Datalogiens Historie

- eller hvordan 0–1 blev til videnskab

De første dataloger vidste slet ikke, at de var dataloger. De programmerede bare løs for at få de første computere til at lave indviklede beregninger. Gennem deres systematiske arbejde med algoritmer og dataproblemer opstod fagområdet Datalogi *i løbet af årene 1950-70.*

Af Christian Gram

Menneskene har længe haft brug for at regne og har også konstrueret hjælpemidler til beregning. Det hele begyndte med kuglerammer for mere end 3000 år siden. Både i Kina og i Babylon har handlende og bogholdere brugt kuglerammen – abacus'en – og siden 1600-tallet er der konstrueret mange mekaniske regnemaskiner, som kunne addere, subtrahere og gange. Maskinerne var så simple, at man brugte dem uden videre og uden at nedskrive planer for beregningerne, der blev lavet. Først da elektronik gjorde det muligt at bygge moderne computere i 1940-erne, blev det nødvendigt at lægge en plan – dvs. skrive et program – for en beregning.

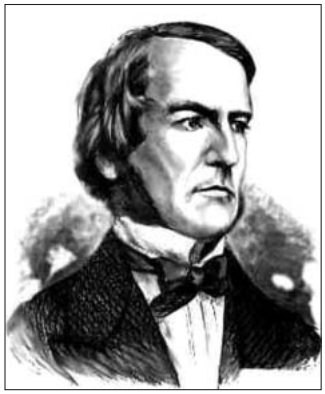
2-talsystemet og logik

Den tyske filosof og matematiker Gottfried Leibniz konstruerede sidst i 1600-tallet den første maskine, der både kunne lægge sammen, trække fra, gange og dividere. Han er mest kendt som filosof og for sit bidrag til udvikling af differentialregning. Men han kom også med flere ideer, der skulle vise sig at blive centrale for datalogien. I en berømt afhandling fra 1703 *Explication de l'Arithmetique Binaire* viser han, at man i stedet for at bruge 10-talsystemet lige så godt kunne lave alle beregninger i 2-talsystemet, hvor man kun behøver cifrene 0 og 1. Det system er blevet brugt i elektroniske computere siden 1950-erne, fordi computerens komponenter dermed kun behøver at have to forskellige tilstande:

- en kontakt er sluttet eller afbrudt.
- en ledning er strømførende eller ikke strømførende.
- et felt på et hulkort er gennemhullet eller ikke hullet.
- et felt på en computerdisk er magnetiseret den ene vej eller den anden vej.

Leibniz skrev også om »En generel metode, hvor alle sandheder om slutninger bliver reduceret til en slags beregning«. Her foregriber han den logiske algebra, som først 150 år senere blev taget op, uddybet og præcist formuleret af den engelske matematiker George Boole. Logisk algebra – eller Boole'sk algebra, som det også kaldes – er læren om vore logiske slutningsregler og ræsonnementer, udtrykt som matematiske ligninger og regler. Det gør det muligt at manipulere matematisk eksakt med indviklede, betingede udsagn som f.eks. denne lovtekst: "Børnetilskuddet udgør 1738 kr., hvis en eller begge forældre modtager folke- eller invalidepension, hvis forsørgeren er enlig og faderskabet ukendt og ingen er kendt bidragspligtig, eller hvis den ene af forældrene er død."

Udsagn kan være sande eller falske; de kan altså have netop to værdier og kan bekvemt repræsenteres med cifrene 0 og 1, og vor dagligdags brug af udsagn og betingelser kan derved oversættes til simple regneregler med 0 og 1. Derved bliver computeren velegnet til at håndtere logik, dvs. "regne med" og kombinere sand og falsk. Det er helt uundværligt i datalogien, fordi der i ethvert



Matematikeren George Boole

```
Far(frede, henrik)
Far(petra, henrik)
Far(hans, frede)
Farfar(X,Y) ← Far(X,Z), Far(Z,Y)
Farfar(hans, X)?
```

Eksempel på et lille logikprogram om familierelationer med tre fakta, en regel og et spørgsmål.

