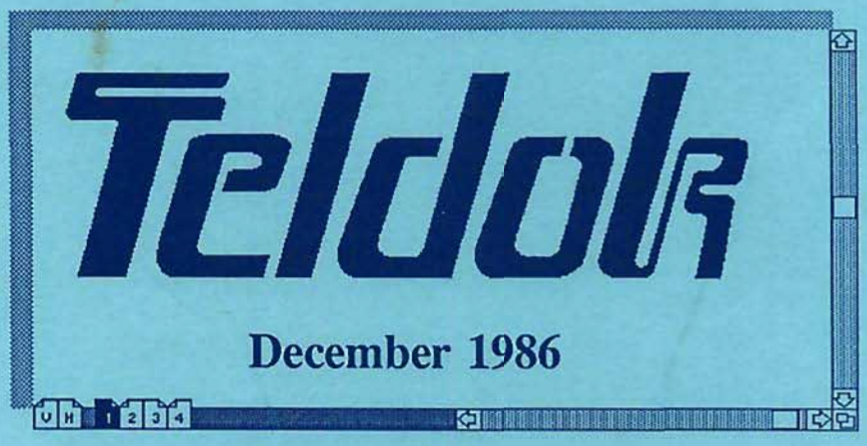


Rapport
26

Datorstödda kunskapssystem i framtidens kontor

Sture Hägglund

Datorstödda kunskapssystem i framtidens kontor

Sture Hägglund

Universitetet och tekniska högskolan i Linköping

Sammanfattning: Denna rapport sammanfattar utvecklingen inom området kunskapsbaserade expertsystem, dvs programvara som självständigt eller i samverkan med en användare löser uppgifter som normalt anses kräva mänsklig expertis. Speciellt betonas synen på sådan programvara som ett kommunikationssystem för förmedling av kunskap från experter till slutanvändare av det datorbaserade systemet. I rapporten redovisas relationen till traditionell informationsteknologi, teknik och metodik för utveckling av kunskapsbaserade system, samt ges exempel på en rad tillämpningar i kontorsmiljö. Speciellt diskuteras hur framtidens informationsteknologiska system kan komma att påverka förutsättningarna för framtidens kontorsarbete.

Rapporten har tillkommit våren 1986 med stöd från Televerket genom TELDOK's redaktionkommitté, under författarens tjänstledighet från institutionen för datavetenskap (IDA) vid Universitetet och tekniska högskolan i Linköping. För värdefulla bidrag och synpunkter tackar författaren Staffan Löf, Pär Emanuelson och Björn Möller, Epitec AB, samt kollegor vid IDA, vilkas samlade erfarenheter i hög grad ligger till grund för denna rapport.

Författarens adress:

Sture Hägglund
IDA
Universitetet i Linköping
581 83 LINKÖPING

Tel 013 - 28 14 31
Computer mail: UUCP: ...enea!liuidalsth

ISSN 0281-8574

Publikationerna kan beställas
gratis, dygnet runt, från
TeleSvar.

Tryckeri: Nyströms Tryckeri AB, Bollnäs

INNEHÅLL

1.

Introduktion

Denna rapports syfte är att sammanställa information om den snabba utvecklingen inom området kunskapsbaserade expertsystem, eller kortare *kunskapssystem*, och att diskutera möjliga och troliga konsekvenser för vårt sätt att använda informationsteknologi i kontorsarbete. Rapporten ska förhoppningsvis kunna läsas med behållning av såväl personer med kunskaper om detta teknikområde, som av personer med en mer begränsad insikt inom området. I framställningen diskuteras framför allt principiella frågor, men försök görs att genomgående illustrera med konkreta exempel från tillämpningar, som finns i dag. Det bör dock påpekas att kommersiellt fungerande produkter har först under de allra senaste åren gjorts allmänt tillgängliga, varför många av de system man hittills har erfarenhet av utgörs av forsknings- eller prototypsystem.

När det gäller nyttiggörande av informationsteknologi knyts för närvarande stora förväntningar till tekniker från det forskningsområde inom datavetenskapen, som benämns *artificiell intelligens (AI)*, och till effektivare sätt att bygga system som understödjer insamling, hantering, utnyttjande och kommunikation av kunskap. Dessa förväntningar är i hög grad knutna till s.k. *kunskapsbaserade expertsystem*, dvs programvara som självständigt eller i samverkan med en användare löser uppgifter som normalt anses kräva mänsklig expertis. Inom alla grenar av informationsteknologin och dess tillämpningar, förväntas dessa system i en nära framtid medföra ett radikalt teknikskifte baserat på de nya möjligheterna att utföra s.k. symbolisk informationsbehandling, dvs att i datasystemen direkt lagra, bearbeta och använda verksamhetsnära kunskap för beslutsfattande och problemlösning.

En av de ledande amerikanska forskarna inom området, professor Edward A. Feigenbaum vid Stanford-universitetet, säger om den utveckling vi nu står inför:

> "*It's not just the second computer revolution . . .*
> *It's the IMPORTANT one!*".
>
> (IVA-symposium, december 1985)

Innebörden av detta yttrande är att hittillsvarande informationsteknologi baserats huvudsakligen på datorns förmåga till effektiva matematiska beräkningar och rutinmässig administration av stora datamängder, medan dess

potential som hjälpmedel för bearbetning och användning av symboliskt representerad kunskap nu öppnar nya perspektiv och möjligheter.

Möjligheten att reproducera intelligensfunktioner i icke-levande system har länge fascinerat och utmanat människan. Egentlig forskning om möjligheterna att använda datorer för att åstadkomma konstgjord intelligens initierades redan på femtiotalet och anmärkningsvärda resultat nåddes tidigt, exempelvis med program för problemlösning, bevis av matematiska satser och frågebesvarande system som arbetade med naturlig engelska. Emellertid visade sig dessa framgångar vara svårare att generalisera än man först trodde, och detta tillsammans med orealistiska förutsägelser om ett snabbt genombrott för praktiskt användbara AI-system, ledde till att AI-forskningen under 70-talet delvis kom att betraktas med en viss skepsis.

Under det tidiga 80-talet har bilden hastigt svängt. Den filosofiska frågan om en dator verkligen kan ges intelligens i en djupare mening kvarstår visserligen obesvarad, men inom en rad områden har man kunnat realisera program som hanterar problemställningar som tidigare låg utom räckhåll. Samtidigt har det snabba prisfallet på datakraften har gjort AI-forskningens verktyg ekonomiskt rimliga i en rad tillämpningar. De tekniker som AI-forskarna utnyttjat i laboratoriemiljö kan nu få en bred praktisk användning.

Kunskapsbaserade expertsystem, eller med den något vidare term som alltmer vinner insteg, *kunskapssystem*, innebär en möjlighet att introducera datorstöd även för uppgifter där svårigheter att exakt formulera lösningsmetoder gör att traditionell datateknik ej varit framgångsrik. Ett sådant datorprogram baseras på att kunskaper, fakta och erfarenheter kan utnyttjas på ett sätt som liknar mänskliga experters sätt att resonera. Typiska egenskaper är:

* *heuristiska lösningsmetoder.* För problem där en analytisk/algoritmisk lösning ej är känd eller är beräkningsmässigt ogenomförbar används någon intelligent strategi för att begränsa antalet alternativ, exempelvis med hjälp av "tumregler" och erfarenhetsmässiga kunskaper.

* *transparens.* Resultat och rekommendationer från systemet kan förklaras och motiveras. Kunskapsbasen, omfattande aktuella fakta och beslutsregler m.m., är hela tiden tillgänglig för inspektion.

* *resonemang under osäkerhet.* Hänsyn kan tas till att såväl kunskap om orsaksamband som aktuella data kan vara ofullständiga, otillförlitliga, eller eljest osäkra, varvid systemet beräknar graden av plausibilitet för en slutsats eller ett resultat.

* *flexibilitet.* Kraftfulla hjälpmedel som stöder stegvis formulering och modulärt underhåll av kunskapen i systemet ingår i den datorstödda utvecklingsmiljön.

Denna typ av programvara kan sägas komplettera traditionell datateknik för numeriska beräkningar och för rutinmässig massdatabehandling. Typer av uppgifter där denna teknik kan erbjuda fördelar är exempelvis:

- *Människa-dator interaktion*, där problemet är att utforma dialogen med användaren så att dennes kunskaper och förmåga utnyttjas på bästa sätt med ett minimum av krav på teknisk "fingerfärdighet, samtidigt som systemet så långt möjligt erbjuder effektiv hjälp och förklaringar när så behövs.
- *utbildning och träning*, dvs att till en användare förmedla kunskaper som lagrats in i ett system och att öva dennes förmåga att tillämpa dessa kunskaper.
- *övervakning*, t.ex. av en process, ett system, eller ett ärende, med uppgift att upptäcka när ett i någon mening onormalt tillstånd uppstår.
- *tolkning*, dvs att värdera uppgifter och data, speciellt med hänsyn till att dessa kan vara oprecisa, ofullständiga eller felaktiga.
- *diagnos, (felsökning)* dvs att utifrån givna symtom dra slutsatser om vilken orsak som ligger bakom.
- *konstruktion*, dvs att utforma en produkt eller process av något slag som uppfyller givna krav och restriktioner.
- *planering*, dvs att välja åtgärder med syfte att nå ett givet mål och att utforma en sekvens av handlingar som leder till målet.
- *styrning*, dvs att reglera en process eller en verksamhet enligt givna kriterier.

Ambitionen är att nå upp till prestandanivån hos de mest framstående mänskliga experterna inom ett givet område. Systemen kan vara *autonoma*, dvs självständigt problemlösande, eller *konsultativa*, varvid man som användare vanligen tänker sig en person som är kunnig, men inte expert på den aktuella uppgiften, och som ges möjlighet höja sin kompetens genom att ta assistans från systemet.

Sammanfattningsvis kan noteras att vi dels kan definiera begreppet kunskapssystem/expertsystem i första hand utifrån deras *prestationer*, dvs att lösa i någon mening svåra (expertiskrävande) problem, dels utifrån *lösningsmetoden*, dvs användning av vissa språk, programorganisationstekniker, utvecklingsmetodik eller problemlösningsmetod. En kombination av dessa synsätt kommer att tillämpas i denna framställning, dvs vi ser kombinationen av lösningsmetod och prestationsförmåga som betydelsefull. Termerna expertsystem respektive kunskapssystem kommer delvis att användas synonymt i det följande. Dock med en betoning på problemlösningsförmågan när det gäller expertsystem medan kunskapssystem föredras när vi ser till det det vidare begreppet.

Som torde ha framgått av det föregående, gör vi en åtskillnad mellan forskningsområdet AI och den tillämpade programvaruteknik som resulterat i möjligheten att bygga praktiska kunskasphanterande syustem för beslutsstöd och problemlösning. Det är i dag knappast meningsfullt att diskutera huruvida expertsystemen är "intelligenta" i någon djupare mening eller inte. Mer intressant är att värdera dessa systems egenskaper i sig, att bedöma vilka

praktiskt fungerande metoder och tekniker som AI-forskningen redan resulterat i och vilka principiella problem som återstår att lösa för denna forskning.

Waterman [WAT86] gör följande illustrativa jämförelse mellan mänsklig och artificiell expertis:

Mänsklig expertis	*Artificiell expertis*
förgänglig	permanent
svår att förmedla	lätt att förmedla
svår att dokumentera	lätt att dokumentera
opålitlig	förutsägbar
hög kostnad	rimlig kostnad

Figur 1.1 *Motiv för datoranvändning.*

Mänsklig expertis	*Artificiell expertis*
kreativ	oinspirerad
anpassningsbar	kräver instruktioner
kombinerar sinnesintryck	symbolisk information
brett tillämpbar	högt specialiserad
sunt förnuft	teknisk kunskap

Figur 1.2. *Människans fördelar.*

Denna översikt visar på de begränsningar som vi i dag har att acceptera när det gäller datorbaserade kunskapssystem. Även med dessa begränsningar finns det en mycket betydande potential för denna teknik, vilket de stora satsningar som görs på många håll ger uttryck för. Följande är ett är försök att sammanfatta dagens förväntningar på denna nya teknik:

- Inom vissa på lämpligt sätt avgränsade områden kan vi hantera komplexitet i beslutsfattande och problemlösning på ett sådant sätt att vi kan reproducera prestationsförmågan hos de bästa mänskliga experterna, t.ex. för finansiellt och ekonomiskt beslutsfattande, felsökning, konstruktions- och produktsammansättningsarbete, säljstöd, m.m.
- Inom allt fler industrigrenar blir förmågan att skräddarsy produkter och lösningar för kundens behov av allt större betydelse. Genom att på olika sätt kunna "paketera" kunskap och ta betalt för den som produkt eller komponent i produkter, hoppas man kunna skapa nya marknader eller förädla allmänt tillgänglig teknik av traditionellt slag.

- Redan i dag har kvaliteten hos gränssnittet människa-dator i interaktiva system som styrs genom en dialog med användaren i många fall en avgörande betydelse för systemets framgång. Utformning av dialoggränssnitt, som på ett optimalt sätt tillvaratar människans unika egenskaper, förutsätter i praktiken att omfattande kunskap om tillämpningen, brukssituationen och användaren byggs in i systemet.
- Ett allmer centralt problem är utformning av metodik och stödjande programvara för formulering och fortlöpande underhåll av den kunskap som ligger till grund för ett systems problemlösande förmåga. Effektiva hjälpmedel inom detta område, med speciell tonvikt på att involvera områdesexperter direkt i systemutformningen, utvecklas underhand.
- Vi ser också nu framför oss informationsbehandlingssystem med förmåga att producera förklaringar och motiveringar till bearbetningar och resultat. Speciellt finns vissa möjligheter att realisera undervisande och utlärande funktioner som ett stöd för slutanvändaren att tillägna sig de kunskaper och den problemlösningsförmåga som ligger till grund för programsystemet.
- Vi kan också uppfatta tekniken med kunskapssystem som en rationellare teknik för utveckling av vissa klasser av (interaktiva) informationssystem i traditionen av den s.k. fjärde generationens språk. Tekniken ger fördelar när det gäller exempelvis att precisera användarkrav med hjälp av prototyper, för att underlätta kundanpassning i levererade system, för att förenkla underhåll av programfunktioner när krav eller förutsättningar förändras, för att strukturera tillämpningar med en uppdelning i välformade normalfall respektive olika typer av undantagssituationer, osv.

Det är vår förhoppning att de följande kapitlen ska ge en djupare förståelse för dessa möjligheter och en illustration av vad som kan åstadkommas i dag eller inom den närmaste framtiden. Det torde också vara oundvikligt att den utveckling som beskrivs i denna rapport, på ett eller annat sätt kommer att genomgripande påverka arbetsuppgifter, arbetsutformning och framtidens kontorsarbete över huvud taget.

2.

Datorstödda informationssystem

Man förknippar vanligen datateknik med införandet av centraliserade, formaliserade system, som medför en rationalisering genom att olika informationsbehandlingsprocesser kan automatiseras. Datateknikens administrativa användning skulle därmed leda till uppbyggande av centrala systemavdelningar med en högt specialiserad kompetens, samtidigt som behovet av lokal arbetskraft för rutinuppgifter reduceras radikalt. En yttring av denna utveckling kan man se i den livaktiga diskussionen runt försäkringskassornas datasystem, där man mötte en stark opposition mot centraliserade systemlösningar. Det är uppenbart att denna utveckling kan komma att beröra stora delar av tjänstesektorn, t.ex. bankvärlden där behovet av decentrala kontor för kontanttransaktioner minskar.

Emellertid representerar denna typ av ADB-system, bara en av flera möjliga användningar av datateknik. För närvarande ser vi en explosionsartat snabb utveckling av distribuerad databehandling baserad på lokala smådatorer och även införandet av kontorsinformationssystem med stark tonvikt på hjälpmedel för den enskilde tjänstemannens arbete. Denna typ av system innebär ett understöd för ärendehantering och konstruktionsarbete snarare än en automatisering av arbetsuppgifter. Stödet är dock huvudsakligen passivt i den meningen att det baseras på sökning, sammanställning, presentation och kommunikation av information. I den följande framställningen ska vi föra diskussionen vidare till den typ av datorbaserade hjälpmedel som på ett aktivt sätt utnyttjar en kunskapsbas för att stödja problemlösning och beslutsfattande inom olika områden. Sådana system har realiserats inom tillämpade grenar av forskningen inom det område som kallas artificiell intelligens och tekniken är för närvarande snabbt på väg ut i praktisk användning.

När datatekniken utvecklades för c:a trettio år sedan, uppfattades datorn som i första hand ett *redskap för att utföra beräkningar*. Snart insåg man dock att även andra slag av informationsbehandlande processer kunde utföras med detta nya hjälpmedel. Administrativa system för löneberäkningar m.m. utvecklades parallellt med tekniska tillämpningar. Gemensamt för synen på datorn var emellertid ett algoritmiskt perspektiv, d.v.s. ett program uppfattas som en beskrivning av hur en given mängd indata ska transformeras till utdata, resultatet av bearbetningsprocessen.

Uppgiften att utforma system för informationsbehandling blir med detta synsätt att definiera reglerna för olika bearbetningar, att beskriva formen på önskade utdata och att sedan insamla och preparera behövliga indata. Systemarbetet koncentreras på problemet att formalisera behandlingsreglerna och att utforma algoritmer för hantering av olika databearbetningar.

Så småningom visade det sig i allt flera sammanhang att det egentliga problemet oftast inte låg i att definiera bearbetningen av data, utan snarare fanns att söka i uppgiften att beskriva och hantera den information som läggs till grund för olika bearbetningar. Ur teknisk synpunkt betydde detta att man inom en verksamhet snarare sökte organisera sin informationsbehandling som en databas av gemensam information, än som en mängd bearbetningsprogram med separat definierade behov av indata. Synen på datorn som ett *redskap för att organisera och hantera verksamhetens information*, kom att allt mer ta över jämfört med synen på dator som ett instrument för bearbetning av data.

De allra senaste årens utveckling har emellertid allt mer dominerats av ett tredje synsätt, nämligen datorn som ett *kommunikationsredskap*. I detta fall ser man datoranvändningen i ett dynamiskt perspektiv. Den information som hanteras i datasystemet får med denna utgångspunkt en mer dynamisk karaktär. De tekniska problemställningarna i samband med systemkonstruktion koncentreras på utformning av kommunikation mellan olika delsystem i en distribuerad arkitektur, på utformning av kommunikation mellan användare och program, samt på informationssystemet som ett stöd för kommunikation mellan dem som tillför information och dem som extraherar information ur systemet.

Den successiva utvecklingen av synen på datorn som ett redskap för *databearbetning, informationslagring* respektive *kommunikation* har naturligtvis starkt påverkats av den förskjutning av kostnadsförhållandet mellan s.k. maskinvara i form av datorer och terminaler å den ena sidan och arbetskraftskostnader å den andra. I tidiga tillämpningar koncentrerades datorutnyttjandet till uppgifter där den enorma beräkningskapaciteten kunde tillvaratas, och nödvändiga kringuppgifter fick utföras av mänsklig arbetskraft. I dagens läge kan datorstöd motiveras också för uppgifter, där även ett begränsat stöd för människans arbete ökar produktiviteten mätt på lämpligt sätt. Därmed ökar incitamentet att utveckla sådan teknik, inte minst på programvaruområdet, som på ett flexibelt sätt kan understödja människans naturliga sätt att arbeta. Detta står i en kontrast till tidigare erfarenheter, där människan ofta tvingats anpassa sig till datateknikens förutsättningar och begränsningar.

2.1 Konventionell ADB-teknik.

ADB-teknik, såsom den typiskt används idag, förknippas vanligen med rutinisering av arbetsuppgifter, formalisering av den information som hanteras och likriktning av handläggningsrutiner. Detta kan också uppfattas som en svårighet, när det gäller att utnyttja datorstöd i ett informationssystem, eftersom det ofta leder till onaturliga förenklingar i de modeller som läggs till

grund för systemet. Relevant men svårformaliserad information måste utelämnas, eller också förloras viktiga nyanser när informationen kodas om till en form som kan läggas till grund för automatiserad bearbetning i datasystemet.

Något karikerat kan vi säga att införande av automatisk databehandling (ADB) i en verksamhet ofta förknippas med följande egenskaper:

* rationalisering genom att informationsbehandlingen automatiseras med eliminering av manuella ingripanden.
* krav på att all information formaliseras, vilket leder till stela handläggningsregler och svårigheter att hantera avvikande situationer.
* användaren av ett ADB-system får i första hand en betjänande roll, med uppgiften att preparera indata, ta hand om utdata och åtgärda felsituationer.
* ändringar i informationshantering och system är svåra att genomföra, bl.a. på grund av att följdeffekter av en lokal ändring är besvärliga att överskåda.
* systemen ger i allmänhet användaren små möjligheter att kunna kontrollera hur utdata härletts ur indata.

Med ett sådant synsätt kan datatekniken uppfattas som ett redskap för centralisering, där kontrollen över systemets funktioner och den underliggande kunskapen samlas i speciella kompetenscentra. Därigenom kan fortlöpande rutinmässigt arbete utföras ute i en organisation, medan det egentliga expertkunnandet koncentreras till mindre grupper av specialister. Man får dock akta sig för att dra slutsatsen att sådana egenskaper är en nödvändig följd av användning av datateknik. I själva verket har vi att göra en avvägning mellan behovet av automatiserad bearbetning av informationen respektive önskemålet om mer realistiska modeller av verkligheten. Dagens ADB-system har i stor utsträckning prioriterat den förra synpunkten. I det följande ska vi diskutera aktuell utveckling inom områden där man i stället söker stödja en mänsklig beslutsfattare som i samspel med datorsystemet löser olika uppgifter.

2.2 Kontorsinformationssystem.

För närvarande diskuteras en hel del kring det förestående införandet av datorstöd i kontorsarbete, som kan förväntas ske i stor skala de närmaste åren. Det som karakteriserer denna utveckling är införande av ord- och text-behandling, informationssökning, meddelandesändning och småskaliga stödsystem snarare än automatisering av kontorets rutiner, för vilket ju redan i dag ADB-teknik utnyttjas i stor utsträckning.

Som en illustrativ kontrast till tradtionell databehandling, kan vi se system för informationsåtervinning (*information retrieval*), *IR-system*. I sådana system konverteras inte den egentliga informationen till en form som kan "förstås", dvs representeras i symbolisk form, av ett program i en dator, utan man

arbetar med fri text eller helt enkelt referenser till dokument eller motsvarande som hanteras vid sidan av det datoriserade systemet. Det som lagras i datorn utgörs i stället av *beskrivningar* eller *klassificeringar* av den tillgängliga informationen. Därmed avstår man också från möjligheten att automatisera mer än processen att söka fram potentiellt intressant information. Uppgiften att extrahera efterfrågade fakta ur en framsökt text eller dokument överlåts till användaren i varje särskilt fall. Med en sådan teknik vinner man å andra sidan flexibilitet, eftersom man inte i förväg behöver exakt fastställa vilket slags information som ska kunna representeras i det datorstödda systemet.

Om vi så går över till att titta på vad som utmärker databehandlingen i ett kontorsinformationssystem, finner vi klara skillnader jämfört med vad som är typiskt för administrativa ADB-system enligt den karakteristik vi gav tidigare. Enligt definitionen i [SAN82] har vi här att göra med:

> . . . en viss typ av administrativa informationssystem, nämligen sådana som användaren upplever som ett redskap för det egna arbetet.

Betoningen av redskap för egen informationshantering är viktig, eftersom den ger en annan intresse-centrering än för system avsedda för hela organisationens samlade verksamhet.

Kontorsinformationssystemen utmärks alltså av att de:

* ger stöd för icke-specialiserade tjänster, som redskap för enskilda personers informationshantering.

* understödjer hantering av såväl ostrukturerad (textbehandling) som formaliserad (formatterade databaser) information.

* utgörs väsentligen av småskaliga system, där rationaliseringseffekter snarare vinns genom integrering av olika arbetsuppgifter än av automatisering med standardiserad handläggning.

* betonar redskapsaspekten, varigenom användarens behov är av primär betydelse vid bedömning av redskapets användbarhet och effektivitet.

* förutsätter hög anpassningsbarhet, bl.a. eftersom omfattande systemutredningar i förväg knappast kan motiveras ekonomiskt.

* fungerar så att dess funktion enkelt kan överblickas och förstås av den enskilde användaren, som själv bär ansvaret för det slutliga resultatet utifrån det beslutsunderlag och den bearbetning som ett datorstött redskap tillhandahåller.

Vi kan alltså se att vi i den klass av informationssystem, som här diskuteras, kommer att kräva delvis annorlunda egenskaper än de som vi vanligen brukar associera med användning av ADB-teknik. Bl.a. kan kunskaper om avancerade IR-tekniker, sökstrategier, textanalys och indexeringsmetoder etc., spela en viktig roll vid uppbyggnaden av sådana system.

Exempel på typiska tjänster som tillhandahålls i ett sådant kontors-informationssystem är:

Ord- och text-behandling

Hjälpmedel för att skriva in och redigera text, med efterföljande utskrift omfattande även typografisk redigering (formattering) är centrala komponenter i ett kontorsdatasystem.

Meddelandesändning och telekonferenser

Datorsystemets roll som kommunikationskanal kan inte överskattas i detta sammanhang. Överföring av "data-brev" till enskilda personer eller grupper av mottagare liksom öppna diskussioner av typen "anslagstavla" motiverar en bred spridning av terminaltillgång respektive sammankoppling av datorer i kommunikationsnät.

Textlagring och sökning i databaser

Eftersom många ärenden som handläggs i kontorsmiljö handlar om hantering av fri-text-dokument, katalogisering och arkivering, har IR-system för informationssökning i textdatabaser sin självklara plats i repertoaren.

Hjälpmedel för dokumentframställning

Utöver text-editorer och -redigerare, kan speciella hjälpmedel för att förenkla produktion av olika dokument finnas tillgängliga, exempelvis program för automatisk stavningsrättning, avstavning respektive stöd för framställning av dokument med fast struktur eller delvis förutbestämt innehåll.

Personliga register

Utöver möjligheten att lagra och katalogisera dokument innehållande fri text, är det av stort värde att enkelt kunna lägga upp register med mer strukturerad information, exempelvis adressregister o.d. Ofta sker detta genom att en blankett-layout för den avsedda typen av information skapas direkt vid terminalen, varefter ifyllda blanketter kan lagras och bearbetas i systemet.

Kalendarium m.m.

Administration av gemensamma resurser, såsom lokaler och bok- och tidskrifts-bestånd kan med fördel skötas med datorstöd. Ett viktigt specialfall här är datorlagrade almanackor för förbättrad samordning vid tidsplanering av sammanträden m.m.

Frågesystem för administrativa databaser

Bekväma och effektiva frågespråk för flexibla frågor mot stora databaser har kommit i allmänt bruk under senare år, främst i samband med de s.k. relationsdatabassystemen.

Kalkylsystem

Principen för dessa system är att dataelement lagrade i en 2-dimensionell matris kan relateras till varandra med olika ekvationer, exempelvis på så sätt att det sista elementet i en rad ska utgöras av summan av de övriga elementen. Systemet sköter automatiskt om att beräkna nya beroende värden så snart något element ändras. Matrisen manipuleras interaktivt med direkta kommandon och någon programmering i traditionell mening behöver ej utföras för uppgifter av typen budgetsimuleringar m.m.

Grafiska presentationsprogram

För presentation av numeriska data är det värdefullt med tillgång till bekväma hjälpmedel för att producera på lämpligt sätt redigerade tabeller eller bilder i form av olika slags diagram.

Utrymmet här medger inte någon mer detaljerad diskussion av denna utveckling. Det är dock i sammanhanget viktigt att observera den betydelse som dessa nya kommunikationssystem kan komma att få i framtiden. Redan används telekonferenser, datorförmedlad post och samordnad dokumentframställning m.h.a. datanät som viktiga hjälpmedel inte bara i det internationella forskarsamhället utan också i många större företag, ofta sammanknytande verksamheter över hela världen.

Allmänt kan sägas att kontorsinformationssystemen i första hand karakteriseras av en passiv användning av tillgänglig information. Systemet hjälper till att producera, redigera, lagra och kommunicera texter och dokument, men saknar förmåga att på ett aktivt sätt utnyttja kunskapsinnehållet i den hanterade informationen. Den teknik som diskuteras i det följande syftar till att även tillhandahålla tjänster som baseras på system med förmåga att dra slutsatser av den kunskap som lagrats in.

2.3 Kunskapsbaserade system.

Mot bakgrund av den diskussion som vi fört tidigare i detta kapitel, kan vi uppfatta användning av kunskapsbaserade programvarusystem som en ytterligare breddning av användningsområdet för informationsteknologi. Ett kunskapssystem är ett datorprogram för problemlösning inom ett väl avgränsat område. Vissa system fungerar som självständiga problemlösare, men vanligare är att systemen ger kvalificerat stöd vid beslutsfattande eller problemlösning. I en viss mening kan alla datorprogram sägas hantera kunskap, men det finns ett antal unika kännetecken på ett genuint kunskapssystem:

- *Graden av formalisering är mindre.* Kunskap som inte låter sig uttryckas i en rent matematisk modell kan formuleras i symbolisk logik, eventuellt med resonemang baserade på sannolikheter och med heuristik ("tumregler").
- *Kunskapen lagras explicit.* Begrepp, regler och fakta hanteras och underhålls separat från det egentliga programmet, vilket ger

exceptionellt goda egenskaper när det gäller att utveckla kunskapssystemet och att anpassa dess funktion till en föränderlig omvärld.

- *Självförklarande program.* Ett kunskapssystem ska kunna förklara hur det kommer fram till en viss slutsats eller rekommendation och användaren måste kunna se vilka regler, fakta och antaganden som beslutet grundar sig på. Generella förklaringsmekanismer är en viktig del i ett kunskapssystem.

Kunskapssystem möjliggör i vissa fall datorstöd där detta tidigare inte var genomförbart, i andra fall ger kunskapssystemen effektivare lösningar än den traditionella programvarutekniken.

Kunskapssystem kan:

- Effektivisera beslutsfattande och problemlösning genom att stödja, komplettera eller i vissa sammanhang ersätta mänskliga insatser.
- Ge fler tillgång till det expertkunnande som ofta bara finns centralt i en organisation. De kan ge personalen på exempelvis ett lokalt kontor eller mannen/kvinnan på fältet tillgång till ett effektivt beslutsstöd.
- Minska den sårbarhet och det beroende som kommer av att kunnande och erfarenhet i en organisation ofta är knuten till ett fåtal experter.
- Frigöra befintlig expertis. Anhopning av rutinärenden kan förhindra experterna att ägna sina krafter åt verkligt kvalificerade problem, vidareutveckla sig själva och skapa ny kunskap. Kunskapssystem kan ge experterna tid att vara just experter.
- Rationalisera utveckling och underhåll av vissa typer av informationssystem, genom att bl.a. minska avståndet mellan den egentliga områdeskunskapen och rutinerna i de implementerade programmen.

Kunskapssystem är lämpliga när:

- Kunskapskrävande rutinarbete kräver koncentration
  (- datorn blir inte uttråkad).
- Många olika data påverkar beslutsprocessen
  (- datorn blandar inte bort korten).
- Snabba beslut måste fattas i svåra situationer
  (- datorn blir inte stressad).
- Expertkunnandet är svårt att hålla ajour
  (- datorn glömmer inte).
- Användarnas egen kompetens måste vidmakthållas
  (- datorn kan förklara).

