Informatik = Matematik $\oplus$ Datalogi $\oplus$ Anvendelse

Dines Bjørner, IT/DTU
Sektion for Programmelsystemer

Indhold

Ord som informatik hører vi sikkert ofte. Men har vi en klar forståelse heraf? Og bruger vi alle denne term med samme indholdsforståelse? Sikkert ikke. Et ord som datalogi bruges alt sjældnere. Og når vi så går videre og bruger det meget lange ord informationsteknologi, ja hvad mener vi så egentligt? Er det teknologi uden videnskab? Skal ingeniører, og da især dataloger, og som vi her vil fremføre det, informatik-ingenører, kunne forske, og hvis så, da hvor meget, og hvad med matematikken og anvendelserne?

I denne populariserende artikel vil jeg forsøge blotlægge hvad det er vi dataloger beskæftiger os med, hvad vi som forskere studerer, som lærere underviser i, og hvorledes vi gerne så vores kandidater praktisere deres ingeniørkunst.

Artiklen er et bidrag til forståelse af faget datalogi og af hvorledes jeg opfatter og gerne vil bidrage til DTUs informatikprofil. Artiklen bunder i et behersket syn på faglighed og professionalisme.

1 Baggrund

Min baggrund for at fremføre alt dette — og på nuværende tidspunkt — er blandt andet følgende:

Jeg er just vendt hjem fra en fem års orlov fra DTU (mere herom siden) og ser derfor med noget andre øjne på DTU's situation, og da især Datalogiens rôlle på DTU. Derudover flg.:

Jeg har arbejdet ved IBMs udviklings– og forskningslaboratorier i Sverige, USA, Schweiz og Østrig i årene 1962–1975 dog afbrudt af et licentiatstudie (1965–1969) hvor jeg dog af og til arbejdede for IBM i Schweiz. Jeg var en hovedkonstruktør af IBM datamaterne IBM 1130 og IBM 1800 (i en vis forstand vel de første RISC maskiner), og også af andet maskinel (IBM 1070). Jeg arbejdede med bl.a. John W.Backus (skaberen af Fortran) —

som den første han arbejde med i over 10 år — og realiserede de første fortolkere for såkaldte funktionssprog, med E.F.Codd — "opfinderen" af relationsdatabase konceptet — og "arkitektede" den fysiske datalagerstruktur for relationsdatabaser, og med en gruppe af forskere i Wien på konstruktion af en PL/1 oversætter baseret på en helt ny programmelkonstruktionsmetode: VDM. VDM står for Vienna Development Method.

VDM er principielt baseret på at udvikling af oversættere [generelt: programmel] sker i tre trin: Først dokumenteres en matematisk semantik af sproget [en matematisk model af centrale domæne-termer] (mere herom siden). Vi vil her karakterisere det at konstruere en sådan sprogsemantik [domænemodel] som at definere domænet. Kravspecifikationer til oversætteren [domæne-anvendelsen] formuleres siden på grundlag af, dvs. med reference til domænebeskrivelsen. Endelig udvikles oversætteren ved en yderligere afledning fra kravspecifikationerne. Vort ærinde i Wien var at sikre at oversætteren genererede korrekt kode [at anvendelsesprogrammelsystemet korrekt støtter og automatiserer udvalgte domænefunktioner]. Hvad det vil sige at være korrekt vil blive et centralt emne i denne artikel.

Efter Wien blev jeg professor ved ID/DTH[1]. I 1980rne var tiden klar til videre udbredelse af Wiener-metoden i dansk industri. Dette blev gjort gennem Dansk Datamatik Center (DDC). Jeg var (1979–1988) den videnskabelige og projektmæssige hovedkraft i DDC. Ved professor Asger Kierbye Nilsen's mellemkomst var vi først istand til at konstruere en formel beskrivelse af CCITT (telefoncentral) programmeringssproget CHILL og siden den første, og vel stadigt den eneste "fuld" Chill oversætter, dernæst af US DoD's programmeringssprog Ada. Lektor Hans Bruun sikrede, gennem sit omhyggelige forskningsarbejde, at både CHILL og Ada projekterne kunne gennemføres. Teleteknisk Forskningslaboratorium (tfl) overtog siden CHILL arbejdet, og kommercialisering af Ada oversætteren overførtes til DDC Intl. DDC Intl. arbejder primært i USA hvor man har 30 ansatte. Herhjemme er der 30 ansatte. Har man gode metoder behøver man ikke megen (som regel dyr) arbejdskraft. Min kollega, professor Jørgen Fischer Nilsson (JFN), dokumenterede nogle fine idéer vedrørende en (typet) variant af Prolog vha VDM og udviklede eksperimentelt en Prolog oversætter. Med hjælp fra docent Tom Østerby overførte tre vakse studerende JNFs oversætter til PC verdenen. De tre studerende skabte firmaet Prolog Development Center (PDC). Deres Prolog system blev i mange år forhandlet af Borland som Turbo–Prolog. En stor success. PDC beskæftiger idag omkring 50 medarbejdere i Rødovre og godt 20 i Skt. Petersborg (Rusland). Dansk Datamatik Center måtte lukke i 1988 pga. fejlsatsning indenfor hvad der kunne betegnes som "Kunstig Intelligens" og "Bruger Datamatik". På det tidspunkt havde de DDC ansatte, der stod bag CHILL og Ada igangsat et større internationalt forsknings & udviklingsprojekt RAISE. RAISE blev videreført i den del af DDC der arbejdede herpå og som overgik som een gruppe på omkring 15 personer til CRI A/S. RAISE er en matematisk baseret metode til udvikling af større programmel, et specifikationssprog (RSL: RAISE Specification Language) med tilhørende bevissystem og et sæt værktøjer. RAISE står for Rigorous Approach to Industrial Software Engineering. Jeg "flyttede" med til CRI som videnskabelig rådgiver, så RAISE blive fuldført og var med til at sikre at CRI A/S sammen med et engelsk firma fik et efterfølger-projekt LaCoS. I LaCoS blev RAISE konceptet afprøvet i syv industriprojekter rundt om i Europa: Danmark, England, Frankrig, Grækenland, Italien og Spanien. Ved DDC Intl., ved tfl, og ved CRI blev der i 1980rne ansat vel omkring 80 af ID/DTH's bedste kandidater.

[1] — hele ID/DTH indgik for nogle år tilbage i det da nyoprettede IT/DTU. Herefter benyttes betegnelsen IT/DTU for "ID/DTH" forhold efter 1994

De fleste er der stadigvæk eller er overgået til Maconomy og The Vision Factory, som begge er meget successrige firmaer startet af ligeledes vakse ID/DTH kandidater. Sammen med en af verdens abslut førende datalogiforskere, professor C.A.R. Hoare (Oxford Universitetet, England) startede og ledte jeg, indtil min afrejse til Macau, et af de mest vellykkede ESPRIT BRA projekter: ProCoS. Det står for Provably Correct Systems. Andre IT/DTU kollegere (Michael Reichhardt Hansen, Hans Henrik Løvengreen, A.P.Ravn og Hans Rischel) førte lokalt ProCos videre — og det er idag internationalt anerkendt som et af de absolut ledende projekter indenfor forståelse af krav til og udvikling af såkaldte indlejrede systemer. Dvs. programmel til sikkerheds– og tidskritiske systemer. I dette projekt deltog også Prof. Zhou Chaochen, det Kinesiske Videnskabernes Akademi, Peking, og nu min efterfølger som direktør for UNU/IIST — se siden. Zhou's bidrag blev helt afgørende for ProCoS's store success. Endelig er min baggrund for denne artikel at jeg i de sidste fem år har bygget et idag internationalt anerkendt forsknings– og "post–doctoral"– center, UNU/IIST, for programmel–teknologi. UNU/IIST står for United Nations University's International Institute for Software Technology. Det beskæftiger idag 6–7 forskere, 8 tekniske-administrativt personale og er til stadighed vært for omking 24 post–graduates and PhDs. Ved dette center — som nærmere omtales in en særskilt artikel — har der i fem år været forsket på netop den baggrund som resten af artiklen vil gøre nærmere rede for!

Det er således med en rimelig solid baggrund i anvendt datalogisk forskning, med over 50 eksamensprojektkandidater, over 10 PhD kandidater, knap 60 publicerede artikler, 2 medforfattede bøger, redaktion af 12 andre, arrangør af over 40 internationale videnskabelige konferenser, heraf flere som "præsident", og med en stor industrierfaring, at jeg fremsætter den følgende beskrivelse af disciplinerne datalogi, praktisk og teoretisk, og af informatik.

2 Ingeniørkunst

Programmel er een blandt mange former for teknologi. En række kritiske programmelsystemer udvikles, som anden kritisk teknologi, af ingeniører. Som de fleste sådanne ingeniører bygger også dataloger på videnskabeligt opnåede resultater og ved hjælp af tilsvarende opnået metodik. Så på den ene side har vi videnskaben og på den anden side teknologien. Imellem dem arbejder ingeniøren: bygger en tovejs bro mellem dem: fra videnskab til teknologi i konstruktionsarbejdet, fra teknologi til videnskab når nye problemer skal erkendes. En teknolog, sådan lidt sat på spidsen, er en tidligere ingeniør hvis virke ikke mere kræver faglig, videnskabelig indsigt og brug af videnskabelige metoder. En tekniker er en person der konstruktions– eller vedligeholdelsesmæssigt beskæftiger sig med teknologi uden at skulle referere til videnskaben.

