COMPLEX
INSTITUTT FOR
RETTSINFORMATIKK

HENNING HERRESTAD

ALDERSTRYGDEN I ET NØTTESKALL ELLER SKALL, SKAL IKKE;

en studie av anvendelsen
av ekspertsystemskall for PC
innen det juridiske domene

NORIS (83)

F*KUS

TANO

10/88

Hovedkontor: Rosenholmvn. 25, 1410 Kolbotn, tlf. (02) 99 80 00
Bergen: Dreggsalmenningen 10/12, tlf. (05) 31 65 00
Stavanger: Auglendsdalen 81, tlf. (04) 58 85 00
Trondheim: Kongensgt. 60, tlf. (07) 53 06 14

Norsk forening for JUS og EDB

Postboks 7557, Skillebekk, OSLO 2
Postgiro 5 13 96 54 Bankgiro 8200 42 49727
Postgiro ComPlex 2 12 16 63

Foreningen står bl.a. for salget av ComPlex-heftene og vedlikeholder abonnements-ordningene for serien.

SamvirkendeDataSystemerSamvirkendeDataSystemer
DataSystemerSamvirkendeDataSystemerSamvirkende
SystemerSamvirkendeDataSystemerSamvirkendeData
SamvirkendeDataSystemerSamvirkendeDataSystemer
DataSystemerSamvirkendeDataSystemerSamvirkende
SystemerSamvirkendeDataSystemerSamvirkendeData
SamvirkendeDataSystemerSamvirkendeDataSystemer
DataSystemerSamvirkendeDataSystemerSamvirkende
SystemerSamvirkendeDataSystemerSamvirkendeData
SamvirkendeDataSystemerSamvirkendeDataSystemer
DataSystemerSamvirkendeDataSystemerSamvirkende
SystemerSamvirkendeDataSystemerSamvirkendeData
SamvirkendeDataSystemerSamvirkendeDataSystemer
DataSystemerSamvirkendeDataSystemerSamvirkende
SystemerSamvirkendeDataSystemerSamvirkendeData
SamvirkendeDataSystemerSamvirkendeDataSystemer
DataSystemerSamvirkendeDataSystemerSamvirkende
SystemerSamvirkendeDataSystemerSamvirkendeData

Samvirkende
Data
Systemer

Statens DataSentral a.s

CompLex nr. 10/88

Institutt for rettsinformatikk
Universitetet i Oslo
Niels Juels gate 16
0272 OSLO 2

Henning Herrestad

ALDERSTRYGDEN I ET NØTTESKALL, ELLER SKALL, SKAL IKKE;

En studie av anvendelsen av ekspertsystemskall for PC innen det juridiske domene

NORIS (83)
F*KUS

Prosjektet har vært støttet av
Norges Teknisk-Naturvitenskapelige Forskningsråd.

TANO

ISBN 82-518-2594-6
Utgivelser i skriftserien CompLex støttes av:

Den norske Advokatforening
Den norske Bankforening
Digital Equipment Corporation A/S
Ericsson Information Systems A/S
IDA, Integrert Databehandling a.s
Industriforbundets Servicekontor
K-link, Kreditkassen
Lovdata
Norges Forsikringsforbund
Norges Statsautoriserte Revisorers Forening
Televerket
Wikborg, Rein & Co

Fotografisk opptrykk ved
Engers Boktrykkeri A/S, Otta

INNHOLDSFORTEGNELSE

1. FORORD.

Denne rapporten er et resultat av arbeidet med F*KUS prosjektet ved Institutt for Rettsinformatikk. Det samlende konsept for F*KUS prosjektet er ideen om *forvaltningsorienterte kunnskaps-baserte systemer.* Som et skritt på veien mot å lage slike systemer fant vi det ønskelig å prøve ut en type kommersielt tilgjengelige programmeringsverktøy kalt "ekspertsystemskall". Dette er verktøy som er ment å gjøre det enkelt å lage regelbaserte programmer slik man kjenner dem fra de såkalte "ekspert-systemer". Ekspertsystemer skal ifølge omtalen ha lagret alle de reglene og faktaene en ekspert bruker for å løse et problem innenfor sitt domene. Systemet bruker denne kunnskapen til å etterligne ekspertens resonnering når han løser problemer. De kalles derfor også "kunnskapsbaserte systemer" og de regnes som produkter av forskningen innen "kunstig intelligens". Et ekspertsystemskall blir gjerne beskrevet som et ekspertsystem med en tom kunnskapsbase hvor programmereren kan legge inn ny kunnskap om det domene han ønsker å automatisere.

Ekspertsystemskallene har etterhvert kommet i mange varianter. Ut i fra hensyn til Instituttets daværende maskinutrustning ble det bestemt å kjøpe inn ekspertsystemskall for PC, og valget falt på "Personal Consultant Easy" (PC Easy) fra Texas Instruments og "Crystal" fra Intelligent Environment Limited[1]. På grunnlag av anbefalinger vi fikk ble hoveddelen av arbeidet utført med Crystal. PC Easy ble bare brukt som sammenligningsgrunnlag, og slik blir det også sporadisk brukt i denne rapporten. For ytterligere å kunne sette vårt program og det skallet vi prøvet ut i perspektiv, har vi gitt en ganske utførlig beskrivelse av TAXMAN, et kunnskaps-basert program laget av Thorne McCarty[2].

[1] Vi har brukt Crystal versjon 2.10 fra 1986. All eksplisitt kritikk av Crystal er med utgangspunkt i denne versjonen av programmet.

[2] L. Thorne McCarty "Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning", Harvard Law Review 90, pp 837-893, 1977.

Vi ønsket å prøve å representere juridisk kunnskap i et slikt ekspertsystemskall. Vi brukte Crystal til å modellere Lov om Folketrygd kap 7, §§ 7-1 og 7-2, om regler for tildeling og utregning av grunnpensjon. Vi hadde ambisjoner om også å modellere reglene for tilleggspensjon, og døpte derfor programmet vi laget "Alderstrygd". Programmet ble senere begrenset til grunnpensjon fordi vi først og fremst ønsket å utforske skallets virkemåte og evne til å modellere den juridiske kunnskap. Noe fullstendig program for utregning og tildeling av alderstrygd var ikke påkrevd for dette.

