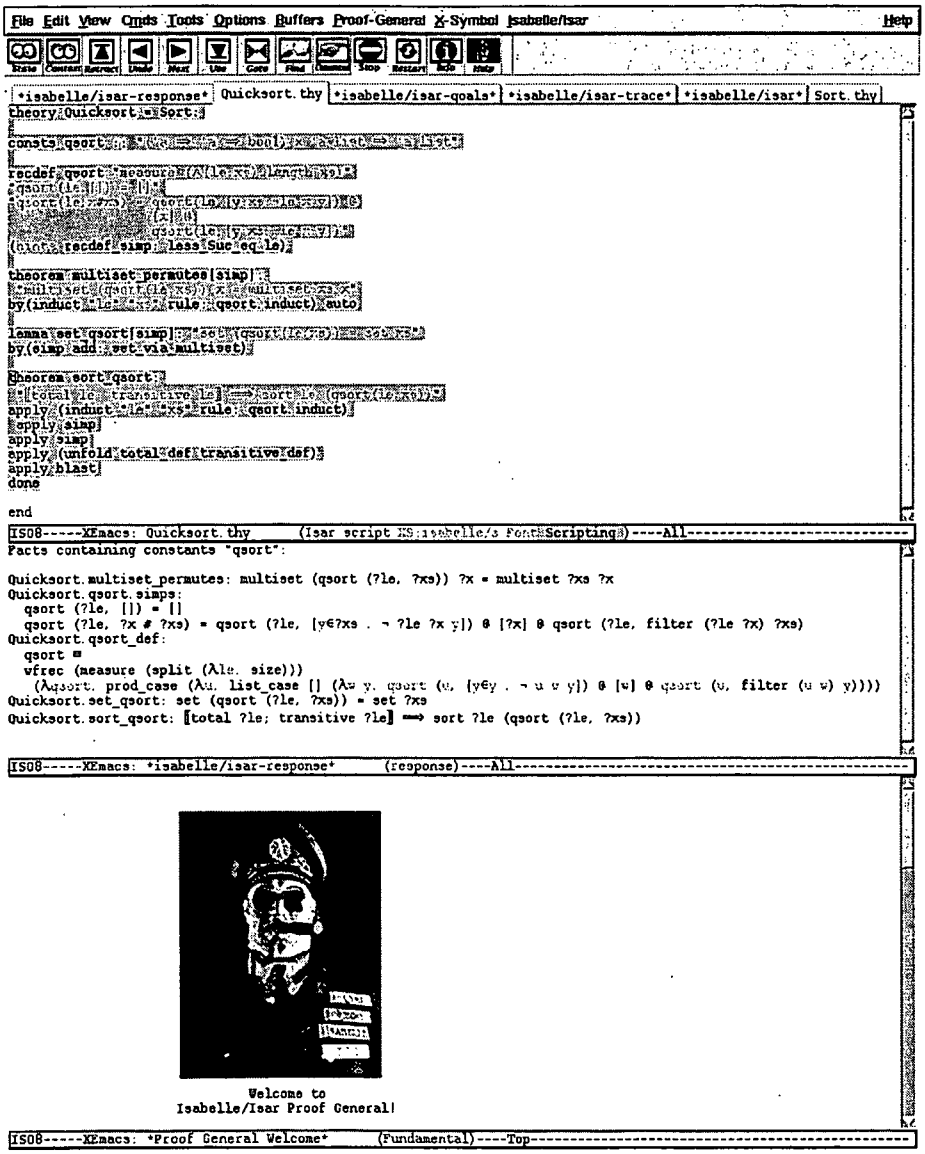
I dette appendiks bringes tre screenshots fra *Isabelle*.



Figur D.1 Et screenshot af vores teoriers bibliotek. Det lille vindue viser træet over teoriafhængigheder. (Tidligere vist som figur 13.2).



Figur D.2 Screenshot af brugergrænsefladen for *Isabelle* i *Proof General*. Øverst er koden for Quicksort, hvor systemet har verificeret den markerede del. Nederst er systemets 'goals' for et af skridtene i beviset for det sidste teorem. (Tidligere vist som figur 13.1).