Den videnskabelige, eller i det mindste formelle, essens i datalogens og informatikingeniørens arbejde er forståelse for og brug af matematik. I selve det datalogiske — som vi lige straks skal indkredse — er det ikke naturvidenskaben, men alene matematikken der er vigtig. Selvfølgelig er der mange som mener at datalogen og informatikingeniøren også bør besidde andre færdigheder udover datalogi og matematik. Der er de der mener at folk der beskæftiger sig med informatik også bør beherske et minimum af psykologi, sociologi mv. Det kan være — men det er ikke den eventuelle mangel på indsigt heri, der har slået mig som værende en væsentligste årsag til dårligt programmel. Ihvertfald vil dette aspekt ikke blive nærmere behandlet i denne artikel. Forstå det ret: Der kan helt sikkert være mange spændende aspekter ved disse andre discipliner set i forhold til informatik, men vi overlader det til andre datalogiske fagfolk at plædere vigtigheden og udforske emnet videnskabeligt.

3 Datalogi

Vi deler faget datalogi op i to overlappende under-discipliner: Teoretisk datalogi (computer science) og datalogisk metodelære (computing science eller programming methodology).

Teoretisk datalogi er læren om og studiet af egenskaberne ved de "størrelser" der kan eksistere inde i datamaskinen.

Datalogisk metodelære er læren om og studiet af hvorledes man konstruerer disse "størrelser".

Og hvad er da "størrelserne"? Traditionelt opfatter vi dem som 'data' og 'beregninger' (computations, processer).

Dette er da hvad en datalog beskæftiger sig med. En dataingeniør opfattes af mig som en form for informatik–ingeniør idet vedkommende sidder "overskrævs" på to heste: datalogi og digital elektronik. I den forstand er, eller var jeg selv dataingeniør.

3.1 Teoretisk Datalogi

Teoretisk datalogi beskæftiger sig derfor med modeller for hvad der kan beregnes og de mange måder man kan beskrive data og beregninger.

3.1.1 Beregnelighed

Først er der selve begrebet beregnelighed. Hvad kan (og hvad kan ikke) beregnes. Hvis vi til en begyndelse alene ser på endelige beregninger — såsom de der normalt, når alt går vel, foreskrives af f.eks. Fortran programmer, ja så ser vi at dette spørgsmål blev hypotetisk besvaret i 1930rne. Da foreslog den engelske matematiker Alan Turing og den amerikanske matematiker Alonzo Church forskellige modeller for "beregnings-maskiner". Turing's blev kendt som Turing Maskinen, Church's som λ–kalkylen. Alt hvad der kan beregnes kan beregnes, blev tesen, enten vha. Turings Maskine eller ved evaluering af et udtryk i λ–kalkylen. Turing's og Church's arbejde falder indenfor meta-matematikken som igen måske bedst kan forstås som en afledning fra den matematiske logik. Derfor er en rimelig dyb indsigt i meta–matematik — nærmere bestemt den rekursive funktionsteori — en vigtig forudsætning for den teoretiske datalog.

3.1.2 Processer

Foruden endelige beregninger er der også de principielt set uendelige beregninger som vi forventer fra det operativsystem der administrerer de fleste datamater. Vi skal her kalde dem processer. Det interessante ved processer er at man kan sammensætte to eller flere processer og derved opnå een process, eller omvendt: at man kan dissekere een process ned i mange underprocesser. Og da opstår behovet for synkronisering og kommunikation mellem processer og for hvilke former for sammensætning man kan operere med: "ægte" parallellitet (samtidighed) mm. Hvor to (eller flere, N) processer begge er istand til at synkronisere med og bevirke kommunikation med en 'tredie' process (for eksempel een blandt M andre processer) — som i givet fald godt kan være een af de N processer — opstår derfor spørgsmålet: hvilke processer udvælges, er selekteringen deterministisk eller såkaldt ikke--deterministisk, afgøres valget udefra processerne (eksternt), eller indefra, blandt processerne: de N+M? Teoretiske modeller som Petri Net og Process Algebraer udgør et hovedemne i den teoretiske datalogi. Studiet af disse benytter sig af matematik som sit vigtigste værktøj — hvor modellerne selvfølgelig er motiveret i rent fysiske forhold.

Der finder en vigtig forskning sted ved IT/DTU af dette område (lektor Hans Henrik Løvengreen).

• • •

Dernæst er der de mange specialdiscipliner indenfor den teoretiske datalogi som beskæftiger sig med beskrivelse af data, beregninger og processer. Man kan dele disse op i model– hhv. bevisteoretiske beskrivelser.

3.1.3 Typeteori

Typeteori er oprindelig udsprunget fra ønsket om at beskrive de langt rigere universer af data man finder i programmer end man finder f.eks. i fysikken. Typeteori opstod således udfra ønsket om at hjælpe programmørerne med at konstruere fejlfri programmer ved først at typebestemme alle data, beregninger og processer — før programmøren beskrev beregningernes algoritmiske, eller processernes opførselsmæssige forløb. Men typeteori er gået langt videre. Som man kan løse mange opgaver i fysik ved almindelige dimensionsbetragtninger, således kan for eksempel udtryk i intuitionistisk typeteori beskrive beregninger. Man må nok idag konstatere at datalogisk typeteori er en af den teoretiske datalogi's fundamentalt nye bidrag til videnskaben — og til matematikken?

3.1.4 Bevisteoretiske Beregninger

Logik programmering, dvs. beskrivelse af løsning på et beregningsproblem alene vha. logiske udsagn i en passende logik, er et centralt emne i datalogien. Der findes mange logikker: udsagnskalkylen (propositional logic), første ordens prædikatkalkylen (predicate calculus), flere modal-logikker såsom (på engelsk): belief, deontic, default, defeasible, [auto–] epistemic, non–monotonic, possibilistic, og andre trænger sig på. Som termen logik antyder bunder studiet af disse data– og beregningsbeskrivelsesformer sig på logikken. Og matematisk logik er derfor en grunddisciplin for alle dataloger.

IT/DTU kan mønstre en effektiv forskning også indenfor dette område (professor Jørgen Fischer Nilsson).

3.1.5 Algebraisk Semantik

Modularitet i strukturen af programmer — de være sig de traditionelle, såsom Pascal's efterfølger som for eksempel Modula, C++ og Java, eller mere utraditionelle, som for eksempel Standard ML — forstås som et begreb blandt andet ved hjælp af algebraiske modeller. Også selve en successivt leddelt programudvikling, med dens første trin af abstraktion efterfulgt af senere trins konkretisering, kan med fordel forstås algebraisk. Studiet af algebraisk semantik er derfor vigtigt for alle dataloger. Dette studie betjener sig af ikke--trivielle resultater fra blandt andet universal algebra og kategoriteori, gode klassiske matematiske discipliner.

Ved IT/DTU har især forskningsadjunkt Anne Haxthausen bidraget med en fin række arbejder indenfor dette basale område.

3.1.6 Denotations Semantik

Church's λ–kalkyle viste sig at være den model for beregnelighed som lå og ligger nærmest de programmeringssprog vi kender som Algol 60, Pascal, m.fl. Man kunne faktisk forklare f.eks. Algol 60 ved omskrivninger til λ–udtryk. Da λ–kalkylen indtil 1970 kun kunne forstås 'mekanisk': ved såkaldt reduktion (dvs. ren syntaktisk omskrivning) af udtryk til en ikke-reducérbar form (hvis den da eksisterer), ja så blev det magtpåliggende at forstå også λ–kalkylen ved dens indlejring i den klassiske matematik. Dette skete i 1970 (Dana Scott). Samtidig kunne Scott's opdagelse bruges dels til at beskrive kommercielle programmeringssprog præcist for derved at opnå mere pålidelige oversættere, dels i ganske almindelig udvikling af generelle programmer. Dette sidste blev "opdaget" og gennemført ved IBM's Wiener Laboratorium omkring 1973.[2] Hvor algebraisk semantik som tyd af udsagn knytter algebraer, der knytter denotations-semantikken — som Scott's arbejde blev kaldt — en tyd i form af en matematisk funktion. Denotations-semantik er derfor en hjørnesten i enhver datalogs uddannelse og arbejde. Scott benyttede, i sin løsning af hvilken fikspunkt-tyd rekursivt formulerede λ-udtryk bestemmer, sig af gitterteori, og siden er mange andre former fra ikke-

[2] Heraf fulgte VDM som nævnt i afsnit 1.

-standard analyse blvet taget i brug. Den teoretiske datalog forventes derfor at have en stærk befæstning i gitterteori, Hausdorfske topologier, metriske rum, mv.

Ved ID/DTH blev der i 1980rne ydet væsentlige bidrag til denotationssemantikker for industrivendte programmeringssprog (CCITTs CHILL og US DoDs Ada) og for det første internationalt (ISO) standardiserede specifikationssprog VDM–SL. Uden først lektor Hans Bruun's og siden lektor Bo Stig Hansen's banebrydende arbejder ville ingen af disse tre sprog idag have en fornuftig formel semantik — og ihvertfald ikke nogle der var lavet i Danmark! ID/DTH opnåede herved en internationalt anerkendt førerplads på området.