I denne rapporten bruker vi begrepene "representasjon", "modell" og "program" uten å skille dem klart ad. Når meningsinnholdet (semantikken) i en rettskildefaktor (i det hele tatt) får et materielt uttrykk, f.eks. i ord i naturlig språk, så har dette meningsinnholdet fått en representasjon. Hvis rettskildefaktoren gis uttrykk i et formelt språk med en fast definert, entydig semantikk, har den fått en formell representasjon. Når vi med et system av flere språklige elementer har dannet en struktur som representerer rettskildefaktorens meningsinnhold, så har vi laget en modell av dette meningsinnholdet. Når denne modellen er laget med et programmeringsspråk som lar en datamaskin bruke modellen til f.eks. å fastsette grunnpensjon, så er denne modellen også et program.

En slik modell innebærer en oversettelse av rettskildefaktoren fra dets representasjon i naturlig språk til et annet (formelt) språk. Enhver oversettelse innebærer en fortolkning, og en oversettelse av rettskildefaktorer innebærer derfor en juridisk tolkning. Denne tolkningen kan ha flere ambisjonsnivå. Vår ambisjon var at den ikke skulle være direkte gal eller mangelfull. Om bruken av systemet ville få juridiske konsekvenser, ville man også måtte kreve at tolkningen var fullstendig, eller ihvertfall tilstrekkelig fyldestgjørende for programmets formål. Hva dette skulle innebære vil måtte fastlegges gjennom en juridisk undersøkelse, noe denne rapporten ikke er et forsøk på. Men ved å gi en relativt fullstendig beskrivelse av hvordan "Aldrstrygd" virker og hvilke mangler det har, håper vi å stimulere den juridisk interesserte leser til å tenke igjennom hvilke krav som måtte stilles til en tilstrekkelig juridisk fyldestgjorende representasjon. En slik representasjon kan f.eks. tenkes brukt i et program til bruk ved trygdekontorene for

å assistere en saksbehandler med å gi de riktige informasjoner for fastsettelse og utregning av alderstrygd.

Rapporten begynner med en drøftelse av ulike formål for bruk av denne teknologien innen juridisk arbeid. For å illustrere enkelte vansker med å lage en god representasjon, gir vi en beskrivelse av trygdeloven. Så følger en introduksjon til de viktigste trekkene ved ekspertsystemskallet Crystal. Programmet "Alderstrygd" blir deretter utførlig beskrevet, og det gjengis en dialog med programmet. Vi har sammenlignet "Alderstrygd" med Thorne McCartys program TAXMAN. Dernest har vi tatt opp vansker Crystal gir en med å lage og oppdatere programmer. Vi diskuterer ulike brukergrensesnitt og ulike måter å gi brukeren forklaringer på. Og vi har til slutt tatt opp spørsmålet om et program laget i Crystal i det hele tatt kan sies å være noe ekspertsystem eller et kunnskapsbasert system. Trygdeloven §§ 7-1 og 7-2 følger som vedlegg.

2. BRUK AV EKSPERTSYSTEMSKALL I JURIDISK ARBEID.

Det er skrevet masse om formålet med ekspertsystemer og om formålet med juridiske ekspertsystemer. Dette er etterhvert blitt et spørsmål nesten like stort og utflytende som spørsmålet om hva datateknologi kan brukes til generelt. For ekspertsystemer blir lett et slags symbol for en mer vag ide om avanserte datasystemer med nye fantastiske egenskaper. Vi vil ikke her drøfte i detalj hva ekspertsystemer er og kan eller burde kunne gjøre. Istedet vil vi ta opp hvordan man i juridisk sammenheng kan bruke de kommersielt tilgjengelige ekspertsystemskallene for PC. Vi vil heretter kalle ekspertsystemskallene ES-skall, og ES-skall for PC kaller vi PC-skall.

Computas Expert Systems har utgitt rapporter med ca. et års mellomrom som oppsummerer ES-skallenes tekniske spesifikasjoner og kommenterer deres fortrinn og begrensninger[3]. Inntrykket etter å ha fått presentert resultatet av et par slike undersøkelser er at ekspertsystemskall for dedikerte LISP-maskiner og kraftige arbeidsstasjoner betraktes som seriøse utviklingsverktøy. ES-systemskall for PC har derimot intil nylig stort sett vært å betrakte som spennende leketøy. Noen PC-skall prøver også å importere alle mulighetene man får med arbeidsstasjonskallene. Dette er nødt til å gå ut over effektiviteten, slik at disse skallene presser grensene for en PC's minne og regnekraft og arbeider ulidelig langsomt.

PC-skallene er likevel blitt avanserte nok til på den ene side å kunne gi nybegynnere en innfallsvinkel til å tilnærme seg den nye teknologien, og på den andre side til å lage mindre demonstrasjonssystemer for at man gjennom et forprosjekt kanskje kan overbevise en oppdragsgiver om at et ekspertsystemprosjekt er

[3] Rolf Nossum, "Ekspertsystem-skall for PC og Macintosh", Nordisk Datanytt, vol. 16 no. 1, 1986.

Roar Fjellheim, "Ekspertsystemer 1987: Status og tendenser", DataTid, Nr. 4 (April) 1987.

verdt å gå videre med[4]. Ettersom de blir laget for lettere å kunne virke sammen med andre typer programmer, er det kanskje også mulig å tenke seg at man med skall kan lage mindre regelbaserte moduler til andre systemer.

PC-skall er en ganske ny form for programvare. I likhet med andre nyere programmeringsomgivelser utnytter de moderne virkemidler som vinduer, menyer og mus. Skallene er ment å være verktøy som skal hjelpe programmereren til å kunne konsentrere seg om innholdet i reglene. Skallet har høynivå kommandoer som hjelper brukeren med å få utført maskinnære ting som hvordan data skal lagres eller skjermbilder tegnes opp, uten å måtte gi mange detaljerte instruksjoner. PC-skall gir dermed inntrykk av å være brukervennlige også for uøvede programmerere.