Universel Turingmaskine

Maskinen: Husker egen tilstand S.
Læser aktuelt tegn T.
T og S tilsammen bestemmer næste trin:
`<ny tilstand, båndflytn, nyt tegn skrives>`

Båndet: Uendelig langt, hvert felt plads til 1 tegn.
Indeholder både program og data.

program indgår en masse logiske beregninger ved siden af de numeriske beregninger. Behovet for at udføre logiske slutninger har endda ført til, at der siden 1970-erne er udviklet særlige programmeringssprog til logikprogrammering. Det er sprog som f.eks. er velegnede til at beskrive følgevirkninger af indviklede love, eller til at forudsige konflikter mellem forskellige lovforslag.

Programmeringssprog og beregnelighed

De allerførste computerprogrammer blev skrevet i binær kode, computerens eget sprog. Men et program består af mange, mange ordrer, og sådanne lange rækker af 0-er og 1-ere er fuldstændig uoverskuelige for mennesker. Derfor blev der hurtigt opfundet specielle sprog – programmeringssprog – til at beskrive den databehandling, der skulle udføres. Så kunne programmøren f.eks. skrive ADD i stedet for 0101101 og decimaltallet 21 i stedet for det binære tal 10101. Sådanne programmer skal oversættes, før databehandlingen kan udføres. Det klarer computeren ved hjælp af særlige oversætterprogrammer, compilere. Deres opgave er at transformere programmer fra forståelig form til binær kode. Udviklingen og brugen af programmeringssprog rejste nogle fundamentale spørgsmål:

- Hvordan beskriver man bedst en beregning, og hvor præcist kan man beskrive meningen (hensigten) med beregningen?
- Hvilke grundoperationer er nødvendige for at kunne beskrive alle beregninger?
- Hvor hurtigt – eller effektivt – kan man udføre en bestemt beregning, f.eks. løse N ligninger med N ubekendte?
- Hvad betyder det at beregne en funktion, og hvad kan man overhovedet beregne?

Det har ført til udvikling af datalogiske teorier for beregningskompleksitet og sprog. Beregningsopgaver kan inddeles i kompleksitetsklasser efter, hvor vanskelige de er, mens programmeringssprog inddeles i klasser efter, hvor "udtryksfulde" de er, dvs. hvor nemt det er at beskrive komplicerede beregningsopgaver. En speciel matematisk notation – Backus-Naur Formen – blev opfundet for at kunne definere programmeringssprog præcist og entydigt.

Turingmaskinen

Længe før computerudviklingen tog fart, var matematikere begyndt at spekulere på, hvad beregning af en funktion egentlig betyder, og hvorvidt man kan beregne alle funktioner. Englænderen Alan Turing opfandt – på papiret – en meget teoretisk, meget primitiv og meget generel "maskine", som bruges til at definere beregnelighed: Turings maskine kan beregne alt, hvad vi kalder beregneligt. Maskinen, som vi i dag kalder en *Universel Turing-*

Datalogi og Peter Naur

Ordet *Datalogi* betyder Læren om data, deres væsen og brug. Det danske ord er opfundet i 1966 af Peter Naur, som allerede i 50-erne begyndte at beregne planetbaner på computer. Peter Naur var uddannet som astronom og interesserede sig for planetbaner, som kræver mange beregninger. Han var en af de første danskere, der brugte computere, og han forlod astronomien for at kaste sig over databehandling. Han er en af de ganske få danske dataloger, der er verdenskendt, især for sit arbejde med programmeringssproget ALGOL.