Figur D.3 Screenshot af brugergrænsefladen for *Isabelle* i *Proof General*. Øverst er koden for Quicksort-beviset, hvor systemet har verificeret den markerede del. I vinduet under det ses de oplysninger systemet har om `qsort` efter endt bevis. Nederst ses et billede af *Proof General*.

Litteratur

A. D. Aleksandrov, A. N. Kolmogorov, and M. A. Lavrent'ev. *Mathematics: Its Content, Methods, and Meaning.* The M.I.T. Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1969. 9

P. B. Andrews. A Reduction of the Axioms for the Theory of Propositional Types. *Fundamenta Mathematicae*, 52:345–350, 1963. 59

P. B. Andrews. General Models, Descriptions, and Choice in Type Theory. *Journal of Symbolic Logic*, 37:385–394, 1972. 73

P. B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Kluver Academic Publishers, 2002. 4, 44, 50, 61, 62, 64, 65, 67, 68, 69, 70, 71, 72, 73, 74, 77, 84

H. Barendregt. The Impact of the Lambda Calculus in Logic and Computer Science. *Journal of Symbolic Logic*, 3:181–215, 1997. 57

H. P. Barendregt. *The Lambda Calculus.* North Holland Publishing Company, Amsterdam, 1981. 57

P. Barrette and A. Duncan. The Bertrand Russell Gallery. `http://www.humanities.mcmaster.ca/~bertrand/`, 2003. 30, 31

E. T. Bell. *Men of Mathematics.* Simon and Schuster, New York, 1937. 13, 14

E. T. Bell. *The Development of Mathematics,* (Second Edition). McGraw-Hill Book Company, New York, 1945. 9

M. Bramley-Moore. Interactive Theorem Proving. *Thesis*, November 2003. Tilgængelig på nettet som `http://www.csse.monash.edu.au/hons/projects/2003/Mostyn.Bramley-Moore/thesis.pdf`. 83, 84

T. A. A. Broadent. Russell, Bertrand Arthur William. In *Dictionary of Scientific Biography*, volume XII, pages 9–17. Charles Scribner's Sons, New York, 1970-80. 30, 31

A. Church. A Set of Postulates for the Foundation of Logic. *Journal of Symbolic Logic*, 33:346–366, 1932. 58

A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940. 59

A. Church. *Introduction to Mathematical Logic, Volume I* (Revised Edition). Princeton University Press, Princeton, 1956. 22, 23, 35, 56

J. N. Crossley, C. J. Ash, C. J. Brickhill, J. C. Stillwell, and N. H. Williams. *What is mathematical logic?* Oxford University Press, 1972. 9

M. Davis. *Computability and Unsolvability.* McGraw-Hill Book Company, 1958. 9

M. Davis. *The Undecidable.* Raven Press, New York, 1965. 50, 51, 58

M. Davis. *The Universal Computer – The road from Leibniz to Turing.* W. W. Norton and Company, 2000. 14, 15, 16, 20, 80

K. Devlin. *The Joy of Sets – Fundamentals of Contemporary Set Theory* (Second Edition). Springer Verlag, New York, 1993. 27, 28

G. Frege. *Collected Papers on Mathematics, Logic, and Philosophy.* Basil Blackwell Publisher Ltd., Oxford, 1984. 9

G. Frege. *Filosofiens, sprogets og matematikkens grundlag.* Forlaget Philosophia, Århus, 2002. 21, 25

M. T. Goodrich and R. Tamassia. *Algorithm Design – Foundations, Analysis and Internet Examples.* John Wiley & Sons, Inc., 2002. 92, 93, 95

J. Harrison. Formalized Mathematics. *Technical Report from Turku Centre for Computer Science (TUCS)*, (36):1–45, August 1996. Tilgængelig på nettet som `http://www.rbjones.com/rbjpub/logic/jrh0100.htm`. 15, 24, 41, 42, 43, 45

W. S. Hatcher. *The Logical Foundations of Mathematics.* Pergamon Press, 1982. 9

K. Helsgaun. Sortering – Forelæsningsslides. `http://www.dat.ruc.dk/~keld/teaching/algoritmedesign_f03/Slides/`, 2003. 92, 93, 94, 95