Den allmänna trenden är att datorsystemen mer och mer utvecklas till *kommunikationssystem*, där funktionerna att utföra beräkningar och att administrera data blir funktioner underordnade uppgiften att aktivt förmedla information och kunskap relativt ett givet behov. I denna process får förmågan att i systemen hantera ostrukturerad information och informella kunskaper en ökande betydelse. Datorsystemens karaktär av redskap betonas allt mer och insikten ökar om behovet att tillvarata människans kompetens och förmåga för att optimalt utnyttja möjligheterna till effektiv problemlösning i samspel människa dator.

3.

Kunskapsteknik

I likhet med många andra områden, där utvecklingen har gått fort och där en snabb kommersialisering av forskningsresultat skett, lider det område som beskrivs i denna rapport av en långt gången begreppsförvirring. Detta gör det tyvärr onödigt svårt för en utomstående att bilda sig en uppfattning om *state of the art* och vad som är underliggande tekniska realiteter. Samma begrepp används ofta för att beskriva helt olika saker, medan i andra sammanhang identiska fenomen ges radikalt olika beskrivningar.

Vi har i nästa avsnitt sammanställt en del definitioner av de grundläggande begreppen AI respektive expertsystem. Den senare termen har på gott och ont fått en slagkraft, som gör att det är svårt att ersätta den med ett lämpligare och mera precist begrepp. Just denna term har fått en uttunnad betydelse i samband med dess starka attraktionskraft, inte minst i kommersiella sammanhang. Utan att göra anspråk på att uttrycka den "sanna" innebörden i berörd terminolgi. vill vi dock ge följande synpunkter på den terminologi som används i samband kunskapsbaserade system:

Begreppet expertis

Begreppet expertsystem går tillbaka både på systemens förmåga att lösa problem för vilka vi vanligen förutsatt tillgång till mänsklig expertis, och på förutsättningen att den modell för problemlösning som utnyttjas baseras på någon enstaka människas kunnande och insikter, snarare än på en matematisk teori eller ett formellt regelsystem. Den förhärskande användningen tycks i praktiken vara för system som löser "svåra" problem. Det är uppenbart att ingen objektiv definitition kan ges av vilka problem som är svåra i den meningen att de borde kräva mänskliga expertkunskaper för sin lösning. I själva verket tenderar vi att per definition anse att de uppgifter som ett dataprogram kan klara *inte* förutsätter egentlig expertis, i varje fall inte i någon djupare mening. Det har ibland framförts att termen *specialistsystem* skulle vara att föredra, eftersom den tydligare markerar systemens roll som problemlösare och rådgivare inom (i praktiken) klart avgränsade områden. Mot bakgrund av denna diskussion förefaller det rimligt att låta termen expertsystem i första hand associera till systemets prestationer och i mindre grad till den utformning och implementering som programmet har fått.

Begreppet kunskap

Innebörden av detta begrepp har ända sedan de gamla grekerna varit ett problem som sysselsatt filosofiska tänkare och det blir på intet sätt eliminerat i samband med införandet av s.k. kunskapsbaserad programvara. I detta sammanhang har emellertid termen fått en pragmatiskt definierad innebörd, som tar fasta på kunskapens generella användbarhet, dvs möjligheten i olika situationer kunna utnyttja denna för att lösa nya typer av problem. Detta står då i kontrast till den traditionella synen på programmering, där vi systematiskt förbereder hur varje upptänklig situation ska hanteras och i detalj förskriver vad som då ska ske. Med termen kunskapsbaserad avses därmed sådan programvara, där man i systemet representerar en i någon mening abstrakt kunskap, som till form och innehåll ligger nära dess naturliga formulering utanför systemet och som dynamiskt kan appliceras i olika uppkommande situationer. Därmed markeras betydelsen av affinitet mellan det som representeras i systemet och det som exempelvis en expert inom ett område uttrycker.

Begreppet regler

Intresset i samband med kunskapsystem fokuseras ofta till användningen av regler för att uttrycka kunskap. Ofta används termen *regelbaserat system* som tämligen synonymt med expertsystem. Detta är olyckligt eftersom det på inget sätt alltid är givet att en formalism baserad på användning av regler är vare sig det naturliga sättet att uttrycka kunskap eller att åstadkomma en problemlösande förmåga på expertnivå. Dock har man haft en notabel framgång inom vissa områden när det gäller att utnyttja en strukturering av kunskap i form av regler som en bas för att åstadkomma modularisering och flexibilitet i problemlösning. Man kan också notera att man ibland associerat termen regelbaserat med att ursprungskunskapen är uttryck i ett regelverk av något slag, t.ex. en lagstiftning. Det är dock svårt att se någon nödvändig koppling mellan sådana tillämpningsområden och tekniken med kunskapssystem implementerade med en regelformalism. Om regelsamlingen är uttömmande och precis brukar man inte ha några svårigheter att implementera dess tillämpning med konventionell programvaruteknik, och om den omvänt snarare ger ramarna för ett beslutsfattande kan man förvänta sig att den intressanta och svårfångade kunskapen ligger utanför det egentliga regelverket. Den dominerande användningen av termen regelbaserat system är för ett program där huvuddelen av informationsbearbetningen definieras av ett antal oberoende regler som appliceras på givna indata av en speciell programmodul som tolkar reglerna som instruktioner.

Mot bakgrund av den allmänt omvittnade centrala betydelse som den i systemet representerade kunskapen har för dess prestationsförmåga, har vi valt att i första hand använda termen *kunskapsssystem* som sammanfattande beteckning. Ovanstående diskussion har syftat till att belysa användning av andra vanliga termer inom området. Läsaren bör vara klar över att många, delvis oförenliga, åsikter råder om vad som är korrekt terminologi inom

området.

I det följande ska vi efter ett avsnitt om definitioner ge en kort översikt över de viktigaste teknikerna som kommit till användning när det gäller att representera och använda områdeskunskap som en bas för problemlösning i datorbaserade system. Framställningen gör inte anspråk på att ge en uttömmande beskrivning av berörda tekniker, utan avsikten är snarare att ge en känsla för inriktning och terminologi inom området.

I begreppet expertis ingår inte bara förmåga att lösa problem inom ett specialgebit, utan också att kunna förklara en lösning eller ett resultat, att kunna assimilera ny kunskap, samt att veta gränserna för när en viss kunskap kan tillämpas. För att kunna uppnå sådana egenskaper i ett datorbaserat system använder man vanligen objektbeskrivande strukturer och någon form av regelspråk för att uttrycka kunskapen om tillämpningsområdet och dess begreppsapparat. Hit hör exempelvis orsakssamband, logiska relationer, restriktioner, handläggningsrutiner, åtgärdsscheman, erfarenheter, etc. Utformningen av denna representation med tillhörande mekanismer för slutsatsdragning kan sägas vara det centrala problemet vid konstruktion av kunskapssystem.

3.1 Något om definitioner

I stället för att försöka ge en formell definition av termerna AI respektive expertsystem, har vi valt att återge några citat ur litteraturen inom området. Dessa citat speglar uppfattningar som delvis avviker något från varandra men som samtidigt uttrycker de dominerande uppfattningarna om innebörden i dessa mångtydiga begrepp.

3.1.1 Artificiell Intelligens

Några framträdande representanter för forskningsområdet *artificiell intelligens* har kommenterat områdesbeteckningen på följande sätt:

"Det finns många sätt att definiera området artificiell intelligens (AI). Här är ett:

- *AI är studiet av ideer som kan möjliggöra för datorer att fungera på ett intelligent sätt.*

Men vad är intelligens? Är det förmågan att resonera? Är det förmågan att tillägna sig och använda kunskap? Är det förmågan att uppfatta och manipulera ting i den fysiska omvärlden? Uppenbarligen är alla dessa förmågor viktiga delar av vad intelligens är, men det är lika uppenbart inte hela sanningen. En definition i konventionell mening tycks omöjlig, eftersom intelligens utgörs av en sammansättning av så många olika informationsrepresenterande och informationsbehandlande förmågor. Icke desto mindre kan målsättningarna för forskningsområdet AI definieras:

- *Ett centralt mål för AI är att göra datorer mer användbara.*
- *Ett annat centralt mål är att förstå de principer som gör intelligens möjlig."*

Patrick H. Winston, Massachusetts Institute of Technology.

Datorn och AI-utvecklingen hotar emellertid den moral som bygger på tesen att människan intar en särställning inom och är fristående från resten av naturen:

"Definitionen av människans särställning har alltid utgjort kärnan i hennes kosmologiska och etiska system. Med Kopernikus och Galileo upphörde hon att vara den art som placerats mitt i universums centrum, omgiven och uppvaktad av solen och stjärnorna. Med Darwin upphörde hon att vara den art som Gud skapat och specialutrustat med själ och förnuft. Med Freud upphörde hon att vara den art vars beteendemönster potentiellt kunde styras av det rationella förnuftet. Allt eftersom vi börjar skapa mekanismer som kan tänka och lära, är människan inte längre den enda art som är utrustad med den unika förmågan att manipulera sin omgivning på ett komplicerat och intelligent sätt."

Herbert Simon, nobelpristagare i ekonomi 1978 och banbrytande forskare inom AI.

3.1.2 Expertsystem

På samma sätt finns olika uppfattningar om hur begreppet *expertsystem* ska definieras:

"Kunskapsbaserade expertsystem, eller kortare kunskapssystem, utnyttjar mänsklig kunskap för att lösa problem som vanligen erfordrar mänsklig intelligens."

Fredrick Hayes-Roth, *IEEE Computer*, September 1984.

"Ett expertsystem är ett regelbaserat AI tillämpningsprogram för att utföra en uppgift som kräver expertkunnande."

Charniak, McDermott: *Introduction to Artificial Intelligence*

"Det var inte förrän under slutet av sjuttiotalet, som forskare inom AI började förstå något särdeles betydelsefullt, nämligen att den problemlösande styrkan hos ett program kommer från den kunskap det besitter, inte bara från de formalismer och resonemangsmetoder som det använder. Det begreppsmässiga genombrottet var därmed gjort och kunde enkelt uttryckas:

> *För att göra ett program intelligent, förse det med mängder av hög-kvalitativ kunskap om det aktuella problemområdet.*

Denna insikt ledde till utvecklingen av specialiserade datorprogram, som var experter på något avgränsat problemområde. Dessa program kallades för expertsystem."

Don Waterman: *A Guide to Expert Systems*

(Ett expertsystem är) *"ett intelligent datorprogram, som använder kunskap och slutledningsprocedurer för problem, som är svåra nog för att kräva betydande mänsklig expertis för sin lösning. Kunskap nödvändig för prestationer på den nivån, tillsammans med använda slutledningsprocedurer, kan uppfattas som en modell av de mest framstående specialisterna inom området.*

Kunskapen i ett expertsystem består av fakta och heuristik. Fakta bildar en informationsmängd som delas av många, är allmänt tillgänglig, och om vilken det inte råder några delade meningar bland experterna inom området. Heuristiken är vanligen personligt betonade, informella principer för gott omdöme (regler för plausibla slutsatser, regler för goda gissningar) vilka utmärker beslutsfattande på expertnivå inom ett område. Prestationsnivån hos ett expertsystem är primärt en funktion av storleken och kvaliteten hos den kunskapsbas systemet

besitter."

Edward A. Feigenbaum, Stanford University

"Det finns ingen enstaka definition av ett expertsystem . . . De huvudsakliga dimensionerna efter vilka jag föredrar att definiera expertsystem är följande:

1. *AI metodik -- expertsystem är AI program, dvs program som arbetar med symbolisk information och använder heuristiska (icke-algoritmiska) slutledningsprocedurer.*
2. *Höga prestanda -- prestationsförmåga på expertnivå är vad konstruktörerna av dessa system eftersträvar, men detta är inte heller särskilt lätt att definiera. . . .*
3. *Flexibilitet -- AI program är i allmänhet mer flexibelt utformade än algoritmiska program, delvis beroende på att denna egenskap krävs för att tillåta modifieringar när problemet blir bättre definierat. I tillägg till flexibilitet vid utvecklingstillfället är det också önskvärt att ett expertsystem uppvisar flexibilitet vid användning. Speciellt gäller att ju mer tolerant ett system är för oförutsedd inmatning, nya sammanhang eller tillämpningar, eller olika typer av användare, desto mer expertbetonat kommer det att uppfattas.*
4. *Förståbarhet -- på samma sätt som en expert kan förklara sina resonemang, ska ett expertsystem kunna förklara sina slutledningar och innehållet i sin kunskapsbas. Detta är likaså viktigt vid utvecklingstillfället, vid felsökning och för ett accepterande av rimligheten i systemets slutsatser."*

Bruce G. Buchanan, Stanford University

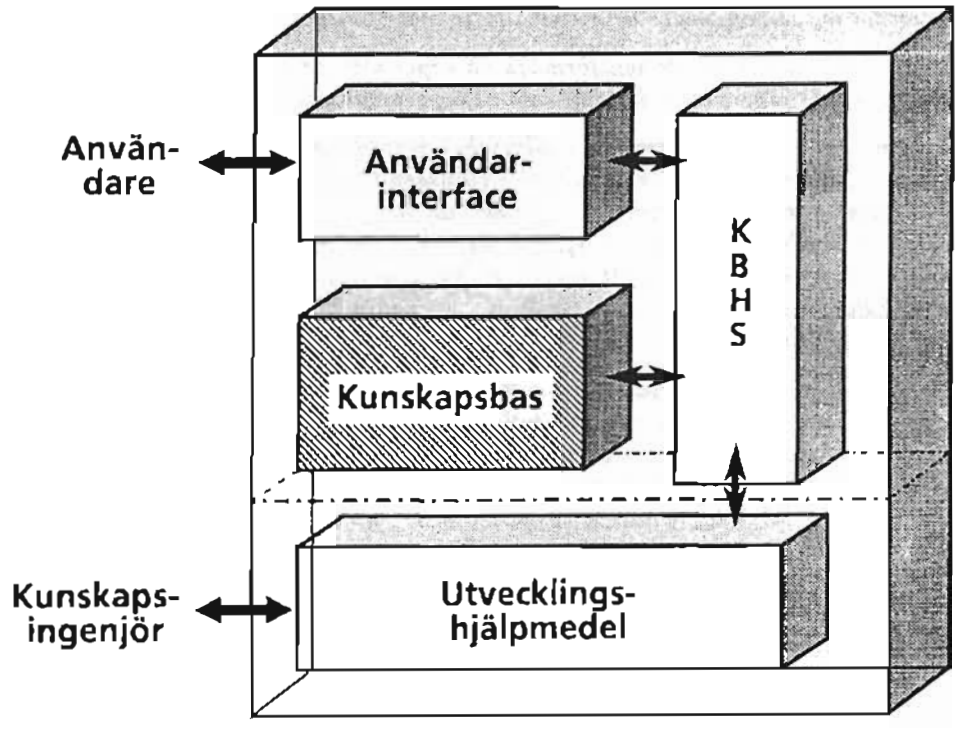
Dessa definitioner, ehuru tämligen samstämmiga, ger i praktiken stort utrymme för olika tolkningar av vilka program som kan anses utgöra expertsystem. I dag finns en tydlig tendens att använda den sammanfattande termen *kunskapssystem*, för den klass av programvara som beskrivs i denna rapport. Starka skäl talar för användning av dennna term, som betonar den centrala rollen för kunskaper i en djupare mening än den traditionella databehandlingens bearbetning av data enligt välspecificerade algoritmer.

Därmed borde termen *expertsystem* reserveras för de program som verkligen uppvisar problemlösningsförmåga i paritet med mänskliga experter inom ett på lämpligt sätt avgränsat område, och detta oavsett vilken teknisk lösning som använts. Termen *kunskapsbaserat* borde i sammanhanget betyda ett system där underliggande kunskaper representeras på ett problemnära sätt, med möjligheter till modulärt underhåll, förklaringar och återanvändning för nya ändamål. Om vår tekniska lösning av detta problem baseras på användning av uttryck av typen *OM A gäller SÅ utför B* (eller *dra slutsatsen B)*, så är beteckningen *regelbaserat system* motiverad.

3.1.3 Kunskapssystem

Vi har i det föregående diskuterat fördelarna med att fokusera intresset på området kunskapssystem, dvs programvara baserad på explicit representerade och modulärt manipulerbara kunskapsfragment. En viktig egenskap i ett sådant system är uppdelningen i kunskapsbas respektive programstyrning av det som konventionellt motsvarar ett enda sammanhållet program. Med denna

uppdelning förenklas väsentligt möjligheten att förstå systemets funktion och underhålla kunskapen och därmed programsystemet. Dessutom får man en möjlighet att återanvända kunskapen för olika ändamål, exempelvis för problemlösning respektive simulering eller undervisning.



Figur 3.1. *Strukturen hos ett kunskapssystem.*

Det har ibland visat sig vara instruktivt att göra en uppdelning av kunskapssystemen i tre klasser, utgående från den huvudsakliga metoden för representation av kunskap:

regel-baserade system
: Detta är den klassiska typen av expertsystem, representerad exempelvis av det berömda MYCIN-programmet för diagnosticering av bakterieinfektioner. Den genomgående problemlösningsmetoden är användning av s.k. produktionsregler, som väsentligen säger om en given situation är för handen ska vissa slutsatser dras eller vissa aktioner utföras.

objekt-orienterde system
: Dessa system betonar möjligheten att organisera kunskapen runt beskrivningar av olika klasser av objekt, inkluderande även hur objekten är relaterade till varandra och hur de reagerar i olika situationer. Viktigt är också stöd för s.k. ärvning av egenskaper från mer generella överordnade begrepp.

logik-baserade system
Till denna klass hänförs vanligen system konstruerade med hjälp ett generellt programmeringsspråk baserat på predikatlogik, exempelvis Prolog.

3.2 Kunskapsrepresentation m.m.

En utgångspunkt för de tillämpningar där kunskapssystem aktualiseras är att områdets fackkunskap endast i begränsad utsträckning kan uttryckas på ett välstrukturerat, algoritmiskt sätt. Expertkunskapen formuleras i stället ofta i form av ett antal relativt oberoende s.k. *produktionsregler*. Man har då ett eller flera *villkor* eller *premisser* och en *slutsats* eller *åtgärdsdel*, som följer om premisserna kan visas vara sanna.

```
RULE015
-------
[This rule is tried in order to find out about whether a community
     permission is necessary]

  If:  1) The transaction is not regarded as a selling, or
       2) A: The receiver is not under age, and
          B: The donator is not under age

  Then:  It is definite (1.0) that a community permission
           is not necessary


PREMISE:  [$AND ($OR (NOTSAME CNTXT KOP)
                     ($AND (NOTSAME CNTXT MOMYNDIG)
                            (NOTSAME CNTXT GOMYNDIG]
ACTION:  (CONCLUDE CNTXT OVERFORM-TILLSTAND YES TALLY -1000)
```

Figur 3.2. *Extern och intern form av en produktionsregel.*

Denna teknik har den stora fördelen att nya kunskaper lätt kan tillföras genom att mängden regler utvidgas. Nackdelen är den ineffektivitet som resulterar om kunskapen inte struktureras ytterligare. Metoder för sådan strukturering med bibehållen enhetlighet och modularitet i representationen är en viktig frågeställning inom området kunskapsbaserade expertsystem. Exempel på lösningar är att införa *metaregler*, som beskriver när en viss regel bör tillämpas eller att samla regler i grupper som aktiveras i olika situationer.

Den formella grunden för konstruktion av sådana regelbaserade system utgörs av den matematiska logiken, speciellt den s.k. predikatlogiken. Denna formalism ger oss en sund teoretisk grund inte bara för att uttrycka

påståenden om en given verklighet, utan också för att applicera olika slutledningsmekanismer för att avgöra om sanningen i en utsaga kan härledas ur kända fakta. Programmeringsspråket Prolog, som baseras på sådan logik, utför automatiskt slutsatsdragning av denna typ. Andra språk som kommit till användning i sammanhanget betonar i högre grad möjligheten att konstruera mer problemnära representationsmetoder än logiska formler och att ge användaren större möjlighet att styra programmets arbete.

En ofta citerad fördel med att genomgående använda regler är att den enhetliga formen ger en effektiv grund för att på olika sätt manipulera den kunskap som reglerna uttrycker. Exempelvis kan man enkelt automatiskt producera förklaringar av resonemang i ett sådant system. Omvänt får man ofta problem med att hålla isär regler som uttrycker olika slags kunskap, t.ex. om orsakssamband, tvingande åtgärder, restriktioner, begreppsklassificeringar, sekvenseringar motiverade av effektivitetsskäl, osv.

3.2.1 Representation av objektstrukturer

Olika system skiljer sig ganska mycket åt när det gäller vilken teknik som används för att representera kunskap om de olika begrepp och typer av objekt som förekommer inom tillämpningsområdet och hur dessa är relaterade till varandra. Den allra enklaste modellen är att arbeta med ett antal namngivna variabler som kan anta olika värden. I praktiken sammanför man dock vanligen sådana variabler till en uppsättning egenskaper som beskriver en viss klass av objekt, vilket innebär att man arbetar med s.k. objekt-attribut-värde tripler. (Exempel: ÄRENDE-1 - HANDLÄGGARE - SVENSSON) Vissa system nöjer sig med detta och förlitar sig på användning av regler eller användarprogrammerade metoder för att uttrycka mer komplicerad kunskap om objekts egenskaper och hur olika begrepp är relaterade till varandra.

Mer avancerade system baseras på användning av s.k. *frames* (bra svenskt ord saknas) för att representera fakta. Med denna teknik kan man till ett objekt (eller begrepp) inte bara knyta enkla egenskapsvärden som beskriver objektet, utan också definiera en uppsättning procedurer (t.ex. med hjälp av av regler) som anger hur det ska reagera i olika sammanhang, samt hur det är relaterat till andra objekt, exempelvis hur kunskap om objektet kan härledas från andra relaterade objekt (ärvning av egenskaper).

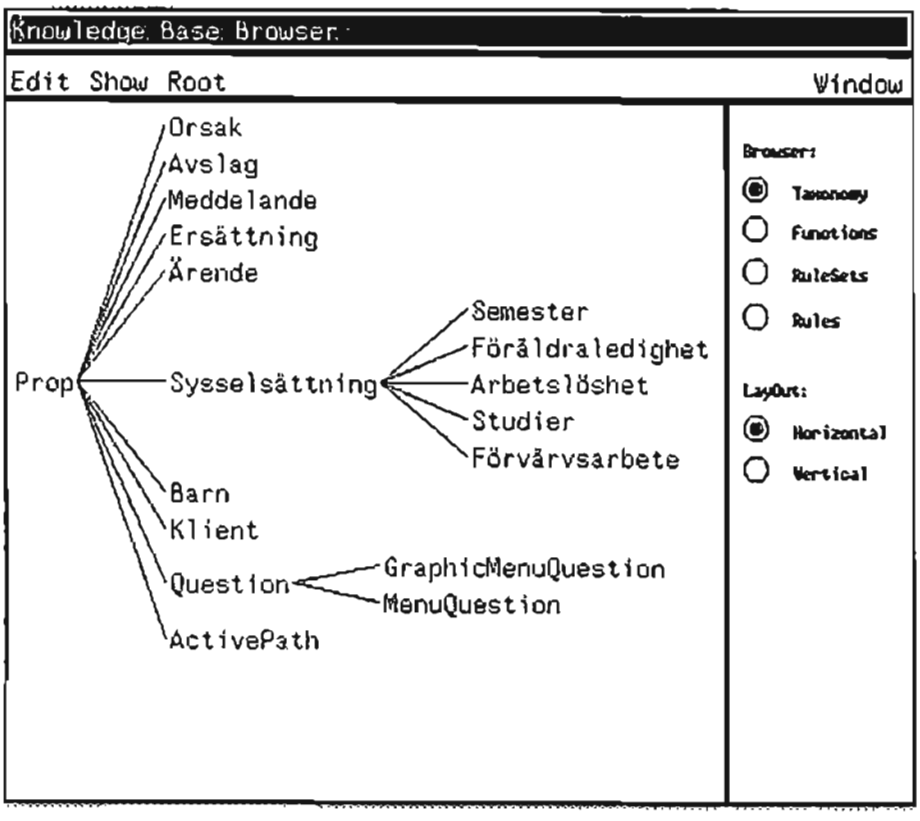
Ofta representeras mer komplicerade utsagor i något som kallas ett *semantiskt nätverk*, dvs associationer mellan exempelvis objekt, begrepp, händelser och andra beskrivande egenskaper. Till kunskapsdatabasen för en viss tillämpning kan också höra en *taxonomi* eller *klassifikationshierarki* som anger hur olika begrepp inom området är relaterade till varandra. Taxonomin kan exempelvis användas för att härleda vilka mer generella eller abstrakta kunskaper som kan appliceras vid en speciell frågeställning.

Det är också möjligt att direkt använda någon form av standard logik (exempelvis satslogik eller predikatlogik) för att uttrycka kunskaper inom ett

```
***** MANUFACTURING-JOINT *****

Concept name: MANUFACTURING-JOINT
Created-by: HANS
Created-at: 20-Jan-86 16:56:57
Modified-by: HANS
Modified-at: 21-Jan-86 08:15:09
Generalizations: JOINT
Specializations: GLUE-JOINT,SOLDERED-JOINT,WELDED-JOINT
Aspects:
  MAX-MANUFACTURING-TEMP
    Type: (DESCR REAL)
    Valuemode: INDVAL
    Askmode: ASK
    Cachemode: NOCACHE
    Dictionary: (QUESTION: WHAT IS THE MAXIMUM PERMITED MANUFACTURING TEMPERATURE? (DEGREE CELSIUS))
  SLIT-WIDTH
    Type: (DESCR REAL)
    Valuemode: INDVAL
    Askmode: ASK
    Cachemode: NOCACHE
    Dictionary: (QUESTION: WHAT IS THE SLIT WIDTH? (MM))
  JOINING-TYPE
    Type: (DESCR INTEGER)
    Valuemode: INDVAL
    Askmode: ASK
    Cachemode: NOCACHE
    Dictionary: (QUESTION: WHAT TYPE OF JOINT? BUTT JOINT (1) , T-JOINT (2) , LAP JOINT (3) , CORNER
JOINT (4) , FLANGED JOINT (5) , EDGE JOINT (6))
  IMPOSSIBLE-JOINT  from: JOINT
    Type: (DESCR STRING)
    Valuemode: INDVAL
    Askmode: NOASK
    Cachemode: CACHE
Request driven inference methods: RLS#15


*** RDIM: RLS#15 ***

Ruleset
     Workspace = MANUFACTURING-JOINT
     Control = DO1
     Stop = FALSE
     Access = DEDUCE
  Rule: R#1
     If (AND (= (THE MATERIAL)
                STEEL-STEEL)
             (> (THE TENSILE-STRENGTH)
                20)
             (> (SHEARING-STRENGTH)
                10))
     Then CONCLUDE (THE IMPOSSIBLE-JOINT)
                   SOLDERED-JOINT
  Rule: R#2
     If (< (THE MAX-MANUFACTURING-TEMP)
          80)
     Then CONCLUDE (THE IMPOSSIBLE-JOINT)
                   (SOLDERED-JOINT WELDED-JOINT)
  Rule: R#3
     If (> (THE SLIT-WIDTH)
          0.5)
     Then CONCLUDE (THE IMPOSSIBLE-JOINT)
                   SOLDERED-JOINT
```

Figur 3.3. *Exempel på en "frame" med tillhörande regler.*

område, t.ex. genom användning av programmeringsspråket Prolog eller motsvarande primitiver i andra system.



Figur 3.4. *Exempel på begreppshierarki.*

3.2.2 Slutledningsmekanismer

För att kunna dra slutsatser från den kunskap som representeras i ett kunskapssystem måste vi implementera någon form av slutledningsmekanism. Givetvis måste dessa båda aspekter utformas i ett sammanhang. För regelbaserade system av den typ som indikerades i föregående avsnitt gäller att det program som tolkar reglerna i databasen brukar kallas för ett *inferenssystem*. Denna mekanism fungerar vanligen i princip så att de storheter som nämns i reglerna jämförs med de fakta som är kända för det aktuella problemet, varefter relevanta regler evalueras och sanningsvärdena för berörda storheter uppdateras.

Vid *framåtriktad* eller *datadriven slutsatsdragning* (*forward chaining*) jämförs kända fakta med reglernas premisser, varvid regeln kan appliceras om samtliga premisser kan tilldelas ett sanningsvärde. Om flera regler är tillämpbara används någon strategi för prioritering mellan dem (*conflict resolution*). Vid *bakåtriktad* eller *målstyrd slutsatsdragning* (*backward chaining*) försöker man verifiera en hypotes genom att hitta regler vars slutsatser styrker hypotesen. Om fakta för att värdera premisserna inte är kända, kan systemet fråga slutanvändaren. Båda metoderna syftar till att härleda sökta storheter ur mera elementära kända fakta och används praktiskt i förekommande expertsystem.

Sådan slutsatsdragning kan också tillhandahållas direkt i ett programmeringssystem, t.ex. i Prolog. Detta språk är baserat på målstyrd slutsatsdragning.

3.2.3 Stöd för plausibla resonemang

En betydelsefull egenskap i många expertsystem är tillgången till ett systematiskt stöd för att hantera såväl inexakta och otillförlitliga data som erfarenhetsmässigt osäkra orsakssamband och beslutsregler. Möjligen kan man hävda att detta är den mest utmärkande egenskapen för ett expertsystem.

Eftersom antaganden om osäkerhet i data och behandlingsregler är något som radikalt avviker från de förutsättningar man normalt arbetar med vid konstruktion av programvara får detta stora konsekvenser för vår syn på det utvecklade programmet. En sund metodik i normal systemutveckling bygger på att man först noggrant specificerar programförutsättningarna och sedan verifierar att det implementerade programmet under givna krav på indata producerar korrekta utdata. Denna teknik fungerar bra i många sammanhang, under förutsättningen att vi verkligen kan ge en exakt formell specifikation av programmets uppgift. Emellertid gäller detta typiskt inte för uppgifter som fordrar expertis för sin lösning. Exempelvis misslyckades grundligt de optimistiska försök som tidigt gjordes att formulera rättsväsendets lagar i algoritmiska språk, bl.a. på grund av omöjligheten att fullständigt formalisera den juridiska tillämpningen.

Ett expertsystem konstrueras ofta för att prestera en rekommendation (t.ex. lämpligt läkemedel och dos), som inte kan verifieras formellt, eftersom korrekthetskriterier saknas eller är okända. I ett sådant system måste tilliten till utdata baseras på empiri snarare än formella bevis. Tillgång till förklaringssystem enligt ovan är då givetvis oundgänglig, eftersom vi måste ha möjlighet att i varje särskilt fall undersöka vilka förutsättningar en rekommendation baseras på.

Typiskt för expertsystemen är alltså att kunskapen vanligen inte är absolut. Premisserna för en regel kan vara etablerade med en viss grad av osäkerhet och slutsatsen kan i sin tur följa endast med en begränsad sannolikhet. Ofta associeras en *visshetsfaktor* med varje regel, som anger med vilken grad av säkerhet som slutsatsen följer ur premisserna. Denna faktor kan uppfattas som ett mått på hur mycket tilltron till slutsatsen stärks när premisserna visats vara sanna.