Programmeringssprog

Et program i en computer består af lange rækker af 0-er og 1-ere (binær kode). Men vi programmerer i sprog, der er lidt mere forståelige for mennesker. Figuren viser et lillebitte program for at danne summen af 50 tal i fire forskellige programmeringssprog. FORTRAN (Formula Translation) er et af de første og mest udbredte sprog. Som navnet siger, kan men der – næsten – beskrive en beregning som en række matematiske formler. Et andet af de tidlige sprog, ALGOL (Algorithmic Language), fik en enorm betydning som standard, fordi det var meget præcist defineret. Danskeren Peter Naur og amerikaneren John Backus udviklede her den særlige Backus-Naur Form, som er en matematisk notation til at definere et programmeringssprog.

Et program til at lægge 50 tal sammen, skrevet i binær kode:

```
00001001   00000001   00110010
00000010   00110010   11111111
00010100   11111111   00000000
00010101   11111110   00000000
00000000   00000000   00000000
```

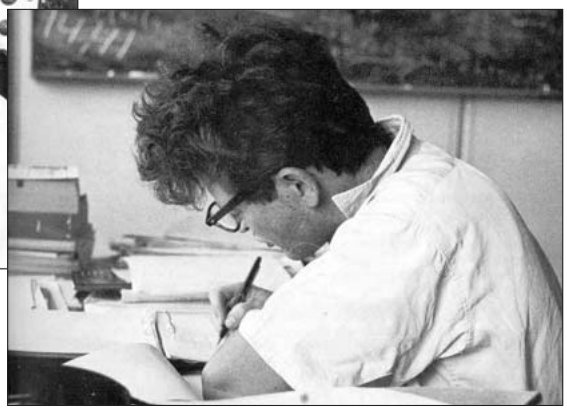
Det samme i et primitivt programmeringssprog (assembler kode):

```
PLACER   REL+1    50
ADD      50       -1
COND     REL-1     0
JMP      REL-2
STOP     0
```

- og det samme i to højere programmeringssprog (Fortran og Algol):

```
      DO 1  I = 1,50
      R = R + A(I)
    1 CONTINUE

for  I:=1 step 1 until 50 do
  R := R + A[I];
```



Fotos/kilde: Chr. Gram

Danmarks første programmør: Jørn Jensen blev ansat som en af de første på Regnecentralen, Dansk Institut for Matematikmaskiner, i 1958, og han udviklede sig hurtigt til en genial programmør. Han lå helst på knæ på sin skrivebordsstol, mens han programmerede på livet løs og spiste mængder af hugget sukker imens. Han opfandt bl.a. "Jensen's Device", en særlig snedig måde at kombinere to argumenter i en funktion, der skulle beregnes. På billedet til venstre ses Jørn Jensen sammen med direktør Niels Ivar Bech og chefingeniør Bent Scharøe Petersen ved DASK's kontrolbord.

Verdens første programmør

Midt i 1800-tallet forsøgte en genial og gal englænder, Charles Babbage, at bygge en hulkortstyret regnemaskine, som skulle kunne beregne alle mulige funktioner. Han blev aldrig færdig med maskinen, men Lady Ada Lovelace blev interesseret i hans arbejde. Hun så, at for at kunne bruge den komplicerede maskine fornuftigt, var det nødvendigt først at nedskrive en plan for hver beregning (som bestod af mange enkelttrin). På den måde blev hun verdens første programmør – uden nogensinde at have haft en fungerende regnemaskine. Hun var i øvrigt datter af den berømte digter Byron.

maskine, består af en blackbox og et uendelig langt bånd med tegn. Maskinen har en begrænset indre "hukommelse", idet blackbox'en kan være i forskellige "stillinger" kaldet tilstande. Desuden kan den læse tegn fra båndet, flytte båndet frem eller tilbage og skrive nye tegn på det. Derved kan båndet fungere på en gang som ind-medium, program- og datalager, samt ud-medium, hvor blackbox'en er en yderst primitiv styreenhed. Det overraskende er, at en så simpel maskine kan gøre så meget.

Enhver moderne computer er faktisk en – mere eller mindre specialiseret – kompleks udgave af den universelle Turingmaskine, dog kun med et endelig langt bånd, og en computer kan derfor teoretisk set udføre en hvilken som helst beregning inden for en grænse, som sættes af størrelsen på dens lager.