L. Henkin. A Theory of Propositional Types. *Fundamenta Mathematicae*, 52: 323–344, 1963. 59

R. Herken. *The Universal Turing Machine – A Half-Century Survey.* Oxford University Press, 1988. 54

J. R. Hindley and J. P. Seldin. *Introduction to Combinators and Lambda Calculus.* Cambridge University Press, 1986. 9

A. Hodges. *Alan Turing: The Enigma.* Anchor Brendon Ltd., Tiptree, Essex, 1983. 48, 50, 51, 53

A. Hodges. The Alan Turing Home Page. `http://www.turing.org.uk/turing/`, 2003. 48, 49, 50, 51

V. J. Katz. *A History of Mathematics – An Introduction* (Second Edition). Addison-Wesley Educational Publishers, Inc., Massachusetts, 1998. 25

A. Kfoury, H. Xi, and S. M. Pericas. The Church Project. `http://types.bu.edu/alonzo-church.html/`, 2003. 56, 57, 58

S. C. Kleene. *Introduction to Metamathematics.* Wolters-Noordhoff Publishing and North-Holland Publishing Company, 1952. 9

S. C. Kleene. *Mathematical Logic.* John Wiley and Sons, Inc., 1967. iii

M. Kline. *Mathematical Thoughts from Ancient to Modern Times.* Oxford University Press, New York, 1972. 34

M. Kline. *Mathematics – The Loss of Certainty.* Oxford University Press, New York, 1980. 9

A. Margaris. *First Order Mathematical Logic.* Dover Publications, Inc., Mineola, New York, 1967. 42, 44, 61

E. Mendelson. *Introduction to Mathematical Logic* (Fourth Edition). Chapman and Hall, London, 1997. 26, 27, 28, 44, 54, 62, 63, 71, 73

H. Meshckowiski. Cantor, Georg. In *Dictionary of Scientific Biography*, volume III, pages 52–58. Charles Scribner's Sons, New York, 1970-80. 9

G. H. Moore. Gödel, Kurt Friedrich. In *Dictionary of Scientific Biography*, volume 17, pages 348–357. Charles Scribner's Sons, New York, 1970-80. 38, 39, 40, 42, 44, 45

A. Mostowski. *Thirty Years of Foundational Studies – Lectures on the Development of Mathematical Logic and the Stydy of the Foundations of Mathematics in 1930-1964.* Societas Philosophica Fennica, 1966. 9

A. Nerode and R. A. Shore. *Logic for Applications* (Second Edition). Springer-Verlag, 1997. 84

T. Nipkow. Structured Proofs in Isar/HOL. *LNCS*, 2646:259–278, 2003. Tilgængelig på nettet som `http://isabelle.in.tum.de/Isar/`. 9

T. Nipkow, L. C. Paulson, and M. Wenzel. Isabelle/hol – a proof assistant for higher-order logic. *LNCS*, 2283:1–220, 2002. Revideret udgave tilgængelig på nettet som `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/dist/Isabelle2003/doc/tutorial.pdf`. 108

J. J. O'Connor and E. F. Robertson. The MacTutor History of Mathematics archive. *http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians.html*, 2003. 14, 19, 29, 37, 39, 43, 45, 47, 48, 55, 56, 58

L. C. Paulson. The Foundation of a Generic Theorem Prover. *Journal of Automated Reasoning*, 5:363–397, 1989. 97, 99

L. C. Paulson and T. Nipkow. Isabelle. *http://isabelle.in.tum.de/overview.html*, 2003. 97

S. G. Shanker. *Gödel's theorem in focus.* Croom Helm Ltd, Provident House, Kent, 1988. 38, 39, 40, 41

H. Sluga. *The Philosophy of Frege Volume 2 – Logic and Foundations of Mathematics in Frege's Philosophy.* Garland Publishing, Inc., New York, 1993. 9

J. van Heijenoort. *From Frege to Gödel - A Source Book in Mathematical Logic, 1879-1931.* Harvard University Press, Cambridge, 1967. 21, 22, 23, 24, 25, 28, 31, 32, 33, 34, 41, 42, 43, 44, 45, 54