Kärnfrågan här är givetvis vad dessa osäkerhetsmått egentligen uttrycker och den därtill kopplade frågan om hur de kan kombineras och fortplantas. Klassisk sannolikhetslära är i praktiken inte särskilt användbar, även om den ligger till grund för en del av de modeller som tillämpats i expertsystem. Allmänt kan sägas att de osäkerhetsmått man arbetar med vanligen uttrycker en subjektiv tilltro till styrkan hos en utsaga eller beslutsregel. Man kan också notera att i detta mått vägs även "risken" eller "värdet" av en slutsats in,

exempelvis kan man vid felsökning vilja prioritera upp diagnoser som är ekonomiskt mer "värdefulla" eller "farliga" jämfört med andra som i och för sig är lika sannolika.

3.2.4 Underliggande logik

Det torde ha framgått av det föregående att användning av matematisk logik är ett viktigt hjälpmedel för att representera kunskap och för att härleda sådana fakta som inte är explicit representerade. Ett sådant logiskt system definierar en uppsättning slutledningsregler med vilkas hjälp vi från en mängd sanna utsagor kan härleda slutsatser eller prova sanningshalten i andra utsagor. I den s.k. *satslogiken* utgörs utsagorna av enkla fakta, medan *predikatlogiken* också tillåter utsagor om egenskaper som gäller för alla individer (eller någon individ), som tillhör en på lämpligt sätt definierad klass. I satslogiken kan man alltså säga "*Sokrates är människa*" och i predikatlogiken dessutom "*Alla människor är dödliga*". Predikatlogikens slutledningsregler ger slutsatsen att "*Sokrates är dödlig*". Emellertid har predikatlogiken, trots sin stora användbarhet i samband med kunskapsbaserade system, också sina begränsningar när det gäller att representera alla de typer av kunskaper som är relevanta i sammanhanget.

Inom den filosofiska vetenskapen har svagheterna hos den klassiska logiken när det gäller att uttrycka mänsklig kunskap länge varit kända. Logiken kan vi uppfatta som ett formellt system, som på intet sätt är beroende av empirisk kunskap. Därmed är det också möjligt att utveckla alternativa logiska system och sedan prova deras förklaringskraft när det gäller olika sidor av mänskligt vetande. Sådana alternativa system, kan endera vara utvidgningar av klassisk logik eller också mer strikta alternativ, dvs att det finns satser som är sanna i den klassiska logiken men falska i den alternativa. Ett typiskt sådant exempel är lagen om *det uteslutna tredje*, dvs att varje påstående antingen är sant eller också falskt.

För närvarande knyts ett stort intresse till användning av icke-standard logik i kunskapsbaserade system. Nedan ges exempel på olika typer av logiker, som är av intresse i sammanhanget. För en mer detaljerad genomgång hänvisas till [TUR84].

Klassisk första ordningens logik

Hit räknar vi alltså sats- och predikatlogiken. Att logiken är av första ordningen innebär (något förenklat) att predikaten uttalar sig om individer men inte om andra predikat. Implementering av resonemangsmodeller baserade på denna logik spelar en stor roll inom området kunskapssystem och även generella programmeringsspråk, exempelvis Prolog, har sin bas i första ordningens predikatlogik.

Modal logik och dynamisk logik

Denna typ av logik kan vi se som en utvidgning av den klassiska logiken med syfte att också kunna hantera utsagor, vilkas sanning är *nödvändig* respektive *möjlig*. Praktiskt ger det oss möjlighet att resonera om s.k. hypotetiska världar, dvs sådana som är förenliga med vad vi vet om den aktuella världen och som kan uppnås från denna på något logiskt konsistent sätt. En nödvändig sanning är då något som gäller i alla dessa möjliga världar. Att kunna bedriva resonemang av denna typ är av den största betydelse i många kunskapssystem, t.ex. sådan som sysslar med planering. Flera av de utvecklingsverktyg som finns på marknaden understödjer resonemang runt hypotetiska världar.

Flervärda logiker

Klassisk logik baseras på två sanningsvärden, dvs att ett påstående endera är sant eller falskt. För praktisk tillämpning är detta antagande tämligen orealistiskt. Flera ansatser att hantera detta problem finns, bl.a. genom att introducera ett tredje sanningsvärde för påståenden som varken är sanna eller falska. Man kan tolka innebörden hos ett sådant sanningsvärde på några olika sätt, exempelvis som:

okänt	dvs vi kan för tillfället inte avgöra om det är sant eller falskt;
obestämt	dvs varken sant eller falskt och alltså i någon djupare mening obestämt;
meningslöst	dvs påståendet kan av principiella skäl varken vara sant eller falskt, exempelvis den kända s.k. lögnar-paradoxen, "jag ljuger nu".

Dessa olika tolkningar svarar mot olika sätt att definiera sanningsvärden för sammansatta utsagor, varför det finns flera olika varianter av 3-värd logik. Vissa kunskapssystem har inbyggda hjälpmedel för att understödja hantering av utsagor med ett okänt sanningsvärde.

Intuitionistisk logik och typteori

Intuitionistisk logik och matematik bygger på kravet att ett bevis av en sats för att godtas måste vara *konstruktivt*, dvs explicit ange härledningen av "objekt" vilkas existens ska bevisas. Därmed kan man också undvara lagen om det uteslutna tredje, vilken visat sig under vissa förutsättningar kunna leda fram till inkonsistenta slutsatser. Det praktiska värdet av den intuitionistiska logiken är att *konstruktiva bevis*, baserade exempelvis på Martin-Löfs typteori, väsentligt förenklar uppgiften att automatiskt framställa och verifiera program från en formell logisk specifikation.

Icke-monoton logik

Klassisk logik är baserad på s.k. monoton slutsatsdragning, vilket innebär att mängden bevisade sanningar är strikt växande, dvs ny information som tillförs kan aldrig falsifiera tidigare etablerade sanningar. För praktiskt bruk är ett sådant krav ofta oacceptabelt. I själva verket använder vi genomgående antaganden om att vissa utsagor betraktas som sanna, såvida inte vi får ett

explicit skäl att konstatera motsatsen. Förmågan att revidera vår bild av verkligheten när vi erhåller sådan information som falsifierar tidigare antaganden är ett viktigt inslag i intelligent beteende. Understöd för att resonera utifrån villkorliga sanningar, inklusive ett effektivt understöd för att härleda alla följdeffekter när sådana revisioner måste göras, är ett viktigt hjälpmedel vid konstruktion av kunskapssystem.

Temporal logik

I många sammanhang är förmågan att resonera om tidsberoende förhållanden och skeenden av central betydelse. Klassisk logik ger knappast något effektivt stöd i detta avseende, varför försök har gjorts att konstruera *temporal logik*, där en en utsagas sanningsvärde kan bero av vilken tidpunkt den avser. Flera varianter har utvecklats och tillämpats i forskningssammanhang, men det finns i dag knappast någon generellt tillämpbar modell som kan användas reguljärt vid utveckling av kunskapssystem. Man får alltså lösa problemet inom ramen för andra tillgängliga formalismer i de sammanhang där behovet uppträder.

Oskarp (fuzzy) logik

Denna ansats, som är påtagligt relevant för arbete med kunskapssystem, avviker i tämligen hög grad från klassisk logik. Utgångspunkten är att de klassiska sanningsvärdena (entydigt *sant* eller *falskt*) inte är användbara för att modellera informella resonemang av den typ som förekommer i all mänsklig verksamhet. Därför inför man, på liknande sätt som vid flervärd logik, multipla sanningsvärden för utsagor. Av mer avgörande betydelse är dock att man introducerar ett osäkerhetsmått för predikaten *sant* respektive *falskt*, vilket tekniskt utförs genom användning av s.k. *oskarpa mängder (fuzzy sets)*. Ett elements tillhörighet till en viss mängd är alltså behäftad med en viss explicit osäkerhet. Den logik man erhåller på denna grund saknar många av klassiska logikens fördelar när det gäller väldefinierade slutledningsregler och konsistensegenskaper. I gengäld erbjuder den uttrycksmedel för att hantera informella resonemang och osäkerhet i utsagor om verkligheten. Metoder för understöd av s.k. plausibla resonemang i expertsystem utnyttjar delvis metoder baserade på oskarp logik.

4.

Utvecklingsmetodik

Ett kritiskt problem vid konstruktion av ett expertsystem är underströd för processen att explicit formulera kunskapen inom det aktuella området och att bygga upp kunskapsdatabasen. Vanligen arbetar en områdesexpert (*domain expert*) tillsammans med en systemexpert, kallad *kunskapsingenjör*, ("*knowledge engineer*") för att reda ut all den, ofta intuitiva, kunskap som ska praktiskt appliceras för att lösa problem inom tillämpningen. Ett exempel på en besvärlig fråga kan vara värderingen av de osäkerhetsmått som är förknippade med olika regler. I praktiken kan själva intrimningen av regeluppsättning och visshetsfaktorer ofta betyda ganska mycket för kvaliteten hos de resultat som presteras.

Vanligen använder man något verktyg i form av stödjande programvara som ett redskap i processen att bygga en kunskapsbas och realisera ett expertsystem. Vissa programmeringsspråk är särskilt väl lämpade för denna uppgift, t.ex. språken Lisp och Prolog. I praktiken utnyttjar man dessutom vanligen påbyggnader på dessa språk som utvecklingshjälpmedel (*expert systems building tool*) eller helt specialiserade språk, som kanske kan användas direkt av områdesexperten (kallas ibland *systemskal*, *shell*). I kapitel 5 ska vi går in mer i detalj på frågor som rör språk och verktyg för utveckling av kunskapssystem.

Det stöd man har av sina verktyg under utvecklingen av ett expertsystem kan variera ganska mycket. De flesta verktyg är endera mycket generellt applicerbara och lämnar därmed mycket åt kunskapsingenjörens uppfinningsrikedom, eller också är de hårt specialiserade för en viss problemtyp, med risk för att avvikelser från normalfallet blir svåra att klara av. Vissa verktyg stöder kunskapsuppbyggnaden genom att extrahera generella regler ur en samling exempel som användaren (domänexperten) ger. Tyvärr har denna teknik fortfarande mycket begränsad användbarhet. Likaså finns det i praktiken knappast några system som själva utvidgar sin kunskapsbas genom att ackumulera erfarenheter från tidigare utfall. (Dock bedrivs en hel del forskning inom detta område (*learning*).)

4.1 Kunskapsuppbyggnad.

Som tidigare framgått är tillgången till en expert inom det aktuella tillämpningsområdet av avgörande betydelse för möjligheten att framgångsrikt genomföra utvecklingen av ett kunskapsbaserat system. Tyvärr är experten dock vanligen inte förmögen att direkt kunna uttrycka den väsentliga kunskapen inom området på ett sådant sätt att den kan läggas till grund för automatiserad problemlösning i ett expertsystem. Ofta är väsentliga och kritiska delar av kunskapen undermedvetna och svåra att artikulera. Kunskaperna måste också preciseras och omformuleras till en form som kan representeras med de tekniska lösningsmetoder som står till buds.

Principiellt kan vi tänka oss åtminstone följande huvudstrategier vid utveckling av ett kunskapssystem:

1. En kunskapsingenjör intervjuar experten och tillsammans arbetar de fram en förståelse för den aktuella problematiken och hur den kan representeras i systemet.
2. Kunskapsingenjören utvecklar systemet efter specifikationer hämtade ur läroböcker eller andra allmänt tillgängliga källor.
3. Experten formulerar själv, eventuellt stödd av ett utfrågande program, sin kunskap och lagrar in den i systemet.
4. Med hjälp av understödjande programvara extraheras generell kunskap ur exempel eller andra kända data inom området.
5. Systemet bygger själv upp kunskap genom att lära sig av sina erfarenheter.

Av dessa strategier kan vi notera att den första är normalfallet och den som vanligen tillämpas i praktiskt arbete. Nummer två är, för de typer av problem som diskuteras här, vanligen orealistisk, åtminstone för icke-triviala uppgifter. Den femte kan sägas vara klart bortom vad som i dag är möjligt att praktiskt klara och man har knappast ens i forskningssammanhang i högre grad kunnat realisera system som själva lär sig i någon djupare mening. Strategierna tre och fyra har en del förespråkare och har i vissa sammanhang varit framgångsrika. Vi ska ge några exempel på detta i nästa avsnitt.

Utöver experten är alltså kunskapsingenjören en nyckelperson. Av denne krävs, förutom teknisk kompetens och skicklighet, förmåga att kommunicera med experten på ett konstruktivt och inspirerande sätt. Detta kan vara en krävande och tålamodsprövande uppgift, eftersom han måste räkna med att experten:

- har ont om tid och ofta kallas bort till andra viktiga uppgifter;
- inte alltid är engagerad och samarbetsvillig;
- försöker undervisa kunskapsingenjören i den "akademiska" kunskapen inom området och utelämnar mycket av de erfarenhetsmässiga "tumregler" m.m. som är av avgörande betydelse för praktisk problemlösning;

- ogärna eller motvilligt preciserar graden av osäkerhet i regler eller bedömningar.
- inte alltid berättar sanningen, medvetet eller omedvetet;
- motsäger sig själv;
- utelämnar praktiska detaljer och idealiserar sina problemlösningsmetoder;
- samt behöver fortlöpande stöd, feedback och uppmuntran.

Processen att formulera kunskap inom ett område omfattar dels att identifiera de viktiga begreppen och objekten, dels att uttrycka hur objekten reagerar i olika situationer, deras inbördes relationer, olika orsakssamband, strategier för problemlösning osv. Systematiska metoder för att locka fram och artikulera den kunskap som experten besitter saknas i hög grad, men följande tekniker har visat sig vara användbara:

- Observation av experten medan han utför sitt arbete och nedtecknande (eller bandning) av observationerna.
- Experten får ett antal uppgifter att lösa och berättar hela tiden om hur han går till väga.
- Experten intervjuas om olika typiska problem och hur man löser dessa.
- Systematisk utfrågning av experten om viktiga begrepp, samband och problemlösningsmetoder inom området.
- Användning av prototyper baserade på de kunskaper som experten hittills lyckats formulera och analys av de tillkortakommanden som systemet visar upp.

4.2 Programstödd uppbyggnad av kunskapsbas

I föregående avsnitt beskrevs olika metoder för att inhämta och formulera kunskap, *knowledge acquisition*. Man brukar ofta referera till denna uppgift som en flaskhals vid utveckling av kunskapssystem. Det är därmed också naturligt att man sökt efter metoder att elimiera detta problem.

En av de vägar man sökt gå är att utnyttja det faktum att en expert ofta lättare kan ge exempel på beslutsfattande inom sitt område än han kan beskriva de abstrakta principer som förklarar hur han fattar sina beslut. Två vägar att utforma sådana indirekta metoder för syntes av begrepp och regler inom ett område beskrivs i det följande.

4.2.1 Induktivt lärande.

Den normala principen för slutsatsdragning från kända fakta i ett kunskapssystem är *deduktion*, dvs härledning av sanningar som är logiska konsekvenser av redan kända eller bevisade utsagor. En annan, ehuru svagare, slutledningsprincip av intresse är *induktion*, dvs generaliseringar från specifika fall. Denna metod kan användas för att åstadkomma ett visst mått av *inlärning*, genom att återkommande mönster antas vara ett uttryck för en bakomliggande regelbundenhet. Givetvis kan man inte garantera att de slutsatser som dras på detta sätt är säkra och inte bara ett slumpmässigt sammanträffande som inte kan extrapoleras till nya fall.

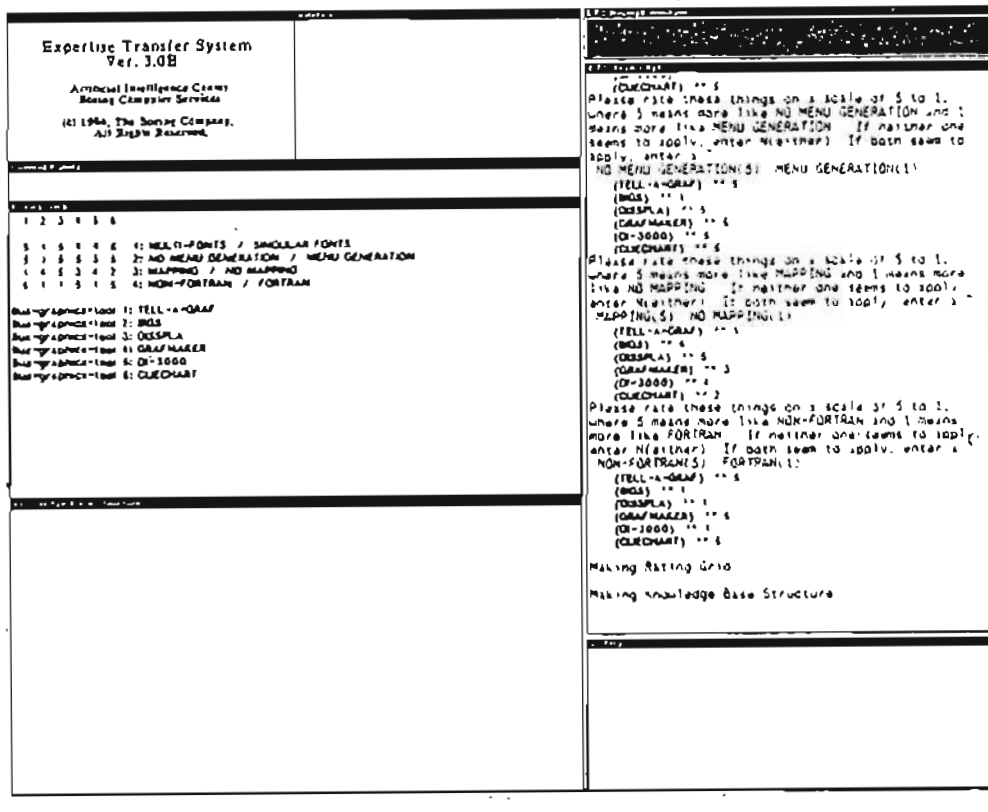
Metoden med induktiv syntes av regler som en bas för beslutsstödsystem som understödjer val mellan en uppsättning alternativ har kommit till praktisk användning i färdiga verktyg för utveckling av kunskapssystem. Exempel på detta ges i avsnitt 5.3.1. Sådana verktyg har fördelen att de automatiserar vissa delar av kunskapsinhämtningen. Praktiskt sker detta genom att experten ombeds redovisa ett representativt urval av exempel på beslutsfattande inom det aktuella området. Stödsystemet kontrollerar motsägelsefrihet och fullständighet (i viss mening) samt producerar en redigerad mängd regler som sammanfattar exempelsamlingens innehåll. Någon induktion av mer generella principer sker dock knappast inom ramen för de hjälpmedel som finns tillgängliga i dag.

Fördelarna med induktionsprincipen i sammanhanget är att den erbjuder ett snabbt och säkert sätt att ta fram vissa typer av kunskap. Svagheterna ligger främst i den begränsade tillämpbarheten, beroendet av att exempelsamlingen är tillräckligt allsidig och heltäckande, samt i svårigheten att värdera tillförlitligheten i den inducerade beslutsmodellen.

4.2.2 Datorstödd intervju

En annan teknik av speciellt intresse i sammanhaget är baserad på en metod för genomförande av intervjuer med syfte att klarlägga hur en person bildar begrepp och teorier för att bringa reda i en kaotisk omvärld. Systemet *ETS, Expertise Transfer System* [BOO85], utvecklat av John Boose med kollegor vid Boeing i Seattle har inspirerats av George Kelly's *personal construct theory* [KEL55].

Denna teknik bygger bl.a. på att experten först ska ange ett antal beslutsmöjligheter som är relevanta i det tänkta expertsystemet, t.ex. rörande ekonomiska beslut, diagnoser vid felsökning etc. Därefter frågar systemet ut experten om olika karakterististiska egenskaper som kan användas för att särskilja mellan alternativen. Detta går till så att man bildar grupper om tre element (beslutsalternativ) och ber experten ange en egenskap som ett element, men inte de övriga två har. Man får också ange vad som är motsatsen till denna egenskap och underhand bildas en tvådimensionell matris där beslutselementen definierar den ena dimensionen och de karakteriserande



Figur 4.1. *Uppbyggnad av bedömningsmatris i systemet ETS*

egenskaperna den andra, varvid varje egenskap ges ett siffervärde på skalan mellan de ytterligheter som anger ändpunkterna för respektive egenskap.

Utgående från denna matris appliceras sedan ett antal analysmetoder för att härleda olika samband i form av regler. I en fortsatt intervju förfinas den insamlade informationen vidare. Exempelvis kan man vilja formulera hur olika begrepp är relaterade till varandra i termer av generaliseringar respektive specialiseringar. Hela denna process syftar till att underlätta för experten att formulera sina kunskaper, eliminera eventuella motsägelser och ofullständigheter och samtidigt få en ökad inblick i de olika samband som råder.

Ur den på så sätt formulerade kunskapen extraheras så ett antal produktionsregler som uttrycker hur olika beslut indikeras av observerade förutsättningar eller önskade resultat. Denna regelsamling kan testas direkt inom ramen för ETS, varefter möjlighet finns att automatiskt producera en representation av regelsystemet för något av de vanligaste expertsystemsskalen, t.ex. EMYCIN, OPS, LOOPS och Prolog. Inom Boeing har hittills c:a 250 regelbaserade system inom en rad olika tillämpningsområden genererats på detta sätt, varav dock de flesta prototyper som inte satts i reguljär drift.

4.2.3 Synliggörande av kunskapen

I många sammanhang har det omvittnats den stora betydelse som genomförandet av ett expertsystemsprojekt kan ha som katalysator för formulering av expertkunskap inom ett tillämpningsområde. Ibland visar det sig att en framtagen regelsamling och den modell av applikationen som artikulerats har ett större värde än själva det implementerade systemet. Man kan därmed tänka sig att tekniken med expertsystem vidareutvecklas också som ett redskap för att formulera modeller av expertkunskap inom olika områden, oberoende av eventuella önskemål om datoriserade expertsystem.

Exempel på sådana tekniker och hjälpmedel är de nyss beskrivna induktionssystemen och intervjusystem baserade på *personal construct theory*. När regelsamlingen är framtagen är det inte ovanligt att expertens reaktion blir "detta borde jag ha insett från början". I detta fall har tekniken fungerat som ett hjälpmedel för att lyfta fram en kunskap som man tidigare inte var medveten om.

4.3 Relation till traditionell systemutveckling

Om vi jämför de tekniker och problemställningar som aktualiseras vid utveckling av kunskapsbaserade system med motsvarande i samband med traditionell systemutveckling kan man konstatera en hel rad beröringspunkter. All vår kunskap om systemanalys, informationsmodellering och databasutformning är relevant också vid arbete med kunskapssystem. Det som skiljer är bl.a. fokuseringen på individens kunskaper i högre grad än vid konventionella datasystem, där man mer söker modellera organisationens eller verksamhetens informationsbehandling.

På senare tid har man vid systemutveckling kommit att i allt högre grad betona behovet av experimentella ansatser, exempelvis i form av försök med tidiga prototyper (*rapid prototyping*) som kan demonstreras för användare. Detta kan ses som ett uttryck för en metodik baserad på *iterativ systemutveckling*, där implementeringar systematiskt används som ett hjälpmedel för att förstå exakt vilka krav som bör ställas på ett system och hur motsvarande lösning bäst kan realiseras. Kunskapssystemen förutsätter i allra högsta grad en sådan metodik, eftersom redan kunskapsformuleringen i praktiken kräver återkoppling från experiment med en fungerande, om än rudimentär, kunskapsbas. Ett kunskapssystem i drift kan också förväntas vara föremål för kontinuerlig utveckling och underhåll av dess kunskapsbas.

Mot denna bakgrund är det kanske inte heller överraskande att det område, där kunskapsbaserade tekniker först väntas komma in som ett stöd i en systemutvecklingsprocess av traditionellt slag, är utformning av kravspecifikation. Att rätt förstå och formulera kraven på ett givet system är ofta mycket betydelsefullt. Historien visar ett antal skräckexempel på projekt som lagts ner med mycket stora kostnader som följd på grund av att kraven ställts fel från början. Även i system som fullföljts kan åtgärdande av fel som

introducerats i specifikationsfasen visa sig bli mycket kostsamma, varför effektiva hjälpmedel som höjer kvaliteten på kravspecifikationerna är mycket värdefulla.

5.

Utvecklingshjälpmedel

Inom området kunskapssystem arbetar man vanligen med icke-konventionella programmeringsspråk och med speciella programvaruredskap, på motsvarande sätt som vid applikationsutveckling med s.k. fjärde generationens språk. Metodiken för systemutveckling är ofta nära knuten till en viss klass av redskap, vilket ytterligare försvårar uppgiften att välja strategi vid införande av tekniken i en given organisation.

Man kan med anspråk på trovärdighet idag knappast göra en objektiv utvärdering av vilken teknik eller vilker redskap som är "bäst". En bärande princip är att det är den i systemet representerade *kunskapen* inom ett applikationsområde, som avgör kvaliteten på ett systems prestationsförmåga, snarare än någon inbyggd generell "intelligens". Därmed är det också avgörande för redskapets värde hur väl det understöder formulering av den relevanta kunskapen, med hänsyn till områdets och uppgiftens speciella karaktär. Det blir då intressant att studera redskapens egenskaper relativt olika typiska problemställningar som kan uppträda. Exempel på områden där tekniken är tillämpbar är beslutsstödssystem för svårformaliserade uppgifter, såsom felsökning och diagnos, ekonomisk rådgivning, planering av olika slag, analysproblem, osv.

Allmänt sett kan vi urskilja åtminstone fyra typiska nivåer i implementeringen av ett kunskapssystem, nämligen

1. Maskinvaran, som endera är en standarddator eller en speciellt konstruerad processor (eventuellt mikroprogrammerad). Valet mellan olika datorer är i dag väsentligen en kostnads- och effektivitetsfråga. Principiellt krävs inte någon viss processor med speciella egenskaper.
2. Generellt programmeringsspråk, som används i systemet och som endera kan vara ett traditionellt kompilerande språk, typ Fortran, PL/1, Pascal eller C, eller ett speciellt s.k. AI-språk, vanligen Lisp eller Prolog. AI-språken erbjuder avgörande fördelar ur många synpunkter, men det är alltid i princip möjligt att i botten ha ett konventionellt språk. (Trivialt sant, eftersom exempelvis Lisp och Prolog i sin tur kan implementeras i Fortran, C, osv.) Det förekommer också att färdigutvecklade kunskapssystem omimplementeras i ett konventionellt språk.
3. Redskap eller systemskal, dvs ett programpaket med ett specialiserat

språk och hjälpmedel för att konstruera kunskapssystem. Sådana redskap kan vara utbyggnader av det underliggande generella programmeringsspråket eller också helt isolera användaren från behovet att behärska exempelvis Lisp, även om detta språk använts för att implementera det aktuella redskapet.

4. Det egentliga kunskapssystemet, med en kunskapsbas som definierar systemets förmåga att lösa problem och ge konsultativt stöd inom ett givet område.

På många håll står man i dag inför problematiken att välja redskap för projekt inom området expertsystem. Det torde ha framgått av ovanstående att denna uppgift vanligen är långt ifrån trivial. Uppgiften förenklas inte heller av de programvaror som kan vara aktuella ligger i prislägen från under tusen kronor (t.ex. vissa Prologsystem på PC) till femhundratusen kronor (t.ex. ART, som dessutom kanske kräver investering i hårdvara till en ännu högre kostnad). Uppenbarligen vore det angeläget med en vägledning för detta val, som kunde motivera vilken typ av redskap och vilket prisläge som är lämpligt och realistiskt i ett givet sammanhang. Tyvärr finns inga enkla svar på denna fråga. Framställningen i det följande är ett försök att åtminstone ge någon vägledning för den som vill bilda sig en uppfattning om vad som skiljer olika ansatser och motsvarande redskap åt.

5.1 Maskinvara

Utmärkande för de tekniker som diskuteras i denna rapport är tonvikten på att flytta rutinarbete till datorsystemet och frigöra människans resurser för egentlig problemlösning och beslutsfattande. Detta har resulterat i programvarutekniker som ställer stora krav på prestanda hos de maskinsystem som utnyttjas. Delvis är också dessa krav annorlunda än de som ställts i traditionell databehandling, vilket bl.a. lett fram till utvecklingen av specialiserade maskinarkitekturer, företrädesvis för understöd av språket Lisp (Lispmaskiner).

AI-forskningen initierade tidigt behovet av tidsdelade datorer, dvs möjligheten för varje enskild användare att arbeta i direkt dialog med datorn. Om många samtidiga användare delar på datorresursen på detta sätt, får man tyvärr ofta oacceptabelt långa svarstider. Detta tillsammans med önskemål om ett användargränssnitt baserat på högupplösande grafik har lett fram till utvecklingen av kraftfulla arbetsstationer som det viktigaste redskapet för arbete med AI och expertsystem. En sådan arbetsstation har typiskt prestanda i klass med en medelstor dator (t.ex. typ VAX), ett primärminne i storleksordningen 2-8 Mbyte, sekundärminne i storleksordningen från 40-200 Mbyte, samt kommunikation med andra arbetsstationer och fillagring via ett höghastighetsnät.

De dominerande leverantörerna av Lispmaskiner hittills har inte varit de traditionella dominanterna inom dataområdet, utan Xerox och de nystartade spinoff-företagen från MIT, Symbolics och LMI. AI arbetsstationer erbjuds

även från DEC, Texas Instruments, Sperry, SUN, Apollo, Tectronix, HP, m.fl. Utbudet av enklare programvaror för konventionella persondatorer ökar snabbt. På stordatorsidan kan noteras att de i allmänhet erbjuder en mindre lämplig utvecklingsmiljö, men också att färdiga tillämpningar i många fall kommer att behöva integreras med databaser och andra befintliga programsystem. Exempelvis marknadsför IBM sedan en tid tillbaka dock ett utvecklingsverktyg som går i stordatormiljö under VM (och som nyligen också annonserats för MVS).

Prisläget för arbetsstationer av den typ som beskrivs ovan varierar från c:a etthundra tusen kr för en arbetsstation som åtminstone kan användas för färdigutvecklade system, till en dryg miljon kr för den kraftfullaste utvecklingsmiljön. Utvecklingen går i riktning mot betydligt ökad kapacitet och prestanda, samtidigt som nerskalade lågprisversioner lanseras bl.a. som leveranssystem för färdiga tillämpningar. Detta leder också till att gränsen mot traditionella smådatorer snart kommer att suddas ut.

5.2 Generella språk

De programmeringsspråk som är ojämförligt mest använda för utveckling av kunskapsbaserade system är Lisp och under senare tid Prolog. Båda dessa språk har rötter förhållandevis långt bak i tiden. Lisp definierades redan runt 1960 och kom i allmänt bruk inom AI-forskningen under sextiotalet. Prolog är av europeiskt ursprung och utvecklades under det tidiga 70-talet, men baseras på den s.k. resolutionsmetoden som utgjorde en central teknik inom AI redan under sextiotalets senare del.