I juridisk sammenheng har man også lagt vekt på det faktum at ES-skall er et hjelpemiddel til regelbasert programmering. Å lage et program i form av serier med utsagn om at "HVIS *det og det er tilfelle* SÅ skal *det og det skje*" synes som et mindre sprang fra tradisjonell juridisk argumentasjon enn om man skulle representere argumentene med instruksjonene i et tradisjonelt programmeringsspråk. I teorien skulle man også kunne legge mindre vekt på programmets overordnede struktur. I en konvensjonell algoritme må programmereren forutse alle logiske slutninger og slutningenes rekkefølge. Deretter må programmereren gi maskinen instruksjon om å gjøre disse slutningene. I et ekspertsystem er tanken at man kan lage en deklarativ representasjon av domenet. Slutningsmekanismen i skallet skal finne fram til hvilke regler som i hvilken rekkefølge er del av den aktuelle slutningskjede. Det kan med andre ord virke attraktivt for folk i det juridiske miljø å begynne å bruke ES-skall.

I navnet "ekspert system" ligger det en hentydning om at produktet man lager er i stand til å gi råd på linje med en ekspert. Det er vanskelig å definere hvem som er en ekspert. Forenklet kan vi si at en ekspert har en spesiell kunnskap andre ikke har om et emne. En av ideene bak å lage ekspertsystemer var å gjøre slik

[4] Dette er det uttalt formål for mange av de prototyper som er blitt produsert. Et eksempel på dette er rapporten til Berg/Theisen, NR rapport 811, jan 1988.

kunnskap tilgjengelig for flere. En juridisk ekspert vil vel ofte være en person med kunnskap og evne til å løse særlig vanskelige juridiske problemer. Slike problemer er igjen ofte knyttet til at rettstilstanden på det aktuelle område er uklar.

Å få en maskin til å etterligne måten en juridisk ekspert løser slike problemer på er meget vanskelig. Det fordrer at alle de rettslige normer som trekkes inn er skrevet om på en entydig (men ikke nødvendigvis presis) måte som maskinen kan forstå. Dagens ES-skall har dessuten bare mekanismer for ren deduktiv resonnering[5], mens man i juridisk metodelære også tillater analogiske resonnementer, utvidende og innskrenkende fortolkning, reelle hensyn etc. Med dagens teknologi er vi m.a.o. bare istand til å fange enkelte begrensede aspekter av juridisk argumentasjon. Programmene vil dessuten bare inneholde en enkelt, fiksert representasjon av rettsreglene. Kommersielle ES-skall vil vanskelig kunne representere tvetydighet, tvil eller rettslig motstrid. Dette innebærer at drømmen om en maskin som gir juridiske råd på ekspertnivå ennå er fjern. Vi kan derfor heller ikke få en maskin som tar ekspert avgjørelser.

Det finnes alt i dag programmer som produserer rettslige avgjørelser ved å beregne tildeling og beløp av f.eks. sosiale stønader[6]. Disse kan enten kritiseres for ikke å ha en fullstendig representasjon av rettsområdet avgjørelsene bygger på, eller de inneholder bare representasjoner av enkle rettsregler som ikke gir noen av de omtalte problemer. Vi vil i denne rapporten ikke komme nærmere inn på hvordan slike "produksjonsprogrammer" i seg selv kan forbedres. Formålet med rettslige ekspertsystemer vil være å assistere et menneske

[5] Flere ES-skall er også istand til å resonnere med usikkerhetsfaktorer.

[6] Eksempler er bostøtteordningen som er drøftet i

- Paul Rynning, "En fremstilling og vurdering av reglene for tildeling av bostøtte", Skriftserien JUS og EDB nr.17/1976,

og INFOTRYGD og NORTRYGD systemene som er drøftet i

- Dag Wiese Schartum "The introduction of computers in Norwegian local insurance offices", CompLex 9/87.

med å gjøre en rettslig analyse. Dette kan forbedre et produksjonsprogram ved å hjelpe en bruker med den analyse som skal til for å gi produksjonsprogrammet riktigere informasjon til beregningene. F.eks. har man på Norsk Regnesentral[7] utviklet en prototype for et ekspertsystem som kan hjelpe en ligningsfunksjonær å fastsette særfradrag. Når dette er fastsatt vil det inngå som nødvendig informasjon til programmet som utregner skattetrekk.

Thorne McCartys TAXMAN program illustrerer en annen anvendelse[8]. Hans mål var å gjøre eksplisitt innholdet og strukturen i et juridisk domene. I forhold til et vanlig resonnement presentert i naturlig språk, er det å lage et program en langsom og møysommelig måte å uttrykke seg på. Men en slik *formalisert* beskrivelse er tilgjengjeld helt entydig og alle deler av resonnementet er gjort helt ekspisitt. I tillegg er resonnementet automatisert, man møter det ikke bare som en lang tekst, men som en dialog med programmet der resonnementet vises fram som forklaringer. Gjennom en dialog med programmet kan man gjøre seg kjent med konsekvensene av programforfatterens argumentasjon. Om man vil gå enda mer i detalj, kan man ved å studere programkoden dissekere dialogens forutsetninger. Dette kan synes som et mulig verktøy til en mer eksakt juridisk argumentasjon, hvilket vel ihvertfall ville kunne få gjennomslag på enkelte områder innen juridisk forskning.

Slike programmer kan også tenkes brukt som datastøttet undervisning i rettslære og kanskje som hjelpemiddel for jusstudenter. Det er mulig å tenke seg at elever og studenter vil synes det er spennende å løse et rettslig problem i dialog med et dataprogram, og at de vil ha nytte av å se hvordan programmet lar reglene komme til anvendelse. Hvis det var tilstrekkelig enkelt å lage nye regler i programmet, kunne det tenkes at de ville kunne eksperimentere med å endre reglene i programmet for å se hvordan slutningene endrer seg.

Å klargjøre for seg selv hvilke beslutningsstrukturer som ligger i rettsreglene ved selv å prøve å utvikle en formalisert beskrivelse av et avgrenset rettsområde kan kanskje også være nyttig. Man

[7] Berg/Theisen, NR rapport 811, jan 1988.