B. van Rootselaar. Frege, Friedrich Ludwig Gottlob. In *Dictionary of Scientific Biography*, volume IV, pages 152–155. Charles Scribner's Sons, New York, 1970-80a. 20, 21, 22, 24

B. van Rootselaar. Turing, Alan Mathison. In *Dictionary of Scientific Biography*, volume XIII, pages 497–298. Charles Scribner's Sons, New York, 1970-80b. 48

A. N. Whitehead and B. Russell. *Principia mathematica Volume I-III* (Second Edition). University Press, Cambridge, 1927. iii, 34, 35

F. Wiedijk. Comparing Mathematical Provers. *LNCS*, 2594:188–202, 2003a. Tilgængelig på nettet som `http://www.cs.kun.nl/~freek/pubs/index.html`. 85, 87, 88, 89

F. Wiedijk. The Fifteen Provers of the World. pages 1–124, 2003b. Tilgængelig på nettet som `http://www.cs.kun.nl/~freek/comparison/index.html`. 86

H. Wussing and W. Arnold. *Biographien Bedeutender Mathematiker.* Volk und Wissen Volkseigener Verlag, Berlin, 1975. 9

Liste over tidligere udsendte tekster kan ses på IMFUFA's hjemmeside: http://mmf.ruc.dk eller rekvireres på sekretariatet, tlf. 46 74 22 63 eller e-mail: imfufa@ruc.dk.

332/97 ANOMAL SWELLING AF LIPIDE DOBBELTLAG
Specialerappoprt af: Stine Korremann
Vejleder: Dorthe Posselt

333/97 Biodiversity Matters
an extension of methods found in the literature on monetisation of biodiversity
by: Bernd Kuemmel

334/97 LIFE-CYCLE ANALYSIS OF THE TOTAL DANISH ENERGY SYSTEM
by: Bernd Kuemmel and Bent Sørensen

335/97 Dynamics of Amorphous Solids and Viscous Liquids
by: Jeppe C. Dyre

336/97 Problem-orientated Group Project Work at Roskilde University
by: Kathrine Legge

337/97 Verdensbankens globale befolkningsprognose
- et projekt om matematisk modellering
af: Jørn Chr. Bendtsen, Kurt Jensen, Per Pauli Petersen

338/97 Kvantisering af nanolederes elektriske ledningsevne
Første modul fysikprojekt
af: Søren Dam, Esben Danielsen, Martin Niss,
Esben Friis Pedersen, Frederik Resen Steenstrup
Vejleder: Tage Christensen

339/97 *Defining Discipline*
by: Wolfgang Coy

340/97 Prime ends revisited - a geometric point of view -
by: Carsten Lunde Petersen

341/97 Two chapters on the teaching, learning and assessment of geometry
by: Mogens Niss

342/97 A global clean fossil scenario DISCUSSION PAPER prepared by Bernd Kuemmel for the project LONG-TERM SCENARIOS FOR GLOBAL ENERGY DEMAND AND SUPPLY

343/97 IMPORT/EKSPORT-POLITIK SOM REDSKAB TIL OPTIMERET UDNYTTELSE AF EL PRODUCERET PÅ VE-ANLÆG
af: Peter Meibom, Torben Svendsen, Bent Sørensen

344/97 Puzzles and Siegel disks
by: Carsten Lunde-Petersen

345/98 Modeling the Arterial System with Reference to an Anesthesia Simulator
Ph.D. Thesis
by: Mette Sofie Olufsen

346/98 Klyngedannelse i en hulkatode-forstøvningsproces
af: Sebastian Horst
Vejledere: Jørn Borggren, NBI, Niels Boye Olsen

347/98 Verificering af Matematiske Modeller
- en analyse af Den Danske Eulerske Model
af: Jonas Blomqvist, Tom Pedersen, Karen Timmermann, Lisbet Øhlenschlæger
Vejleder: Bernhelm Booss-Bavnbek

348/98 Case study of the environmental permission procedure and the environmental impact assessment for power plants in Denmark
by: Stefan Krüger Nielsen
project leader: Bent Sørensen

349/98 Tre rapporter fra FAGMAT - et projekt om tal og faglig matematik i arbejdsmarkedsuddannelserne
af: Lena Lindenskov og Tine Wedege