5.2.1 Lisp

Som programmeringsspråk är Lisp i vissa avseenden att betrakta som jämförbart med andra generella programspråk, dvs ett procedurellt språk som opererar på datastrukturer lagrade i datorns minne. Emellertid har Lisp, eller snarare programmeringssystem baserade på detta språk, också egenskaper som har gjort det till helt dominerande när man ser till fungerande system baserade på AI-teknik, exempelvis följande:

- Lisp är ett interaktivt och inkrementellt språk (liksom exempelvis APL), dvs program kan byggas upp och exekveras steg för steg i direkt dialog. Språket kan interpreteras, dvs tolkas direkt uttryck för uttryck, eller kompileras, dvs först översättas till en form som kan exekveras mer effektivt.
- De fundamentala datatyperna i språket är lämpade för att representera symbolisk snarare än numerisk information. Man kan med fördel uppfatta Lisp som ett (databas)system för att skapa och manipulera representationer av olika slags objekt, deras egenskaper och relationer.
- Det finns ingen inbyggd åtskillnad mellan program och data i språket. Alla strukturer (program och data) har en väldefinerad extern

representation (list-struktur) som direkt kan användas för inmatning och utskrift. En vanlig missuppfattning är att Lisp har en besvärlig syntax, men sanningen är snarare att syntaxen är extremt enkel vilket dock gör program svårlästa för den som är ovan.

Nedanstående exempel visar hur en funktionsdefinition i LISP kan se ut:

```
(DE APPEND (L1 L2)
    (COND ((NULL L1) L2)
          ((ATOM L1) (CONS L1 L2))
          (T (CONS (CAR L1) (APPEND (CDR L1) L2)))))
```

Denna definition av funktionen APPEND med argumenten L1 och L2 illustrerar också hur den interna strukturen ser ut, eftersom varje parentespar markerar en i datorns minne lagrad, länkad lista av element, vilka i sin tur kan utgöras av nya listor eller atomära symboler.

- Det är naturligt och enkelt att använda Lisp som implementeringsspråk för språk på en högre eller mer applikations-orienterad nivå. Detta är också det typiska sättet att använda Lisp i olika tillämpningar och många redskapssystem är baserade på Lisp.
- Lisp-system (exempelvis Interlisp) har i allmänhet en mycket utvecklad och avancerad programmeringsmiljö. Språkets interaktiva, interpreterande natur gör det också lämpligt som generellt styrspråk på operativsystemsnivå, vilket bl.a. utnyttjas i arbetstationer baserade på Lisp (t.ex. Xerox, Symbolics, LMI, Texas Instruments).

5.2.2 Prolog

Prolog skiljer sig på ett helt annat sätt än Lisp från de traditionella programmeringsspråken. Språket bygger på första ordningens predikatlogik (något begränsad) och kan ur vissa synpunkter sägas realisera en programmeringsteknik där ett program samtidigt utgör en specifikation av problemet uttryckt i formell logik. Emellertid kan Prolog också uppfattas som ett generellt programmeringsspråk och vid användning i realistiska tillämpningar krävs i praktiken insikter i och hänsynstagande till hur språkets interpretator arbetar. En stor fördel är dock, med föregående reservation, att programstyrningen i hög grad sköts av systemet själv (*"back-tracking"*), vilket bl.a. innebär att man direkt i språket har tillgång till en *inferens-mekanism* för evaluering av regler av den typ som är karakteristisk för många realiserade expertsystem.