3.1.7 Operationel Semantik

Til forståelse af semantikken af bl.a. programmeringssprog til beskrivelse af maskin--baserede (dvs. fysisk bestemte) systemer benyttes nok en beregneligheds-teknik kaldet operationel semantik. Operationel semantik kan siges at være nært beslægtet med bevisteori. Dataloger som arbejder med parallelle, samtidige og distribuerede systemer har derfor brug for et godt kendskab til operationel semantik.

3.1.8 Algoritmelære og Komplexitet

I begyndelsen var datamaskinen! Og det meste arbejde omkring den bestod i at skrive "små" (Fortran, Algol 60 og andre) programmer til løsning af tekniske-videnskabelige beregninger — sådan som vel er de fleste af de programmer der udvikles på DTU! Algoritmelære — det at vide hvorledes man konstruerer programmer, og at sikre sig at sådanne "små" programmer var korrekte og at de førte til rimeligt resoursse-venlige beregninger — er derfor en vigtig grundpille i datalogien. Opdagelsen af at man faktisk kunne knytte et kompleksitets-tal til bestemte problemstillinger (såsom søgning. sortering, sproggenkendelse, kryptering, mmfl.) må betragtes som en ganske genial opdagelse (Hartmanis m.fl.).

De fleste ansatte på DTU betragter datalogi, og dermed det som de fleste på det tidligere Institut for Datateknik på DTH bedrev (og som mange nuværende på Institut for Informationsteknologi bedriver), som i det væsentligste det at programmere, dvs. nærmest at ligne med algoritmelære. Som jeg forsøger vise ovenfor, og siden, mere, nedenfor, er dette en noget forenklet opfattelse. Hvis den er generel blandt de toneangivende på DTU, i forskningsråd mv., ja så har Institut for Informationsteknologi et meget stort kommunikationsproblem.

• • •

Relationer mellem de forskellige teorier er et vigtigt emne i den teoretiske datalogi — idet man i forskellige situationer af beskrivelsen af store beregnings– og process-systemer benytter sig af samstemmende teknikker der så skal relateres. For eksempel er korrelation, eller kongruens, mellem model– og bevisteoretiske beskrivelser et vigtigt emne idet programmer oftest udvikles model-teoretisk, men påvises korrekte bevis-teoretisk.

Ovenfor har jeg nævnt blot de vigtigste teoretisk datalogi-emner. Mange andre kunne nævnes.

Det er kendetegnende for studier i teoretisk datalogi at de kan afgrænses: forskningsemner har en tendens til at gå meget i dybden, og jo dybere des mere fjernt kommer de at ligge fra informatikingeniørens dagligdag. Og måske kommer visse emner og resultater til slet og ret at falde ved siden af: blot være en frustreret matematikers forsøg på at tiltrække sig opmærksomhed og at sikre sig et job i det ellers populære datalogi-område. Datalogens eneste virkelige berettigelse er at konstruere programmer der beskriver brugbare beregninger på datamaskiner. Om dette handler næste afsnit.

3.2 Metodelære

Ved en metode forstås
En række principper for analyse,
udvælgelse og brug af teknikker og værktøj
til effektivt at konstruere en effektiv genstand.

Hvor genstandsområdet i teoretisk datalogi er de "størrelser" der kan eksistere inde i datamaten, indbefatter genstandsområdet i datalogisk metodelære også de personer der skal konstruere datamatikken. Dette forhold kan nemt føre til et "skred": at man istedetfor at interessere sig primært for data–, beregnings– og processkonstruktion fokuserer på den menneskelige størrelse. Det sidste er, især i vores navlebeskuende, selvrealiserende tider, måske vigtigt, men det ville være katastrofalt hvis en programmørs eller brugers stort set ikke--definérbare trivsel sker på bekostning af en raket der eksploderer, et tog der kører ind i et andet, et skattetræk der medfører telefonstorm og lignende.

Vi skal derfor, i god overenstemmelse hvad vi er uddannet til, i det følgende koncentrere os om det rent faglige i datalogisk metodelære. Men helt glemme mennesket vil vi så sandelig ikke gøre — al den stund metodelærens forskrifter skal tale til dette mennneske hvis kreativitet ikke gerne skulle begrænses men snarere tværtimod: gennem gode principper, teknikker og værktøj at give ingeniøren mulighed for at koncentrere sig om det kreative, herunder de menneskelige hensyn.

3.2.1 Programmeludvikling

Lad os antage at vi skal udvikle programmel der skal koble alt DSB's (Banestyrelsen, DSB Passager og DSB Gods's) strategiske, taktiske og operationelle funktioner sammen: fra de der blot understøttes til de der automatiseres.

For at kune gøre dette er det rimeligt først at sikre sig at informatikingeniøren ved DSB (etc.) forstår alle de fagtermer som berører de ønskede programmel. Hertil kræves at udviklerne først konstruerer en del-semantik for alle disse termer (af visse kaldes dette en 'ontologi'). Termerne dækker over individder (observerbare tilstande mv.) og fænomener (begivenheder, handlinger, forløb). Jeg kalder dette for domæne–arbejdet, andre for videnindsamling og bearbejdning, atter andre for 'ontologi'.

For dernæst at sikre sig at kundens og brugernes krav indkredses på bedst mulig måde, konstrueres en såkaldt kravspecifikation. Denne bygger på domæne-modellen. Denne sidste valideres vis-a-vis kunden og brugerne.

Endelig fastlægges programmellets arkitektur: de ydre grænseflader der bestemmes af brugerens eller andet programmels' eller andre maskiners brug, og programmellets struktur: de indre grænseflader der bl.a. tilsikrer modularitet, programmets evne til at blive flyttet mellem datamatiske "platforme", til at videreudvikles (evolvere), dets robusthed, sikkerhed, fejl-tolerance, mmm. Til slut sker der yderliere en række trin af forfining af programmet, fra arkitekturens og strukturens abstraktioner til den eksekverbare kodes konkretiseringer.

I de næste tre afsnit vil vi nærmere beskrive forhold i forbindelse med denne tredelte udviklingsprocess.

3.2.2 Domæne–ingeniørkundskab

Når Sir Isaac Newton undersøger det nærmeste univers og opdager tyngdekraften oa. fænomener og så beskriver disse matematisk, ja da er det at han bedriver domæneanalyse. Beskriver det som er, eller måske snarere som synes at være. Newtons love blev ikke udviklet for siden at kunne formulere krav til mekaniske konstruktioner, men for at forstå dette nærmeste, inklusive det planetære, univers. For at kunne "regne på det" og forudsige effekt. Tilsvarende, før informatikeren skal udspørge kunder og brugere om hvad de forestiller sig at deres indforskrevne datamatik skal kunne yde, hvilke faciliteter det bør tilbyde — før dette — skal der først opstilles en teori for, en model af, domænet.

Eksempler på et lidt større, og derfor muligvis et mere interessant, domæne er: Et jernbanesystem (forestil dem DSB for 2–3 år tilbage). *Hvad er DSB?* Hvorfor ønsker vi at stille dette spørgsmål? For det første fordi DSB bruger datamatik. Her og der og alle vegne. For det andet fordi DSB stedse ønsker at sammenkoble denne datamatik i alt større, sammenhængende systemer for derved bedre (i) strategisk at styre sine resoursser (opgradering, nedlæggelse "downsizing"), for bedre taktisk at planlægge disse spatialt (allokering) og tidsmæssigt ("schedulering"), og for bedre operationelt at tage disse i brug (afsending af tog, varetagelse af gods, salg af billetter, etc.). Jo mere disse mange meget forskellige begreber griber ind i hver andre, desto større et behov er der for at opnå en "total" forståelse. For blot tilnærmelsesvis at skitsere problematikken lad mig nævne nogle af komponenterne i en domæneteori for DSB. Vi kunne for eksempel begynde med det faste inventar: jernbanenettet. Af hensyn til signallering skal netbeskrivelsen bygges op fra de mindste dele: sporene, de ubrudte (lige eller kurvede), sporskifter, kryds, mv. Dernæst disses sammensætning til stationer og linier mellem stationer. Indenfor stationer er vi interesseret i både passager og godstrafik. Derfor skal de spor beskrives hvor passagertog holder (perroner og klargøring) og hvor godstog laster og losser gods. Også de særlige rangérarealer skal beskrives. Enkle og sammensatte sporbeskrivelser er dels statiske: Hvorledes hænger det hele sammen?, dels dynamiske: Hvorledes opfører tog sig? Beroende på tilgangsvinklen og abstraktionsniveauet beskrives signalering og tilstand af sporskifter, eller dette beskrives ikke. Det var nettet — og der er meget at tage vare på! Så til trafikken: togs dynamiske positioner på sporlegemet. Vores beskrivelse bør ikke idealiseres: også togsammenstød skal modelleres for at vi siden kan kravspecificere, ja blot tale om det uønskelige i dette. Dernæst tilkommer så f.eks. passagersiden: rejseforespørgsler, billet og pladsreservation, køb, annullering og brug (eller ikke–brug!). Tilsvarende for gods: Hvor befinder et specifikt stykke indskrevet gods sig pt? Vi "domæner" videre: Regler og forskrifter for afsending af tog, for varetagelse af gods, mv. Allokering og schedulering af jernbaneresoursser: de flytbare såvelsom de stationære: de materielle såvelsom staben. Og videre igen: Udbygning eller indskrænkning af disse resoursser: Planlægning af nye tjenester, linjer, stationer, spor, tog, time tabeller, osv. Nedlæggelse af samme. Osv. Vi begynder at se billedet. Et stort system, megen naturlig tekstbeskrivelse, mange matematiske formler. Og alt dette skulle vi gerne have på plads, valideret og dokumenteret, før vi kan gå videre med DSB og tale om dets IT Strategi! Dvs. også før vi kan tale om krav til bestemt datamatik. Og for at sikre os at vi har forstået dele af hvad DSB er ("informatisk" set) bør vi også "lige" have afklaret et passende antal mere teoretiske forhold såsom: "Kirschoffs lov for jernbanedrift": Over et passende, af problemstillingen, afledt tidsinterval bør antallet af tog der ankommer til en station, minus det antal tog der, i dette tidsinterval, afslutter deres rejse ved denne station plus de der påbegynder en rejse være lig det antal der forlader denne station i dette interval. Eller: ("Gud spiller ikke med terninger"): to tog på vej ned af en linje, f.eks. i samme retning, kan ikke pludselig byttes om (således at det tog der før var forrest nu bliver bagerst)! Eller (ingen "spøgelsestog"): Hvis et tog er på en banestrækning til tiderne t og t', hvor f.eks. t' er blot nogle få sekunder senere end t, ja da vil toget være på denne banestrækning til enhver tid mellem t og t'.