Parallelprogrammering

Omkring 1970 var elektronikken blevet så hurtig, at en computer kunne udføre flere programmer parallelt. Det skal forstås sådan, at computeren meget hurtigt skifter mellem flere programmer: Først udføres program A i 50 millisek., så program B i 50 millisek., osv. Da indlæsning af data og udskrivning af resultater tager lang tid målt med computerens regnehastighed, kan computeren udnytte ventetid i ét program til at lave noget nyttigt med andre programmer.

Men så skete det sommetider, at parallelkørende programmer gik i baglås, dvs. gik totalt i stå, fordi de ventede på hinanden: Program A kunne ikke komme videre, før B var færdig med at skrive på terminalen; men B kunne ikke blive færdig med det, fordi B først skulle hente data på harddisken, som A stadig havde reserveret. I de første udgaver af Microsoft's Windows-system optrådte den slags situationer ikke helt sjældent. Studiet af baglås – på engelsk "deadlock" eller "deadly embrace" – blev startskuddet til udvikling af teori for samspil mellem parallelle programmer og udvikling af særlige programmeringssprog til parallelprogrammering. Disse sprog har specielle ordrer til baglås-fri kommunikation mellem programmer. Det sikrer, at programmerne ikke blokerer hinanden, selvom de kører parallelt.

Store og forsinkede Edb-projekter

Det har altid været svært at planlægge edb-projekter. Englænderen Charles Babbage, som allerede i midten af 1800-tallet forsøgte at bygge en mekanisk "Analytical Engine", blev aldrig færdig med sin maskine. I de sidste 50 år har vi set utallige edb-projekter blive forsinket og fordyret. Og når et edb-program endelig erklæres færdigt og køreklart, viser det sig alligevel ofte at rumme fejl. Det skyldes den store kompleksitet og det mylder af data, der præger edb-programmer. Alle detaljer skal være på plads og spille rigtigt sammen, og det er meget svært at holde overblikket over det hele.

Det har ført til udvikling af

- Systematiske testmetoder for edb-programmer, hvor programmøren kan sikre sig, at alle hjørner af programmet fungerer, og at alle programdelene spiller rigtigt sammen.
- Metoder til matematisk beskrivelse af meningen med programmer. Det kaldes formel semantik og er et hjælpemiddel under planlægningen.

- Bevisførelse, dvs. matematisk-logiske ræsonnementer for at edb-programmer virkelig gør det, der var meningen.

Derfor er avanceret programmering i dag stærkt præget af matematik og logik. Almindelige edb-programmer – som f.eks. bankers kontosystemer eller kommunernes skattesystemer – kan nogenlunde let ændres og rettes, hvis de rummer fejl. Det er meget værre med alle de programmer, der indbygges i apparater som mobiltelefoner, og programmer til processtyring. En programfejl i en mobiltelefon kan man ikke let komme til at rette, og fejl i de programmer, der styrer en atomreaktor eller S-togssignalerne, kan få meget alvorlige følger. Derfor planlægges og testes den slags programmer overordentlig grundigt med brug af al mulig matematisk og logisk hjælp.

Edb-systemer og kirkebyggeri

De store datamængder, der indgår i store edb-systemer – som f.eks. et pengeinstituts edb-system – gør det nødvendigt på effektiv vis at kunne søge efter bestemte data. Derfor har datalogerne udviklet

- Teori for databaser, dvs. teori for, hvordan store datamængder kan struktureres og opdeles.
- Avancerede søge-metoder samt teorier for, hvor hurtigt man kan finde en "nål" i en "høstak" af data.
- Avancerede sorterings-metoder, så man effektivt kan indsætte nye data, slette "døde" data, og omordne registre. Det er f.eks. nyttigt at kunne udskrive et personregister efter navn eller adresse eller telefonnummer eller fødselsår, alt efter hvad det skal bruges til.