350/98 OPGAVESAMLING - Bredde-Kursus i Fysik 1976 - 1998
Erstatter teksterne 3/78, 261/93 og 322/96

351/98 Aspects of the Nature and State of Research in Mathematics Education
by: Mogens Niss

352/98 The Herman-Swiatec Theorem with applications
by: Carsten Lunde Petersen

353/98 Problemløsning og modellering i en almendannende matematikundervisning
Specialerapport af: Per Gregersen og Tomas Højgaard Jensen

354/98 A Global Renewable Energy Scenario
by: Bent Sørensen and Peter Meibom

355/98 Convergence of rational rays in parameter spaces
by: Carsten Lunde Petersen and Gustav Ryd

356/98 Terrænmodellering
Analyse af en matematisk model til konstruktion af digitale terrænmodeller
Modelprojekt af: Thomas Frommelt, Hans Ravnkjær Larsen og Arnold Skimminge
Vejleder: Johnny Ottesen

357/98 Cayleys Problem
En historisk analyse af arbejdet med Cayleys problem fra 1870 til 1918
Et matematisk videnskabsfagsprojekt af: Rikke Degn, Bo Jakobsen, Bjarke K.W. Hansen, Jesper S. Hansen, Jesper Udesen, Peter C. Wulff
Vejleder: Jesper Larsen

358/98 Modeling of Feedback Mechanisms which Control the Heart Function in a View to an Implementation in Cardiovascular Models
Ph.D. Thesis by: Michael Danielsen

359/99 Long-Term Scenarios for Global Energy Demand and Supply
Four Global Greenhouse Mitigation Scenarios
by: Bent Sørensen (with contribution from Bernd Kuemmel and Peter Meibom)

360/99 SYMMETRI I FYSIK
En Meta-projektrapport af: Martin Niss, Bo Jakobsen & Tune Bjarke Bonné
Vejleder: Peder Voetmann Christiansen

361/99 Symplectic Functional Analysis and Spectral Invariants
by: Bernhelm Booß-Bavnbek, Kenro Furutani

362/99 Er matematik en naturvidenskab? - en udspænding af diskussionen
En videnskabsfagsprojekt-rapport af: Martin Niss
Vejleder: Mogens Nørgaard Olesen

363/99 EMERGENCE AND DOWNWARD CAUSATION
by: Donald T. Campell, Mark H. Bickhard, and Peder V. Christiansen

364/99 Illustrationens kraft - Visuel formidling af fysik
Integreret speciale i fysik og kommunikation
af Sebastian Horst
Vejledere: Karin Beyer, Søren Kjørup

365/99 To know - or not to know - mathematics, that is a question of context
by: Tine Wedege

366/99 LATEX FOR FORFATTERE - En introduktion til LATEX
og IMFUFA-LATEX
af: Jørgen Larsen

367/99 Boundary Reduction of Spectral Invariants and Unique Continuation Property
by: Bernhelm Booss-Bavnbek

368/99 Kvartvejsraport for projektet SCENARIER FOR SAMLET UDNYTTELSE AF BRINT SOM ENERGIBÆRER I DANMARKS FREMTIDIGE ENERGISYSTEM
Projektleder: Bent Sørensen

369/99 Dynamics of Complex Quadratic Correspondences
by: Jacob S. Jalving
Supervisor: Carsten Lunde Petersen

370/99 OPGAVESAMLING - Bredde-Kursus i Fysik 1976 - 1999
Eksamensopgaver fra perioden 1976 - 1999. Denne tekst erstatter tekst nr. 350/98

371/99 Bevisets stilling - beviser og bevisførelse i en gymnasial matematik undervisning
Et matematikspeciale af: Maria Hermannsson
Vejleder: Mogens Niss

372/99 En kontekstualiseret matematikhistorisk analyse af ikke-lineær programmering: Udviklingshistorie og multipel opdagelse
Ph.d.-afhandling af Tinne Hoff Kjeldsen

373/99 Criss-Cross Reduction of the Maslov Index and a Proof of the Yoshida-Nicolaescu Theorem
by: Bernhelm Booss-Bavnbek, Kenro Furutani and Nobukazu Otsuki