[8] L. Thorne McCarty, 1977.

er nødt til å skaffe seg en meget grundig forståelse av hva en tekst egentlig sier for å kunne å lage en formal modell. Hvis man også implementerer en formell beskrivelse, eller bruker et programmeringsverktøy til modelleringen, kan man teste hvilke slutninger som kan deduseres fra modellen. Spørsmålet om programmet fungerer til nytte eller til å skape storre forståelse for andre blir da mindre viktig. Man kan godt tenke seg at en person lager noe helt ubrukelig, men at dette likevel gir ham en tilbakemelding om hva som er feil i hans forståelse av rettsområdet. Hvis han får nye innsikter er målet nådd. Om han så vil dokumentere dette for seg selv eller andre med å lage et bedre program, blir en ny vurdering.

En begrensning er likevel at en vanlig argumentasjon ofte tar hensyn til svært mange faktorer og kan ha mange underforståtte forutsetninger. I et program må man eksplisitt beskrive alle hensyn. Hvis det dermed blir svært mange regler, vil reglenes klare eksplisitte formuleringer og sammenhenger drukne i manglende oversiktbarhet. Denne begrensning gjør det nødvendig å nøye avveie nytteverdien av å lage slike programmer. Hvis resonnementet er enkelt, er det spørsmål om ikke det langt enklere lot seg forklare med naturlig språk, eller eventuelt ved å tegne et flytdiagram. Hvis det er antydninger om at resonnementet egentlig er ganske omfattende, bør man antagelig vokte seg for i det hele tatt å prøve å lage noen modell som krever et så stort detaljeringsnivå som et program.

Av denne rapporten vil det framgå hvorfor vi ikke vil anbefale Crystal til noen av de nevnte formål.

3. LOV OM FOLKETRYGD - LITT OM MODELLENS KONTEKST.

Det kan være på sin plass med en beskrivelse av den rettslige kontekst for den relativt begrensede modell vi har laget av alderstrygden. Følgende beskrivelse bygger på et foredrag holdt av professor Asbjørn Kjønstad i egenskap av formann for trygdelovutvalget som arbeider med forenkling av folketrygdloven[9].

Dagens trygdelov er resultat av at man i 1966 og 1970 slo sammen 10-12 trygdelover med ulike formål, stønadsregler og historisk bakgrunn. Utviklingen av slike lover er preget av over 100 års sosialpolitisk og juridisk arbeid. Kjønstad uttaler:

> "Trygderettslig lovgivning er i stor grad en realisering av politske lofter. Lovene er ofte blitt til under hastverk, og dette preger mye av lovgivningen. Den er lite systematisk oppbygd, og det er vanskelig å finne fram i den. De enkelte ord og uttrykk er ikke alltid valgt med største omhu."

Med tilhørende forskrifter omfatter trygdeloven over 1000 paragrafer. Loven er i kontinuerlig forandring. Kjønstad påpeker at i de senere år har over en fjerdedel av Stortingets lovvedtak vært endringer i trygdelovgivningen eller nye trygdelover. Uten tilgang til Lovdata er forskriftsverket vanskelig å få tak i for folk utenfor trygdeetaten, og det er ofte vanskelig å se hvordan regler i forskriftene skal kombineres med reglene i loven. Mange viktige løsninger på trygderettslige spørsmål er bare å finne i trygdeetatens interne rundskriv.

Kjønstad understreker at et særlig problem ved folketrygdloven er alle de interne henvisningene. Mange henvisninger er diffuse eller upresise. Andre steder er det regelsett som bygger på store deler av et annet regelsett. Kjøsntad uttaler:

> "For å kunne beregne attføringspengenes størrelse etter kapittel 5, må man kunne reglene om uførepensjon i kapittel 8. For å kunne reglene om uførepensjon, må man kunne reglene om

[9] Asbjørn Kjønstad "Vi får en ny trygdelov som mange flere kan forstå", Sosial Trygd nr 7, 1982.

alderspensjon i kapittel 7. Og for å kunne reglene om alderspensjon, må man beherske reglene om folketrygdens beregningsfaktorer i kapittel 6."

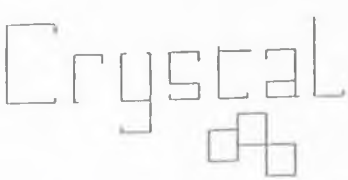
Kjønstad kaller dette med en viss forakt for "aktuarjuss". Andre eksempler på aktuarjuss finner han i lange og tunge setninger, vanskelige ord og uforståelige forkortelser. Kjønstad leder som nevnt et utvalg som skal legge fram et forslag til forenkling av loven. Han understreker at dette kunn er et redigeringsarbeid, ikke et forslag om innholdsmessige endringer. I vår terminologi kan dette sies å være en ny representasjon av lovens innhold. Hvor tidkrevende og vanskelig denne prosessen er kan belyses ved tidsplanene for trygdelovutvalgets arbeid. Arbeidet, som ble påbegynt i 1980, regner Kjønstad at kan resultere i vedtagelse av en ny trygdelov tidligst i 1991.

Som det fremgår er ikke trygdeloven noen enkel lov hverken å forstå eller lage modeller av. Regelverket er så omfattende at å lage en fullstendig modell vil være svært ressurskrevende. For enklere å håndtere regelverket bør man nok sette sin lit til arbeidet med regelforenkling og rettslige informasjonssystemer som Lovdata. Med hjelp fra personer med ekspertkunnskap om dette regelverket vil man kunne lage modeller av deler av regelverket, f.eks. reglene om alderspensjon. En slik modell kan hjelpe personer med mindre kunnskaper på dette området å anvende denne delen av reglene.

Dette var ideen bak vårt forsøk på å lage en modell av reglene om alderspensjon. Det viste seg at også disse reglene ville utgjøre et ganske omfattende program. Til vårt demonstrasjonsprogram fant vi det derfor tilstrekkelig å modellere hovedreglene om grunnpensjon i paragrafene § 7-1 og § 7-2. Teksten er gitt som vedlegg etter rapporten.