Exempel på definition av ett litet "program" i Prolog:

```
append([],L,L).
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

Dessa två relationer anger hur en lista (en ordnad sekvens av element) kan bildas genom sammanfogning av två andra. Relationerna kan uppfattas som regler, där högerledet anger en förutsättning och vänsterledet en utsaga som är sann om givna argument satisfierar högerledet.

(Den första regeln har ett tomt högerled och uttrycker väsentligen att en lista inte förändras av att sammanfogas med en tom lista.)

Språket kan implementeras även på mycket små datorer och är förhållandevis lätt att lära. Nackdelar är att programmeringsmiljön inte är så utvecklad som för Lisp och att de prestanda som erhålls på dagens datorer ofta inte är acceptabla. Förhoppningsvis kan prestandabilden bli en annan när datorer baserade på nästa generations arkitekturer blir tillgängliga. Dock bör man komma ihåg att även när man får datorer som baseras på parallell bearbetning kommer lösningstiderna för exponentiella problem fortfarande att vara lineära i antalet processorer, varför lösning av typiska AI-problem fortfarande kommer att vara en metod- snarare än en hårdvaru-fråga.

5.2.3 Användning av konventionella språk

Traditionellt har de redskap som använts för utveckling av expertsystem baserats på specialiserade språk, med få undantag Lisp. Tendensen för närvarande är dock att man i viss grad omimplementerar redskapen, eller åtminstone de delar som behövs i reguljär användning av ett utvecklat system, i något konventionellt språk, t.ex. C eller Pascal. Detta ger uppenbara fördelar ur flyttbarhets- och marknadssynpunkt, men kan leda till svårigheter när man som användare tvingas gå utanför de funktioner som redskapet kan erbjuda. Det kan också visa sig att man tvingas omimplementera alla det ursprungligen underliggande språkets egenskaper (såsom hantering av pekarstrukturer, dynamisk minneshantering, etc) i den nya miljön, med uppenbara risker för sänkt effektivitet jämfört med de mikro- eller assembly-kodade rutiner som finns i ett Lispsystem.

5.3 Verktyg för kunskapssystem

Begreppet expertsystem är suggestivt och har fått stor genomslagskraft. Den åtföljande breda användningen har också lett till en viss förvirring när det gäller avgränsning av området. Bl.a. har en ganska varierad samling programvara förts ut på marknaden under rubriken verktyg för utveckling av expertsystem. Ibland har till och med beteckningen expertsystem kommit att betyda ett program som utvecklats med något av de språk eller verktyg som vanligen används i sammanhanget.

Tyvärr är det en mycket svår uppgift att ge en mer precis karakteristik av vad som utmärker ett redskap för utveckling av denna typ av programvara. Vi ska i det följande försöka att på olika sätt klassificera och beskriva de tekniker och verktyg som är aktuella vid utveckling av kunskapsbaserade expertsystem. En distinktion, som kan göras även om gränserna ibland är svåra att dra, är mellan:

- *"små", "enkla" redskap*, avsedda att även kunna användas direkt av en områdesexpert. Dessa verktyg är ofta PC-baserade och utmärks av att de företrädesvis är starkt specialiserade för en viss problemtyp

och/eller en viss lösningsmetod. Begränsningar ligger naturligt i svårigheter att hantera svårstrukturerade problemställningar, problem med att utvidga en lösning utanför de ramar som systemet tillhandahåller.

- *"stora", komplexa redskap*, avsedda i första hand som verktyg för professionella kunskapsingenjörer som arbetar tillsammans med områdesexperter vid konstruktion av system. Dessa redskap ger ofta ett mångsidigt understöd vid val av lämpliga representations- och resonemangsmetoder, men ställer samtidigt större krav på förmåga att rätt utnyttja dessa möjligheter.

Ibland används begreppen verktyg respektive systemskal (eng: *tool*) resp. *shell* synonymt för programvara avsedd att fungera som hjälpmedel för utveckling av expertsystem. Vanligen används dock begreppet systemskal om en mer begränsad klass av hjälpmedel, nämligen sådana som implementerar en viss specifik resonemangsmetod och därmed förutsätter att områdeskunskapen ska representeras i en fast form som passar denna metod.

I de följande avsnitten ska vi diskutera en delvis annorlunda strukturerad indelning i olika klasser av redskap. Vi tar där fasta mer på den typ av understöd som systemet tillhandahåller. De grupper som vi tar upp är långt ifrån uttömmande och alternativa indelningar är givetvis också möjliga.

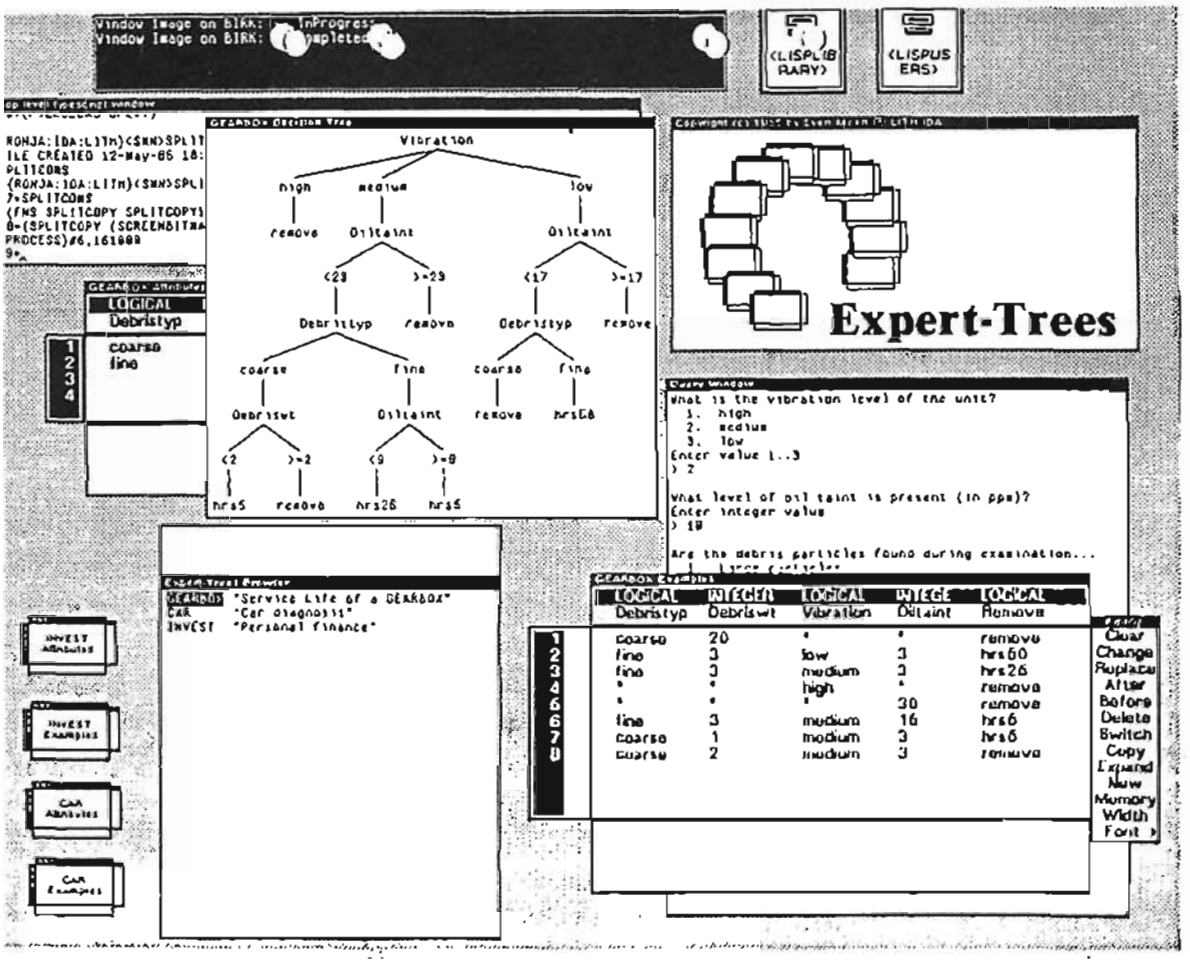
5.3.1 Induktionssystem.

Denna klass av verktyg utmärks bl.a. av den starka tonvikten på stöd för processen att formulera kunskapen inom ett givet område. Principen för detta är att experten uppmanas formulera typiska exempel för beslutsfattande och att systemet sedan understödjer härledning (induktion) av generella regler ur en sådan exempelsamling. Användbarheten av denna ansats motiveras av den i försök påvisade svårigheten för en expert att direkt formulera exakta och uttömmande regler för en viss uppgift. I ett berömt exempel [MIC80] demonstrerades hur en expert på att diagnostisera sjukdomar hos soyabönor trots idogt arbete med ett utveckla ett regelsystem misslyckades med att nå upp till precisionen hos ett beslutsträd som inducerats från exempel som han formulerat.

Induktionssystemen kan användas för klassifikationsproblem (eller uppgifter som kan formuleras i motsvarande termer) där man har ett antal oberoende variabler som kan anta ett ändligt antal distinkta värden och en beroende beslutsvariabel, vars värde sökes. De som utgångspunkt givna exemplen används för att ange vilket beslut som indikeras av en viss kombination av variabelvärden. Systemet bygger upp ett (eller flera sammanlänkade) beslutsträd från exemplen och kontrollerar fullständighet och motsägelsefrihet. Ofta används någon algoritm (t.ex. den s.k. ID3-algoritmen) för att under vissa (vanligen förenklade) antaganden optimera beslutsträdet ur informationsteoretisk synpunkt.

Utvecklingsverktyg av typ Expert-Ease, Extran-7, RuleMaster och TIMM baseras på att man först definierar en beslutsvariabel och dess värdeförråd samt identifierar de oberoende variabler som kan tänkas påverka det sökta beslutet. Därefter bygger man upp en exempelsamling, som bör vara allsidig och någorlunda heltäckande. Systemen genererar sedan regler, vanligen i form av ett beslutsträd. Denna struktur kan därefter användas endera som ett flödesschema för beslutsfattande eller som en bas för ett program som avger en rekommendation efter att ha frågat ut användaren om den aktuella situationen.

Nedanstående illustration [MOE84] visar exempel på de olika faserna i en sådan process. I de fönster som visas på skärmen kan man se definition av värdeförråd för de olika variablerna, exempelsamling (ett per rad, där * som variabelvärde anger ett irrelevant värde), genererat beslutsträd, exempel på konsultation, m.m.



Figur 5.1. *Exempel på induktionssystem.*

Induktionssystemens svaghet är naturligtvis den mycket begränsade klassen av tillämpningar, där det förutsätts att alla beslut entydigt bestäms av en kombination av värden för en given uppsättning variabler (beslutsparametrar). Systemet TIMM ger dock möjlighet att markera styrkan hos ett exempel genom användning av "tillförlitlighetsmått". Detta system arbetar också med

en modell för att avgöra "närheten" av en given situation till de olika situationer och med dem förknippade beslut som finns registrerade i systemet. Därmed undviker man att enstaka missbedömningar av någon beslutsparameter leder fram till ett helt felaktigt beslut (eller inget beslut alls). Induktionssystemens styrka ligger givetvis i det effektiva stöd som ges för formulering av kunskap i form av regler inom områden som är begreppsmässigt välstrukturerade.

5.3.2 Regelbaserade system

Genombrottet för praktiska tillämpningar av kunskapsbaserade expertsystem kom med introduktionen av regelbaserade formalismer för att uttrycka kunskapsfragment, som sedan utnyttjas av ett program som härleder slutsatser med hjälp av någon generell teknik för slutsatsdragning. I själva verket har expertsystem i många sammanhang kommit att uppfattas som synonymt med regelbaserade system, något som dock inte kan anses vara motiverat i dagens läge.

Tekniken att använda s.k. produktionsregler går ytterst tillbaka på metoder utvecklade inom den kognitiva psykologin med syftet att modellera vissa aspekter av mänskligt beteende i termer av simulus-respons. Den allmänna strukturen hos en sådan regel är att den består av en villkorsdel (OM-ledet) som används för att identifiera en viss situation eller om givna förutsättningar är uppfyllda, och en slutsatsdel (SÅ-ledet) som anger en slutsats som följer eller en handling som ska utföras om villkorsledet är uppfyllt. Ofta är regeln förknippad med ett *tillförlitlighetsmått* som anger med vilken grad av tilltro som slutsatsen följer av förutsättningarna.

RULE401

C is a car.
[This rule is tried in order to find out about whether the cause
of the problem with the car C is voltage.regulator.bad]

If: 1) The pattern observed by attaching an oscilloscope
to the charging circuit of the car C is
fluctuating.arches, and
2) the alternator of the car C responds properly to
the different loads,

Then: There is strongly suggestive evidence (0.9) that the
cause of the problem with car C is voltage.regulator.bad

Figur 5.2. *Exempel på en regel.*

De första redskapen som framgångsrikt fick en bred användning var baserade på denna teknik. Systemet EMYCIN, utvecklat vid Stanford, har fått en bred spridning och en efterföljd i form av kommersiella produkter, t.ex S1 och Personal Consultant. Denna familj av regelbaserade redskap har ett relativt stereotypt regelspråk, men en stor styrka i förmågan att understödja resonemang under osäkerhet och automatiskt genererade förklaringar av

systemets resonemang. Den helt dominerande resonemangsmetoden i denna familj av system är målstyrd (*backward chaining*) slutsatsdragning, dvs systemet söker verifiera olika möjliga hypoteser genom att söka bakåt mot kända fakta som kan styrka hypotesen.

Exempel på andra typer av regelbaserade verktyg är OPS5 (med varianter), som understödjer en mer flexibel, effektivt implementerad resonemangsmodell (men ej osäker kunskap och förklaringar) respektive Prolog, som snarast är att betrakta som ett generellt programmeringsspråk där program skrivs i form av en samling regler. OPS är ett typiskt exempel på ett produktionssystem med framåtriktad slutsatsdragning. Alla kända fakta är tillgängliga i ett arbetsminne. I varje resonemangssteg appliceras någon regel vars villkorsdel satisfieras av fakta i arbetsminnet, varvid innehållet i arbetsminnet kan förändras enligt instruktioner i regelns slutsatsdel. Denna process fortsätter tills det aktuella problemet är löst eller tills det inte längre finns några regler som kan tillämpas.

Den enhetliga representationsmetoden i form av regler, som är en klar styrka när det gäller att tillhandahålla standardiserade förklaringar m.m., utgör samtidigt en svaghet vid lösning av komplexa problem med behov av mer sammansatta begrepp och av strukturerade resonemangsmetoder. Redskap i EMYCIN-familjen är främst tillämpbara på diagnosproblem (eller s.k. strukturerade val), där man har ett statiskt problem som ska klassificeras i termer av ett antal fördefinierade kategorier. Detta betyder också att man inte utan vidare kan hantera problem som har en tidsdynamisk aspekt. Språk av typen OPS och Prolog har inte på samma sätt sådana begränsningar, men kräver i gengäld mer kvalificerade kunskaper hos kunskapsingenjören. En genomgående svårighet i detta sammanhang är att när man bygger system för realistiska, någorlunda komplexa, tillämpningar blir regelsamlingarna oöverskådliga, bl.a. beroende på att ingen skillnad görs mellan regler som uttrycker den egentliga områdeskunskapen, respektive sådana som definierar "sunt förnuft" kunskap, begreppsdefinitioner, explicit programstyrning för att förbättra effektiviteten, osv.

5.3.3 Objekt-orienterade system

Denna klass av redskap tar fasta på behovet att kunna beskriva enskilda begrepp (objekt) och deras inbördes relationer på ett mera allsidigt och nyanserat sätt. Därmed bygger man också upp systemen runt primitiver (*frames*) för att representera objekt, deras egenskaper och beteenden i olika situationer. Ofta används regler för att beskriva dessa beteenden, men till skillnad från de regel-baserade systemen struktureras kunskapen efter objekten snarare än i form av en generellt giltig samling regler.

I systemet hanteras klasser bestående av alla individer av en given typ. En klass definierar därmed ett begrepp eller en objekttyp. Begrepp kan sedan struktureras i en hierarki av successiva generaliseringar, där exempelvis begreppet *bil* är en specialisering av begreppet *fordon*. Systemets uppgift är

bl.a. att vid behov härleda uppgifter om enskilda objekt eller klasser av objekt genom "ärvning" av egenskaper i denna begreppshierarki. Objektegenskaper kan vara lagrade som explicita värden eller som procedurer som aktiveras exempelvis när ett värde efterfrågas eller förändras.

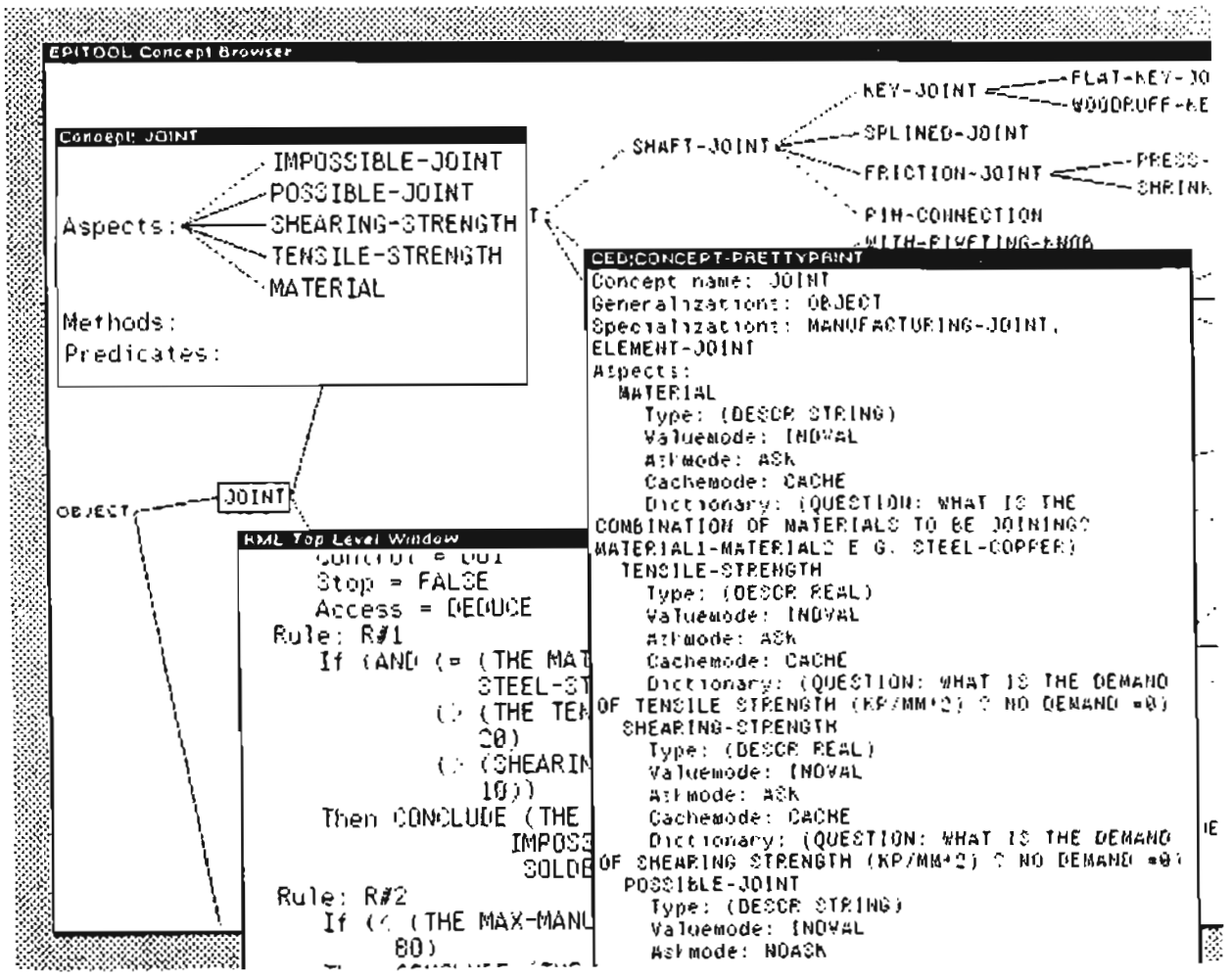
Objekt-orienterade metoder har använts i en rad olika sammanhang. Anknytning finns även till databasvärlden och till användningen av abstrakta datatyper i avancerade programmeringsspråk. Exempelvis är Simula och Smalltalk exempel på generella programmeringsspråk med stark betoning på en objekt-orienterad problemlösning. I samband med praktiska redskap för expertsystem har denna ansats i allmänhet inkorporerats i s.k. hybrida utvecklingsmiljöer.

5.3.4 Hybrida utvecklingsmiljöer.

Utmärkande för de hybrida systemen är understöd för objekt-orienterad kunskapsrepresentation, inkluderande begreppshierarkier med ärvning av egenskaper, samt alternativa metoder för utförande av regelbaserade resonemang. Genomgående innehåller dessa system också en mer utvecklad programmeringsmiljö samtidigt som man förutsätter att användaren (kunskapsingenjören) har kvalificerade kunskaper inom området. För utveckling av nya kunskapssystem krävs vanligen en kraftfull arbetsstation.

De helt dominerande systemen i denna kategori är KEE, som får sägas vara marknadsledande f.n., ART, LOOPS (Common Loops) och Knowledge Craft. Nyligen har också ett svenskutvecklat system, EPITOOL från Epitec AB, introducerats på marknaden. EPITOOL hanterar en begreppshierarki med multipel ärvning. Regelmängder kan knytas till objekten och aktiveras för slutsatsdragning endera när egenskaper förändras eller när de efterfrågas. En intressant egenskap är att systemet kan hantera även egenskapsvärden som är obestämda (okända) eller delvis bestämda, dvs det är känt att det aktuella värdet tillhör en på lämpligt sätt specificerad delmängd av variablens tillåtna värdeförråd. Dessa beskrivningar kan utnyttjas vid resonemang och beräkningar, varvid beskrivningen ibland kan vara tillräckligt precis för att medge en viss slutsats medan i andra fall successiva operationer kan leda fram till ett bestämt värde för den aktuella egenskapen.

En tendens i sammanhanget är att man i dessa system i allt högre grad gör sig oberoende av det underliggande språket (i praktiken Lisp) och på så sätt gör det möjligt både att implementera systemen direkt i språk av typ C och även att uppnå hög effektivitet, exempelvis vid övergång till reguljär driftsmiljö, genom analys av kunskapsbasen för en tillämpning. De hybrida miljöerna tar bl.a. fasta på den genomgående erfarenheten vid utveckling av kunskapsbaserade expertsystem, att man ofta måste räkna med förkasta tidiga prototyper och kanske välja påtagligt reviderade lösningsmetoder allt eftersom man får en fördjupad insikt i det aktuella problemet.



Figur 5.3. *Skärmutseende vid arbete i hybrid utvecklingsmiljö.*

5.3.5 Jämförande analys.

Några böcker och tidskrifter inom området har publicerat jämförande översikter av olika redskap för utveckling av expertsystem, t.ex. [HAR85]. De vanligaste beskrivningsmetoderna är endera att beskriva karakteristiska egenskaper för varje redskap för sig eller också att identfiera ett antal funktioner/faciliteter som kan finnas respektive saknas och i tabellform redovisa dessa aspekter, eventuellt med användning av en viktning som anger "kvaliteten" på den aktuella funktionen. Sådana tabellsammanställningar tenderar att vara av begränsat värde, och i vissa fall missvisande, eftersom en summering av enkla egenskaper långt ifrån ger någon rättvisande bild av ett systems användbarhet. Den förra metoden har nackdelen att jämförelser försvåras eller omöjliggörs av att olika system beskrivs ur olika synvinklar och delvis på helt olika sätt.

Beroende på en given tillämpnings krav kommer givetvis betydelsen av olika egenskaper hos ett redskap att variera, varför det är svårt att göra någon objektiv rangordning av deras användbarhet. Nedan anges dock ett antal exempel på aspekter, som kan vara intressanta vid utvärdering och val av redskap för utveckling av kunskapssystem.

- Tekniker för organisation av kunskapsbas, grundläggande fakta, regler, begreppshierarkier, m.m.
- Metoder för slutsatsdragning som understöds (*forward, backward* osv).
- Understöd för resonemang under osäkerhet.
- Förklaringsmekanismer samt hjälp- och undervisnings-funktioner.
- Funktioner för felsökning, editering och underhåll av kunskapsbas m.m.
- Användargränssnitt för kunskapsingenjören, samt hjälpmedel och tekniker för utformning av interaktion med slutanvändaren.
- Teknisk miljö i termer av datorer och underliggande programmeringsspråk, flyttbarhet, m.m.
- Gränssnitt mot andra språk och system.
- Implicit eller explicit metodstöd samt eventuella begränsningar i systemets praktiska tillämpbarhet för vissa problemtyper.
- Tekniska begränsningar, avseende prestanda, kunskapsvolymer, etc.
- Stöd från leverantör och tillgänglighet till utbildning och konsulttjänster.

Ett utkast till en klassificeringsstruktur har utarbetats vid institutionen för datavetenskap, IDA, vid Tekniska högskolan i Linköping [HAG85]. Denna struktur förtecknar ett antal aspekter som kan vara relevanta vid beskrivningen av ett specifikt redskap. Tanken är att enhetligt uppställda beskrivningar över olika redskap ska påtagligt underlätta uppgiften att göra jämförelser och avgörande om respektive redskaps lämpligt för en given problemställning.

Denna korta översikt leder fram till det något pessimistiska konstaterandet att man i dag knappast kan formulera några generella kriterier för ett problemoberoende val av utvecklingsverktyg för expertsystem. Viktigt är att skaffa sig en klar bild av de problemklasser man avser att angripa och av de krav som en reguljär tillämpning i drift ställer. Många enkla redskap har fördelen att mycket snabbt kunna resultera i små demonstrationssystem, men saknar förmåga att växa med uppgiften. Det är också viktigt att vara klar över att redskapen i sig inte löser några problem. Av helt avgörande betydelse är förmågan att förstå den centrala kunskapen inom en given applikation, att formulera den explicit och att på ett effektivt sätt kunna representera den i systemet, som en bas för aktiv problemlösning. Redskapens uppgift är därför i första hand att på ett så effektivt sätt som möjligt understödja denna process.

5.3.6 Standardisering m.m.

I detta sammanhang är det naturligt att efterlysa någon form av standardisering, formell eller *de facto*. När det gäller språk noterade vi att Common Lisp har alla förutsättningar att bli basen för en sådan standard. När det gäller programmeringsmiljön och redskapssystem är situationen mer öppen.

Man kan notera att systemet KEE, som sålts i mer än 500 exemplar hittills, har en klar marknadsmässig dominans i USA. Det måste dock fortfarande anses vara en öppen fråga om någon egentlig standardisering kan förväntas inom överskådlig tid. Därtill varierar troligen krav och behov alltför mycket från område till område. Kraftfulla satsningar från företag av typen IBM eller DEC kan givetvis påverka bilden.

När det gäller prisläget för redskapssystem, upplevs ofta priset för system i den större klassen som anmärkningsvärt högt. (Priser över 50.000 USD för programvaran till en arbetsstation är inte ovanliga.) Man bör dock komma ihåg att citerade priser avser ett enstaka utvecklingssystem. För ytterligare lincenser inom en organisation är priset normalt väsentligt lägre, och prisbilden för de s.k. *run-time system*, som krävs vid reguljär drift av en färdig tillämpning ligger normalt på en mycket lägre nivå. Prisbilden framöver beror givetvis på volymtillväxten och konkurrensen på marknaden. Vid kalkyler över systemkostnad vid drift av utvecklade tillämpningar bör man alltså inte utgå från dagens priser för utvecklingssystem. För enklare redskap avsedda för exempelvis persondatorer, ligger priserna för språk och systemskal normalt i ett läge från några tusen till några tiotusen kronor. Mer allsidiga utvecklingsmiljöer finns dock knappast att tillgå här.

6.

Beslutsstödsystem

Som tidigare diskuterats ligger i begreppet expertsystem en stark association till programvara som reproducerar förmågan hos mänskliga experter när det gäller problemlösning inom ett givet område. Med termen kunskapssystem ville vi bredda perspektivet en del och markera att denna nya tekniks användbarhet i hög grad ligger i att tillhandahålla underlag för mänskligt beslutsfattande. Därmed har den också en naturlig anknytning till tidigare arbeten som studerat beslutsstödssystem, t.ex. för beslutsfattande inom företag.

Vi ska i detta kapitel kort sammanfatta grundläggande ansatser inom området och ge några exempel på datorstödda system för beslutsstöd baserade på kunskapsteknik.

6.1 Principer.

När det gäller användning av datorstöd för beslutsfattande kan man grovt göra följande indelning i olika typer av ansatser:

- Flödesscheman och beslutstabeller. Denna ansats bygger på att man i förväg rangordnar olika kriterier och avgör vilka beslut som bör följa på en viss kombination av förutsättningar. Datorstöd kan användas för härledningen av aktuellt beslut, eller i vissa fall för att syntetisera beslutsträd m.m. ur en uppsättning exempel som täcker typiska beslutssituationer.
- Analys och presentation av aggregerad information ur databaser. Denna teknik representerar ett passivt hjälpmedel där man skaffar sig överblick över exempelvis ett företags operativa data som ett underlag för beslutsfattande i strategiska frågor.
- Matematiska modeller. I viss begränsad utsträckning kan sådana modeller formuleras och tillämpas på ett beslutsproblem. Till denna klass hör såväl rena optimeringsmodeller som modeller som identifierar orsakssamband mellan olika ekonomiska faktorer.
- Statistisk mönsterigenkänning. Denna teknik bygger på att man konstruerar ett måttsystem som kan användas för att avgöra hur nära en aktuell situation ansluter till någon eller några tidigare kända situationer. Därigenom kan man genom en analogislutledning erhålla ett förslag till hur den aktuella situationen bör hanteras.

- Bayesianska statistiska ansatser. Denna klass av tekniker bygger på Bayes' regel, som kan användas för att beräkna sannolikheten för att en viss situation är för handen, givet att vi känner olika situationers apriori sannolikheter och hur starkt observerade evidenser är relaterade till olika möjliga situationer. (Dvs, om vi har observerat ett eller flera symtom, vilken orsak kan vi då sluta oss till ligger bakom.) Sådana tekniker har med framgång tillämpats t.ex. inom medicinskt beslutsfattande, men en svaghet är att metoden är mycket känslig för att sannolikheten även för ovanliga händelser kan uppskattas med hög precision.
- Beslutsteoretiska ansatser, innefattande också bedömningar av risker och positiva värden förknippade med olika beslut och därmed förknippade sammanvägning av olika faktorer.
- Symboliska resonemang, dvs det som vi i denna framställning har kallat kunskapsbaserade expertsystem.

För närvarande knyts de största förväntningarna inom området till metoder baserade på den sista av dessa ansatser. Vi ska nedan kort redovisa inriktningen av några typiska projekt av forskningskaraktär, för att ge en känsla för teknikens användningsområden. De presenterade system har i allmänhet ej kommit till stadigvarande användning i produktionsmiljö. Man bör dock betänka att projekten ligger några år tillbaka i tiden och att kommersiella produkter för kostnadseffektiv implementering av expertsystem börjat dyka upp först under det senaste året. Vidare kan man notera vissa företag, som bevisligen utvecklat system av denna typ, valt att inte externt berätta ens om vilket område man arbetar inom.

6.2 Exempel på system inom juridik/ekonomi.

Nedanstående översikt redovisar några tidigare arbeten med expertsystem inom det juridisk/ekonomiska området. Inom forskningsvärlden finns en hel del arbeten rapporterade. Det är också känt att flera av de företag som i USA avknoppats från universitetens forskningsgrupper specialiserat sig på system för bank- eller försäkrings-världen. Dessa företag har dock ännu ej i någon större omfattning annonserat sina produkter/system.

Ett allmänt intryck är att beslutsstöd inom området utgör ett mycket lovande tillämpningsområde för expertsystem, eftersom man i hög grad arbetar med resonemang baserade på analogier och osäker eller ofullständig bakgrundskunskap. Nackdelarna är att området fortfarande anses "svårt" och att den ekonomiska potentialen för många tillämpningar är begränsad.

6.2.1 LUCKY - gåvobrev vid fastighetsöverlåtelser.

Ett mindre demonstrationssystem togs under hösten 84 fram inom samarbetsprogrammet mellan institutionen för datavetenskap (IDA) vid Universitetet i Linköping och S-E-Banken i Stockholm. Systemet är baserat på

verktyget EMYCIN och syftade till att illustrera hur expertsystemsteknik skulle kunna utnyttjas inom en juridisk tillämpning i banken. LUCKY har utvecklats av Lars Bengtsson, S-E-Banken och Henrik Nordin, IDA, med assistans av juristen Ann-Cathrine Strömberg, S-E-Banken. Arbetet med att ta fram demonstrationsexemplet utfördes med några veckors arbetsinsats. Utöver att systemet har fungerat som ett effektivt hjälpmedel för att få igång en diskussion bland jurister om möjligheter och konsekvenser av tekniken med expertsystem, har det också inspirerat till forskningsinsatser kring metodikfrågor, nämligen hantering av den situation där klienten ändrar sig under en konsultation respektive för återanvändning av kunskapsbasen för utbildning.

Tillämpningsområdet för LUCKY är att ge råd vid upprättande av gåvobrev innefattande fastighetsöverlåtelser. Systemet har i nuvarande form ingen ambition att vara på något sätt uttömmande. I stället har några typiska delproblem beaktats och enbart regler som har med dessa frågeställningar finns i systemet. Det använda verktyget har också medfört vissa begränsningar på hur uppgiften kunnat lösas, bl.a. har man tvingats använda engelska som grundspråk i dialogen med användaren.

Följande är ett något redigerat exempel på dialog med användaren i systemet LUCKY. Användarens inmatning är markerad med *kursiv* stil. Exemplet visar bl.a. användning av möjligheten att begära förklaringar och förtydliganden (användaren frågar *WHY*).

. . .
. . .
7) *What is the reciever's relation to the donator?*
** ?
What is the receiver's relation to the donator?
Expected responses are: CHILD, PARENT, KLASS2 or KLASS3
Enter HELP for list of user options.
** CHILD
8) How big is the compensation ("vederlag"), for receiving the estate?
** 300000
9) How big is the tax assessment value ("taxeringsvärdet")?
** WHY

[i.e. WHY is it important to determine the tax assessment value?]

[5.0] This will aid in determining whether the transaction is regarded as a selling.

If
[5.1] the compensation is less than the tax assessment value
then
it is definite (1.0) that the transaction is not

regarded as a selling
[RULE032]

[back to question 9...]
** 200000
10) Is the donator married (and hasn't he/she written a marriage settlement ("äktenskapsförord"))?
. . .
. . .

Figur 6.1. *Dialog i konsultationssystem för fastighetsöverlåtelser.*

Som resultat från ett fullständigt utbyggt system skulle man kunna få:

- ett förslag till utformning av det aktuella gåvobrevet;
- redovisning av ekonomiska konsekvenser i form av en beräkning av kostnaden för den valda lösningen;
- angivelse av de tillstånd m.m. som krävs för att genomföra den avsedda gåvohandlingen;
- samt kommentarer i form av allmänna resonemang kring frågor som rättviseaspekter inom familjen, kommunala förköpsrätter, önskemål om kvarboende, etc.

I den framtagna demonstrationsversionen finns ovanstående funktioner med i begränsad utsträckning. Speciellt finns några regler som behandlar förköpssrätt, hänsynstagande till givarens ålder, förenklad beräkning av kostnaden, några typer av tillstånd (god man, makemedgivande, överförmyndartillstånd, förvärvstillstånd).

Den genomförda lösningen bygger på antagandet att man har en situation där kunden kommer med en mer eller mindre uttalad avsikt att genomföra en fastighetsöverlåtelse. Eventuellt finns också ett förslag till utformning av gåvobrev eller principer för genomförandet. Den juridiske expertens uppgift är då att informera sig om bakgrunden för att kunna förvissa sig om att genomförandet kommer att motsvara kundens verkliga intentioner och avsikter. I denna process sker informationsinsamling och kontroll av olika risker och icke-önskvärda konsekvenser. Reaktion från systemet kommer i form av fortlöpande kommentarer om konsekvenser av bakomliggande förhållanden och önskad utformning av överlåtelsehandlingen, samt i form av en kontroll att den slutliga lösningen allmänt sett är lämplig och genomförbar.

I ett fortsatt arbete har en reviderad lösning till problemet studerats. Huvuduppgiften har varit att dels introducera s.k. domän-beroende programstyrning för att ytterligare öka graden av naturlighet i dialogen under en konsultation, dels medge återanvändning av kunskapsbasen för att lära ut och öva snarare än egentlig problemlösning. (Systemen POZZO och WATT, [NOR85]). Detta arbete baseras på representation av problemlösningsstrategier i form av *prototyper* baserade på typiska fall. Genom att implementering gjorts i ett system som tillåter icke-monotona resonemang, dvs att nya uppgifter kan

göra tidigare utsagor ogiltiga, har man här också på ett elegant sätt klarat problemet att klienten ändrar på förutsättningarna i ljuset av redovisade konsekvenser, t.ex. av skattemässig art.

6.2.2 Kunskapsbaserad återsökning av juridiska dokument.

I detta projekt (**Hafner**, Michigan) har man sökt förädla tekniken med informationsåtervinning (nyckelordsbaserad dokumentsökning) genom att med hjälp av s.k. semantiska nätverk representera det egentliga kunskapsinnehållet i dokumenten. Därigenom ska man också kunna återfinna referenser med utgångspunkt från deras relevans för ett ett specifikt juridiskt problem, snarare än enbart genom enbart en text-mässig likhet. Experimentsystemets innehåll baseras på *Negotiable Instruments Law, an area of Commercial Law that deals with checks and promissory notes* och utgörs av c:a 200 författningar (*statutes from the Uniform Commercial Code*) och 200 relaterade fall.

6.2.3 Regel-baserad sammanställning av dokument.

Syftet med detta system (**Sprowl**, Chicago) är att producera standardiserade dokument (exempelvis ett avtal) genom att i ett specifikt (dator)språk definiera en dokumentmall, som sedan fylls i genom en datordriven dialog med användaren. I denna dialog hämtar man dels in aktuella uppgifter, dels utförs procedurer som implementerar författningstexter och resulterar i härledda uppgifter som också kan inkluderas i det producerade dokumentet. I fälttester har man använt systemet för testamenten, fastighetsöverlåtelser, bodelning etc.

6.2.4 Datorstödd juridisk analys.

I detta projekt (**Meldman**, MIT) har man utformat ett prototypsystem för att avgöra om en viss lagstiftning kan appliceras på en given samling fakta, endera direkt (genom syllogism) eller indirekt (genom analogi). Systemet baseras på ett förenklat språk för att formulera kända fakta och engagerar användaren i en dialog för att ytterligare klargöra förutsättningar för det aktuella fallet. Denna typ av system ligger troligen fortfarande bortom vad som kan anses vara state-of-the-art för praktiska tillämpningar.

6.2.5 Utveckling av formellt språk för lagstiftning.

LEGOL-projektet (**Stamper**, London School of Economics) syftar till att utveckla ett speciellt språk (LEGOL), ett datasystem och en analysteknik som tillåter att lagstiftning kan uttryckas i en form som kan tolkas av en dator. En sådan formalisering kan användas både för "automatisering" av lagtolkning och för en effektivare återsökning av den egentliga lagtexten. Språket avviker från normala datorspråk exemplevis när det gäller förmågan att uttrycka tidsberoenden på ett konsistent sätt.

LEGOL-systemet har provats inom en rad områden (t.ex *intestate succession, purpose, right, duty, judgement, privilege, liability*, etc).

I ett relaterat arbete har man använt språket Prolog för att representera innehållet i lagen om brittiskt medborgarskap (*British Nationality Act*) [SER86]. En sådan aktivitet kan motiveras både med möjligheten att ge en precis definition av lagen och av önskemålet att kontrollera olika konsekvenser av regelsystemet, genom logisk härledning av slutsatser. I det aktuella fallet har man dock valt att inte mekanisera resonemang runt begrepp som är vagt uttryckta i lagen ("... har tillräckliga kunskaper i engelska"). I stället kvalificerar man slutsatser genom att också uttryckligen ange de (vaga) förutsättningar som måste vara uppfyllda. Allmänt har man noterat att den aktuella lagen innehöll färre oklarheter och flertydigheter än man hade räknat med, och de som fanns kunde tämligen enkelt redas ut. I flera avseenden visade sig uttryckskraften i den använda logiken (Prolog's) vara för svag för att på ett bra sätt representera skrivningen i lagen.

6.2.6 Bolagsbeskattning.

TAXMAN-projektet (McCarthy, SUNY at Buffalo) arbetar med tillämpning av AI-teknik när det gäller juridiska resonemang och argumentation med bolagsbeskattning som experimentell problemdomän. Problemställningen är uppenbarligen komplicerad och kräver resonemang på flera olika nivåer. Man hävdar att man har lyckats korrekt representera alla fakta i ett autentiskt amerikanskt skattemål, men pekar också på en rad av fundamentala problemställningar som återstår att lösa.

6.2.7 Regelbaserade modeller av juridisk expertkunskap.

Ett projekt som (i motsats till flera av de ovan relaterade) arbetar med en typisk regelbaserad resonemangsmodell är LDS (Waterman och Petersen, Rand Corp.). Man har då använt redskapssystemet ROSIE och bl.a. identifierat följande fem typer av regler som behövs i en sådan modell:

1. *formella doktriner*, dvs den egentliga lagstiftningen, etc.
2. *informella principer*, sådana regler som ehuru ej formellt definierade allmänt tillämpas i juridiska resonemng. Exemplevis innebörd i begrepp som "rimlig", "betydande omfattning", osv., eller rent kvantitativa tolkningar (t.ex. i amerikansk praxis beräknas värdet av *pain and suffering* till 3 gånger den medicinska vårdkostnaden.)
3. *strategier*, metoder som används av jurister för att åstadkomma ett önskat resultat.
4. *subjektiva bedömningar*, regler för att förutsäga hur exempelvis människor involverade i ett mål kommer att reagera (i exempelvis besluts- och värderingssituationer).
5. *bieffekter*: regler som beskriver hur olika regler kan påverka varandra inbördes.

Ett mindre pilotsystem rapporteras implementerat inom området juridiskt

beslutsfattande i samband med förlikningsavtal i samband med felaktiga produkter (*settlement strategies and practices in connection with prodcut liability litigation*).

6.2.8 Skatterådgivning.

Ett system TAXADVISOR (Michaelson, Lincoln, Nebraska), som implementerats med hjälp av redskapet EMYCIN har till uppgift att bistå med råd när det gäller skatteplanering för inkomst och gåvoskatt (fastigheter). Av rapporterna att döma har man i detta fall delvis gått ifrån den s.k. måldrivna resonemangsmetod som EMYCIN normalt baseras på, och med hjälp av egna modifieringar konstruerat en mer kontrollerad dialog med användaren. Ett sätt att se på detta system är att uppfatta det som en kompilerad checklista, som inte bara baseras på hårda fakta utan också väger in informella erfarenheter.

6.2.9 Portföljhantering.

Systemet *FOLIO: an expert system for portfolio managers.* (Cohen, Stanford.) assisterar vid fördelning av tillgångar över nio typer av värdepapper. (Systemet gör grupperingar i termer av högavkastande, låg-risk, tillväxtaktier, statspapper, etc., men går inte ner på nivån av enskilda papper.) I detta system genomgås tre faser: i en intervjufas utreds fasta förutsättningar, i en målformuleringsfas dras med hjälp av ett regelsystem slutsatser om placerarens mål när det gäller riskspridning, likviditetsbehov, osv, samt i en avslutande fas utförs en optimering relativt den givna målstrukturen. Fastställande av placerarens mål är en väsentlig del av problemställningen. Man värderer 14 olika målparametrar, var och en av dessa definierad av 5 värden (idealvärde, övre och undre gräns, "kostnad" för avvikelser uppåt respektive neråt från idealvärdet).

6.2.10 Arbetsstation för ekonomisk analys.

A Graphical Interface to an Economist's Workstation. (Williams, Wagner, Stott and Company.) Detta system demonstrerar användbarheten av en kraftfull arbetsstation med högupplösande grafik som bl.a. understödjer kommunikation med användaren i flera fönster. Den beskrivna tillämpningen utgörs av ett system för den analys och tolkning av makroekonomiska data för enskilda länder som görs inom internationella valutafonden. Systemets tyngpunkt ligger snarare på att visa värdet av ett effektivt och bekvämt användargränssnitt, snarare än att demonstrera kunskapsbaserad problemlösning.

6.2.11 Beskattning av fåmansbolag.

Som ett exempel på ett system för att stödja en texeringsmyndighet finns ett system utvecklat av Roycroft och Loucopoulos för den engelska taxeringsmyndigheten [ROY85]. Uppgiften är begränsad till rådgivning när det gäller att taxera fåmansbolag och att avgöra när vinster undanhålls beskattning genom att inte delas ut till ägarna. Regelsystemet för detta tillämpas sällan och är tämligen komplicerat, varför det utgör ett gott exempel där ett konsultationssystem baserat på expertsystemsteknik kan underlätta såväl precision som likformighet i lagtillämpningen. Det utvecklade systemet synes fungera väl men har inte tagits i drift eftersom taxeringspersonalen normalt inte har den tillgång till terminal, som skulle krävas.

6.2.12 Riskbedömning och premiekalkyl i försäkringsbolag.

Ett flermiljonsprojekt inom ett större amerikanskt försäkringsbolag, som inte publicerats offentligt, har beskrivits informellt i en rapport från *Gartner Group*. Uppgiften är att göra riskbedömning och premiekalkyl bl.a.i samband med mer komplexa försäkringsobjekt. Systemet ska assistera den ordinarie personalen snarare än ersätta den. I skrivande stund är testkörning av systemet nära förestående. Vid ett lyckat projekt kommer försäkringsbolaget att behålla äganderätten till den aktuella kunskapsbasen, medan leverantören kan återanvända det underliggande bassystemet för andra kunder.

6.2.13 Nyckelfärdiga produkter

Under det senaste året har det kommit ut några produkter inom området färdiga expertsystem för finansiellt beslutsfattande, t.ex. Palladians *Finacial Advisor* för investeringsrådgivning och APEX's (Applied Expert Systems) Planpower. Liknande programvara, utvecklad med traditionell programvaruteknik, har dock även tidigare funnits tillgänglig. Den avgörande principiella skillnaden är att ett kunskapsbaserat system ger mycket större möjligheter att påverka systemets funktion genom tillförande av personliga preferenser, bedömningar och lokal kunskap. Det är ju självklart att om olika aktörer på en marknad har tillgång till exakt samma (datorförpackade) expertis, så kan värdet av standardrekommendationen bli ringa. Dessutom gäller det ju ofta att fortlöpande kunna anpassa systemets funktion till förändrade förutsättningar av olika slag. Det tycks dock som om de generella produkter, som nu finns allmänt tillgängliga på marknaden, fortfarande är relativt begränsade när det gäller lokal anpassning.

6.2.14 Sammanfattande analys.

Thorne McCarthy (TAXMAN-projektet), som är en av ledande forskarna inom området, har pekat ut några viktiga punkter när det gäller möjlighterna att realisera konsultationssystem baserade på juridisk expertkunskap. Han gör

följande grovklassificering:

1. *Juridiska informationssökningssystem.* Inom detta område kan AI-tekniken medföra väsentligt utbyggda möjligheter när det gäller att hitta relevant lagstiftning och tidigare praxis m.m.
2. *Juridiska analys- och planerings-system.* Denna klass av system stöder dessutom direkt processen att framställa ett önskat dokument eller fatta ett juridiskt beslut. Vanligen sker detta efter en dialog med användaren där relevant information efterfrågas allteftersom. Enklare system av denna typ kan i dag skapas med någorlunda välkänd teknik.
3. *Integrerade juridiska informationssystem.* I detta fall ser man mer till hela problematiken runt lagstiftning, dess tolkning och tillämpning.

Sammanfattningsvis kan man peka på en svårighet i många juridiska tillämpningar, som gör området svårare än en del av de medicinska och tekniska områden där expertsystem hittills framgångsrikt implementerats. I juridiskt beslutsfattande är det nämligen ofta svårt att göra det "closed world"-antagande som man i dag ofta är tvungen att basera sig på. För ett sådant beslut är ofta inte bara juridisk praxis relevant, utan också vad som ur allmänmänsklig och samhällelig synpunkt kan anses vara rimligt. En intressant slutsats som McCarthy drar ur detta är att man inte nödvändigtvis ska börja med de juridiskt enkla problemen när man gör ett expertsystem, eftersom dessa ofta har starka inslag av "sunt förnuft", vilket är utomordetligt svårt att modellera i en dator. I stället kan många problem som av människor upplevs som komplexa vara bättre lämpade att lösa med hjälp av ett datorbaserat expertsystem.

6.3 Beslutsstöd inom socialförsäkringen

Ett mycket ambitiöst projekt inom detta område genomförs f.n. inom det engelska Alvey-programmet. Detta 5-års projekt har en budget på närmare 75 MSEK och utförs av ICL tillsammans med en rad universitet och andra företag i samarbete med DHSS (socialdepartementet). Huvudproblemställningen är att belysa hur kunskapsbeserade system för beslutsstöd kan användas inom en organisation vars verksamhet i hög grad baseras på ett system av lagstiftning och förordningar. För detta ändamål ska ett antal s.k. demonstrationsprojekt genomföras under den aktuella perioden.

Tillämpningar ligger inom två huvudområden, nämligen:

- handläggningstillämpningar som har att göra med rutiner för beslutsfattande, respektive
- stöd för utformning och underhåll av själva regelsystemet (lagstiftning och tillämpningsföreskrifter, m.m.) och undersökning av dess konsekvenser i olika sammanhang.

Man bedömer att utsikterna för framgångsrika projekt inom den senare av dessa båda inriktningar är betydligt mer osäkra i dagens läge, varför

huvuddelen av studerade tillämpningar rör det förra området, t.ex.

- beslut om socialbidrag
- flyttningsbidrag
- stöd för kontroll mot bedrägerier
- utbildning av handläggare
- informationssystem för klienter.

Inom dessa områden har man i princip att göra med komplexa handläggningsproblem där konventionella ADB-metoder haft klara begränsningar och dessutom i många fall påverkat arbetsrutiner och klientkontakter på ett negativt sätt. Ovanstående exempel hänför sig till situationer där uppgifter ska insamlas under svåra förhållanden, registreras, analyseras och jämföras med uppgifter i olika befintliga register. Vidare ska beslut fattas baserat på ett omfattande regelsystem av lagar och förordningar (som dessutom ofta förändras eller utökas), där reglerna ofta är oklara och där deras tillämpbarhet på en given situation är svår att avgöra. Mot denna bakgrund ska ett mycket stort antal (mer än sju miljoner om året) ärenden avgöras, motiveras och verkställas, förhoppningsvis på ett likformigt sätt i en organisation som är distribuerad över femhundra kontor. Dessutom finns önskemål om att tillhandahålla möjligheter för klienterna själva att kunna förstå sina rättigheter och möjligheter att utnyttja socialförsäkringssystemet, att utveckla och vidmakthålla personalens kompetens och att ge stöd för specialisternas arbete med att avgöra svåra frågor.

I projekten ska följande mål eftersträvas:

- att skapa ökad effektivitet och kvalitet i producerade tjänster.
- att skapa förbättrade möjligheter att tillgodose önskemål om hänsynstagande till speciella omständigheter i enskilda fall och att reagera på krav på förändringar uppifrån.
- att skapa system som fungerar med hänsynstagande till de olika användarnas behov och kunskaper på ett konstruktivt sätt.
- att skapa system som innehåller mekanismer för att skydda den enskildes rätt samtidigt med att säkerhet och tillförlitlighet garanteras på en övergripande nivå.

Programarbetet inleddes 1984 och de första demonstrerbara system-prototyperna har nu kunnat visas upp. Dessa demonstrationer befinner sig fortfarande på ett respektingivande avstånd från de ultimativa visionerna, något som i och för sig inte är överraskande. En viktig del i projektarbetet är ju också att identifiera och initiera den forskning som behövs för att föra oss närmare de utsatta målen.

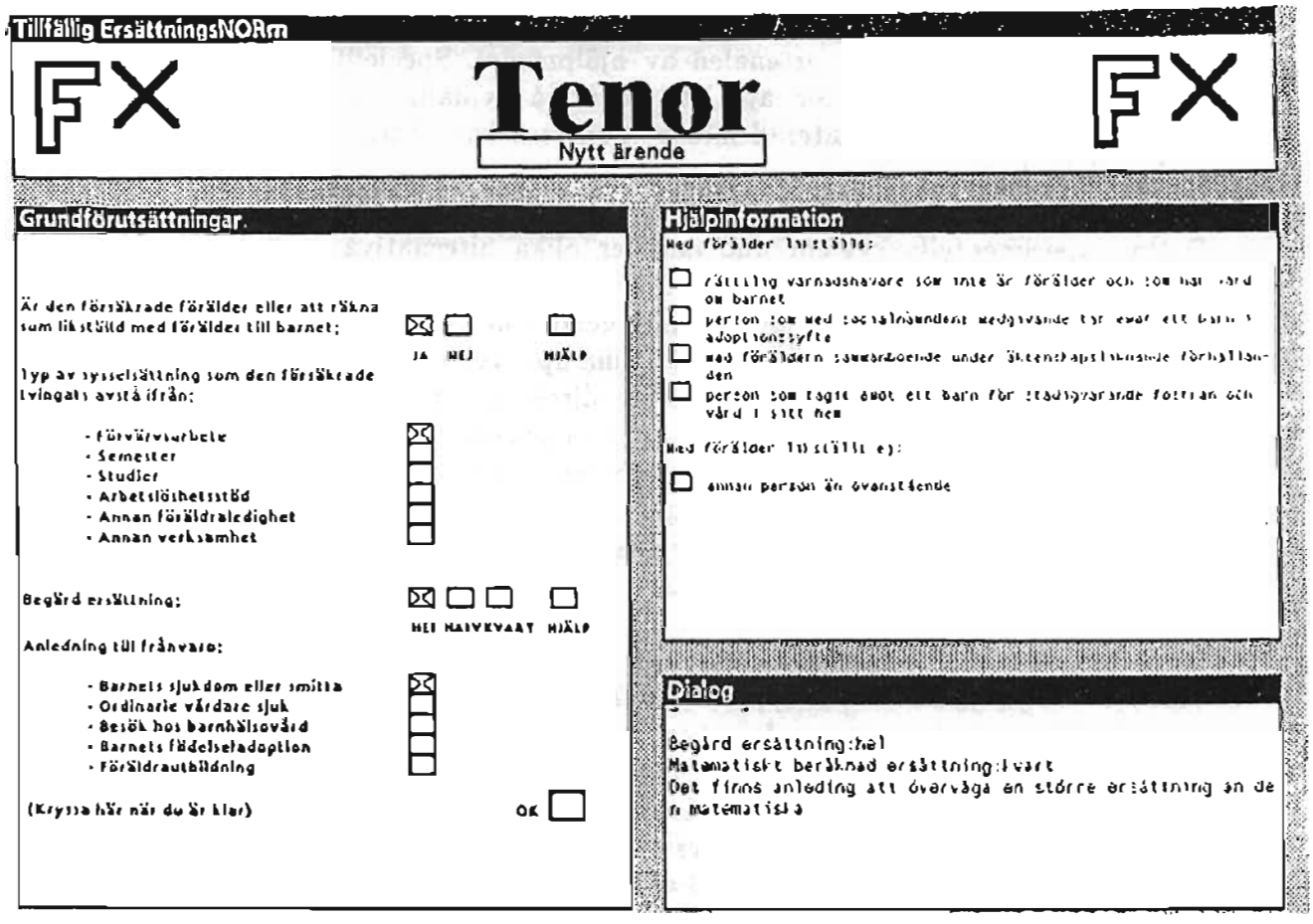
Ett av de första projekten som har förts fram till detta stadium berör beslutsstöd vid utredning om socialbidrag. I prototypsystemet arbetar man med Xerox 1108 arbetsstation ("Lispmaskin") och man har som grundmodell för användargränssnittet valt att ligga nära de blanketter som normalt

används vid handläggning. Samtidigt ger arbetsstationens understöd för grafik och parallella "fönster" på skärmen goda möjligheter att söka information och ta del av förklaringar m.m. samtidigt som datainmatning pågår. Tillgången till effektiva hjälpmedel för att "genomsöka" register och regelsamlingar (*browsing*) ingår som en viktig del i arsenalen av hjälpmedel. Speciellt viktig är denna möjlighet vid understöd för att leta reda på typfall, praxis, tillämpningsanvisningar och annan relaterad information som kan bidra vid avgörandet av en besvärlig fråga.

Detta experimentella system understödjer olika alternativa metoder för att fatta beslut, exempelvis genom att först inspektera grunddata om klienten, genom att ta fram relevant lagstiftning, genom att söka genom liknande fall, genom att låta systemet automatiskt tillämpa relevanta regler på aktuella data, eller genom att handläggaren själv direkt tar beslutet. Behovet av ett sådant system baseras inte bara på önskemålet att effektivisera själva handläggningen, utan fastmer på möjligheterna att minimera problemen med att införa och tillämpa nya förordningar och allt vad det för med sig i form av ändring av handböcker, utbildningsprogram, olikformighet i tillämpningen innan en stabil praxis hunnit utvecklas, osv.

Som tidigare nämnts har projekten ännu inte hunnit särskilt långt. De är dock värda att följas med största uppmärksamhet eftersom det är första gången som man i större skala satsar på experimentellt arbete med att utveckla och utvärdera denna nya teknik inom den offentliga sektorn. Viktigt är också understödet för användaren som beslutsfattare baserat på möjligheterna att utforma informationsteknologiska system där kunskapen sätts i centrum med bevarad handlingsfrihet för människan att utnyttja denna kunskap för olika ändamål. Det ovan beskrivna projektet med socialbidrag kommer att utvärderas med följande framgångsmått:

- Förändring i lagstiftning m.m. ska kunna hanteras genom modifiering enbart av systemets kunskapsbas.
- Användning av systemet ska medföra att lagstiftning och tillämpningsföreskrifter följs på ett korrekt och ändamålsenligt sätt.
- Systemets precision i beräkningar och redovisning ska uppfylla alla revisionskrav.
- Den datorstödda beslutsprocessen ska vara minst lika snabb som nuvarande manuella system.
- Åtkomsttiden för data om en klient ska understiga 30 sekunder.
- Berörda handläggare ska uttrycka sin tillfredställelse som användare av systemet.
- Systemet ska kunna visas vara effektivt när det gäller att få fram korrekta och fullständiga uppgifter från klienten.
- Man ska kunna visa att den demonstrerade metoden att organisera och utföra arbetet är tillfredställande för den berörda allmänheten.



Figur 6.2. *Beslutsstöd vid utredning om föräldrapenning*

Även inom den svenska socialförsäkringssystemet har man börjat titta på de långsiktiga möjligheter och konsekvenser som ny informationsteknologi kan medföra. Bl.a. mot bakgrund av erfarenheterna från det senaste systembytet inom riksförsäkringsverket har man insett detta är frågor som måste studeras och behandlas i ett långsiktigt perspektiv. För att vinna erfarenheter liknande dem som diskuterats ovan inleds nu vissa försök med användning av kunskapsbaserade system för beslutsstöd inom föräldraförsäkringen.

7.

Självförklarande programvara

Vi har tidigare kraftigt betonat den viktiga roll som *kunskap* representerad och utnyttjad i systemets interna arbete har för prestationsförmågan hos ett givet expertsystem. Denna kunskap kan vi samtidigt se som en resurs i sig och inte enbart som en förutsättning för effektiv problemlösning. Den process som genomförs vid uppbyggnad av ett kunskapsbaserat system är ju i själva verket en mödosam rekonstruktion av expertkunskap inom ett givet område och en explict formulering av denna kunskap på ett sätt som i många avseenden liknar den som man måste utföra vid författandet av en lärobok eller handledning. Det är därför rimligt att tänka sig att kunskapssystem i framtiden kan få en betydande roll som kommunikationsmedia för förmedling av kunskap från experten till olika användare. Vidare kan man redan i dag se hur kunskapen i ett expertsystem kan återanvändas för sådana viktiga funktioner som förklara och motivera resultat och för att lära ut kunskaper till användaren.

Vi ska i detta kapitel diskutera dessa aspekter på kunskapsbaserade expertsystem.

7.1 Kunskapssystemet som kommunikationsmedium

Ett möjligt och viktigt synsätt på kunskapsbaserade expertsystem är som ett kommunikationssystem vars funktion är att förmedla kunskap mellan den person som bygger upp kunskapsdatabasen och den som använder det färdiga systemet. Vi har här att göra med en kommunikation som utgör ett mellanting mellan en direkt dialog och en skriven bok, när det gäller att förmedla upplysningar baserade på en bred områdeskunskap till en annan person. I viss utsträckning förenar expertsystemet precisionen i den direkta dialogen med bokens bevarande av kunskaper från någon som kanske inte längre är tillgänglig. Behovet att säkerställa en fortsatt tillgång till intern expertis, som är knuten till enstaka personer i organisationen, anförs ofta som ett skäl för intresse för expertsystemsteknik.

Som ett exempel kan vi se ett konsultationssystem, vars uppgift är att hjälpa en handläggare att ta fram relevant information som ett underlag för beslutsfattande i de fall där de egna kunskaperna inte riktigt räcker till, t.ex. vid konstruktionsarbete, inom bank- och försäkringsverksamhet, eller inom den offentliga sektorn. De klassiska lösningarna att tillhandahålla handböcker

(motsvarande) och/eller möjlighet att rådfråga en mer erfaren kollega. Den förra metoden har fördelen att vara billig, alltid tillgänglig och lätt att tillhandahålla i stor skala. Nackdelen är svårigheten att i förväg strukturera och uttrycka kunskapen på ett sådant sätt att den kan förstås och tillämpas på alla de upptänkliga situationer som kan uppstå. Fördelen med den senare metoden är givetvis den erfarna människans förmåga att tillämpa sin sakkunskap och erfarenhet på ett flexibelt sätt med hänsyn till de speciella förutsättningar som kan gälla i ett särskilt fall. Nackdelen är i gengäld svårigheten att alltid ha en expert tillgänglig och de kostnader m.m., som en sådan lösning drar med sig.

Mellan dessa båda alternativ har vi hittills haft ett visst tomrum. I dag börjar man dock i större organisationer i allt större utsträckning ta i anspråk system för dataförmedlad post, dvs system där den enskilde tjänstemannen från sin terminal direkt skickar meddelanden till kollegor inom och utom den egna organisationen i ett ofta globalt nätverk. Med sådan teknik kan man på ett effektivt sätt bygga upp exempelvis rådfrågningssystem, där en hierarki av lokala och centrala experter kan ta hand om förfrågningar från fältet allt eftersom de kommer in och tämligen snabbt leverera svar.

Denna metod har dock fortfarande svagheten att varje enskild fråga tar expertens tid i anspråk. Den utveckling vi kan se framför oss är nu att experterna på olika nivåer i högre grad kan ägna sig åt att bearbeta och paketera sin kunskap i en sådan form att den kan mellanlagras i ett kunskapssystem, där den ligger till grund för de speciellt anpassade svar som kan levereras vid olika typer av rådfrågning. Med detta synsätt är det naturligt att uppfatta kunskapssystemet som ett sätt att förmedla kunskap, i viktiga avseenden analogt med handböcker. Därmed får också exempelvis frågan om ansvarsfördelning mellan människan och datasystemet en klarare belysning. Anvisningar som kommer från kunskapssystemet får den tyngd som är förenlig med organisationens auktorisation av systemet och det personliga ansvar som människan i sista hand alltid har.

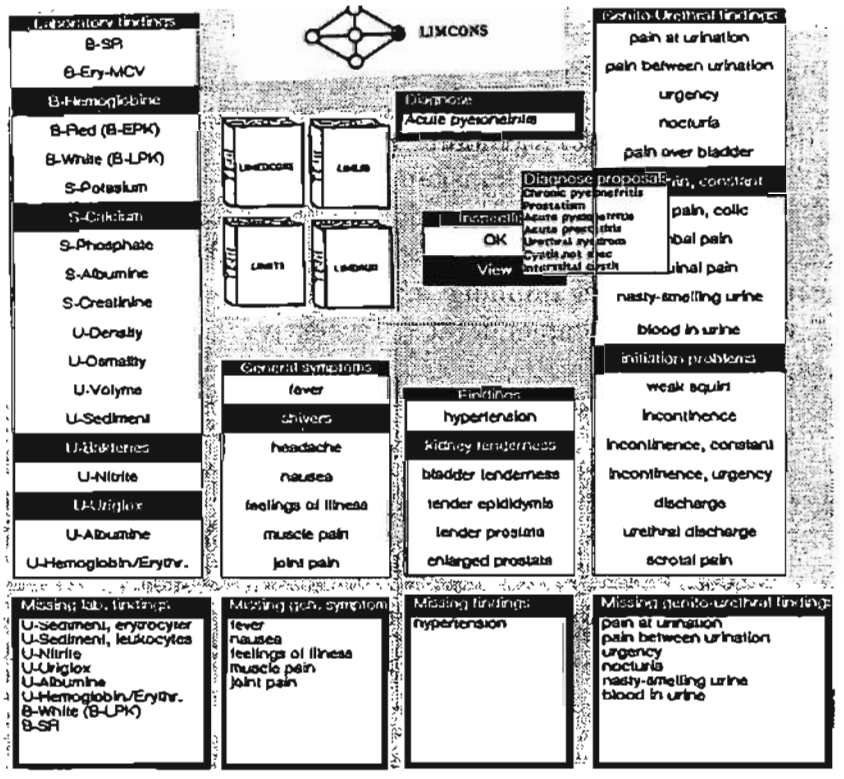
För exempel på praktiska system hänvisas till genomgången i föregående kapitel, där t.ex. målsättningen i delar av det brittiska *DHSS Demonstrator* projektet direkt syftar till effektiv genomsökning (*browsing*) av kunskapsbaser som bl.a. innehåller delar av socialförsäkringens regelsystem i en form som tillåter intern slutsatsdragning.

7.1.1 Kunskapssystem och informationssökning

En viktig användning av datorbaserad informationsförmedling är system för informationsåtervinning, *information storage and retrieval* (IR). Här är principen att i databaser lagra fulltext dokument, som ju ofta redan finns maskinläsbara från datoriserade tryckerier, eller sammanfattningar i form av löpande text eller nyckelord. Önskemålet är sedan att kunna återfinna så precist och fullständigt som möjligt de dokument som svarar mot ett visst givet informationsbehov. Systemet förmedlar i detta fall referenser till

efterfrågad kunskap, snarare än svaret på en ställd fråga.

I Linköping genomförs ett projekt inom det medicinska området, där man undersöker möjligheten att förbättra kunskapsförsörjningen i primärvården genom en kombination av kunskapssystem och klassisk teknik för informationsåtervinning genom sökning i textdatabaser. I systemet LIMCONS [TIM86] ges beslutsstöd för s.k. differentialdiagnostik och för snabb åtkomst till färdiga vårdprogram, med tonvikt på att ge underlag för beslut, snarare än en konkret rekommendation om åtgärder.



Figur 7.1. *Exempel på medicinskt beslutsstöd: systemet LIMCONS*

7.1.2 Kunskapssystem och videotex

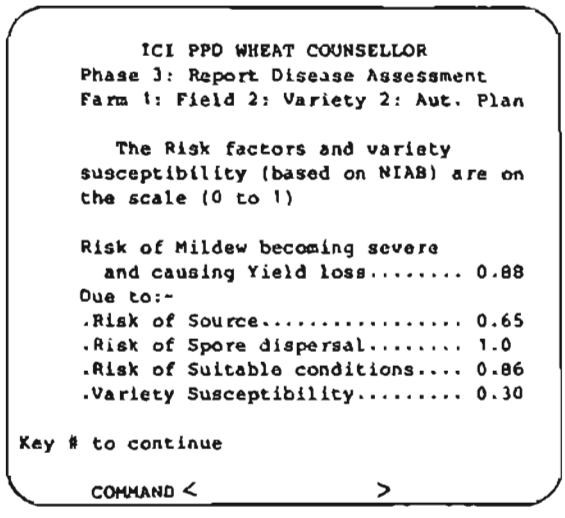
Videotex (datavision, teledata, viewdata, m.m.) är en teknik för att tillhandahålla information till företag och allmänheten, där exempelvis en vanlig text-TV apparat (med tillsats) uppkopplad över det vanliga telefonnätet kan användas för frågor mot databaser med olika slags information. Vanligen söker man sig fram till efterfrågad information i form av en eller flera textsidor genom att successivt välja ur menyer med alternativ. Man kan också direkt beställa upp sådana textsidor, vilkas beteckning i form av exempelvis ett sidnummer, som man känner till i förväg.

I ett videotexsystem kan man också ersätta textsidorna med program som exekveras när man har valt motsvarande sida. För sådan kunskap, som är svår att uttrycka i en enhetlig form som svarar mot olika användares behov, har

man i några fall provat att utveckla kunskapssystem som ska kunna utnyttjas inom ramen för ett allmänt tillgängligt videotexsystem.

Ett exempel på ett sådant system är *Counsellor* som utvecklats av ICI i England [JON85]. Detta system ska kunna användas för att ge råd till jordbrukare om lämpliga metoder och dosering av kemiska bekämpningsmedel mot sjukdomar vid spannmålsodling. Syftet är att ge en ökad precision i bekämpningen, baserad på kännedom om lokala förutsättningar, aktuell vädersituation, aktuell sjukdomsrisk och kännedom om tillgängliga bekämpningsmedel. Systemet motiveras av att tillgänglig expertis blir hårt belastad under korta perioder, möjligheten till förbättrad ekonomi samt minskade miljörisker.

Systemet *Wheat Counsellor* genomför en dialog med den uppkopplade användaren för att hämta in kunskap om dennes odlingssituation. Denna information kompletteras från interna databaser, varefter systemet producerar en kommenterad bedömning av risker för olika sjukdomar, föreslår bekämpningsmetod (om bekämpning är rekommendabel), samt redovisar en kostnadskalkyl för rekommenderat respektive eventuella alternativa bekämpningsprogram.



Figur 7.2. *Exempel på riskbedömning i systemet Counsellor.*

7.2 Förklaringshjälpmedel

Expertsystem har också bl.a. inneburit ett genombrott för möjligheten att konstruera *transparenta* program, där beslutsregler och annan kunskap direkt kan inspekteras av användaren. Även om mycket återstår att göra, när det gäller att presentera relevant information ur systemets kunskapsbas på ett effektivt sätt, så har expertsystemen på ett övertygande sätt demonstrerat hur

producerade resultat kan förklaras och motiveras på ett för den normale användaren begripligt sätt.

Vidare tillhandahåller många redskapssystem fortlöpande hjälp- och stödinformation på ett effektivt sätt, som kräver ett minimum av extra arbete vid uppbyggnad av expertsystemet. Konflikter uppstår dock lätt vid val av intern representationsteknik, eftersom ett mer kraftfullt system ofta ökar komplexiteten i uppgiften att konstruera ett förklaringssystem. Tillgång till förklaringsfaciliteter borde vara av stort värde, även i mer konventionellt utformad programvara, och den är av central betydelse i system som understödjer resonemang med hänsyn till osäkerhet i data och beslutsregler.

De stora betydelse som förmågan att redovisa den interna kunskapen och resonemangsprocessen i ett expertsystem har, kan motiveras på bl.a. följande sätt:

- *förståelse*, hos såväl den som bygger upp och underhåller systemet, som hos den slutlige användaren.
- *felsökning*, inklusive intrimmning av den kunskap som systemet baseras på.
- *utbildning*, av användaren som genom att ta del av kunskaper och resonemang kan tillägna sig systemets problemlösande förmåga i viss utsträckning.
- *acceptans* för resultat, dvs att övertyga om trovärdigheten av de rekommendationer som systemet producerar.
- *visa systemets begränsningar* genom att synliggöra under vilka förutsättningar systemets kunskap är tillämpbar.
- *lokal hjälpinformation* under en dialog där användaren genom att ta del av den interna resonemangsmodellen bättre kan förstå innebörden av frågor som systemet ställer.

Sammanfattningsvis kan vi identifiera tre olika kategorier användare med delvis olika behov av förklaringar:

slutanvändaren,
som kan få förutsättningar och motiveringar för producerade resultat, allmän information genererad ur den kunskap som finns representerad i systemet, samt hjälp och assistans under dialog med systemet.

områdesexperten,
som kan få hjälp med att kontrollera att hans kunskaper återges på ett korrekt sätt i systemet och med att successivt formulera konsistenta tillägg till kunskapsbasen, exempelvis vid fortlöpande underhåll där en förändrad omvärld kräver motsvarande förändringar i systemet.

kunskapsingenjören,
som kan få hjälp att kontrollera fullständighet och frihet från motsägelser i den representerade kunskapen, och speciellt få ett effektivt understöd för att felsöka kunskapsbasen när man inte är nöjd

med systemets prestationer.

Viktiga frågor i samband med produktion av förklaringar har att göra med metoder för att identifiera vilka uppgifter som är relevanta och intressanta med ett högt förklaringsvärde för en given användare, samt hur man gör presentationen så informativ som möjligt, t.ex. genom att generera en text i naturligt språk. Tekniska metoder för detta baseras på omstrukturering av härledningskedjor och "intelligent" selektering av de viktigasten stegen. Vidare kan man tänka sig formuleringen av en förklaring styrs av användarens förmodade bakgrund och erfarenheter.

Förklaringar kan produceras direkt från programkoden, vilket dock leder till kodberoende och försvåras av att domänkunskap ofta blandas med styrande information. Tyvärr uppfattas ibland möjligheten till förklaringar som synonymt med att kunna reproducera spårutskrifter (*trace*) av resonemangskedjor som svar på *hur*- och *varför*-frågor (se nedan), trots att dessa kan vara tämligen svåra att genomskåda om man inte har en mycket "ren" tillämpning. Givetvis är det ändå en styrka, om man vid utvecklingen av ett system automatiskt får tillgång till förklarande texter, som på ett för slutanvändaren begripligt sätt kan redovisa varje steg i programmets beräkningar eller bearbetning av data.

I praktiken baseras förmågan att förklara och motivera ett expertsystems arbete i dagens tillämpningar ofta på handkodade förklaringsfunktioner och förpreparerade texter. De utvecklingsverktyg som allmänt används ger vanligen ett gott stöd för denna uppgift. Man kan vänta sig ett snabbt ökande utbud av hjälpmedel för denna uppgift.

Av grundläggande betydelse för att ett expertsystem ska accepteras är alltså möjligheten att kunna redovisa och motivera den kunskap som ligger till grund för systemet, de resonemang som genomförs och hur ett givet resultat uppnåtts. Speciellt viktigt är detta i de fall där systemet tjänar som ett stöd för mänskligt beslutsfattande. Vissa system använder till och med en s.k. kommenterande ansats (*critiquing*), där användaren kan få sitt eget förslag till beslut värderat av systemet och argument för alternativa beslut redovisade.

De vanligaste förklaringsmekanismerna är de s.k. HUR- och VARFÖR-frågorna, som kan användas endera direkt i samband med den dialog där användaren tillför systemet ny information, eller också i en separat fas efter det att systemet levererat sitt resultat:

* HUR kom programmet fram till en viss slutsats? En förklaring genereras genom att följa resonemangskedjan bakåt till ursprungligen kända eller av användaren lämnade uppgifter.

* VARFÖR frågor programmet efter en viss uppgift? Den hypotes som programmet söker verifiera redovisas. Vid behov kan också hela den kedja av resonemang visas, som leder framåt från den efterfrågade uppgiften till slutmålet.

Utöver dessa frågetyper, som ofta utnyttjas direkt under dialogen med ett konsultationssystem, kan finnas möjligheter att ställa frågor direkt mot kunskapsbasen eller till frågor efter en konsultation för att utröna hur systemets slutsatser härletts ur indata, exempelvis:

Hur användes en viss uppgift?
Vilken slutsats drogs beträffande ett givet subproblem?
Varför användes inte en viss uppgift?
Varför avvisades en hypotes, som kanske användaren funnit rimlig
Etc.