374/99 Det hydrauliske spring - Et eksperimentelt studie af polygoner og hastighedsprofiler
Specialeafhandling af: Anders Marcussen
Vejledere: Tomas Bohr, Clive Ellegaard, Bent C. Jørgensen

375/99 Begrundelser for Matematikundervisningen i den lærde skole hhv. gymnasiet 1884-1914
Historiespeciale af Henrik Andreassen, cand.mag. i Historie og Matematik

376/99 Universality of AC conduction in disordered solids
by: Jeppe C. Dyre, Thomas B. Schrøder

377/99 The Kuhn-Tucker Theorem in Nonlinear Programming: A Multiple Discovery?
by: Tinne Hoff Kjeldsen

378/00 Solar energy preprints:
1. Renewable energy sources and termal energy storage
2. Integration of photovoltaic cells into the global energy system
by: Bent Sørensen

379/00 EULERS DIFFERENTIALREGNING
Eulers indførelse af differentialregningen stillet over for den moderne
En tredjesemesters projektrapport på den naturvidenskabelige basisuddannelse
af: Uffe Thomas Volmer Jankvist, Rie Rose Møller Pedersen, Maja Bagge Pedersen
Vejleder: Jørgen Larsen

380/00 MATEMATISK MODELLERING AF HJERTEFUNKTIONEN
Isovolumetrisk ventrikulær kontraktion og udpumpning til det cardiovaskulære system
af: Gitte Andersen (3.moduls-rapport), Jakob Hilmer og Stine Weisbjerg (speciale)
Vejleder: Johnny Ottesen

381/00 Matematikviden og teknologiske kompetencer hos kortuddannede voksne
- Rekognosceringer og konstruktioner i grænselandet mellem matematikkens didaktik og forskning i voksenuddannelse
Ph. d.-afhandling af Tine Wedege

382/00 Den selvundvigende vandring
Et matematisk professionsprojekt
af: Martin Niss, Arnold Skimminge
Vejledere: Viggo Andreasen, John Villumsen

383/00 Beviser i matematik
af: Anne K.S.Jensen, Gitte M. Jensen, Jesper Thrane, Karen L.A.W. Wille, Peter Wulff
Vejleder: Mogens Niss

384/00 Hopping in Disordered Media: A Model Glass Former and A Hopping Model
Ph.D. thesis by: Thomas B. Schrøder
Supervisor: Jeppe C. Dyre

385/00 The Geometry of Cauchy Data Spaces
This report is dedicated to the memory of Jean Leray (1906-1998)
by: B. Booss-Bavnbek, K. Furutani, K. P. Wojciechowski

386/00 Neutrale mandatfordelingsmetoder – en illusion?
af: Hans Henrik Brok-Kristensen, Knud Dyrberg, Tove Oxager, Jens Sveistrup
Vejleder: Bernhelm Booss-Bavnbek

387/00 A History of the Minimax Theorem: von Neumann's Conception of the Minimax Theorem - - a Journey Through Different Mathematical Contexts
by: Tinne Hoff Kjeldsen

388/00 Behandling af impuls ved kilder og dræn i C. S. Peskins 2D-hjertemodel
et 2. moduls matematik modelprojekt
af: Bo Jakobsen, Kristine Niss
Vejleder: Jesper Larsen

389/00 University mathematics based on problemoriented student projects: 25 years of experience with the Roskilde model
By: Mogens Niss
Do not ask what mathematics can do for modelling. Ask what modelling can do for mathematics!
by: Johnny Ottesen

390/01 SCENARIER FOR SAMLET UDNYTTELSE AF BRINT SOM ENERGIBÆRER I DANMARKS FREMTIDIGE ENERGISYSTEM Slutrapport, april 2001
Projektleder: Bent Sørensen
Projektdeltagere: DONG: Aksel Hauge Petersen, Celia Juhl, Elkraft System[#]: Thomas Engberg Pedersen[##], Hans Ravn, Charlotte Søndergren, Energi 2[#]: Peter Simonsen, RISØ Systemanalyseafd.: Kaj Jørgensen[*], Lars Henrik Nielsen, Helge V. Larsen, Poul Erik Morthorst, Lotte Schleisner, RUC: Finn Sørensen[**], Bent Sørensen
[#]Indtil 1/1-2000 Elkraft, [##] fra 1/5-2000 Cowi Consult
[*] Indtil 15/6-1999 DTU Bygninger & Energi, [**] fra 1/1-2001 Polypeptide Labs.
Projekt 1763/99-0001 under Energistyrelsens Brintprogram