4. PRESENTASJON AV CRYSTAL.

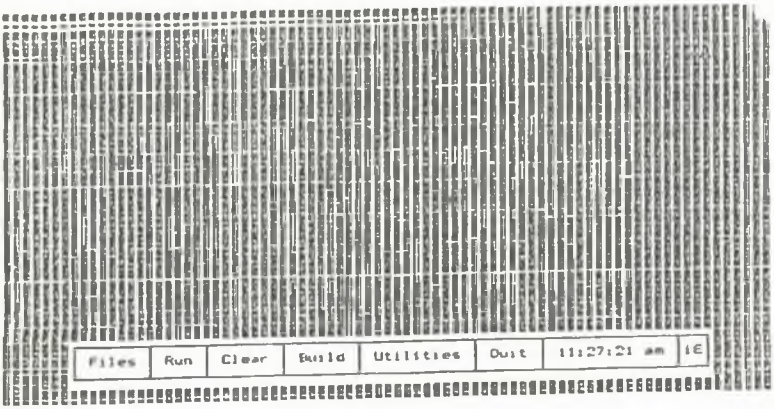
4.1 Oppstart.
Slik starter Crystal:



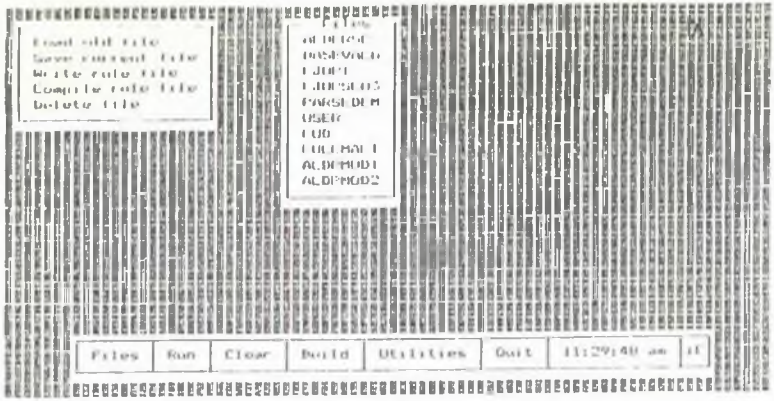
1A

(Vi har nummerert eksempel bildene fortløpende med felles bokstav for felles avsnitt. Her fra (1A).)

Man får ført hovedmenyen (2A) gjennom hvilken man håndterer filer, utskrift, start av programmer (run), bygging av programmer (build) og avslutning (quit). I runtime versjoner (operative programmer) antar vi at programmer kjøres uten disse omgivelsene. De er del av programmeringsverktøyet Crystal.



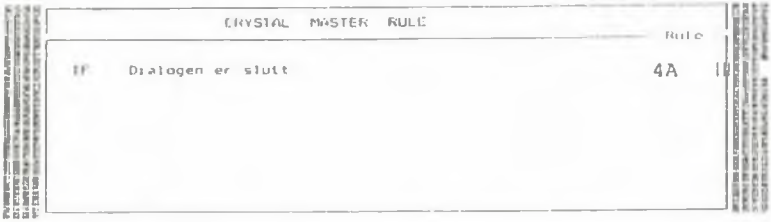
Vi laster inn (fra Files menyen) programmet ALDERSTRYGD (3A).



Det kan nå kjores ved å gå til (run), eller bygges videre ved å gå til (build).

4.2 Å bygge programmer.

Når vi har valg å bygge flere regler, (build) (rules), kommer vi alltid til toppen av trestrukturen. I Crystal kalles toppen CRYSTAL MASTER RULE (4A). Herfra kan vi bevege oss nedover i grenene av treet med funksjonstasten F10 og oppover i treet igjen med F8. Disse tastene brukes hele tiden for å flytte seg rundt i treet.



Det vi gjør i Crystal er kort og godt å beskrive en konklusjon, og så gi denne en kjede av betingelser. Betingelsene kan hektes sammen med OG. Da må de sammen være sanne for at konklusjonen skal være oppfylt. Om en av dem ikke kan oppfylles vil programmet ikke prove fler, fordi straks en betingelse ikke oppfylles kan konklusjonen ikke folge. Betingelsene kan også være hektet sammen med ELLER. Betingelsene er da alternative til hverandre slik at den neste proves om ikke den forste blir oppfylt. Bare dersom ingen av alternativene kan oppfylles kan konklusjonen

ikke folge. Konklusjoner kan også hektes sammen med ELLER slik at om konklusjonen ikke er sann, prøver programmet å oppfylle den alternative konklusjonen istedet.

Alle operasjoner vi ønsker å bruke i programmet må hentes fra en forholdsvis kort liste.se (7A) [10]. Vi vil her gi en oversikt over de viktigste operasjonene i vår versjon av Crystal:

4.3 Kommandolisten:

Conclusion Display	: Viser brukeren toppen av treet/hoved-konklusjonen kalt Crystal Master Rule.
Display Form	: Viser brukeren en melding. Bruk-eren kan måtte fylle inn variabelverdier. Brukeren må alltid kvittere med return tasten.
Yes/No Question	: Et spørsmål som må besvares Ja eller Nei.
Graphics File	: Om brukeren trykker F1 vises brukeren et bilde hentet inn fra en egen grafikk fil. Dette må lages med et tegneprogram. Crystal inneholder ikke noe eget tegneprogram.
Fail	: Tvinger en betingelse til aldri å oppfylles.
Succeed	: Tvinger en betingelse til alltid å oppfylles.
Begin Explain	: Viser brukeren en tekstfil laget utenfor Crystal istedet for CrystalS regeltre i tilfelle brukeren bruker F1 til å be om forklaring.
End Explain	: Gjør at ingen forklaring vises om man bruker F1.

[10] I nyere versjoner av Crystal som først ble tilgjengelige etter at vi var ferdige med vår modell er det noen flere kommandoer.

Test Expression : Gjør oppfyllelsen av en betingelse avhengig av at en variabel har en bestemt verdi.

Assign Variable : Brukes til å gi en variabel verdier, evt. ved hjelp av logiske eller aritmetiske operasjoner.