Ovenfor har jeg alene koncentreret mig om det basale i domænet. Dertil kommer dets allerede eksisterende teknologiske understøttelse. Åbne og lukkede banestrækninger er markerede, ikke ved bomme, men bl.a. ved tilsvarende åbne eller lukkede signaler. Signaler er blot een måde teknologisk at realisere spors brugstilstande. Tilsvarende: Hvorfra véd vi om togs positioner? Måske fra optoelektriske observationer. Og også sådant udstyr kan svigte (og vi har reelt set "spøgelsestog"). Så alt, hvad der findes af tilfældige, tidsbestemte måder at understøtte jernbanedriften på, skal ligeledes beskrives. Og vi er slet ikke kommet til datamatikken endnu.

Det er klart at beskrivelsesværktøjerne: Specifikations– og definitionssprog skal kunne meget. Meget mere end den klassiske analyse kan tilbyde. Ihvertfald tilbyde på en måde, der gør det forholdsvist enkelt at bruge og ekspressivt at læse. Fra sprogbeskrivelser skal vi gerne formelt kunne bevise en lang række egenskaber. Så beskrivelsessprogene skal forsynes, hvis de ikke allerede er logiske, med et bevissystem, osv. I den klassiske analyse, og i de varianter der udforskes i f.eks. adaptiv og stokastisk styreteori, kan man beskrive et togs opstart fra en station samt dets ordinære rejse ned ad sporet: langs lige strækninger, i kurver, over broer og gennem tunneller. Under alt dette forventes toget accelerere, decellerere eller køre med konstant hastighed. Men hvad nu hvis elektriciteten svigter og motoren standser, eller en ko kommer på tværs af skinnerne? Ja, her er det at datalogien kan tilbyde sprog, der tillader en logisk sammenhængende, konsekvent, komplet og konsistent beskrivelse af de mange diskrete tilstandsovergange som klassisk kontinuert analyse ikke specielt er indrettet på.

Mao.: Domæne–ingeniørarbejdet har mange facetter, betjener sig metodisk af mange principper, teknikker og værkøjer. Ved IT/DTU arbejder forskere i SEKTION FOR PROGRAMMELSYSTEMER på at udvikle domæne–modellerings–metoder. For øjeblikket påbegynder man projekter der forsøger forstå hvorledes man besvarer bl.a. flg. spørgsmål: Hvad er en Bank? (eller, for den sags skyld, en hel finansverden med bank–, forsikrings–, børsmægler– og mange relaterede tjenester). Hvad er Hovedstadens Trafik? Hvad er Lufttrafik? Hvad er en fremstillingsindustri? Det sidste er ikke bare en mindre variant af CIM (Computer Integrated Manufacturing), men omfatter en hel industri med dens (på engelsk) 'suppliers, consumers, producers & traders'. Hele markedet skal med før vi opnår en troværdig model af ethvert enkelt firmas funktioner.

3.2.3 Krav–ingeniørkundskab

Før vi kan begynde at udvikle programmel må vi først, til fulde, forstå kundens ønsker og brugernes forventninger. At opstille krav, som i tilfældet med domæne–modellering, kræver et snævert samarbejde mellem udviklerne (informatikerne) og kunde + brugere. Hvorledes dette snævre arbejde kan foregå skal jeg ikke komme ind på i denne artikel. Istedet skal jeg koncentrere mig om det teknisk–videnskabelige idet jeg samtidig glæder mig over at der er mange andre der studerer ko–operations problematikken.

Ved opbygning af kravspecifikationer projicerer vi, så at sige, en del af domænemodellen ind i datamaten. I kravspecifikationsarbejdet er vi derfor interesserede i at forstå hvilke dele af domænetilstanden, der skal indlæses i maskinen, hvor ofte den skal justeres, sikring af dataløødighed, mmm. Dernæst er vi interesserede i at afbilde — også ind i datamaten — de af domænets funktioner, begivenheder eller opførsler der skal understøttes eller helt eller delvist automatiseres. Udover dette, som vi kan kalde domæne–specifikke tilkommer det programmel–specifikke: Krav til dets robusthed overfor svigt i de tilkoblede fysiske (herunder elektroniske) systemer, dets egen pålidelighed (korrekthed), dets evne at blive flyttet fra værts–maskine til anden, dets evne til at blive videre–udviklet. Endelig, når disse krav er formuleret, da at sikre passende menneske–maskine grænseflader: dialog, "display", og meget andet.

I metodelære for moderne kravspecifikationsteknik indgår at disse specifikationer formelt videreudbygger domænespecifikationerne. Relationer såsom korrekthed i trin af udvikling fra domæne til krav, konsistens, og meget andet, er derfor vigtige forsknings–såvelsom undervisnings–objekter. Grænser for hvad der kan kræves og hvad der idag kan specificeres undersøges også: Hvad forstås ved fejlsikkerhed, pålidelighed, oma., er centrale emner for datalogen.

IT/DTU's SEKTION FOR PROGRAMMEL-SYSTEMER udforsker pt. dette område intenst.

3.2.4 Programmel–ingeniørkundskab

Og endelig er vi da ved den "go'e gamle" programmering. Det som de fleste forstår ved datalogi. Men også her spiller abstraktion og trinvis forfining af sådanne abstraktioner mod konkret, eksekverbar kode, en alt afgørende rolle. Evnen til at abstrahere, og til sikkert, ubesværet og forståeligt at nedskrive abstraktioner, denne evne er helt central for den praktiserede datalog's success. Meget kan læres, men ikke alt. En del tilbagestår som en kunst. I det følgende vil jeg forsøge opridse blot nogle få af de mange overvejelser datalogen må gøre sig i det daglige arbejde, og for hvilken der derfor finder en løbende forskning sted, også på IT/DTU.

- Abstraktion:

  Ved abstraktion forstår vi det at fokusere på visse egenskaber ved en genstand samtidig med at der således ses bort fra andre egenskaber. De første "udnævnes" til at være de væsentlige, de andre at være "mindre betydningsfulde". Ved abstrakt modellering forstår vi det at finde et passende matematisk model eller et passende matematisk logisk udsagn der beskriver den eller de udnævnte, vigtige egenskaber.

  Som dataloger betjener vi os af abstraktion som et bedste middel til at betvinge kompleksitet af systemer. Ved at abstrahere kan vi troværdigt tackle endog de største systemer. Matematikken tilbyder de hidtil eneste midler til utvetydigt at abstrahere.

  Et jernbanespor modelleres f.eks. som et ikke nærmere beskrevet matematisk objekt der siden, algebraisk, kan udbygges med en række egenskaber. Først, f.eks., at sporet kan besidde et antal sammenføjningspunkter. Et par skinner har således to: et i hver ende. Et almindeligt sporskifte tre. Et kryds fire. Sammenføjningspunkterne defineres passende abstrakt således at det er enkelt at tale om kombinationer af skinner, som for et Märklin sæt. Dernæst tilføjes egenskab af en skinnetilstand således at vi kan tale om åbne og lukkede sporstrækninger. Nu indføres tog som værende noget abstrakt noget, der (blot) er positioneret [knyttet], i tid, over [til] en sekventiel (sammenhængende) serie skinner. Osv. Altsammen alene karakteriseret ved matematiske (herunder især matematisk logiske) egenskaber snarere end ved grafteoretiske, kombinatoriske eller operationsanalytiske egenskaber. Disse senere tilkommer som en del af en muligvis afledt 'jernbane'–teoridannelse. Funktioner over det abstrakte domæne (af f.eks. jernbaner) kan nu defineres som almindelige diskret–matematiske funktioner eller ved deres matematisk–logiske egenskaber (herunder ved axiomer).

  Som nævnt tidligere kan man udtrykke abstraktioner algebraisk, denotationsmæssigt, operationelt, etc. Man kan tackle abstraktion på modelteoretiske eller på bevisteoretiske præmisser, og man kan kombinere disse.