391/01 Matematisk modelleringskompetence – et undervisningsforløb i gymnasiet
3. semesters Nat.Bas. projekt af: Jess Tolstrup Boye, Morten Bjørn-Mortensen, Sofie Inari Castella, Jan Lauridsen, Maria Gøtzsche, Ditte Mandøe Andreasen
Vejleder: Johnny Ottesen

392/01 "PHYSICS REVEALED" THE METHODS AND SUBJECT MATTER OF PHYSICS
an introduction to pedestrians (but not excluding cyclists)
PART III: PHYSICS IN PHILOSOPHICAL CONTEXT
by: Bent Sørensen.

393/01 Hilberts matematikfilosofi
Specialerapport af: Jesper Hasmark Andersen
Vejleder: Stig Andur Pedersen

394/01 "PHYSICS REVEALED" THE METHODS AND SUBJECT MATTER OF PHYSICS
an introduction to pedestrians (but not excluding cyclists)
PART II: PHYSICS PROPER
by: Bent Sørensen.

395/01 Menneskers forhold til matematik. Det har sine årsager!
Specialeafhandling af: Anita Stark, Agnete K. Ravnborg
Vejleder: Tine Wedege

396/01 2 bilag til tekst nr. 395: Menneskers forhold til matematik. Det har sine årsager!
Specialeafhandling af: Anita Stark, Agnete K. Ravnborg
Vejleder: Tine Wedege

397/01 En undersøgelse af solvents og kædelængdes betydning for anomal swelling i phospholipiddobbeltlag
2. modul fysikrapport af: Kristine Niss, Arnold Skimminge, Esben Thormann, Stine Timmermann
Vejleder: Dorthe Posselt

398/01 Kursusmateriale til "Lineære strukturer fra algebra og analyse" (E1)
Af: Mogens Brun Heefelt

399/01 Undergraduate Learning Difficulties and Mathematical Reasoning
Ph.D Thesis by: Johan Lithner
Supervisor: Mogens Niss

400/01 On Holomorphic Critical quasi circle maps
By: Carsten Lunde Petersen

401/01 Finite Type Arithmetic
Computable Existence Analysed by Modified Realisability and Functional Interpretation
Master's Thesis by: Klaus Frovin Jørgensen
Supervisors: Ulrich Kohlenbach, Stig Andur Pedersen and Anders Madsen

402/01 Matematisk modellering ved den naturvidenskabelige basisuddannelse
- udvikling af et kursus
Af: Morten Blomhøj, Tomas Højgaard Jensen, Tinne Hoff Kjeldsen og Johnny Ottesen

403/01 Generaliseringer i integralteorien
- En undersøgelse af Lebesgue-integralet, Radon-integralet og Perron-integralet
Et 2. modul matematikprojekt udarbejdet af: Stine Timmermann og Eva Uhre
Vejledere: Bernhelm Booss-Bavnbek og Tinne Hoff Kjeldsen

404/01 "Mere spredt fægtning"
Af: Jens Højgaard Jensen

405/01 Real life routing
- en strategi for et virkeligt vrp
Et matematisk modelprojekt af: David Heiberg Backchi, Rasmus Brauner Godiksen, Uffe Thomas Volmer Jankvist, Jogvan Martin Poulsen og Neslihan Saglanmak
Vejleder: Jørgen Larsen

406/01 Opgavesamling til dybdekursus i fysik
Eksamensopgaver stillet i perioden juni 1976 til juni 2001
Denne tekst erstatter tekst nr. 25/1980 + efterfølgende tillæg

407/01 Unbounded Fredholm Operators and Spectral Flow
By: Bernhelm Booss-Bavnbek, Matthias Lesch, John Phillips

408/02 Weak UCP and Perturbed Monopole Equations
By: Bernhelm Booss-Bavnbek, Matilde Marcolli, Bai-Ling Wang