Menu Question : Lar brukeren velge verdier fra en meny/liste.

Print Form : Lar en melding bli skrevet ut på printeren.

User Program : Gir en mulighet for innhenting av variabel verdier fra eksterne databaser, regneark eller ASCII-filer.

Restart Rule : Gjør at programmet prøver alle betingelser på grenene utenfor denne kommandoen om igjen.

Output File : Lar en melding bli skrevet ut til fil.

Load File : Henter inn et nytt Crystal program. Dette vil fortrenge det programmet hvor kommandoen er brukt.

4.4 Funksjonstastene:

F1 : Forklaringsfunksjon. Funksjonen viser betingelsene den prøver å oppfylle når den brukes når programmet prøver å oppfylle betingelser. Den viser regeltreet som leder til en konklusjon når programmet har kommet til en konklusjon. Den viser en grafikkfil eller en tekstfil istedet hvis man har lagt inn dette med Graphics File eller Begin Explain.

F4 : Gjør en regel til Special Rule. Det markerer at denne regelen skal prøves om igjen hvis man i programmet har brukt kommandoen Restart Rule.

F5 : Lar en forandre, flytte eller slette regler.

F6 : Gir en liste over kommandoene beskrevet over. Alle kommandoer man vil bruke må velges fra denne listen, se (7A).

F7 : Gir en listen over betingelser, eller listen over variabler eller en kopi av innholdet i den siste ruten man laget.

F8 : Flytter en oppover mot toppen/konklusjonene i regeltreet.

F9 : Setter NOT foran en betingelse. Den er da bare oppfylt hvis det betingelsen krever *ikke* er oppfylt.

F10 : Flytter en nedover mot forfiningen av betingelsene i regeltreet.

4.5 Ekstra- og alternative betingelser:

AND : Knytter en ny betingelse til den første. Man får AND straks man trykker return etter å ha gitt forrige betingelse.

OR : Gir en alternativ betingelse til den første. Man får OR hvis man trykker return to ganger etter hverandre etter å ha gitt forrige betingelse.

4.6 Hovedprogrammet.

Vi fant det mest hensiktsmessig bare å ha en helt generell frase såsom "Dialogen er slutt" som endelig konklusjon i Crystal Master Rule fordi denne bare nås ved en egen kommando, "Conclusion Display". Vi kan tenke oss selve programmet som en kjede av konklusjoner på nivået under Master Rule, som programmet i tur og orden søker å oppfylle. En begynner her å lage strukturer ved

å angi kjeder av betingelser for at konklusjonene skal oppfylles[11]. Crystal Master Rule er i vårt program bare et tak over det hele.

For ALDERSTRYGD ser denne kjeden av konklusjoner slik ut (5A):

```
Dialogen er slutt                                                    < Rule >

  IF    Klienten har krav på alderspensjon                              5A
    AND grunnpensjonen er fastsatt
      AND NOT tilleggspensjonen er fastsatt
      AND :Conclusion Display

  OR    :Display Form
```

Når programmet har funnet at:

IF (1) Klenten har krav på alderspensjon
AND (2) Grunnpensjon er regnet ut
AND (3) Tilleggspensjon er regnet ut
AND "Conclusion Display". Vi leser her: SÅ vises konklusjone (Master Rule) "Dialogen er slutt"

OR "Display form". Vi leser her: ELLER SÅ vises "Programmet er slutt". Dette viser den lager programmet at en av konklusjonene 1 - 3 ikke ble oppfylt for dialogen sluttet.

Crystal går helt sekvensielt igjennom sine regler ovenfra og nedover. Den betingelse som er satt først prøves forst, deretter de andre som er hektet på med OG og ELLER. Crystal går også hele tiden ut og prøver betingelsene ytterst i treet for den går videre. Denne strukturerte virkemåten gjør Crystal effektivt ved at kombinatoriske eksplosjoner unngås[12]. Det gir oss en struktur

[11] Strukturen av betingelser kalles en tre-struktur. I motsetning til trær i naturen, forgrener disse tre-strukturene seg nedover.

[12] En kombinatorisk eksplosjon ville oppstå om Crystal fritt kunne prøve enhver regel i kombinasjon med enhver annen regel. Antall forsøk Crystal måtte gjøre for å utføre de ønskede operasjoner ville da øke eksponensielt med antall regler.

som lett lar seg manipulere. Strukturen gir imidlertid Crystalprogrammer en datastruktur helt forskjellig fra det man vanligvis vil kalle ekspertsystemer eller kunnskapsbaserte systemer[13].

4.7 Forfining av betingelsene.

Vi begynner med å forfine den første betingelsen (1) "Klienten har krav på alderspensjon". Først trykker man F10. (1) rykkes da opp på konklusjonsnivå over linjen der det står <Rule>, og vi kan skrive en ny betingelse; "klienten er fyllt 67 år" (6A). Vi kan lese dette som HVIS "klienten er fyllt 67 år" SÅ "klienten har krav på alderspensjon".

```
klienten har krav på alderspensjon                    < Rule >
                                                        6A
  IF    klienten er fylt 67 ar
    AND :Display Form
```

I eksempelet (6A) er det også med en linje AND :Display Form. Vi har brukt funksjonstasten F6 for å få KOMMANDOLISTEN, se eks. (7A). Fra kommandolisten valgte vi Display Form og fikk et skjermbilde hvor vi kunne skrive meldinger til brukeren av programmet. Toppen av skjermbildet vises i ruten (6A)[14]. Crystals behandler også kommandoer som betingelser. Dersom vi onsker å gi brukeren en melding, så er det en betingelse for Crystal at meldingen blir vist på skjermen og brukeren kvitterer for dette med å trykke Return[15].

[13] Jfr. avsnittet "Er Crystalprogrammer kunnskapsbaserte programmer".

[14] Ruten brukeren får se begynner med "$ 7-1".