  Værktøjer — til udvikling og syntaktisk, og iblandt semantisk, afprøvning, og ihvertfald analyse, af abstrakte specifikationer — udgør en ret så avanceret datamatisk teknologi. Idet abstrakte specifikationer ofte er formuleret i et specifikationssprog, der tillader formulering af ikke–beregnelige systemer opstår således afprøvningsproblematikken: hvor går grænserne for, hvad der kan analyseres.

Dette sidste — og mere generelt, principper og teknikker for konstruktion af abstraktioner — er vigtige emner både i den datalogiske praksis som i dens udforskning.

Ved ID/DTH er der siden 1976 forsket og undervist i abstraktion og det næste emne (forfining). IT/DTU nyder internationalt en meget høj anseelse heri.

- Forfining:

  Abstraktionerne tillader os at separere væsentlige, "evige" egenskaber fra de af den til enhver tid gængse teknologibestemte egenskaber. Forfining fra abstraktioner mod konkrete realisationer er derfor styret udefra af øjeblikkelige teknologivalg og indefra af ønsket om korrekthed i relation til basale, vedvarende egenskaber.

  Der findes i dag flere forfiningsteknikker. Man kan "opfinde" en forfining, eller man kan mekanisk transformere en abstraktion hen imod en konkretisering. De første teknikker benævnes "invent & verify, posit & prove". De sidste "refinement calculi".

  Ved IT/DTU studeres begge facetter af formel udvikling.

  De første kræver gode specifikationsværktøj, de sidste, foruden de forrige værtøj, kræver endvidere gode bevis– eller model–afprøvningsværktøj. Se punkt 3.2.4 nedenfor.

  Som et eksempel på en forfining kan vi anføre flg.: som information kan man abstrahere en telefonbog som en matematisk funktion, en afbildning, fra abonnenters navne til deres adresser og telefonnumre. I en første konkretisering, i form af en algebraisk bestemt (dog stadig abstrakt) data type, opfattes en telefonbog som et binært træ. I roden af et vilkårligt træ er anført een abonnent, og disse abonnenter er spredt ud over det binære træ således at abonnenter i "venstre" undertræer kommer før, i en lexicografisk ordning, abonnenten i træets rod, og abonnenter i "højre" undertræer kommer efter. Man kan intuitivt forestille sig denne abstrakte data type — hvis operationer (aflæs, indsæt, fjern, m.fl.) vi ikke skal berøre — som det vi allerede har kaldt det: træer. Og vi kan derfor tegne et sådant, tilfældigt træ. Men bemærk: det er ikke et træ. Der er ikke noget lagerbegreb knyttet, endnu, til disse træer. Således er der ingen repræsentation for træets rødder, endsige grene. I næste trin bevæger vi os nu mod den teknologi, der muliggør at vi kan tale om lageradresser (pegere, lænker). Det gør vi ved at opfatte enhver rod i træet og dets undertræer som en post (record) med bl.a. følgende felter: abonnentdata (navne mv., adresser og telefonnumre), og "venstre" hhv. "højre" lænker. Disse peger på rødder i undertræer eller er "tomme" pegere, dvs. at roden er et blad! I dette trin af forfining kan vi opfatte også pegerne som algebraiske størrelser der, ligesom de almindelige operationer, er underkastet axiomer. Sluttelig afbilder vi denne abstrakt formulere model af en konkret datastruktur ind i det valgte programmeringssprog's datastrukturer.

  Vi har antydet en fire trins forfining. Ofte kan en udvikling af en domænemodel indebære typisk et dusin forfiningstrin, en kravspecifikation tilsvarende et dusin trin, og endelig kan selve programmelkonstruktionen indebære op mod en snes sådanne trin. Vi taler altså typisk af størrelsesordenen 30–50 niveauer af abstraktion!

- Korrekthed:

  Korrekthed er ene afgørende. Hvor often hører vi ikke om at der er fejl i programmellet og at disse, foruden dybt at irritere dets daglige bruger, kan afstedkomme at raketter eksploderer, at operationspatienter dør under narkose, osv. Når man så betragter at det antal tilstande et mellemstort programmelsystem kræver 30–40 cifrede tal for at udtrykke deres antal, ja så kan man hurtigt aflede umuligheden af at afprøve sådant programmel for fejl. Det er faktisk nemmere — kan man med større rimelighed påstå — at udvikle det således at det apriori er korrekt: gennem at forstå domænet, forstå kravene og ved trinvist at udvikle disse og programmellet kan man komme et langt stykke. Det meste af resten kan man komme ved at føre formelle beviser og ved såkaldt model–afprøvning. Formelle bevismetoder kan mere end model–afprøvning. Model–afprøvning er mere eller mindre automatisk hvorimod et formelt bevis som oftest kræver at datalogen selv opstiller bevisstrategier og –taktikker.

Hvad der her er sagt lyder meget optimistisk. Virkeligheden, idag, 1997, er dog knap så rosenrød. Meget kan opnås hvad angår korrekthed, men fuld sikkerhed får man muligvis kun hos paven i Rom!

Man kan sige at korrekthed er et af datalogiens vigtigste hensyn: programmels fejlfrihed og teknikker og teorier herfor. Området bygger på både teoretisk datalogi og metodelære.

Værktøj til hjælp ved udvikling af beviser, subsidiært deres fulde automation, såvelsom model-afprøvningsværktøj hører til blandt den teknologisk set mest avancerede datamatik idag.

Der forskes mao. intensivt i dette område, også ved IT/DTU. Flere PhD afhandlinger omhandler dette.

- Troværdighed: 'Dependability'

  De flg. definitioner er ret så diskutable, men tjener til at påpege vigtige forskningsovervejelser.

  En lang række øvrige metodelære og systemanalyse principper, teknikker og værktøj er nødvendige for at sikre bl.a. følgende programmel– & systemfacetter:

  – Pålidelighed 'Reliability'

    Et systems (inklusive dets programmels') pålidelighed måles ved den sandsynlighed systemet har for at fejle — målt f.eks. i relation til antal brug af systemfunktioner per år.

  – Sikkerhedskritisk: '... Criticality'

    Her drejer det sig om at programmel, pga. svigt i tilgrænsende fysiske systemer, ikke, i sig selv medfører at det misforstår, herunder ikke opfatter disse svigt, med tab af menneskeliv eller ødelæggelse af vigtige resourcer.

  – Tilgang 'Availability'

    Dvs. at programmelafvikling vedbliver selv i, på anden vis frembragt, fejlbehæftede situationer. Tilgangsproblematikken sikres som oftest ved replicering af maskinel.

  Troværdigheds–facetter udgør et sandt eldorado for udforskning. Dels er der et stort økonomisk betinget behov for sikrere metoder, dels er emnerne rimeligt skarpt afgrænsede.

  Med ID/DTH's ProCoS projekt er der ved DTH, nu ved IT/DTUs Sektion for Datamatbaserede Systemer, og særlig begrundet i professor Zhou Chaochen's, lektor Hans Rischel's og docent Ander P. Ravn's arbejde, forsket og undervist i dette helt centrale område. Problematikken er langt fra afklaret.

- Generel Sikkerhed: 'Secure'

  Lad os postulere begrebet ikke–autoriserede brugere.

  Et datamatisk system er ultimativt sikkert såfremt en ikke–autoriseret bruger som forsøger at bruge systemet (i) ikke kan finde ud af om systemet tilbyder bestemte faciliteter (eksternt), (ii) ikke kan finde ud af hvorledes systemet er opbygget (internt), og (iii) ikke ved om han eller hun ved dette! Det sidste indebærer at et sikkert system "bluffer" ikke–autoriserede brugere — for ellers ville de vide noget, hvormed de eventuelt kunne bryde systemets sikkerhed.

- Distribution:

  Store domæneanvendelser er i sig selv fysisk, dvs. geografisk vidt fordelt, og visse anvendelser benytter sig af sådan distribution for at sikre troværdighed.

  Distribution er et svært forskningsområde, men det er vigtigt. Ved IT/DTU udforskes og undervises der i distribuerede systemer, bl.a. i forbindelse med næste emne.

- Parallellisme:

  Parallellisme er et centralt emne, især i forbindelse med distribuerede domæneanvendelser (generel parallellisme), men også i forbindelse med såkaldt massivt parallelle datamater (speciel parallellisme).

  Der henvises iøvrigt til afsnit 3.1.2.

  Generel parallelisme dyrkes intensivt ved IT/DTU: undervisnings– såvelsom forsknings–mæssigt (ved lektor Hans Henrik Løvengreen).

  Generelt kan vi sige dette: ved konstruktion af parallelle programmelsystemer har det vist sig at man opnår de sikreste resultater ved at se bort fra hver enkel process' udførelseshastighed, herunder også hastighed af tilknyttet maskinel. At bero, i sin konstruktion, på absolutte tidsantagelser har vist sig at være en sikker vej til katastrofe: i projekter såvelsom i produkter.

- Sandtidshensyn:

  Det forrige caveat bærer over også til nærværende deldisciplin. I indlejrede systemer, f.eks. i mikroinstrumenter, i medico–teknik, våbensystemer, process styring, mmm., spiller koordinering mellem maskinelle og programmelle — de "hårde" og de "bløde" — processer en altafgørende rolle for systemernes ydeevne.

  Man skal ikke overse at et system er sandtidskritisk uanset om der er tale om milisekunders eller ti års reaktionstid ('response time').

  Igen forskes og undervises der ihærdigt på dette område ved IT/DTU (ved bla. lektor Michael R. Hansen i tæt samarbejde med professor Zhou Chaochen, UNU/IIST).