```
Enter Debug/review phase, or other option (? for help) : EXPlanation

** WHY DIDN'T YOU CONSIDER THE RESULT OF THE ABO-CLASSIFICATION OF REFERRAL-1?
During the preceding consultation, I never had reason to try to find out the
     result of ABO-classification.  It would have been necessary to know this
     information in order to use any of the following:  66, 67, 65, 56, 53, 52.
     However, none of these succeeded in the context of 123.  If you would
     like an explanation for why any of these rules failed, please enter
     their numbers:
      ** 56
RULE056 could be applied to:  REQUESTED-ANTIBODY-1
Clause 1 of RULE056 äcoldantibodies is one of the actual antibodiesä was
     already known to be false for REQUESTED-ANTIBODY-1, so the rule was
     never tried.

** HOW DID YOU USE THE RESULT OF THE AAL-TEST OF SCREENING TEST-1?
RULE022 used:
     the result of the AAL-test in the SCREENING TEST-1
This information also would have been used in:  15, 16, 14, 19, 20, 24, 25, 31,
     32, 88, 92, 70, 71, 93.  However, none of these succeeded in the context
     of SCREENING TEST-1.  If you would like an explanation for why any of
     these rules failed, please enter their numbers:
      **

** WHAT DID RULE 22 TELL YOU ABOUT 123?
RULE022 could be applied to:  SCREENING TEST-1

RULE022 was used to conclude that your screening test was negative.  The last
     question asked before the conclusion was made was 11.
```

Figur 7.3. *Exempel på förklaringar i systemet A³ för antikroppsanalys.*

Möjligheten att realisera sådana självförklarande egenskaper i ett programvarusystem förenklas väsentligt av det enhetliga användandet av produktionsregler, som vi tidigare beskrivit. I system av s.k. EMYCIN-typ (t.ex. TI's Personal Consultant och Teknowledge's S1) genereras förklaringar

av ovanstående typ uttryckta i naturligt språk direkt från systemets interna representation. För andra verktyg varierar det i vilken grad dessa typer av automatiserade förklaringar kan erbjudas.

7.3 Undervisande system

Det kan vara värt att observera att en intressant tillämpning för expertsystem i praktisk användning är för undervisningsändamål. Exempelvis inom det medicinska området återstår ännu en hel del att utreda beträffande lämpliga former för ianspråktagande av datorbaserade konsultationssystem, men i väntan på detta kan befintliga system i varje fall användas som effektiva utbildningshjälpmedel.

Vidare kan man notera den enorma potential som ligger i möjligheten att direkt realisera utbildningsfunktioner för slutanvändare inom ramen för ett kunskapsbaserat system. Genom att behandlingsregler och informationssamband ligger explicit tillgängliga, kan man lätt ge fortlöpande tillgång till förklarande information och med små extra insatser kan rena undervisningsmoment införas inom ramen för ett driftssystem.

En bestickande analogi som framförts i sammanhanget är möjligheten att vi framtiden kommer att tillhandahålla intellektuella tränings- och utbildningsmoment i datasystemen, på samma sätt som vi har motionsslingor och andra anläggningar för fysisk träning i dagens samhälle där vi i stor utsträckning har ersatt kroppsarbete med maskiner.

Datorstödd undervisning introducerades tidigt som en lovande användning av datateknik. Av olika skäl har dock endast begränsade framsteg gjorts inom området. Ett viktigt problem har därvid varit resursåtgången för att bygga upp en "lektion", ofta många hundra timmar för en lektion som räcker en timme. Att bygga undervisningsprogram med användning av befintliga kunskapsbaser löser uppenbarligen en stor del av detta problem. Omvänt ger de utvecklingsverktyg som finns för att bygga kunskapssystem också ett mycket effektivt stöd för att direkt konstruera undervisningsprogramvara.

Huvudmetoderna för datorstödd undervisning är

1. **Programmerad undervisning.** Principen är att ett kunskapsmaterial struktureras i moduler, som presenteras av systemet i en ordning som beror på elevens resultat när det gäller att tillägna sig och redovisa kunskaperna. De kunskaper som kan beröras inom ett typiskt tillämpningsområde kan motsvara dem som behövs för att producera förklaringar eller hjälpinformation till en slutanvändare av ett expertsystem. Därmed reduceras problemet till att definiera den villkorliga sekvenseringen av respektive kunskapsfragment.

2. **Simuleringar.** Denna metod baseras på möjligheten till lärande genom att prova sig fram, exempelvis genom att ansätta olika

parametervärden för en simuleringsmodell. En i sammanhanget mer intressant typ av simulering är övningar i beslutsfattande där systemet fortlöpande simulerar en situation där eleven får agera. Inom läkarutbildning används exempelvis denna metod för att öva elevens förmåga att tillämpa sina basala kunskaper i situationer, där man av olika skäl inte kan använda riktiga patienter. Sett ur perspektivet expertsystem har vi här att göra med det inverterade problemet, systemet sitter inne med kunskaper om den aktuella situationen och användaren försöker lösa problemet. Flera projekt har framgångsrikt visat att sådana simuleringsövningar med enkla medel kan åstadkommas inom ramen för ett kunskapssystem [NOR85].

3. **Dubbelriktad dialog.** En effektiv undervisning förutsätter normalt att eleven kan ställa frågor och få saker förklarade. Traditionell datorstödd undervisning har knappast klarat att tillgodose det kravet. Däremot är det uppenbart att kunskapssystem, gärna kompletterade med någon typ av förståelse av naturligt språk, ger vissa möjligheter inom detta område.

4. **"Learning-by-doing".** Denna teknik baseras på att man genom att observera någon som löser ett problem, eller genom att själv lösa problem under sakkunnig ledning, successivt tillägnar sig kunskaper inom området. Ofta har man ansett att datorbaserad problemlösning leder till att berörda kunskaper går förlorade, men det borde vara möjligt att motverka detta och rent av att öka användarens kunskaper genom utforma system som arbetar med målmedvetna strategier och som förklarar och motiverar sitt arbete.

Av dessa metoder kan vi säga att den sista automatiskt tillhandahålls vid problemlösning med hjälp av ett kunskapsbaserat konsultationssystem, åtminstone om vissa rimliga förutsättningar är uppfyllda. Med antagande om att systemets problemlösning baseras på någorlunda intelligenta strategier och med en lämplig utformning av förklaringsfunktioner kan man ofta förvänta sig att en användare så småningom kan tillägna sig förmågan att lösa motsvarande uppgifter på egen hand.

Metoden med simuleringar har också demonstrerats i flera sammanhang, bl.a. för medicinska expertsystem, som i vissa fall konverterats till hjälpmedel för att lära ut kunskaper. Figur 7.4 visar ett exempel där ett regelbaserat system för rådgivning om fastighetsöverlåtelser (se avsnitt 6.1.1.) "inverterats", så att systemet och användaren bytt roller. I stället för att lösa användarens problem, genererar systemet här en hypotetisk problemställning. Användarens uppgift är nu att försöka lösa problemet, varvid systemet kan intervenera när användaren kör fast eller avviker från den lösningsstrategi som implementerats i kunskapssystemet. Med denna teknik kan man på ett mer effektivt sätt återanvända en kunskapsbas för att tillägna sig systemets problemlösningsförmåga. Denna möjlighet är speciellt intressant i sammanhang där små förändringar i bakomliggande förutsättningar kan få stort genomslag i resultathänseende, varför även en någorlunda kunnig användare kan ha hjälp

av ett aktualiserat system för att kontrollera sin förmåga att rätt lösa uppgifter efter att sådana förändringar inträffat.

```
**5 I know the cost of the transfer

   You have not enough information to conclude
   the cost of the transfer. You know:

TAXED.VALUE = 125000
COMPENSATION = 100000

   There is 1 parameter more which is needed...

The student thinks he knows a value which cannot yet be determined. He is
then given a summary of what he knows and the number of parameter values
lacking to conclude what he thought he knew.


We skip a number of interactions....


**12 HINT
   You are currently pursuing the strategic goal
   Necessary permissions for the transfer.
**13 HINT
   You should determine whether
   a permission from the wife/husband
   is necessary
The student can always ask for a HINT. The first time he asks, he is just
informed what strategic goal is pursued. The second time, he is given the name
of the nearest goal pursued.

**14 Is the donor married
YES
**15 Does he have a marriage settlement
YES
**16 I know wife husband permission
OK, is it necessary?
** 15 NO
Wrong, it is necessary:

When the student gives a wrong answer, the system shows the rules that have
been used by LUCKY to conclude another answer.

   We try to determine whether a permit from the
   wife/husband is necessary, the rule is [Rule005]:

     If 1) the donor is married, and
        2) the donor has a marriage settlement,

   Then 1) a permission from the wife/husband is necessary.
```

Figur 7.4. *Exempel på en genererad undervisningssimulering.*

7.4 Stöd för textproduktion

Konventionella datorbaserade hjälpmedel för framställning av dokument involverar ordbehandlingssystem och texteditorer för att skriva in, modifiera och redigera text för utskrift. Hjälpmedel för stavningskontroll kan ibland erhållas och det finns till och med vissa system som kontrollerar en text ur vissa stilistiska synpunkter, såsom längd på meningar och upprepningar m.m. I nästa kapitel ges ytterligare exempel på system för textproduktion i samband med användning av system som hanterar naturligt språk.

Om vi emellertid ser ett kunskapssystem som ett nytt sätt att förpacka information, som sedan kan presenteras för "läsaren" / konsumenten på ett individualiserat sätt, så kan vi förvänta oss ett behov av mer sofistikerade hjälpmedel för denna process [HJE86]. Existerande metoder för att konstruera kunskapssystem är i första hand orienterade mot att understödja problemlösning snarare än kunskapsförmedling. Vi kan emellertid vänta oss en fortsatt utveckling av metoder och hjälpmedel för att exempelvis dynamiskt generera problem- och användaranpassade texter ur en kunskapsbas.

8.

Människa-dator interaktion

I många tillämpningar spelar utformningen av gränssnittet människa - dator (i vid mening) en stor och kanske avgörande roll. Uppskattningar när det gäller typiska (administrativa) applikationsprogram ger vid handen att kanske två tredjedelar av programtexten avser hantering av in- och ut-matning, inklusive hantering av fel och undantagssituationer m.m. Samtidigt anser man också ofta att en ytterligare förbättring av gränssnittet mot användaren skulle vara en av de mest angelägna uppgifterna.

I dagens tekniska och administrativa datasystem arbetar man ofta med någon form av hjälpmedel för att specificera och implementera användardialoger, t.ex. tillståndsdiagram eller olika typer av grammatiker för att definiera kommandospråk, formulärsystem för layout av bildskärmar, tabellstyrda menyval, osv. I typiska kunskapssystem genereras dialogen på motsvarande sätt från abstrakta beskrivningar. Men det är dessutom vanligen så att dialogen här genereras direkt ur systemets arbete med att härleda en lösning med hjälp av kunskapsbasen i stället för att programmeras upp i detalj. Detta minskar såväl programmeringsarbetet som underhållsproblematiken.

Många realiserade expertsystem baserar dialogen med användaren på (begränsat) naturligt språk (i skriven form). Språkhanteringen omfattar både inmatning från användaren och presentation av resultat respektive redovisning av systemets interna kunskap i form av fakta och regler. Emellertid överskattas lätt betydelsen av att kunna genomföra en dialog i naturligt språk. Det är ofta effektivare att använda andra interaktions- och presentations-tekniker, som också kan vara naturliga för användaren även om de inte baseras på människans vardagliga språk.

8.1 Dialogutformning.

I expertsystemen läggs typiskt stor vikt vid uppgiften att understödja en bekväm dialog såväl med kunskapsingenjören under utveckling och underhåll av ett system, som med en slutanvändare. De viktigaste teknikerna härvid är utnyttjande av grafisk presentations- och interaktionsteknik på rastergrafiska arbetsstationer (t.ex. s.k. Lisp-maskiner) och användning av naturligt språk. Att genomföra en dialog i helt naturlig svenska eller engelska ligger fortfarande långt bortom vad som är realistiskt att förvänta sig av ett datorprogram, men lyckligtvis är en sådan språkförmåga också av begränsad betydelse i

sammanhanget. Vad som är viktigare än en fullständig språklig frihet är att dialogens struktur upplevs som naturlig av användaren, och att man på ett effektivt sätt kan arbeta sig fram till det slutliga resultatet.

En aspekt på denna fråga är fördelningen av *initiativ* i dialogen, d.v.s. i vilken grad användaren kan styra sekvenseringen av interaktioner med systemet. Vi kan här notera att system baserade på målstyrd slutsatsdragning (se föregående avsnitt) i högre grad än de som understödjer framåtriktad slutsatsdragning kontrollerar den ordning i vilken information kan överföras till systemet. I praktiken leder detta också till mera begränsade behov av understöd för tolkning av fritt inmatade uppgifter, eftersom programmet i varje givet läge begär svar på en specifik fråga till användaren. Man får alltså här en förenklad teknisk lösning på problemet att hantera dialogen med slutanvändaren till priset av att dennes möjligheter att styra systemet begränsas.

Uppgiften att utforma gränssnittet mot användaren utgör i ett typiskt konsultationssystem, liksom i all interaktiv programvara, en stor del av arbetsuppgiften. Man bör dock notera att i motsats till konventionella dialogsystem specificeras inte dialogflödet i detalj på förhand. En poäng är ju att problemlösningen, och därmed dialogen, hela tiden ska styras på mest ändamålsenligt sätt av den information som i varje steg är tillgänglig för systemet. Mängden möjliga vägar genom ett problem är vanligen alldeles för stor för att i detalj kunna förutses, varför man i stället koncentrerar sig på uppgiften att förse systemet med generell kunskap som kan underlätta uppgiften att generera förståeliga frågor och resultatpresentationer. De kunskapssystem som utvecklas i dag är ofta baserade på användning av arbetsstationer med högupplösande grafik som understödjer en effektiv och problemanpassad dialogteknik, bl.a. med användning av multipla fönster som motverkar nackdelarna med ett rent sekventiellt dialogflöde.

8.1.1 Intelligenta användargränssnitt.

I många sammanhang, inte minst i framtidens kontorsarbete, kan vi förvänta oss att behöva utnyttja olika datorbaserade tjänster som innebär användning av en rad olika program, initiering av olika funktioner inom dessa program, inmatning av data och tolkning av utdata från systemen. Även om vi för närvarande kan se en mycket snabb kvalitetshöjning när det gäller utformning av användargränssnitt i interaktiva system, framför allt på persondatorsidan, måste vi räkna med att användargränssnitten blir en kritisk och begränsande faktor i många tillämpningar.

Med ett *intelligent användargränssnitt* (*intelligent frontend*) menar vi i detta sammanhang ett program som sköter dialogen mellan en användare och ett underliggande programsystem. Uppgiften för detta gränssnittsprogram är att:

- ge användaren en likformighet i kontaktytan mot ett antal olika underliggande programsystem, som kanske sinsemellan har mycket varierande kommandospråk och dialogkonventioner.

- underlätta för användaren att välja rätt programsystem eller rätt programfunktion (t.ex. vid tillgång till ett antal likartade programpaket för statistisk analys) och att formulera sitt problem på det sätt som systemet kräver.
- assistera vid bedömning av relevans, giltighet och kvalitetsegenskaper hos tillgängliga datamaterial.
- hjälpa till att ge kommandon, parametrar och data den syntaktiskt korrekta form som det valda programmet kräver och att vid behov föreslå lämpliga defaultvärden.
- initiera exekvering av det underliggande systemet med hänsyn till lokala systemstrukturer och namnkonventioner.
- understödja tolkning av de utdata som systemet producerar.

Tillgången till intelligenta gränssnittssystem av denna typ kommer att vara av största vikt i framtiden, eftersom utbudet av disparata programvaror snabbt ökar. Vi har också den snabbt ökande risken för felaktig användning av programvara på grund av otillräcklig kännedom om ett specifikt program och bristande insikt i förutsättningarna för dess tillämpbarhet. Som ett exempel kan vi ta användning av olika standardsystem för statistisk bearbetning, där en okunnig användare kan dra totalt felaktiga slutsatser av en analys beroende på att han otillräckligt förstått vilka kvalitetskrav som måste ställas på indata och hur resultatet korrekt ska tolkas. Utformning av ett kunskapsbaserat system inom detta område, i form av "statistikerns arbetsstation", pågår för närvarande i Linköping [IDA85].

Ett annat exempel på system inom detta område är det expertsystem för assistans till konsoloperatören i en IBM datacentral, YES/MVS, som utvecklats och tagits i drift vid IBM Yorktown Heights [GRI84]. YES/MVS övervakar och filtrerar trafiken av meddelanden till operatörskonsolen och åtgärdar själv vissa typer av situationer, som normalt kräver ett rutinmässigt operatörsingripande. För mer komplicerade fall kan systemet ge hjälp och rekommendationer till operatören.

8.1.2 Hjälpfunktioner

Vi har flera gånger tidigare berört frågor om att assistera användaren och ge information om ett system och dess användning. I detta avsnitt ska vi kort sammanfatta de viktigaste teknikerna för att ge hjälp till en användare i ett datoriserat informationssystem och speciellt kommentera anknytningen till kunskapsbaserade system.

Typer av hjälp:

- Lokala hjälptexter som förklarar och underlättar korrekt inmatning av kommando eller data. I ett kunskapsbaserat system kan en sådan hjälpfunktion delvis automatiseras genom möjligheten att generera

förklaringar som visar hur systemet kommer att använda de uppgifter som efterfrågas.

- Global hjälpinformation, som beskriver systemets funktioner och struktur, vilka undantagskommandon som finns (för hjälp, avbrott, etc)
- Allmän hjälpinformation, som beskriver systemets uppbyggnad och användning (typ handbok, undervisningssystem, m.m.) Vissa möjligheter finns att i ett kunskapsbaserat system återanvända och/eller utvidga kunskapsbasen för dessa uppgifter.
- Information om aktuellt tillstånd (*mode*) i dialogen, dvs i termer av hur inmatning kommer att tolkas respektive hur man kan ta sig tillbaka till exempelvis toppnivån.
- Illustrerande bilder, figurer och ledtexter som automatiskt skrivs ur för att orientera och vägleda användaren.
- Information om feltillstånd som uppstått, hur felmeddelanden ska tolkas och vad man kan göra föra att komma till rätta med det aktuella felet.
- Möjligheter att prova sig fram, exempelvis genom använda ett system för övning, utan att förorsaka några oönskade externa effekter. En viktig funktion i detta sammanhang är möjligheten att göra *"undo"*, dvs att backa i dialogen och återställa alla förändringar som gjorts. När det gäller kunskapssystemen tillhandahåller en del utvecklingsverktyg möjligheter att resonera om s.k. *hypotetiska världar*, dvs att parallellt undersöka olika alternativa lösningar.
- Förslag om alternativa sätt att lösa en given uppgift. T.ex. att känna igen att en serie kommandon ingår i en plan för att lösa en uppgift som man kan klara med ett enstaka kommando [FIN83].

Dessutom har vi tidigare berört möjligheten att konstruera intelligenta användargränssnitt, som ju kan ses som ett sätt att organisera extern hantering av hjälpfunktioner för ett visst system. I det följande beskrivs också frågesystem baserade på naturligt språk, där en viktig tillämpning är att kunna ge assistans till en oerfaren användare, som kanske inte ens behärskar de mest elementära kommandona i ett system. Ett exempel på ett sådant hjälpsystem för operativsystemet Unix visas i figur 8.4.

Inom det europeiska ESPRIT-samarbetet pågår bl.a. ett projekt, EUROHELP, för att studera hur intelligenta hjälpsystem kan utformas. Ett prototypsystem har utvecklats för att understödja användare av system för datorförmedlad meddelandesändning [MOH85]. I det fortsatta arbetet avser man att utveckla generella hjälpmedel för konstruktion användarvänliga hjälpsystem i olika tillämpningar.

```
User: How do I delete a file?
      (or remove a file)
UC:   Typing 'rm filename' will remove the
      file with name filename from your
      current directory.
User: a directory?
UC:   Typing 'rm -r directoryname' will
      recursively delete the directory with
      name directoryname and all the files and
      directories contained in the directory.
User: How should one edit a file?
      (or change a file)
UC:   Typing 'ex filename' will get you into
      the ex editor; typing 'emacs filename'
      will get you into the emacs editor.
User: What is the best way to create a file?
      (or the right way or the proper way, or
      make a file)
UC:   The best way to create a new file is to
      use the editor, i.e. - 'ex filename'.
User: a directory?
UC:   Typing 'mkdir directoryname' will
      create a directory with name
      directoryname in your current
      directory.
User: Do you know how to read a file?
      (or list a file)
UC:   Typing 'pr filename' will print the file
      with name filename on your terminal.
User: How can I find out how full the disk is?
UC:   Typing 'df' will tell you how full the
      disk is.
```

Figur 8.1. *Hjälpdialog i (begränsat) naturligt språk.*

8.2 Hantering av naturligt språk.

Vi ska i detta avsnitt kort redovisa problem och möjligheter med användning av naturligt språk, dvs "vanlig" engelska eller svenska, etc. i samband med datoranvändning. Utöver datorstödd översättning och möjligheten att använda naturligt språk i själva dialogen med datorsystemet, tar vi också upp områdena textgenerering och textanalys, dvs möjligheten att översätta mellan externa texter och interna kuskapsstrukturer.

8.2.1 Maskinell översättning

Redan under 60-talet satsades stora resurser på att utveckla system som kunde översätta en text från ett språk till ett annat. Tyvärr visade det sig att man överskattat möjligheterna att generalisera från de inledande framgångarna med små enkla system, som kunde klara överraskande mycket. Försöken att ta fram kommersiellt gångbara system för maskinell översättning misslyckades i stort sett och hela området kom i vanrykte.

Behovet av hjälpmedel för att underlätta hantering av den snabbt växande volymen av översättningsarbete, t.ex. av vetenskaplig litteratur, tekniska manualer och officiella dokument inom exempelvis internationella organisationer har emellertid ökat undan för undan. Under senare år har ett antal större projekt startats inom området, samtidigt som en del någorlunda användbara produkter har börjat dyka på marknaden, bl.a. för användning på persondatorer. Stora projekt finns exempelvis i Japan, där man av uppenbara skäl har stort behov av system för översättning till och från västerländska språk. Inom EG pågår projektet EUROTRA 1983-88 där prototypsystem för översättning mellan sju språk ska finnas klart under 1986. Detta projekt är bl.a. motiverat av den stora volymen översättningsarbete till de olika språken inom gemenskapen som krävs för de dokument som produceras i samarbetet.

Dagens teknik inom området bygger på insikten att rent syntaktiska metoder inte är tillräckliga. I stället försöker man använda analystekniker som kombinerar morfologisk, lexikalisk, syntaktisk och semantisk analys, dvs man försöker arbeta delvis parallellt med analys av ändelser, ordstammar, grammatik och tolkning av "betydelse". Dessutom har man övergivit ambitionen att helt automatisera översättningen och siktar i stället på att göra system som stöjer och effektiviserar en mänsklig översättare. De produktivitetsökningar som rapporteras från de system som i dag är i drift rör sig om förbättringar med mellan 25% och 150%. En annan väg som man också provar är att underlätta maskinell översättning genom att i förväg markera och märka upp texten på lämpligt sätt. Det finns all anledning att räkna med att vi inom de närmaste fem åren kommer att få se en kraftig tillväxt av hjälpmedel både för högkvalitativ mänsklig översättning och för massöversättning av exempelvis tekniska dokument.

8.2.2 Frågespråk mot databaser

Det område där system med förmåga att hantera naturligt språk först har kommit i allmänt, om än fortfarande begränsat, bruk är frågespråk mot databaser. Modern databasteknik för administrativa och tekniska tillämpningar är baserade på den s.k *relationsmodellen*, vilket innebär att databasen uppfattas bestående av matematiska mängder av likformiga objekt. Dessa mängder har en naturlig och enkel tolkning i form av tabeller, som användaren kan genomsöka eller kombinera på olika sätt för att få fram information. Genom anknytningen till den matematiska logiken har man fått en bas för användning av formella språk, t.ex. i form av en algebra, för att operera på databasen. Därmed elimineras i många sammanhang behovet av konventionella programmeringskunskaper och av insikter om exakt hur data är tekniskt lagrade i systemet.

Sådana formella frågespråk kräver dock fortfarande att man lär sig en precis syntax och framför allt att man känner till exakt vilken information som finns lagrad och de namnkonventioner som gäller i databasen. För tillfälliga användare, t.ex. en företagsledare som vill ta fram underlag för en marknadsbedömning ur företagets administrativa databas, är det ofta önskvärt

att frågor kan formuleras i (ett på lämpligt sätt begränsat) naturligt språk, varefter systemet tolkar frågan och översätter till det formella frågespråket.

```
LIST THE SALESMEN IN NEW YORK WHO ARE UNDER QUOTA.

YOUR REQUEST IS AMBIGUOUS TO ME.  DO YOU WANT:
 (1): CITY = NEW YORK.
 (2): STATE  = NEW YORK.
PLEASE ENTER THE NUMBER OF THE INTERPRETATION YOU INTENDED.

2


PRINT THE LAST NAME AND 82-YTD-ACT-% QUOTA OF ALL
SALES PEOPLE WITH STATE = NEW YORK & 82-YTD-ACT-%
QUOTA < 100.00.

THE NUMBER OF RECORDS TO BE SEARCHED IS 60
                        1982
LAST                     YTD
NAME            % OF QUOTA

BAHN                   96.24
DYKES                  96.10
ELEY                   96.60
GRAYSON                97.42
```

Figur 8.2. *Tolkning av databasförfrågan i naturligt språk.*

Exemplet i figur 8.2 är hämtat från systemet Intellect, som finns tillgängligt bl.a. för IBM och DEC-datorer. I detta system kombineras följande metoder för analys av innebörden i inmatade frågor:

- konventionell syntaktisk analys med användning av olika lexikon,
- tillgång till beskrivningen av den aktuella databasen med namn på objekttyper och objektegenskaper,
- kännedom om de egenskapsvärden (t.ex. namn på personer) som finns för tillfället i databasen, och
- i sista hand används en s.k. klargörande dialog om systemet inte lyckas hitta en entydig tolkning av användarens fråga.

System av denna typ förutsätter att språktolkningen kan ske relativt ett klart avgränsat tillämpningsområde, vilket ju är fallet när det gäller frågor mot en given databas. Exempel på tillämpningar där man kan dra fördel av ett frågesystem som accepterar naturligt språk är för hantering av oplanerade frågor från tillfälliga användare, intelligenta hjälpssystem och naturligtvis för system som accepterar talad inmatning.

8.2.3 Talad in- och utmatning

I många sammanhang är kravet på användning av tangentbord för inmatning av instruktioner och data till ett system (och presentation av utdata på papper eller bildskärm) en besvärande faktor. Det kan gälla tillämpningar där:

- användaren måste kunna röra sig fritt;
- synsinnet behöver avlastas;
- kommunikationen med datorsystemet kan endast ske via en vanlig telefonanknytning utan datautrustning;
- användaren har svårigheter att tillräckligt effektivt och tillförlitligt skriva korrekta kommandon och tolka textmeddelanden.

En talare är normalt dubbelt så snabb som en skicklig maskinskrivare, varför talad inmatning potentiellt kan ge stora vinster. Dagens teknik kräver dock klara pauser mellan orden, varför en sådan uppsnabbning är svår att praktiskt realisera. Man väntar dock att de första talstyrda skrivmaskinerna, s.k. "*talkwriters*", ska kunna börja testas under någorlunda realistiska förhållanden under de närmaste åren. Ett sådant system tar emot diktamen och genererar skriven text. Forskningssystem av denna typ har demonstrerats av bl.a. IBM. Kurzweil Applied Intelligence har utvecklat ett system som enligt uppgift är baserat på ett lexikon med 10.000 ord och alltså borde kunna få ganska intressanta prestanda. Under överskådlig tid torde dock system av denna typ dels vara relativt långsamma och dels ge relativt hög felfrekvens, exempelvis när det gäller namn som inte finns i lexikon.

Talanalys och talsyntes handlar i praktiken om text-till-tal omvandling, dvs vi förutsätter inte som vid behandling av naturligt språk att systemet i någon mening ska "förstå" innehållet i ett yttrande. Som vi såg när det gäller talanalys för en "talkwriter" krävs dock kunskaper i form av ett lexikon och det visar sig att för att uppnå riktigt goda prestanda krävs samverkan mellan olika kunskapskällor t.ex. även från syntaktisk och semantisk analys av språket. Riktigt intressant blir givetvis denna teknik när man kan genomföra hela analysen från talat språk till en intern tolkning av yttrandets betydelse med en snabbhet som är jämförbar med människans. Forskningsprojekt med denna målsättning pågår f.n., bl.a. i England.

8.2.4 Textgenerering vid analys av stora datamängder.

Textgenerering från intern representation är i princip ett enklare problem än språkanalys. Den svåra delen ligger närmast i valet av vad som ska uttryckas, än i valet av språklig utformning. Som ett exempel på ett lyckat projekt inom detta område kan vi ta det medicinska expertsystemet *PUFF*. (Aikins, Stanford.) Detta system producerar tolkningar i klartext av mätvärden och signaler från ett system för övervakning av patienter med lungproblem.

Detta system kan ses som en effektiv illustration av möjligheten att ur stora siffermaterial härleda en presentation i textform som framhäver intressanta

```
PRESBYTERIAN HOSPITAL OF PMC
 CLAY AND BUCHANAN, BOX 7999
 SAN FRANCISCO, CA. 94128
  PULMONARY FUNCTION LAB

WT 48.8 KG,  HT 161 CM,  AGE 69   SEX F
REFERRAL DX-
**********************************************************TEST DATE 05/13/88
                                PREDICTED                          POST DILATION
                                (+/-SD)    OBSER(%PRED)            OBSER(%PRED)
INSPIR VITAL CAP (IVC)  L       2.7        2.3    ( 86)            2.4    ( 98)
RESIDUAL VOL     (RV)   L       2.0        3.8    (188)            3.0    (148)
TOTAL LUNG CAP   (TLC)  L       4.7        6.1    (130)            5.4    (115)
RV/TLC                  %       43.        62.                     56.

FORCED EXPIR VOL (FEV1) L       2.2        1.5    ( 68)            1.6    ( 73)
FORCED VITAL CAP (FVC)  L       2.7        2.3    ( 86)            2.4    ( 90)
FEV1/FVC                %       73.        65.                     67.
PEAK EXPIR FLOW  (PEF)  L/S     7.1        1.0    ( 25)            1.9    ( 26)
FORCED EXP FLOW 25-75% L/S      1.0        0.7    ( 30)            0.7    ( 39)
AIRWAY RESIST(RAW) (TLC= 6.1)   0.0(0.0)   1.5                     2.2

DF CAP-HGB=14.5    (TLC= 4.0)   24.        17.4   ( 72)  ( 74%IF TLC = 4.7)
**********************************************************************

INTERPRETATION: ELEVATED LUNG VOLUMES INDICATE OVERINFLATION.  IN ADDITION, THE
RV/TLC RATIO IS INCREASED, SUGGESTING A MODERATELY SEVERE DEGREE OF AIR TRAPPING.
THE FORCED VITAL CAPACITY IS NORMAL.  THE FEV1/FVC RATIO AND MID-EXPIRATORY FLOW
ARE REDUCED AND THE AIRWAY RESISTANCE IS INCREASED, SUGGESTING MODERATELY SEVERE
AIRWAY OBSTRUCTION.  FOLLOWING BRONCHODILATION, THE EXPIRED FLOWS SHOW MODERATE
IMPROVEMENT.  HOWEVER, THE RESISTANCE DID NOT IMPROVE.  THE LOW DIFFUSING
CAPACITY INDICATES A LOSS OF ALVEOLAR CAPILLARY SURFACE, WHICH IS MILD.

CONCLUSIONS: THE LOW DIFFUSING CAPACITY, IN COMBINATION WITH OBSTRUCTION AND A
HIGH TOTAL LUNG CAPACITY IS CONSISTENT WITH A DIAGNOSIS OF EMPHYSEMA.  ALTHOUGH
BRONCHODILATORS WERE ONLY SLIGHTLY USEFUL IN THIS ONE CASE, PROLONGED USE MAY
PROVE TO BE BENEFICIAL TO THE PATIENT.