[15] f.eks. gis i (6A) en melding om at brukeren fyller kravet til å få alderspensjon. At det kvitteres for meldingen er en betingelse for å oppfylle at "klienten har krav på alderspensjon".

```
klienten har krav på alderspensjon

IF    klienten er fylt 67 år
   AND :Display Form

Conclusion Display
Display Form
Yes/No Question
Graphics File
Fail
Succeed
Begin Explain
End Explain
Test Expression
Assign Variable
Menu Question
Print Form
User Program
Restart Rule
Output File
Load File
```

7A

Dersom en ønsker å lage en ny Display Form, kan man kopiere innholdet av den forrige Display Form med tasten F7. Dette gjelder også de ruter man får opp med Yes/No Question og Menu Question. Dersom man derimot er i ferd med å skrive betingelser, gir F7 en liste over alle betingelser som tidligere er brukt (8A). Man kan kopiere inn betingelser fra listen, men samtidig overtas alle forfininger av den betingelsen man kopierer inn. Betingelser trenger dog bare oppfylles en gang, idet programmet bare spør en gang for å få en betingelse oppfylt. Unntaket gjelder dersom man merker den Special Rule og anvender kommandoen Restart Rule.

```
klienten har krav på alderspensjon
                                                    < Dictionary >
                andel av grunnpensjon som utbetales er forklart
IF    klienten er| Dialogen er slutt
   AND :Display Fo| grunnbeløpets relative størrelse er utregnet
                  grunnpensjonen er fastsatt
                  hvilken trygdetid det skal regnes etter er avgjort
                  informasjon om evt. ektefelles pensjoner er gitt
                  klienten er fylt 67 år
                  klienten har krav på alderspensjon
                  Max tellende trygdetid 40 år er prøvet
```

8A

Om vi ønsker å forfine betingelsen "klienten er fyllt 67 år" ytterligere, bruker vi igjen F10. Nå vil vi spørre brukeren om klientens navn og fødselsdato (9A). Vi har først satt to linjer med Assign for å få en blank plass å fylle inn navn og 0-stille fødselsdatoen, dernest har vi brukt en Display Form som vi ser toppen av over kanten. Etter "Oppgi klientens navn:" står det <klientens_navn$>. Dette angir en tekstvariabel som får som verdi det navn klienten skriver inn. $-tegnet markerer at variabelens verdi blir behandlet som en tekst(-streng) av Crystal. Tilsvarende

er det en tall-variabel <klientens_F_aar> som får som verdi det årstall som oppgis som klientens fødselsår.

```
                                                          < Test >
Oppgi klientens navn:
                    <klientens_navn$          >

Oppgi klientens fødselsår:

klienten er fylt 67 ar
                                                          < Rule >
                                                                  9A
  IF    :Assign klientens_navn$:=""
    AND :Assign klientens_F_aar:=0
    AND :Display Form
    AND :Assign alder:=1988 - klientens_F_aar
    AND :Test   alder>=67

  OR    :Display Form
    AND :Fail
```

Etter Display Form har vi en Assign setning som setter klientens alder lik inneværende år minus fødselsåret. Deretter finnes en Test setning som prøver om verdien i variabelen alder er større eller lik 67. Dersom dette er tilfellet, er betingelsen om at "klienten er fyllt 67 år" oppfylt. Alternativt prøver programmet å oppfylle de betingelsene som kommer etter OR. Vi benyttet dette til å sende en melding til brukeren (med Display Form) om at han ikke fyller betingelsene for å få alderspensjon (10A). Når brukeren kvitterer med Return prøver programmet å oppfylle den siste betingelsen, men dette er kommandoen Fail som aldri vil være oppfylt. Dersom programmet ikke fikk oppfylt de første betingelsene og derfor prøver å få oppfylt betingelsene etter OR, vil programmet ikke nå konklusjonene "klienten er fylt 67 år" og "klienten har krav på trygd". Derved vil heller ikke betingelsene for å få oppfylt konklusjonene "grunnpensjonen er fastsatt" og "tilleggspensjon er fastsatt" bli prøvet. Istedet vil programmet vise den alternative konklusjon etter OR (5A) som er en Display Form som sier at programmet er slutt.

```
< Test >
Da [klientens_navn$                    ]
ikke er fylt 67 ar har
[klientens_navn$                       ]
ikke rett til alderspensjon.

klienten                                  < Rule >
                                            10A
  IF
    AND
    AND
    AND :Assign alder:=1988 - klientens_F_aar
    AND :Test   alder>=67

  OR    :Display Fore
    AND :Fail
```

En tredje bruk av funksjonstasten F7 har vi når vi skal fylle ut variabelnavn i en Assign eller Test kommando. F7 vil da gi oss en liste over alle tidligere brukte variabelnavn (11A). Fra denne listen kan vi kopiere inn det variabelnavn vi ønsker.

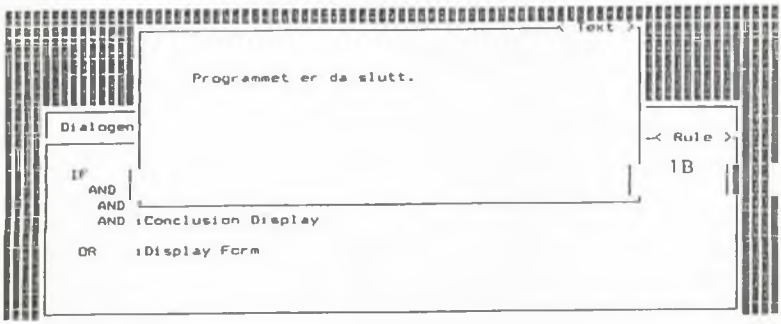
```
< Assign >
alder:=1988 - klientens_F_aar

klienten er fylt 67 ar
                                          < Variables >
  IF    :Assign kli   alder
                      efp_andel                          11A
    AND :Assign kli   ektefellens_navn$
    AND :Display Fo   EKTEFELLES_ALDER
    AND :Assign ald   ektefelles_F_aar
    AND :Test   ald   EKTEF_MED_PENSJON$
                      EKTEF_TRYGDETID
  OR    :Display Fo   flyktning$
    AND :Fail         gjenglfte$

Files | Run | Clear | Build | Utilities | Quit | 11:47:45 am | :E
```

5. MODELL AV LOV OM FOLKETRYGD §§ 7-1 og 7-2.

Når man lager en modell i Crystal begynner man med konklusjonene og angir deretter betingelsene for disse. Dette krever at man strukturerer sin tenkning helt uavhengig av strukturen i lovteksten. Det er viktig å frigjøre seg fra den rekkefølge som er angitt i lovteksten.