- *&c.*

3.2.5 Programmeringsprog

Når alt det ovenstående er overstået: når domænet og kravene er afklarede, når arkitekturen og programmelstrukturen er fastlagt, ja da er det på tide at begynde den egentlige programmering.

Forskellige problemstillinger kræver forskellige værktøj og det bør derfor ikke undre os, at der er flere bud på disse værktøj. En kort gennemgang kan være belærende.

- Funktions–programmering:

  For de der er mere fortrolige med Fortran's subrutiner eller C++'s procedyrer kan vi vel nemmest forklare funktionsprogrammering ved dets fokusering på funtioner (f, g): deres definition ($f = \lambda x.\mathcal{E}$), sammensætning ($f \bullet g$), og brug ($f(a)$). Almindelige — f.eks. aritmetiske — udtryk opfattes som funktionsudtryk.

  Det første funktions–lignende sprog var LISP (ca. 1962). Fra omkring midten af 1970rne skete der en renæsance. Der er idag flere udmærkede — ja nærmest intellektuelt spændende, elegante og inspirererende — funktionsprogrammeringssprog: Standard ML, Russell, Haskell, Miranda, etc.

Funktionssprog er især velegnede til beskrivelse af beregninger over strukturerede data. Specifikations– og bevisførelsesværktøj kan typisk med fordel udvikles i et funktionssprog. Alle almindelige, simple beregninger ligeledes: sortering, søgning, etc.

Da funktionssprog knytter sig tæt til den rekursive funktionsteori under meta–matematikken, dvs. til de vigtige begreber som λ–kalkyle og fikspunktlæren, egner funktionssprog sig også som et generelt første programmeringssprog i undervisning af dataloger.

Lektor Jørgen Steensgaard Madsen er blandt de, der forsker i funktionsprogrammering ved IT/DTU.

- Logik–programmering:

  I Danmark har vi en endog meget sløj tradition indenfor matematik hvad emnet matematisk logik angår. Og blandt ledende skolekredse er det vel nærmest et fyord om man nævner logik som noget allerede børn bør blive fortrolige med. Dansk matematik er, som svensk, domineret af analysen. Selv diskret matematik, grafteori og kombinatorik, synes at være stedbørn — trods det at vi her på DTU har en af verdens så absolut førende graf– & kombinatorik–teoretikere.

  Hvorom alting er, så spiller logikprogrammering en alt vigtigere rolle i dagens datamatik. Det bør være med nogen stolthed at vi kan erkende at vi i PDC, Prolog Development Center, har et verdensfirma indenfor logikprogrammering — som udsprang fra ID/DTH. Historien herom er nævnt i afsnit 1 side 2.

  Det første logiksprog var Prolog. Senere er tilkommet såkaldte "constraint" (dvs. begrænsnings) –baserede logiksprog.

  I logikprogrammering beskriver vi løsning af et beregningsproblem primært ved de matematisk logiske egenskaber et svar bør have. En fortolker af logikprogrammer finder nu alle de løsninger som opfylder de specificerede egenskaber ved en form for matematisk bevis.

  Logikprogrammering er en ligeledes fascinerende teknik til formulering af programmer. I logikprogrammer kan man "vende beregningerne om": det man troede var inddata-parametre kan opfattes som uddata, og omvendt. Herved kan man finde alle de inddata som svarer til hvad man før troede var en løsning!

  Ved IT/DTU er der forsket meget omkring logikprogrammering — jævnfør professor Jørgen Fischer Nilssons og docent Tom Østerbys arbejde nævnt i afsnit 1 side 2.

- Imperativ programmering:

  Klassisk programmering: fra Fortran, over Cobol, Algol 60 og Pascal til vore dages C, C++ og Java er i det væsentlige imperativ programmering. Vi kalder det imperativ da programmer kan siges at foreskrive: gør hint, siden gør dette — imperativ bydemåde.

  Imperative programmeringssprog bygger på at datamatens lager tillader programmer at indeholde såkaldte variable, dvs. navngivne størrelser hvor navnet står for en fast lagercelle hvis indhold, ved gentagne tilskrivninger kan variere. Hermed tillades at også lagercellernes adresse kan bruges som værdi, dvs. at vi kan søge langs en kæde af peger-lænkede data, at vi kan ændre en f.eks. graf–lignende datastruktur: indsætte og fjerne "nye", hhv. "gamle" data, mv.

  Hvor funktionsprogrammeringens basale størrelser var funktioner, og hvor de i logikprogrammering var logiske udsagn og sandhedsværdier (herunder afbildninger fra identifikatorer til sådanne), ja der er pegere og lagertilstande (relationer mellem variabel navne, deres tyd og disses værdi), de basale størrelser.

Såfremt man, som forøvrigt i al programmering, omhyggeligt iagttager en række basale love, ja da vil imperative programmer ligeså elegant som f.eks. funktions– og logikprogrammering kunne beskrive mange beregninger.

Studiet af bevislogikker for imperative programmer begyndte med C.A.R. Hoare's banebrydende arbejde i 1969. Hoare's arbejde byggede bl.a. på danskeren Peter Naur's (og amerikaneren Robert Floyd's) idéer.

- Parallel programmering:

  Der henvises, nok engang, til afsnit 3.1.2 side 4.

  I parallelprogrammering er de centrale størrelser: processer, deres sammensætning, på forskellig vis, og deres indbyrdes synkronisering og udveksling af data (kommunikation).

  Ved IT/DTU varetager lektor Hans Henrik Løvengreen et centralt kursus i parallelprogrammering ligesom han udforsker hvorledes man praktisk og teoretisk udvikler parallelle programmer.

  Store mammut–sprog som CCITT's CHILL og US DoD's Ada, se afsnit 1 side 2, indeholder faciliteter for parallelprogrammering. Vigtige teori–dannelser omkring disse bygger på de mere "teoretiske" sprog: CSP (C.A.R. Hoare) og CCS (A.J.R. Milner). CSP udgør basis for det praktiske, "lille" og elegante sprog 'occam' — som også er det sprog der bruges ved IT/DTU i forbindelse med parallelprogrammering.

- *&c.*

3.2.6 Problem–rammer

Bestemte læsere måtte, indtil dette sted, have bemærket at artiklen — udover flere referenser til oversættere og fortolkere (blandt datalogiens ypperste teknologiske frembringelser) — endnu ikke nævneværdigt har behandlet problemet: databasestyresystemer, datakommunikations– og netværksstyresystemer, endsige operativsystemer. Også disse er jo vigtige elementer i et brugbart datasystem. Hvad skyldes det at jeg ligesom har "fortrængt", eller i det mindste udsat, til et så sent 'tidspunkt', i det mindste en systematisk indpasning af denne — som tidligere implicit nævnt — datalogisk navlebeskuende datamatik? Svaret er ganske enkelt, at disse former for systemer "blot" er eksempler fra hver sin klasse af datamatisk teknologi. Jeg vil nu prøve en systematisk indpasning. Dvs. en form for "opremsning" som tilføjer ikke-trivial viden.

Generelt er det så, at der ikke findes een metode, der sikkert bringer datalogen frem til ethvert af de mange datamatiske systemer denne artikel omtaler. Man kan opdele den datamatiske verden i flere klasser systemer, hver klasse bestemt ved sin såkaldte 'problem–ramme'. Klassedelingen er bestemt ved at alle eksempler på instantieringer af klassens elementer (som hver for sig udgør et større stykke programmel) stort set udvikles vha. en nærmere bestemt metode; samt ved at to forskellige klasser repræsenterer væsensforskellige metoder. Eksempler er:

- Oversættelse:

  Dette er det klassiske datalogiske område. Dels findes der siden begyndelsen af 1970rne en rimelig sikker teori for hvorledes man ingeniørmæssigt udvikler en oversætter. Dels opbygges oversættere af stort set de samme delfunktioner: En lexikalsk passage indlæser programteksten tegn for tegn og genskaber fra disse tegn programmernes atomiske bestanddele. En syntaksanalysator konstruerer (ved såkaldt syntaksanalyse) fra de atomiske dele programmets syntaktiske struktur. En syntaks–fejl passage forsøger, udfra en nu omfattende forståelse for programmets struktur, at rette trivielle og, i tilfælde, mindre trivielle, som oftest omgivelsesbetingede fejl. Efterfølgende oversætterpassager kan omstrukturere programtekst–syntakstræet med henblik på en eventuel optimering (dvs. minimering af den eksekveringstid som den sidenhen genererede kode afstedkommer). Og endelig afsluttes oversætteren logisk set af kodegenereringspassagen.

Danmark har siden 1960 hørt til blandt de mere successrige lande hvad angår udvikling af traditionelle og af avancered oversættere. Fra Regnecentralens Gier Algol oversætter fra omkring 1961, via Turbo–Pascal, til Dansk Datamatik Centers avancerede oversættere for CHILL og Ada og PDCs Turbo–Prolog! Sidstnævnte tre systemer nyder udbredelse på verdensplan, og er alle udviklet under brug af rigorøse, formelt baserede metoder. Dette har medført en betydelig fejlfrihed.

Problemrammen ‘oversættere’ omfatter også fortolkere, JSP (Jackson Structured Programming), såkaldte gen– eller omskrivningssystemer, oa.