Men et er teori, et andet praksis! Selvom datalogerne har systematiseret programmering og databehandling via matematisk-logisk tankegang og fortsat finder på nye smarte teorier, vil vi stadig høre om edb-skandaler, hvor edb er forsinket, for dyrt og fejlbehæftet. Det skyldes, at vi ønsker at lave så store og så komplekse edb-systemer, at teorierne er for svære at anvende i praksis. På en måde opfører vi os med de store edb-systemer som kirkebyggere i middelalderens Europa: De byggede kæmpestore, himmelstræbende katedraler uden at kende lovene for træk og tryk i så store, bærende konstruktioner. Derfor styrtede mange katedraler sammen under byggeriet, og først mange år efter blev man i stand til på forsvarlig vis at forudberegne, hvordan man kan bygge så himmelstræbende. ■

Søgninger på Internettet

Når du søger på Internettet med søgeprogrammet Google, får du svar på få sekunder, næsten ligegyldigt hvad du spørger om. Programmet sorterer også svarene, så de (sandsynligvis) mest relevante svar vises først. Samtidig fortæller programmet, at det søger blandt over 3 milliarder hjemmesider. Det er kun muligt, fordi Google bruger meget avancerede søge- og sorteringsmetoder. For det første har Google forberedt sig ved på forhånd at have kikket alle siderne igennem, noteret sig stikord og lavet hjælpeindekser mv., som reducerer søgearbejdet, når du spørger om noget. For det andet bruger Google forfinede statistiske metoder til at afgøre, hvilke svar der er mest relevante, så de vises først.

DASK

DASK var Danmarks første computer. Udviklingen af DASK startede i 1953, da nogle danske ingeniører tog til Stockholm for at følge arbejdet med konstruktion og programmering af dens forbillede, den svenske computer BESK. DASK blev under betegnelsen elektronhjerne første gang præsenteret for offentligheden ved en demonstration i Forum i september 1957 og blev et par år senere for alvor offentlig kendt, da den ved folketingsvalget i 1960 blev brugt til at forudsige og analysere valgresultatet. DASK lignede i sin konstruktion sine to forbilleder, IAS computeren ved Princeton University og BESK i Stockholm. DASK var baseret på radiorør og havde et ganske enkelt maskinsprog. Senere blev en ALGOL-oversætter implementeret. Dens lager bestod af et ferritkernelager suppleret med et tromlelager. Indlæsningen af data foregik på papirstrimmel, udlæsningen på papirstrimmel eller på elektrisk skrivemaskine. I 1958 blev der koblet en enhed til DASK, så hulkort kunne bruges både til ind- og uddata. I årene 1959-60 blev der endvidere etableret tilslutninger til fire magnetbåndstationer. Dele af DASK, der i sin helhed fyldte en hel spisestue og vejede 3,5 t, kan ses på Teknisk Museum i Helsingør.

GIER

I 1961 blev den anden danske datamat, GIER, færdigudviklet og sat i drift. GIER, der står for Geodætisk Instituts Elektroniske Regnemaskine, blev udviklet i et samarbejde mellem Regnecentralen og Geodætisk Institut, og var en typisk andengenerations computer, det vil sige baseret på transistorer frem for radiorør. Der blev til GIER udviklet en meget smart ALGOL-oversætter, der blandt andet gjorde GIER 10 gange hurtigere end DASK. Samtidig var GIER langt mindre end DASK. GIER blev oprindelig udviklet som en ren prototype, men opnåede at blive produceret i 50 eksemplarer, herunder adskillige til eksport.

Om forfatteren
Christian Gram
Bondehavevej 135
2880 Bagsværd
e-post: chr.gram@ddf.dk

Hjemmesider:
www.computer.org/history
www.dr.dk/videnskab/tidslinier
www.datamuseum.dk

H.B. Hansens side om DASK
www.dat.ruc.dk/~hbh

En bog:
Heide, Lars: Hulkort og EDB i Danmark 1911-1970. Århus: Systime, 1996.