PULMONARY FUNCTION DIAGNOSIS:
1.  MODERATELY SEVERE OBSTRUCTIVE AIRWAYS DISEASE.
      EMPHYSEMATOUS TYPE.
```

Figur 8.3. *Automatiskt genererat journalutlåtande*

observationer och olika slags avvikelser från ett "normaltillstånd". Programmet baseras alltså på ett antal tolkningsregler, som automatiskt producerar en sammanfattande text med relevanta observationer till patientjornalen. Det är helt klart att en liknande typ av presentationsteknik bör kunna vara ett värdefullt komplement till tabeller och diagram även inom det administrativa området, exempelvis i ekonomitillämpningar när det gäller fortlöpande bevakning av utvecklingen för företag och värdepapper eller för att på ett mera begripligt sätt presentera exempelvis komplicerade redovisningar, etc.

8.2.5 Automatisk klassificering av textmeddelanden.

Ett snabbt ökande problem i det informationsteknologiska kommunikationssamhället är svårigheten att överblicka det allt mer omfattande utbudet av information och att identifiera och prioritera betydelsefulla dokument och meddelanden. Ofta tvingas man utse särskilda personer, som får till uppgift att snabbt klassificera, värdera, filtrera och vidarebefordra inkommande meddelanden. Exempel på situationer av denna typ är genomgång av referat av

tidskrifter, böcker, m.m., hantering av förfrågningar och felrapporter, övervakning av nyhetsmeddelanden, telex, datorförmedlad post, osv. Traditionella mekanismer för klassificering och prioritering baserade på layout, typografi, medium, osv fungerar inte för sådan information som flyter runt i ett datanätnät i standardiserat format och skrivs ut på en terminal eller lokal skrivare.

Flera kommersiella produkter som syftar till att hantera detta problem har kommit fram under senare tid. *Cognitive Systems* har utvecklat Atrans för analys av exempelvis bank-telex, dvs korta meddelanden som utmärks av bl.a. användning av specialiserad terminologi, förkortningar, och ett språk som inte alltid är syntaktiskt korrekt. Uppgiften är att översätta fri text (från aktuellt språk) till en standardiserad form som kan behandlas med ett minimum av manuella insatser.

Ett annat system som utvecklats för att bistå i sådana situationer är TESS (*Text extraction and Summary System*) från Carnegie Group. Detta system är tänkt att kunna användas för att behandla exempelvis nyhetstelegram, underrättelseinformation, vidaresändning av (datorförmedlade) ärenden, telex (t.ex. i banker), juridiska texter, medicinsk litteratur och flerspråkiga sammanfattningar. En förutsättning för teknikens applicerbarhet är att:

- texten som ska klassificeras och summeras finns tillgänglig i maskinläsbar form.
- den typ av information som ska extraheras kan specificeras i förväg.
- meddelandetexterna ska kunna hänföras till någon av ett antal väldefinierade kategorier.

Ett sådant system arbetar med en kombination av nyckelordsigenkänning, syntaktisk analys och sannolikhetsteoretiska metoder. Utöver den inbyggda förmågan att hantera (det engelska) språkets syntax, krävs för varje ny tillämpning en kunskapsbas som innehåller bl.a. hierarkiskt uppbyggda begreppsdefinitioner. Med dagens teknik klarar man bl.a. följande:

- Klassificering och summering av nyhetstelegram inom specifika områden.
- Klassificering och extraktion av faktauppgifter ur banktelex (på engelska) med automatisk konvertering till SWIFT-koder eller annat fast format för automatisk bearbetning.
- Vidaresändning av meddelanden inom ett givet område i en organisation (företag, myndighet, etc.) baserat på innehåll.
- Klassificering av fel- eller problemrapporter, t.ex. för datorsystem, samt sammanställning av alla rapporter som berör en viss aspekt på systemet.
- Produktion av flerspråkiga sammanfattningar från engelskspråkiga nyhetstelegram eller telexmeddelanden.

System av denna typ kan för de flesta tillämpningar i dagens läge knappast

uppvisa samma precision som en noggrann människa och är därmed naturligtvis mest intressanta inom områden där man inte hinner (eller har råd) att manuellt behandla alla texter eller där kravet på precision är måttligt. (Exempelvis i samband med prioritering av olika ärendens angelägenhetsgrad.)

8.2.6 Sammanfattande diskussion

Användning av naturligt språk som metod för in och utmatning blir givetvis särskilt intressant när man kan kombinera språkförståelse med effektiv användning av tal-till-text analys. Ett annat viktigt område när det gäller att minska beroendet av användarens förmåga att snabbt och felfritt skriva in respektive tolka meddelanden i textform är användning av grafisk teknik med bilder och symboler. Moderna arbetsstationer, av den typ som allmänt används för avancerat arbete med kunskapsbaserade system, utmärks bl.a. av omfattande användning av grafisk presentationsteknik med hög upplösning, multipla "fönster" för parallella aktiviteter, samt användning av s.k. ikoner, dvs enkla symboliska bilder för att referera till olika objekt och kommandon i systemet. Sådan teknik ger möjligheter att väsentligt minska den mentala belastningen på användaren under en dialog och även att underlätta för operatörer som har funktionella läs- och skriv-svårigheter.

9.

Avslutande diskussion

Vi ska i detta avslutande kapitel, dels summera dagens tekniska realiteter när det gäller utveckling av kunskapsbaserade system, dels diskutera ett antal problemställningar som aktualiseras av denna teknik.

9.1 State of the art.

När det gäller utveckling av kunskapsbaserade expertsystem kan vi kort sammanfatta vad som i dag är läget i följande punkter:

1. Inom avgränsade tillämpningsområden kan vi i vissa fall utveckla problemlösande eller rådgivande system som presterar resultat i nivå med de bästa experterna inom området.
2. För vissa klasser av typiska problem, t.ex. diagnosticering ("strukturerade val"), har vi färdiga tekniker för problemlösning och utvecklingsverktyg som väsentligt underlättar konstruktion av kunskapssystem inom nya tillämpningsområden.
3. Arbetsgång och metodik för systematisk uppbyggnad av kunskapsbaser och kunskapssystem är fortfarande ofullständigt kända. Vissa, om än begränsade, hjälpmedel för att exempelvis härleda generella regler från exempel med hjälp av induktiva metoder finns.
4. I viss utsträckning finns fungerande metoder för att genomföra s.k. plausibla resonemang, dvs slutsatsdragning under antaganden om osäkerhet i tillgängliga data eller i de regler som ligger till grund för resonemanget. Det är dock ofta förenat med betydande svårigheter att få områdesexperter och slutanvändare att uttrycka konsistenta och tillräckligt precisa mått på denna osäkerhet.
5. Förklaringar av resonemang och resultat kan i vissa system genereras automatiskt. Dessa förklaringar är dock tämligen stereotypa och ibland av begränsat värde.
6. Med få undantag krävs i dag att kunskapen förmedlas och underhålls av människor. Ett expertsystem kan alltså normalt inte självständigt förbättra sina prestationer genom att lära av sin egen "erfarenhet".
7. Systemen har ingen naturlig kunskap om sina egna begränsningar, dvs användaren måste ta ansvaret för att ett systems kompetens inte är

föråldrad eller inte tillämpbar på den aktuella situationen.

Beroende på hur man definierar och avgränsar begreppet expertsystem kan man få varierande uppskattningar av hur många lyckade tillämpningar inom området som hittills rapporterats. Man notera att seriösa bedömare anser att det fortfarande rör sig om färre än hundra kända kvalificerade expertsystem, som går i rutinmässig drift eller i fälttester [BUC86]. Det kan tyckas vara ett förvånansvärt litet antal med hänsyn till den stora uppmärksamhet som ägnas området. Man kan dock notera att det knappast finns några kända fall där ett företag eller en organisation utvärderat tekniken och inte funnit den värd ett fortsatt studium eller vidareutveckling. Det är dessutom känt att många företag, som inte själva är leverantörer av informationsteknologi, har föredragit att avstå från att informera externt om sina projekt och system inom området.

Troligen kommer ett allmänt genombrott för praktisk användning av kunskapssystem att behöva invänta ett bredare utbud av personer med en gedigen utbildning inom området, en beredskap att acceptera radikalt nya system och tekniker i företagens dataorganisationer, ett ytterligare prisfall på utrustning och programvara, ökad insikt om hur kunskapssystemen kan integreras med befintliga databaser och applikationer, samt ytterligare kunskapsuppbyggnad runt utvecklingsmetodik och konsekvenser av nya sätt att använda informationsteknologi i arbetslivet. Eftersom dessa förutsättningar kan förväntas snart vara uppfyllda, verkar det nu vara rätt tid att förbereda sig för de nya möjligheter som kommer.

9.2 Diskussionspunkter.

Kunskapssystemen aktualiserar en rad problemställningar, som vi behöver tänka över och ta ställning till. En del av dessa är specifika för området medan andra ligger implicit i all användning av informationsteknologi eller teknik över huvud taget.

9.2.1 Ansvarsfrågor.

Det är svårt att se att någon annan än användaren av ett kunskapssystem skulle vara den som också får ta ansvaret för följderna av ett beslut baserat på en systemkonsultation. Det är redan gängse praxis att leverantörer av generell programvara frånsäger sig allt ansvar för följder som kan uppstå på grund av fel i systemet eller dess användning. Vi bör väl närmast jämföra användning av ett kunskapsbaserat konsultationssystem med rådfrågning av handböcker eller mänskliga experter, där tilltron till erhållna uppgifter ytterst måste baseras på förtroendet för den aktuella källan. Här kan vi dock notera att i en del tillämpningar kan vi förvänta oss att en enskild handläggares konsultation med en expert tjänar just syftet att övervältra ansvaret för ett beslut på denne. I sådana tillämpningar är det alltså knappast meningsfullt att ersätta experten med ett kunskapssystem, såvida inte systemet i någon mening är "auktoriserat" som beslutsfattare i organisationen.

I [YAZ84] diskuteras bl.a. den intressanta frågan om datorer kan ha rättigheter och skyldigheter på liknande sätt som människor. Ett tänkvärt exempel behandlar en situation där ett system anförtrotts en mängd resurser för någon eller några människors räkning, exempelvis ett system som övervakar patienter på en intensivvårdsavdelning. För ett sådant system skulle man kunna tänka sig att systemet har rätt att kräva personalens uppmärksamhet vid behov eller att ge order om åtgärder, som inte bör ifrågasättas. Givetvis är detta synsätts giltighet en öppen fråga.

9.2.2 Informella regelsystem.

I många tillämpningar baserade på komplicerade regelsystem visar det sig att de egentliga svårigheterna kommer sig av bristande precision i reglerna eller av att det finns ett informellt regelsystem, som inte nödvändigtvis överensstämmer med det officiella. I det senare fallet hamnar man ofta i svårigheter om man försöker reproducera expertens problemlösningsförmåga inom området i ett kunskapssystem. Detta beror på att man i så fall måste göra de informella reglerna explicita, vilket i sin tur leder till att de kan få en icke avsedd officiell status. Ett exempel på en sådan situation är bedömningar av när man kan ta sig friheten att överträda en officiell regel. Man är alltså ofta inte beredd att legitimera praxis genom att registrera den i ett datoriserat system, och därmed faller också möjligheten att ersätta den direkta konsultationen med experten.

9.2.3 Transparent problemlösning

En central princip i kunskapssystemen är att kunskaper och problemlösningsmetoder ska vara transparenta, dvs inspekterbara och begripliga för en användare. Tyvärr leder detta ibland till konflikter. Det är inte alltid ekonomiskt rimligt att i driftsmiljö använda en problemlösningsmetod, som har denna egenskap. Det kan alltså krävas att vi i stället använder en optimerad algoritm som är effektiv ut teknisk synpunkt men svår att förstå för en människa. Vidare visar det sig i många tillämpningar att leverantören av kunskapsbasen inte accepterar att en användare kan ta del av alla uppgifter och information som den innehåller. Bakgrunden kan vara konkurrensskäl, omsorg om uppgifter av känslig natur, önskemål att skydda beslutskriterier (taxeringsmyndigheten kan ha ett system för att avgöra vilka avdrag som ska släppas igenom, där man givetvis inte vill avslöja sina principer), eller att systemet innehåller inofficiella regler, som man inte vill legitimera. Det visar sig också att vissa system av konsultationstyp, som under utvecklingsfasen uppvisar goda egenskaper när det gäller att redovisa genomförandet av ett resonemang, blir mindre transparenta vid övergång till ett driftsystem där flertalet uppgifter kanske hämtas direkt från databaser eller andra informationssystem utan mänskligt ingripande.

9.2.4 Behövs experten?

En ofta ställd fråga är hur det går med experterna när vi väl har representerat deras kunskaper i ett expertsystem. Kommer de att göras överflödiga i framtiden? Allmänt gäller givetvis att ny teknik kan göra specilistkunskaper umbärliga. Förmågan att minnas långa berättelser blev mindre värd när skrivkonsten uppfanns, en vacker handstil blev ointressant när skrivmaskiner infördes, snabbhet i aritmetiska beräkningar förlorade sitt värde när räknemaskiner togs i bruk, osv. När det gäller sådant som vi idag betraktar som expertkunskaper finns det givetvis inslag, som kan förväntas bli mindre värdefulla när vi får datorstödda kunskapsssystem som klarar motsvarande uppgifter. Vi bör dock komma ihåg att kunskapssystemen, i varje fall inom en överskådlig framtid, kommer att behandla mycket snävt avgränsade områden och inte på något sätt kan tävla med en kvalificerad mänsklig expert när det gäller bredd, allsidighet och anpassningsförmåga. Möjligen kan vi i stället vänta oss att expertrollen renodlas och koncentreras till uppgiften att utveckla, sammanställa och kommunicera kunskap, samt att övervaka kunskapens användning bl.a. för att avgöra under vilka förhållanden den inte längre är tillämplig.

9.2.5 Risken för kunskapserosion.

Ett välkänt problem vid automatisering av arbetsuppgifter är att manuella färdigheter och kunskaper tenderar att gå förlorade, när vi inte längre själva behöver utföra arbetet. Detta problem är särskilt allvarligt i samband med kunskapssystem, eftersom vi sällan kan räkna med att kunskaperna är särskilt långlivade och att vi kan fortsätta att lösa ett visst givet problem på ett enahanda sätt för all framtid. Det är alltså viktigt att experter och andra berörda fortlöpande underhåller och vidareutvecklar sin kompetens, speciellt som vi inte i dag har några generellt fungerande metoder för expertsystem att själva förbättra sin prestationsförmåga. Frågan om hur detta vidmakthållande av en hög kompetensnivå kan åstadkommas är värd att uppmärksammas, men vi bör också notera att jämfört med det gängse sättet att använda datateknik för informationssystem, erbjuder kunskapssystemen utomordentliga fördelar när det gäller att kommunicera underliggande kunskaper och probelmlösningsförmåga till en användare.

9.3 Slutord.

Lekmannen som i dag får höra talas om möjligheten att introducera expertsystem inom olika yrkesområden reagerar ofta med en oro för en framtid där människan alltmer reduceras till en underordnad faktor i det datoriserade produktionssystemet. Det finns dock all anledning att se tekniken med kunskapssystem som en möjlighet att bättre ta till vara människans unika förutsättningar genom minskade krav på anpassning till stela metoder för datorbaserad problemlösning. Expertsystem, så som de kan realiseras i dag, erbjuder möjligheter att förpacka och kommunicera specialistkompetens med

hjälp av datateknik och ger därmed en möjlighet att bredda basen för en enskild människas problemlösning och beslutsfattande, exempelvis genom att minska kravet på specialisering och hård uppdelning av arbetsuppgifter.

Referenser och rekommenderad läsning:

[BAR81] **Barr, A., Feigenbaum, E.A.,** *The Handbook of Artificial Intelligence, vol 1, & 2*, W. Kaufmann, Calif., 1981, 1982.
Cohen P.R., Feigenbaum, E.A., *The Handbook of Artificial Intelligence, vol 3*, W. Kaufmann, Calif., 1982.

[BUC86] **Buchanan, B.G.,.,** Expert systems: working systems and the research litterature. *Expert Systems, vol 3*, no 1, Jan. 1986.

[BOO85] **Boose, J.H.,** A knowledge acquisition program for expert systems based on personal construct theory, *Int. J. Man-Machine Studies, vol 23*, pp 495-525, 1985.

[COO81] **Cook, S. et al.,** The applications of artificial intelligence to law: A survey of six current projects. *Proc. AFIPS National Computer Conf.*, 1981.

[FEI83] **Feigenbaum, E.A., McCorduck, P.,** *The Fifth Generation. Artificial Intelligence and Japan's Computer Challenge to the World*, Addison-Wesley, 1983.

[FIN83] **Finin, T.W.,** Providing Help and Advice in Task Oriented Systems, *Proc. IJCAI-83*, 1983.

[GRI84] **Griesmer, J.H.,** et al., YES/MVS: A Continuous Real Time Expert System. *Proc. AAAI-84*, 1984.

[HAG85] **Hägglund, S.,** Feature Catalogue of Tools for Building Expert Systems. Draft version. ASLAB Memo 85-07, IDA, Universitetet i Linköping, 1985.

[HAG86] **Hägglund, S., Grunditz, H.,** Kunskapsbaserade expertsystem, Mekanresultat 86001, Sv. Mekanförbund, 1986.

[HAR85] **Harmon, P., King, D.,** *Expert systems.* Wiley, 1985.

[HAR86] **Hart, A.,** *Knowledge Acquisition for Expert systems.* Kogan Page, 1986.

[HEI83] **Hein, U., Wikström, T.,** Artificiell Intelligens: Vad datorn kan och inte kan - ännu. *Forskning och Framsteg*, no 7, 1983.

[HJE86] **Hjerppe, R.,** Electronic Publishing: Writing Machines and Machine Writings,

[HOP84] **Hopkin, V.D.,** Some human-factors implications of expert systems, *Behaviour and Information Tecchnology, vol 3*, no 1, pp 79-83, 1984.

[IDA85] **Sandewall, E., et al.,** *IDA Annual Research Report 1985, Inst. f.* datavetenskap, Universitetet i Linköping, 1986.

[JAC86] **Jackson, P.,** *Introduction to Expert Systems*, Addison-Wesly, 1986.

[JON85] **Jones, M.J., Crates, D.T.,** Expert systems and videotex: An application in the marketing of agrochemicals., in Bramer (ed.) *Research and Development of Expert Systems*, Cambridge University Press, 1985.

[KEL55] **Kelly G.A.,** *The Psychology of Personal Constructs*, New York: Norton, 1955.

[MIC80] **Michalski, R.S., Chilausky, R.L,** Knowledge acquisition by encoding expert rules versus induction from examples: a case study involving soybean pathology. *Int. J. Man-Machine Studies, vol 12*, pp 63-87, 1980.

[MIC83] **Michaelsen, R., Michie, D.,** Expert Systems in Business, *Datamation*, Nov, 1983.

[MOE84] **Moen, S.,** En implementering av Expert Ease under Interlisp-D. ASLAB Memo 84-04, IDA, Universitetet i Linköping, 1984.

[MOH85] **Molich, R., Anker-Möller, B.,** Intelligente hjaelpesystemer, *Proc. NordDATA 85*, Köpenhamn, 1985.

[NOR85] **Nordin, H.,** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, Lic. Thesis LiU-Tek-Lic 1985:13, IDA, Universitetet i Linköping, 1985.

[REI84] **Reitman, W (ed.),** *Artificial Intelligence for Applications in Business*, Ablex, 1984.

[ROY85] **Roycroft, A.E., Loucopoulos, P.,**The Development of an Expert Tax System, in *Proc of the 5th Int. Conf. on Expert Systems and their Applications*, Avignon, 1985.

[SAN82] **Sandewall, E.,** Ny teknologi i kontorsdatasystem, *Proc. NordDATA 82*, Göteborg 1982.

[SER86] **Sergot, M.J., et al.,** The British Nationality Act as a Logic Program, *Comm. ACM. vol 29*, no 5, May 1986.

[TIM86] **Timpka, T. et al.,** Decision Support for General Practitioners; Design and Implementation by Integrating Paradigms: Hypertext, Knowledge Based Systems and Online Library. Avd. f. medicinsk informationsbeh., Universitetet i Linköping, 1986.

[TUR84] **Turner, R.,** *Logics for Artificial Intelligence*, Ellis Horwood, Chichester, 1984.

[WAT86] **Waterman, D.A.,** *A Guide to Expert Systems*, Addison-Wesley, 1986.

[WIN84] **Winston, P.H., Prendergast, K.A. (eds.),** *The AI Business. Commercial Uses of Artificial Intelligence*, The MIT Press, (1984).

[YAZ84] **Yazdani, M., Narayanan, A.,** (eds.) *Artificial Intelligence and Human Effects*, Ellis Horwood ltd, Chichester, 1984.

Appendix A
Ordlista

I nedanstående översikt över termer har vi valt att utgå från de engelska terminologin, eftersom etablerade svenska översättningar ofta saknas. Avsikten är att ordlistan även ska tjäna som en översikt över centrala begrepp inom området och som ett stöd vid läsning av engelsk-språkig AI-litteratur.

active value — *Aktivt värde.* En speciell typ av värde vars förändringar under en konsultation initierar anrop av en procedur. Detta kan ske när värdet läses eller ändras, t.ex. för användning i samband med grafisk interaktionsteknik.

AI — Förkortning för *artificial intelligence.*

algorithm — *Algoritm.* En beräkningsmetod som för varje uppsättning indata definierar en ändlig sekvens av operationer som leder fram till det önskade resultatet.

antecedent — *Antecedent, premiss.* Förutsättningsledet (OM-delen) i en produktionsregel.

artificial intelligence — *Artificiell intelligens, konstgjord intelligens.* Ett delområde av datavetenskapen som studerar möjligheterna att konstruera system, som uppvisar egenskaper vilka normalt förknippas med intelligent beteende, exempelvis förmåga till problemlösning, kommunikation i naturligt språk, inlärning från erfarenheter, tolkning av visuell information, etc.

attribute — *Attribut, egenskap.* En typ av information som används för att karakterisera objekt, t.ex. *färg, ålder*, etc.

backtracking — Processen att återta tidigare gjorda antaganden under ett resonemang och därvid återställa alla förändringar som gjorts på vägen. Används som teknik för att söka igenom ett problemlösningsträd, varvid man provar gren efter gren tills man hittar en som leder till målet.

backward chaining — *Målstyrd slutsatsdragning* Problemlösningsteknik karakteriserad av att hypoteser verifieras genom att man startar med den slutsats som man försöker bevisa och sedan arbetar bakåt mot kända fakta.

blackboard — En databas som används av flera oberoende processer (t.ex. motsvarande olika kunskapskällor) för att kommunicera information i en samordnad problemlösning.

breadth-first search — *Bredden-först sökning.* En sökstrategi för en regel- eller objekt-hierarki som baseras på att man först beaktar alla element på närmast lägre nivå innan man går djupare ner i trädet.

CAI — *Datorstödd undervisning.* I detta sammanhang vanligen ICAI (*Intelligent Computer-Aided Instruction*).

certainty factor — *Visshets-faktor.* Refereras ibland som CF-värde. Ett numeriskt mått (i intervallet -1 till 1) som används för att uttrycka en subjectiv bedömning av tilltron till en regel eller en utsaga. Avser vanligen den teknik som bl.a. används i MYCIN, EMYCIN baserad på Shortliffe's *uncertainty theory.*

Common Lisp — En dialekt av Lisp som förväntas ha förutsättningar att bli standard. Implementeringar finns på stordatorer, arbetsstationer och PC.

concept — *Begrepp.* Används ibland som en samlande beteckning för konkreta och

abstrakta objekt och termer, som ska beskrivas och representeras i ett kunskapssystem, t.ex. med hjälp av *frames*.

conclusion *Slutsats.* Slutsats- eller åtgärds-ledet (SÅ-delen) i en produktionsregel.

conflict resolution Avser det problem som uppstår i ett regelbaserat system när flera regler samtidigt är applicerbara. Vanligen har man då någon form av prioriteringsteknik som får avgöra i vilken ordning reglerna ska appliceras.

consequent *konsekvens, slutsats.* Slutsats- eller åtgärds-ledet (SÅ-delen) i en produktionsregel.

data driven *Datadriven.* Betecknar allmänt en strategi där styrningen av ett systems arbete initieras av de fakta som är kända i varje givet ögonblick. I samband med regelbaserad programstyrning avses detsamma som framåt-riktad (*forward chaining*) slutsatsdragning.

database *Databas.* En datorlagrad informationsmängd, som vanligen är permanentlagrad på sekundärminne och kan bearbetas av olika program. (Jfr *knowledge base.*)

declarative *Deklarativ.* En programorganisationsteknik som baseras på att man anger vilket resultat som önskas, snarare än hur det steg för steg ska beräknas. Motsats *procedural.*

deduction *Deduktion, härledning.* Härledning av slutsatser från förutsättningar, som antas vara sanna, genom användning av ett logiskt slutledningssystem.

default Ett värde som antages gälla så länge inte något annat explicit angivits.

delivery system *Leveranssystem, driftssystem.* Den maskin- och program-vara som används för ett kunskapssystem i reguljär rutinmässing användning.

demon *Databasprocedur, trigger.* En procedur som aktiveras automatiskt när en viss händelse inträffar, t.ex. vid lagring, läsning eller ändring av ett värde på ett givet element i en databas.

domain *Domän, kunskapsområde.* Vanligen avses den samlade kunskapen, terminologi, erfarenheter, m.m. inom den tillämpning för vilken man utvecklar ett kunskapssystem. Ibland har termen den snävare betydelsen *definitionsområde* (t.ex. för en variabel).

EMYCIN Tillämpningsoberoende verktyg för implementering av expertsystem för konsultativ rådgivning, speciellt för diagnos och "strukturerade val". Härlett ur MYCIN.

expert system *Expertsystem.* (Ursprungligen) ett programsystem som inom en given tillämpning presterar resultat på en nivå jämförbar med mänskliga experter. Ofta används termen om all programvara som utvecklats med teknik inspirerad av AI-forskning eller med de vertyg som tagits fram för ändamålet.

explanation *Förklaring.* Denna funktion i ett expertsystem tillhandahåller information som motiverar och belyser resonemang och slutsatser. Vanliga funktioner är s.k. *varför-* och *hur*-frågor, som används för att redovisa den aktuella kedjan av slutsatser som dragits.

forward-chaining *framåt-riktad slutsatsdragning.* Problemlösningsteknik karakteriserad av att hypoteser verifieras genom att man startar med kända fakta och sedan arbetar framåt genom att dra slutsatser från dessa.

frame En struktur som används för att beskriva och representera objekt, dess fasta egenskaper såväl som definitioner av hur objektet reagerar vid olika aktiveringar och i olika situationer. Egenskaperna knyts ofta till

attribut (*slots, aspects*), där man kan lagra *values, defaults, active values, rules, inheritance*, etc.)

fuzzy logic *Oskarp logik.* Ett logiskt system där sanningsvärdena "sant" och "falskt" ersatts med en "graderad" sanning. Används som en bas för approximativa resonemang (resonemang under osäkerhet).

generate and test Problemlösningsteknik baserad på en generator som producerar möjliga lösningar och en evaluator som testar om lösningen är korrekt.

goal *Mål.* Det sökta resultatet vid problemlösning, representerat av det tillstånd i sökrymden (*search space*) som uppfyller alla kriterier på en korrekt lösning. I ett regelbaserat system avses den utsaga vars sanningsvärde ska avgöras genom en slutsatsdragningsprocess (*inference*).

graphics *Grafik.* Användning av bilder och bildelement (oftast uppbyggda av punktmönster) för in- och ut-matning över en bildskärm.

heuristic *Heuristik.* En teknik för problemlösning som baseras på användning av någon intelligent strategi ("tumregler") för att begränsa sökrymden. Det kan dock inte alltid garanteras att man erhåller en lösning, i varje fall inte den optimala.

hypothetical worlds *Hypotetiska världar.* Möjligheten att i ett kunskapssystem konstruera och hantera olika alternativa situationer och utvecklingsalternativ.

ICAI *Intelligent datorstödd undervisning.* Exempelvis simuleringssystem där eleven lär sig genom aktivt arbete med problemlösning, varvid systemet kan föra en insiktsfull dialog runt den underliggande kunskapen.

induction system *Induktionssystem.* Ett system för att ur en mängd exempel härleda ett regelsystem (t.ex. ett beslutsträd) som stöd för beslutsfattande.

inference *Slutsatsdragning.* Den logiska process som ur fakta och regler härleder slutsatser. Ofta i sammansättningen *inference engine*, som avser den programmodul som utför slutsatsdragningen, eller *inference method*, som avser den teknik som därvid tillämpas, t.ex. framåt- eller bakåt-riktad slutsatsdragning.

inheritance *Ärvning (av egenskaper).* En process där objekt relaterade i någon form av hierarkisk struktur kan få egenskapsvärden härledda från överordnade, mer generella, objekt.

instantiation *Instantiering.* Den process genom vilken en ny individ av en given typ skapas, t.ex. ett individuellt objekt i ett *frame*-system.

intelligent frontend *Intelligent användargränssnitt.* Ett program som innehåller kunskap om en tillämpningsmiljö och ett eller flera tillgängliga applikationsprogram och som underlättar användning av dessa program genom att sköta dialogen med användaren.

interactive *Interaktiv.* Egenskapen hos ett system att användarens instruktioner utförs i direkt dialog med omedelbar återkoppling efter varje steg.

Interlisp En dialekt av Lisp, som framför allt är känd för sin väl utvecklade och avancerade programmeringsmiljö, speciellt för utveckling och underhåll av stora system. Interlisp-D innehåller dessutom integrerad hantering av fönster på högupplösningsskärm.

interpreter *Interpretator, tolkare.* Ett program (en programmodul) som tolkar satser i ett språk och utför motsvarande instruktioner. I ett

regelbaserat system har man normalt en interpretator som väljer och applicerar regler på aktuella fakta. En kompilator kan översätta programmet (regelmängden) till ett direkt exekverbart program, varvid interpretatorn inte längre behövs.

kernel *kärna.* En term som i sammanhanget ibland används för att beteckna de tillämpningsoberoende systemdelar som används som bas vid utveckling av ett nytt kunskapssystem.

knowledge acquisition
Kunskapsformulering, kunskapsinhämtning. Den fas i utvecklingen av ett expertsystem, där domänkunskaper formuleras på ett sådant sätt att en kunskapsbas för området kan ges struktur och fyllas med information.

knowledge base *Kunskapsbas, kunskapsdatabas.* Den del av ett kunskapssystem som innehåller domän- och problem-specifik information, fakta, regler, heuristik, m.m. Kunskapsbasen är separerad från programstyrning (*control*), slutsatsdragning (*inference*) och dialoghantering.

knowledge engineering/engineer
Kunskapsingenjör. Den person (med gedigen utbildning för uppgiften att bygga kunskapssystem), som hjälper domänexperten att formulera sin kunskap och därefter ansvarar för uppbyggnad och underhåll av kunskapsbasen.

knowledge representation
Kunskapsrepresentation. Den process eller den metod, genom vilken kunskapen inom ett område struktureras och lagras som en bas för datorbearbetning. Typiska metoder är *semantic networks, frames, predicate logic, production rules and object-attribute-value-triplets.*

knowledge system *Kunskapssystem.* En samlande beteckning som alltmer ersätter termerna kunskapsbaserade system, expertsystem och regel-baserade system. Ett program som använder datorlagrade "kunskapsmoduler" och någon form av generell slutsatsdragning för problemlösning inom ett avgränsat område.

learning *Inlärning.* Tillförande av ny kunskap till ett system, vars problemlösande förmåga därvid ökar. Vanligen avses en förmåga att lära från erfarenheter, t.ex. genom generalisering.

lips Antal logiska slutledningar (*inferences*) per sekund. Ett mått på effektiviteten i program- och maskin-vara när det gäller att utföra den elementära operationen vid logikprogrammering.

Lisp Ett programmeringsspråk för behandling av strukturerade symboliska data (*list processing*), utvecklat runt 1960. Det dominerande språket för forskning och tillämpningar inom AI-området.

list structure *Liststruktur.* En sekventiellt ordnad, länkad (rekursiv) struktur av objekt, som externt representeras med parentetiserade listor. Grundläggande komponent i språket Lisp.

logic programming *Logikprogrammering.* Metodik och tekniker för användning av programmeringsspråk baserade på (första ordningens) logik, t.ex. språket Prolog.

meta knowledge *Metakunskap.* Kunskap om kunskap, t.ex. insikter om när och hur en viss kunskap kan användas. Begreppet *meta rule* avser således en regel som uttalar sig om andra regler, snarare än om elementära objekt i domänen.

monotonic reasoning

Monotont resonemang. Ett system för logisk slutsatsdragning som bygger på att gjorda utsagor eller härledda slutsatser ej kan förändras under resonemangsprocessen, dvs att mängden (härledda) sanningar är monotont växande.

natural language *Naturligt språk.* Betyder i detta sammanhang bearbetning i dator av något icke-artificiellt språk såsom engelska eller svenska, t.ex. för användardialog eller textanalys. I praktiken hanteras en begränsad delmängd av språket, ofta knutet till ett visst tillämpningsområde.

non-monotonic reasoning
Icke-monotont resonemang. Ett logiskt system som tillåter att ny information kan leda till att tidigare dragna slutsatser blir ogiltiga. Betydelsefullt i expertsystem där uppgifter kan vara föränderliga under en konsultation, eller där man vill kunna dra slutsatser utan att initialt behöva kontrollera alla tänkbara undantag som kan påverka slutsatsen.

object-attribute-value-triplets
Objekt-attribut-värde-tripler. En metod för representation av fakta som baseras på att man beskriver ett givet objekt genom att ange värden för de olika egenskaper som karakteriserar objektet. Ett visst objekt definieras alltså av alla de tripler som hänför sig till detta objekt.

predicate logic *Predikatlogik.* (Även *predicate calculus, first order logic.*) En gren av den klassiska logiken som bl.a. baseras på användning av predikat-symboler för att uttrycka relationer mellan olika objekt. Formell grund för vissa programspråk, t.ex. Prolog.

premise *Premiss.* Förutsättningsledet (OM-delen) i en produktionsregel.

probability *Sannolikhet.* Ett statistiskt mått på hur trolig en viss händelse är under givna förutsättningar. Det matematiska begreppet sannolikhet har dock endast begränsad tillämpbarhet när det gäller att hantera resonemang under osäkerhet i kunskapssystem, varför alternativa modeller har utvecklats, t.ex. *uncertainty theory.*

problem solving *Problemlösning.* Den process genom vilken ett önskat sluttillstånd härleds ur en en given uppsättning förutsättningar genom successiv applicering av tillåtna tillståndsförändrande operationer. Utformas ofta i praktiken som en *sökning* i mängden av alla möjliga tillstånd.

procedural *Procedurell.* En programorganisatonsteknik som baseras på att man explicit, steg för steg, anger hur ett visst result ska erhållas. Motsats *declarative.*

production rule *Produktionsregel.* En regel (se *rule*) i ett produktionssystem (*production system*).

production system *Produktionssystem.* Ett system av regler där förutsättningsleden hela tiden jämförs med kända fakta, varefter slutsatsleden används för att modifiera dessa fakta så snart en given förutsättning uppfyllts. Utvecklades ursprungligen som en modell för vissa aspekter av levande organismers beteende.

programming environment
Programmeringsmiljö. Den totala uppsättning verktyg och hjälpmedel som understödjer utveckling, test, drift och underhåll av ett programvarusystem.

Prolog Ett programmeringsspråk baserat på predikatlogik. Används ofta för arbete inom AI och expertsystem, speciellt utanför USA.

prototype *Prototyp.* En implementerad version av ett system, som syftar till att testa lösningen, demonstrera egenskaper och tjäna som en grund för

initala tester.

pruning — *Beskärning.* En teknik för att begränsa sökrymden vid problemlösning, t.ex. det antal grenar i ett beslutsträd som behöver undersökas.

reasoning — *Resonemang.* Den process för slutsatsdragning som genomförs exempelvis genom att generella regler appliceras på kända fakta.

recognize-act — Den process i ett *production system* eller ett *forward-chaining* system, där först jämförelse med aktuella fakta i *working memory* görs och därefter någon regel som befunnits tillämplig får sin slutsatsdel evaluerad.

resolution — *Resolution.* En praktiskt användbar metod för den form av bevisföring som ligger till grund för härledning av slutsatser i logikprogrammeringssystem av typ Prolog.

rule — *Regel.* En formalism, vanligen av s.k. OM-SÅ-typ (*if-then*), för att uttrycka orsakssamband, direktiv, objektrelationer, mönsterigenkänning och andra former av slutsatsdragning. OM-ledet (premiss, villkor, antecedent) specificerar de förutsättningar som ska vara uppfylld för att regeln ska kunna tillämpas och SÅ-ledet (slutsats, åtgärd, konsekvens) vad som i så fall ska utföras.

search — *Sökning.* Betyder i detta sammanhang den process, genom vilken en lösning på ett givet problem härleds ur den aktuella sökrymden. Används som en generell strategi för problemlösning.

search space — *Sökrymd.* Mängden av alla tillstånd för ett givet system, som kan uppnås genom successiv applicering av tillåtna operationer.

semantic — *Semantisk.* Avser *betydelsen* hos ett uttryck. (Ofta i motsats till *syntaktisk*, som avser formen.)

semantic network — *Semantiskt nätverk.* En formalism för kunskapsrepresentation som baseras på en struktur av noder som representerar object (i vid mening) och namngivna bågar som definier objektens egenskaper och inbördes relationer.

shell — *Skal.* Ett programssystem som understödjer konstruktion och exekvering av expertsystem. Vanligen avses produkter som föreskriver en relativt standardiserad form för hur en viss kunskap kan representeras. Begreppet tool används ibland synonymt men indikerar oftast en mer fullständig utvecklingsmiljö med flexibla uttrycksmedel.

slot — De "fack" där man kan lagra egenskaper som beskriver ett objekt, såsom *values, defaults, active values, rules, inheritance*, etc. Vanligen i samband med användning av *frames.*

syntactic — *Syntaktisk.* Avser den *form* som ett uttryck kan ha. (Motsatsen *semantisk* avser uttryckets betydelse.)

taxonomi — *Taxonomi, begreppshierarki.* En struktur som visar hur olika termer och begrepp är relaterade till varandra, speciellt avseende successiva specialiseringar av ett begrepp. Utgör vanligen en bas för automatisk ärvning (*inheritance*) av egenskaper, genom att mer speciella (underordnade) begrepp implicit antages besitta alla mer generella (överordande) begrepps egenskaper.

tool — *Verktyg, redskap.* Programsystem som fungerar som ett hjälpmedel för att bygga upp och exekvera ett expertsystem. Ibland används termen *shell* synonymt.

uncertainty — *Osäkerhet.* Ett uttryck för den begränsade tilltro man kan ha till att ett visst faktum är sant eller att en given slutsats följer ur observerade

	fakta. Olika formella modeller, som kan tjäna som en praktisk grund för resonemang under osäkerhet i kunskapssystem, finns utvecklade, t.ex. *fuzzy logic*, *uncertainty theory*, etc.
user interface	*Användargränssnitt.* Den programdel (eller den teknik) som används för hantering av användarens dialog med systemet. Utgör ofta en volymsmässigt stor, för att inte säga dominerande, del av ett kunskapssystem.
value	*Värde.* De elementära data som lagras för ett objekts attribut för att beskriva dess egenskaper, t.ex attributet **färg** kan anta värdet **röd**, **grön**, **blå**, etc.
window	*Fönster.* Den sektion av en bildskärm som ur interaktionssynpunkt betraktas som en enhet av systemet. *AI workstations* har vanligen understöd för hantering av multipla fönster, i vilka användaren kan ha ett antal oberoende aktiviter igång samtidigt.
working memory	*Arbetsminne, korttidsminne.* Avser i ett expertsystem den mängd fakta som bearbetas under en konsultation, speciellt i samband med *production systems* där reglerna hela tiden matchas (jämförs) med data i arbetsminnet.
workstation	*Arbetsstation.* En datorenhet avsedd för en individuell användare, vanligen med mycket höga prestanda och ett användargränssnitt baserat på grafisk interaktionsteknik.

Teldok

Telestyrelsen har inrättat ett anslag med syfte att medverka till snabb och lättillgänglig dokumentation beträffande användningen av teleanknutna informationssystem. Detta anslag förvaltas av TELDOK och skall bidraga till:

Dokumentation vid tidigast möjliga tidpunkt av praktiska tillämpningar av teleanknutna informationssystem i arbetslivet

Publicering och spridning, i förekommande fall översättning, av annars svåråtkomliga erfarenheter av teleanknutna informationssystem i arbetslivet, samt kompletteringar avsedda att öka användningsvärdet för svenska förhållanden och svenska läsare

Studieresor och konferenser i direkt anknytning till arbetet med att dokumentera och sprida information beträffande praktiska tillämpningar av teleanknutna informationssystem i arbetslivet

Ytterligare information lämnas gärna av TELDOK Redaktionskommitté. Där ingår:

Bertil Thorngren (ordförande), televerket, 08-713 3077
Göran Axelsson, statsrådsberedningen, 08-763 4205
Birgitta Frejhagen, LO, 08-796 2500
Peter Magnusson, TCO (ST), 08-790 5100
Agneta Qwerin, SSI/statskontoret, 08-738 4862
Nils-Göran Svensson, Riksdataförbundet, 08-24 85 55
Bengt-Arne Vedin, KTH/SNS, 08-23 25 20
P G Holmlöv (sekreterare), televerket, 08-713 4131

Adress: TELDOK, KP, televerkets hk, 123 86 Farsta
Telefaxnummer: 08-64 46 72

Teldok

har utgivit:

TELDOK Rapport

1. Om kontorsautomation i USA. December 1981.
2. Telebild. Erfarenheter från näringslivets teledataförsök. December 1982.
3. ADB, telekommunikationer och juridiskt arbete. April 1983.
4. Meddelande att läsa. Datorbaserade textkommunikationssystem på sex svenska företag. Maj 1983.
5. Videokonferenser och tillämpningar av bredbandskommunikation i Nordamerika. September 1983.
6. The automated office. Med sammanfattning och några artiklar på svenska. November 1983.
7. Det framtida kontoret. Bearbetade föredrag från ett IVA-symposium. November 1983.
8. Kontorsautomation. Trender och tillämpningar i USA, Japan och Europa. December 1983.
9. Datakommunikation i företag. December 1983.
10. Telematik i Frankrike. Juni 1984.
11. Ny teleteknik - ny organisation? Juni 1984.
12. Telemöten i USA - en öppenhjärtig rapport. December 1984.
13. Persondatorer i USA. December 1984.
14. Informationsteknologi i Storbritannien. April 1985.
15. Datorbaserad information i småföretag. Juni 1985.
16. Grannskap 90. Närarbete på distans i informationssamhället. September 1985.
17. Datorisering i u-land. April 1986.
18. Kontorsautomation i USA. April 1986.
19. Digitalisering i Västtyskland. April 1986.
20. Telematik och organisationsstyrning. Rapport från ett kollokvium. Augusti 1986.
21. Telebild. Erfarenheter och slutsatser från tre års kommersiell videotexverksamhet. Augusti 1986.
22. Fjärrarkivering. Oktober 1986.
23. Stress och kontorsautomation. Oktober 1986.
24. Meddelanden att använda. November 1986.
25. Ny teleteknik i Sverige - användning i dag. November 1986.
26. Datorstödda kunskapssystem i framtidens kontor. December 1986.

TELDOK Referensdokument

A. Informationssystem på svenska kontor. Juni 1982.
B. Office automation in Europe. Februari 1983.
C. Office automation in Japan. Februari 1983.
D. Office automation and related technologies in Japan. Februari 1985.
E. Office automation in the US. Februari 1985.
F. Office automation in Europe. Oktober 1985.

TELDOK-Info

1. Talteknologi. November 1982.
2. Ett textnummer. Maj 1984.
3. Ett bredbandsnummer. Januari 1986.