IT/DTU står bag disse senere systemer og har leveret afgørende forskningsbidrag til deres sikre udvikling. Uden ID/DTH’s forskning indenfor formel udvikling af oversættere ville vi idag ikke have haft så succesrige firmaer som DDC Intl. og PDC. Tre danske institutioner: to firmaer (DDC Intl., PDC) og et, nu, af andre grunde, desværre nedlagt forskningscenter (tfl = TeleDanmark Forskning) har tilkæmpet sig internationale positioner og kvalitetsry for deres arbejde med Ada, Prolog og CHILL (respektive).

- Styreteknik:

  Programmel til overvågning og styring af processer udgør en anden klasse. Fra de datamatiske bestemte operativ– til de fremstillingsindustri bestemte processtyringssystemer går der en lige linje. Mange større elektroniske instrument eller elektromekaniske fremstillingssystemer indeholder i deres kerne et styresystem. Brüel & Kjær forsøgte sig i begyndelsen af 1980rne på et større instrumentsystem. Man forstod ikke at problemet var et rimeligt klassisk datalogisk problem. Man ville ikke lytte. Og resultatet blev en af de meget store pinde til B&K’s ligkiste.

  IT/DTUs forskning indenfor troværdige og sandtids-systemer er en vigtig forudsætning for etablering af sikre styretekniske systemer (A.P. Ravn, Hans Rischel m.fl.).

- Information:

  Vel den klasse systemer der bliver brugt mest idag, i administration, erhverv og industri, er de der baserer deres afvikling på underliggende databasestyresystemer. Produkter som IBMs IMS, dBase I–II–II, osv. kendetegner denne klasse.

  IT/DTU har ingen særlig forskningsaktivitet indenfor dette vigtige område. Ved den franske stats’ nationale forskningscenter, INRIA, for informatik og automation, udgør database– og informationssystem–forskningen en betragtelig del. Måske er der her noget vi forsømmer generelt i Danmark?

- Arbejdsark:

  Systemer som CAE, CAD, CAM, almindelige regneark, bogholderi & finanssystemer, er typiske eksempler på “arbejdsark” — een til lejligheden sammenstrikket betegnelse (engelsk: workpiece).

  Arbejdsarksystemer er kendetegnet ved at hvad man før udførte stort set alene vha. papir og blyant men muligvis understøttet at datamatiske mellemberegninger, ja det bliver nu altsammen indkorporeret i arbejdsarksystemet.

Ved IT/DTU er man begyndt en udforskning af dette område. En kombination af styreteknisk og arbejdsark problemramme findes i såkaldte 'enterprise integration', dvs. i modeller af større, typisk fremstillingsvirksomheder. Der henvises til punkt 1 side 19.

- Transaktion:

  En bank foretager dagligt millioner af enkeltstående transaktioner, og udveksling af personlige checks mellem banker, det være sig nationalt eller internationalt, kalder idag på nogle af de allerstørste datamatiksystemer.

  Alene Master Card's center i South Dakota, der varetager formidling af kreditkortstransaktioner mellem modtagende og udstedende banker, behandler per time millioner af kreditkortstransaktioner som dækker "The Americas"; fra ildlandet i syd to Alaska i nord!

  Ved IT/DTU er man begyndt en udforskning af dette område som et led i forståelsen af "Hvad er en Bank?".

- Beslutnings–støtte–systemer:

  Mange ønskede beregninger kan ikke effektueres idet man ikke er istand til præcist at angive nødvendige relationer mellem tilgængelig inddata og ønskede svar — ofte fordi man ikke tilfulde, algoritmisk eller logisk, forstår problemet. Men man forstår så meget, at det kan "betale" sig at udvikle et system til understøttelse af de beslutningsprocesser, som beregningerne ellers ville indgå i. Ved hjælp af såkaldt videnbaserede delsystemer kan man komme en lang vej imod automatisering.

  Ved IT/DTU er man begyndt en udforskning af dette område som et led i forståelsen af de strategiske og taktiske beslutningsprocesser hvad angår resoursestyringsproblematikken i f.eks. fremstillings– og transporterhvervene.

  Afsnit 6 nævner yderligere to (punkterne 3 og 4 side 20).

- *&c.*

Jeg vil iøvrigt overlade det til min gode kollega professor Jørgen Fischer Nilsson nærmere at beskrive hele det store, meget spændende og komplementære område: videnbaserede systemer. Vi har ved een lejlighed, umiddelbart før jeg rejste på min 5 års orlov til Macau, arbejdet sammen på en dengang internationalt fremlagt artikel om en mulig forståelse af overgange mellem algoritmiske og videnbaserede systemer.[3]

3.3 Abstraktion og Sprog

Fra domæne–, via krav– til programmelbeskrivelser, og indenfor hver af disse, f.eks. fra abstrakte, programmelarkitektur–, og via mindre abstrakte programmelstruktur–. og til stedse mere konkrete programmeldesign–beskrivelser, er der i hele datalogens virke primært tale om sproglige beskrivelser. Specifikations–, definitions–, design– og programmerings–sprog spiller således en altafgørende rolle: som genstand for forskning og som værktøj i ingeniørkunsten. Det er ikke kun de formaliserede sprog som datalogen skal beherske, men også, og især, det uformelle, naturlige sprog. De forrige sprog for at sikre korrekt udvikling. Det nationale sprog for præcist at kunne kommunikere.

Hvor vi "gamle" på forskellig vis brugte to dage om ugen på om eftermiddagen, gennem to førstedelsår på DTH i 1950rne, til at lære at tegne efter Helge Nielsens noget særegne forskrifter — og hvor datamatiserede tegneprogrammer idag fuldt har overtaget denne

[3] Algorithmic vs. Knowledge–based Systems — Do they Unify? Inviteret foredrag ved 10 års konferencen for det Japanske Femte Generations Datamat Projekt, Tokyo, Oktober 1991.

funktion, ja der kan man vanskeligt forestille sig nogen automatisering af det menneskelige sprog. Evnen til at fokusere på det essentielle, til pædagogisk forståeligt og didaktisk afklaret at fremstille en problemstilling og dens løsning, ned gennem ofte 15–20 niveauer af abstraktion, en sådan evne er altafgørende for at en datalog skal kunne blive succesrig. Når man så dertil lægger hvad vedkommende ingeniør skal præstere som informatiker, hvor det ofte kommer på tale at kombinere det datalogiske, ikke–naturvidenskabelige, med et eller flere af de af naturvidenskaberne bestemte ingeniørfag, ja så skal der ikke megen fantasi til at forestille sig informatikerens kommunikationsproblematik.

4 Informatik

Vi formulerer informatik, lidt slagordsagtigt, som:

- Informatik =
  Matematik ⊕ Datalogi ⊕ Anvendelse

⊕ operatoren er ikke kommutativ — og kan ikke lede til at man flytter om på leddende, og da slet ikke tværs over ligningssymbolet (=)!

Vi har set på datalogien, og gennem det har vi nævnt dets matematiske ståsted og en lang række anvendelser.

Tilbage står at afgrænse en række roller: Hvad vil forskning i informatik sige, hvem forsker i informatik, hvad er en informatiker — i kontrast til en datalog? Osv.

4.1 Datalog Først

Dataloger — som forskere — brugte de første 30 år af fagets eksistens med at "bringe orden i eget hus": dvs. "navlebeskuende" at sikre en både dyb og bred forståelse af de mange forhold jeg har nævnt ovenfor. At sikre fundamentet, beskrivelsesværktøjernes syntaks og semantik, metodelæren i almindelig, mv. Dette nødvendige arbejde er vel ikke rigtigt blevet påskønnet på et sted som DTU — men ID/DTH's arbejde blev dog kraftigt påskønnet i vigtige, internationalt orienterede dele af den danske programmelindustri.

Idag står datalogen, som forsker og underviser, godt udrustet til at tackle anvendelserne — sådan som det også fremgår ovenfor (afsnit 3.2.2).

4.2 Domænespecialist Først

Hvad er så en informatiker? Det kan være en datalog, der specialiserer sig i et specielt anvendelsesdomæne, eller det kan være en domænespecialist, f.eks. en maskiningeniør. Fælles for begge er at de — gennem kurser eller selvstudie — behersker både væsentlige dele af datalogisk metodelære som væsentlige dele af f.eks. de maskinelle fremstillingsprocesser. Derved kan de specialisere sig som informatikere indenfor fremstillingsindustri, eller transportvæsenet, eller elektroniske mikroinstrumenter — som det nu sker ved IT/DTUs ingeniørvidenskabelige Mikroinstrument Center.

Gode datalogiske løsninger har det med at abstrahere. Først, til eksempel, lavede man oversættere, siden lavede man værktøj til at lave oversættere, og nu laver man værktøj til at lave disse værktøj. Maconomy, f.eks., har en sådan familie af værktøj hvormed det hurtigt, sikkert og billigere end konkurrenterne, kan skræddersy løsninger til administrative sturesystemer. Tilsvarende indenfor snart sagt hver branche.

De fleste datamatiske og informatiske ingeniøropgaver er derfor af en sådan beskaffenhed, og vil for stedse vedblive at være det, at det kræves af datalogen og af informatikeren at de kan forske i anvendelserne: at de kan sikre nye, matematiske notationer — sådanne som bedst egner sig til løsning af det forhåndenværende problem, oma.