409/02 Algebraisk ligningsløsning fra Cardano til Cauchy
- et studie af kombinationers, permutationers samt invariansbegrebets betydning for den algebraiske ligningsløsning *før* Gauss, Abel og Galois
Videnskabsfagsprojekt af: David Heiberg Backchi, Uffe thomas Volmer Jankvist, Neslihan Saglanmak
Vejleder: Bernhelm Booss-Bavnbek

410/02 2 projekter om modellering af influenzaepidemier
Influenzaepidemier- et matematisk modelleringsprojekt
Af: Claus Jørgensen, Christina Lohfert, Martin Mikkelsen, Anne-Louise H. Nielsen
Vejleder: Morten Blomhøj
Influenza A: Den tilbagevendende plage – et modelleringsprojekt
Af: Beth Paludan Carlsen, Christian Dahmcke, Lena Petersen, Michael Wagner
Vejleder: Morten Blomhøj

411/02 Polygonformede hydrauliske spring
Et modelleringsprojekt af: Kåre Stokvad Hansen, Ditte Jørgensen, Johan Rønby Pedersen, Bjørn Toldbod
Vejleder: Jesper Larsen

412/02 Hopfbifurkation og topologi i væskestrømning – en generel analyse samt en behandling af strømningen bag en cylinder
Et matematisk modul III professionsprojekt af. Kristine Niss, Bo Jakobsen
Vejledere: Morten Brøns, Johnny Ottesen

413/03 "Elevernes stemmer" Fysikfaget, undervisningen og lærerroller, som eleverne opfatter det i det almene gymnasium i Danmark
Af: Carl Angell, Albert Chr. Paulsen

414/03 Feltliniediagrammer En vej til forståelse?
Et 1. modul fysikprojekt af: Ditte Gundermann, Kåre Stokvad Hansen, Ulf Rørbæk Pedersen
Vejleder: Tage Emil Christensen

415/03 FYSIKFAGET I FORANDRING Læring og undervisning i fysik i gymnasiet med fokus på dialogiske processer, autenticitet og kompetenceudvikling
Ph.d.-afhandling i fysikdidaktik af: Jens Dolin

416/03 Fourier og Funktionsbegrebet
- Overgangen fra Eulers til Dirichlets funktionsbegreb
Projektrapport af: Rasmus Brauner Godiksen, Claus Jørgensen, Tony Moyer Hanberg, Bjørn Toldbod
Vejleder: Erik von Essen

417/03 The Semiotic Flora of Elementary particles
By: Peder Voetmann Christiansen

418/03 Militærnatematik set med kompetencebriller
3. modul projektrapport af: Gitte Jensen og specialerapport af: Jesper Thrane
Vejleder: Tine Wedege

419/03 Energy Bond Graphs – a semiotic formalization of modern physics
By: Peder Voetmann Christiansen

420/03 Stemning og Musikalsk Konsonans
Et matematisk modelleringsprojekt af: Claus Jørgensen
Vejleder: Johnny Ottesen

421/03 OPGAVESAMLING
Bredde-kursus i fysik 1976 – 2003.
Denne tekst erstatter tekst nr. 370/99

422/03 Vurdering af dynamisk blodstrømningsmodel
- ved numerisk simulering med FEMLAB
Et 2. modul matematikprojekt af: Sofie Inari Castella, Ingunn Gunnarsdóttir og Jacob Kirkensgaard Hansen
Vejleder: Johnny Ottesen

423/03 Fysikkens historie i en almendannende fysikundervisning
- Eksemplificeret med Millikan Ehrenhaft kontroversen
Specialerapport af: Marianne Wilcken Bjerregaard
Vejleder: Albert Chr. Paulsen

424/03 Dielectric and Shear Mechanical Relaxation in Glass Forming Liquids
- A thorough analysis and experimental test of the DiMarzio-Bishop model
Master thesis in physics by: Kristine Niss and Bo Jakobsen
Supervised by: Niels Boye Olsen

425/03 Fysiske forklaringer i undervisning
Specialerapport af: Kirsten Ringgard Jensen
Vejleder: Jens Højgaard Jensen