4.3 "Teamwork"

Men det er sandsynligvis ikke nok. Det sikreste for enhver større virksomhed er nok omhyggeligt at bestykke sin ingeniørstab med dataloger, informatikere og f.eks. maskin–, eller elektronik–, og bygnings– ingeniører osv.

5 Generelle Betragtninger

5.1 Historisk Udvikling

Jeg har ovenfor struktureret min fremstilling af datalogiens væsen udfra et logisk synspunkt: dvs. udfra hvad vi idag ser som dens "naturlige" bestanddele. Men historisk set har det udviklet sig stort set omvendt af min fremstillingssekvens. Først interesserede man sig for de programmeringstekniske forhold der så at sige lå nærmest datamaten. Sidenhen har man kastet sit forsknings– såvelsom ingeniørmæssige net i større og større cirkler — for nu i de sidste år at udforske domæne–teorier.

5.2 Datalogiens Univers

Sådan skåret lidt skarpt ud: hvor de af naturvidenskaberne bestemte ingeniørfag bestemmer universer af material kvantitet bestemmer datalogien et intellektuelt univers af kvalitet. Hvor man, til eksempel, i den almindelige — den af elektronik mv. bestemte — informationsteknologi, taler om flere transistorer per arealenhed, mindre energiforbrug per funktionsenhed, lavere omkostningsniveau, hurtigere reaktionstid, større hukommelsesenheder, etc., taler man i datalogien om logiske egenskaber: korrekthed af et stykke programmel mht, dets kravspecifikationer, affinitet til anvendelsesdomænet, "tilpasningsevne" overfor den menneskelige bruger, osv.

Samtidig synes datalogien at rejse fundamentale spørgsmål af næsten erkendelsesteoretisk natur. Dataloger har god nytte af indsigt i videnskabsteori og — filosofi — fra Marcuse, via Kuhn til Popper!

Disse postulerede forskelle gør det ofte vanskeligt at finde fælles fodslag med kolleger på DTU, ja selv indenfor den politisk bestemte konstruktion IT! Stakkels institutleder: hvorledes skal vedkommende dog kunne forlige så ulige sjæle?

6 DTUs Informatikprofil

DTU har strategisk lagt grunden til en spændende fagprofil indenfor informatik. Informatik, som defineret og nærmere indholdsbestemt i denne artikel, passer som fod i hose, til denne profil.

Nu gælder det så om ikke at skuffe de studerende. Det kan nemt ske. Især hvis vi ikke sætter os ned sammen for nærmere at definere en række fælles forskningsopgaver, der kan danne grundlag for en meningsfuld undervisning. Det kan også ske, hvis de indgående partnere ikke har gensidig respekt for hverandres fagområde og dets dybde. Tværvidenskabelig forskning og undervisning bliver meget "tynd" hvis ikke den går i dybden indenfor enhver af de deltagende discipliner.

I dene artikel har jeg forsøgt skitsere en lang række mulige projekter: indenfor transport, fremstilling, m.fl. Nogle passer fint på DTUs arbejdsområder.

1. Virksomhedsmodellering:

   IT/DTU har nogen erfaring i opstille modeller til besvarelse af spørgsmålene: *Hvad er en fremstillingsindustri, Hvad er dets marked?* I det første skabes der en overbygning på begrebsdannelsen CIM: computer integrated manufacturing, i det andet modellers 'suppliers & consumers, producers & traders'.

   I dette, som i de næste eksempler på mulige samarbejdsprojekter, tilfører datalogien ret så nye måder at tackle problemstillingerne på. I en lille årrække endnu vil det være frugtbart for både forskning og undervisning at samarbejde omkring disse emner indtil institutter udenfor IT selv har den fornødne "datalogiske pli" (om man må være så ubeskeden at sige dette?).

2. Transport:

   Som antyder i afsnit 3.2.1 har IT/DTU også en del erfaring med modellering af transportdomæner (til eksempel: jernbaner, hovedstadstrafik, lufttrafik). ID/DTU's interesse i disse domænemodelleringsprojekter, såvelsom alle øvrige sådanne, er ikke at opstille den form for operationsanalytiske modeller som IMM er afgjort bedst placerede til, men snarere logiske og model–teoretiske modeller, der siden kan lede til programmelsystemer der aktuelt styrer virksomheder, transport, mv. Men IMM's og IT's modeller bør samstemmes. Og så er der jo også andre institutter ved DTU som engagerer sig i transport-studier (IFP?).

3. Beslutningsstøttesystemer for Bærbar Udvikling.

   Typiske beslutningsstøttesystemer er sådanne for 'Bærbar Udvikling'. Her interesserer man sig for flg. områder, der siden knyttes sammen: Først: 'Hvad forstår man ved "bærbar udvikling"?'. Herunder: 'Hvilken information skal indsamles?' 'Indenfor hvilke grænser bør informationernes værdier ligge?' 'Hvilke er mao. indikatorerne?' Dernæst skal denne information repræsenteres som data. Typisk i såkaldt fødererede geo– og demografiske informationssystemer (GIS + DIS). Endelig skal der tilvejebringes "standard", operationsanalytiske modeller for undersøgelse og opstilling af problem–specifikke udviklingsforhold.

   IT/DTU har arbejdet noget også med denne problemstilling.

   Dette er typisk noget man måske kunne arbejde med Institut for Miljøteknologi og IMM omkring?

4. "IT Policy Studies":

   Nok en kategori af beslutnings-støtte-systemer kan eksemplificeres: "IT Policy Studies". Det er meget populært for de nyligt industrialiserede lande (Malaysia, Indonesien, Sydafrika, m.m.fl.) at ønske deres land's fremtidige industrialisering baseret væsentligt på brug og udvikling af informationsteknologi. Og det er ligeledes populært at tro at alt sligt kan nøje studeres, overvejes, besluttes, sættes igang og siden successrigt afstedkommet. Hertil kræver man, for det ligger jo i hele denne planlægnings ånds' væsen, at man først foretager et såkaldt "IT Policy Study". Hvordan gøre man u dette: dvs. hvori består studiet, hvorledes valideres data, hvilke data, hvem skal spærges, hvilke socio–økonomiske hensyn skal modelleres, etc.?

   Måske IT/DTU kunne arbejde sammen med Institut for Teknologi og Samfundsfag (ITS) og Institut for Matematisk Modellering (IMM) på dette. IT/DTU står parat — og har nogen erfaring hermed.

7 "Katastrofer" & "Hypes"

Datalogens hverdag foregår, foruden i den ovenfor beskrevne videnskabelighed, i en atmosfære af ingeniørmæssig faglighed og professionalisme: i et stadigt virke, der har til hensigt at sikre at etablerede metoder også vinder indpas i hverdagen. Det kan ofte være svært. Datalogi er stadig et så ungt virke at de fleste af de kandidater vi idag genererer fra IT/DTU — og som ankommer i erhvervslivet, fuldt opladede — ikke bliver fagligt og eller administrativt ledet af professionelle dataloger, data-ingeniører eller informatik-ingeniører. Det betyder blandt andet at den helt reelle viden, der idag findes mht. udvikling af f.eks. meget stort programmel endnu ikke er trængt ud i erhvervet — idet den fortrænges af den eksisterende masse af "gamle" programmører. Kun få danske virksomheder evner det (herunder DDC Intl.).

Vi bevidner derfor jævnligt at udvikling af programmel kræver langt flere resoursser (tid, penge) end lovet, at programmel ikke holder, hvad der blev lovet, og at det er fuld af fejl! Ja man skønner i industrien at omkring halvdelen af alle større programmel-projekter afbrydes sent i udviklingsfasen.

Samtidig bombarderer fagpressen os med at “hypes” som objekt-orienterede “metoder”, Windows, Internet, Java, oma., er svaret på industriens trængsler.

Dette virker som en afledning fra det reelle problem: At erhvervet endnu ikke forstår at udvikle og levere så fejlfrit programmel, som det dog lader sig gøre professionelt at udvikle. At erhvervet ikke altid opfører sig tilstrækkeligt professionelt. Og når så de er i knibe, ja så uddanner vi, på DTU, ikke nok og relevante nok ingeniører. Man læser herom i dagspressen fra direktører — tit op til bestyrelsesmøder. Ved nærmere efterprøvning viser det sig så at de ikke ansætter vores fagfolk, og da slet ikke vores PhD'er!

Industriens misére opfattes sikkert, af fagkolleger på DTU, som vores, IT/DTU's, fejl.

8 Afslutning

Der er fremstillet et fagområde: Datalogi, og dets nære kobling til andre discipliner er kort beskrevet. Der fremstår billedet af en rig, multifacetteret disciplin, nært beslægtet med matematik, og kun gennem informatik–koblingerne knyttet til naturvidenskaberne.

En stor del af de ingeniører der tager eksamensprojekt ved IT/DTU (tidl. ID/DTH) har søgt beskæftigelse udenfor DTUs klassiske aftagerkreds: fremstillingsindustrien og rådgivningserhvervet. Istedet finder de velønnet beskæftigelse i servicefagene: Banker, forsikringsselskaber, m.fl. En ny industri har været på vej siden 1960rne — men man kan ikke just sige at DTU har tænkt særligt på disse servicefag og denne industri. Der refereres til programmelindustrien. ID/DTH kunne med stor ret siges at stå bag flere af de der er tilkommet siden 1980: CRI A/S, DDC Intl., Maconomy, PDC og The Vision Factory.