5.1 Toppen av regeltreet.

Vår modell av trygdelovens bestemmelser blir brukt i et program som skal kunne konsulteres for å finne ut om en person har krav på grunnpensjon og hvor mye vedkommende i tilfelle vil få utbetalt. Det siste som meddeles brukeren er at programmet er ferdig. Dette lar vi være programmets hovedkonklusjon; Crystal Master Rule har vi kalt "Dialogen er slutt", se (4A) [16]. Hvis programmet ikke klarer å få oppfyllt hovedkonklusjonen vises "Programmet er da slutt" (1B). Så følger betingelsene for at "Dialogen er slutt". "Dialogen er slutt" HVIS "Klienten har krav på alderspensjon" OG "Grunnpensjonen er fastsatt", se (5A) [17]. Hvis programmet ikke finner at "Klienten har krav på alderspensjon" eller at ikke "Grunnpensjonen er fastsatt" gir det istedet konklusjonen "Programmet er da slutt".



[16] Jfr. avsnittet "Presentasjon av Crystal".

[17] Vi har også laget en tom betingelse AND NOT: "Tilleggspensjon er fastsatt". Fordi vi ikke har gitt noen betingelser for å fastsette tilleggspensjon ennå, er denne nå oppfylt når tilleggspensjon IKKE er fastsatt, dvs den er alltid sann slik programmet er nå.

5.2 Betingelser for krav på alderspensjon.

Vi har i avsnittet "Presentasjon av Crystal" vist hvordan vi forfinet betingelsen "Klienten har krav på alderspensjon". HVIS "klienten er fylt 67 år" SÅ "klienten har krav på alderspensjon", se (6A). HVIS - klienten oppgir navn og fødselsår og alder er satt til inneværende år minus fødselsår og TEST setningen har funnet at alder er større eller lik 67 SÅ "klienten er fylt 67 år", se (9A). Vi viste også hvordan Crystals sekvensielle virkemåte gjorde at vi kunne gi som en alternativ betingelse en melding om at "da klienten ikke er fylt 67 år har klienten ikke krav på alderspensjon" og så en FAIL kommando slik at den alternative betingelsen ikke kan oppfylle konklusjonen "klienten er fylt 67 år", se (10A). FAIL kommandoen utelukker også oppfyllelsen av konklusjonene "klienten har krav på alderspensjon" og "Dialogen er slutt" ,og vi får "Programmet er da slutt".

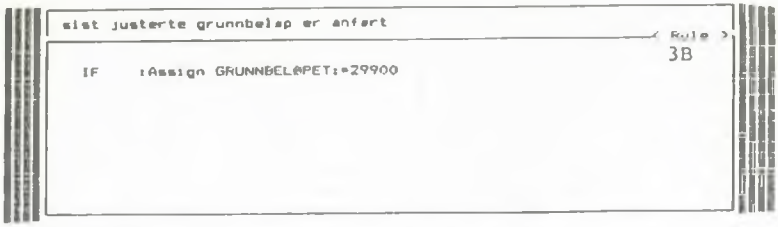
```
grunnpensjonen er fastsatt                                        < Rule >
                                                                    2B
  IF    sist justerte grunnbeløp er anført
    AND sivilstatus er fastlagt
    AND informasjon om evt. ektefelles pensjoner er gitt
    AND spørsmål om minimum trygdetid er tatt hensyn til
    AND grunnbeløpets relative størrelse er utregnet
    AND andel av grunnpensjon som utbetales er forklart
    AND størrelsen på grunnpensjonen er oppgitt klienten
```

5.3 Betingelser for å fastsette grunnpensjon.

Hvis programmet får oppfylt "Klienten har krav på alderspensjon" går det igang med å prøve å få oppfylt betingelsene for "Grunnpensjonen er fastsatt" (2B). Vi har her en hel liste med betingelser: "Grunnpensjonen er fastsatt" HVIS

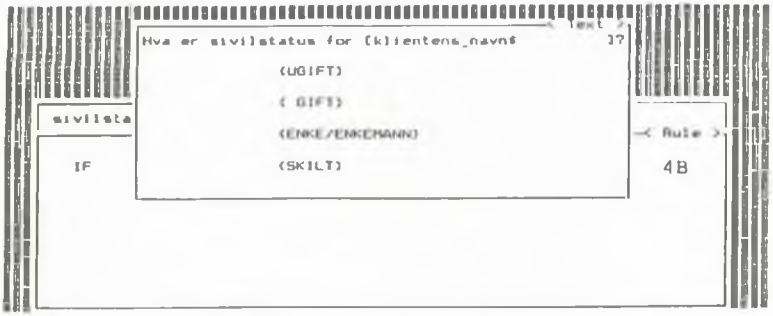
(A) "sist justerte grunnbeløp er anført" OG
(B) "sivilstatus er fastlagt" OG
(C) "informasjon om eventuell ektefelles pensjoner er gitt" OG
(D) "spørsmål om minimum trygdetid er tatt hensyn til" OG
(E) "grunnbeløpets relative størrelse er tatt hensyn til" OG
(F) "andel av grunnpensjon som utbetales er forklart" OG
(G) "størrelsen på grunnpensjonen er oppgitt klienten".

Denne listen angir en modell for hvordan man går fram for å fastsette grunnpensjon. Prosessen "å fastsette grunnpensjonen" er m.a.o. splittet opp i en rekke delprosesser. Vi vil vise hva som gjøres i hver av disse delprosessene.



5.4 Del (A) og (B).
Sist justerte grunnbeløp er anfort HVIS grunnbeløpet er satt lik 29900 (3B)[18]. Denne betingelsen er skilt ut som en egen delprosess for at det skal være lett å finne hvor grunnbeløpet er satt hvis man skal justere beløpet senere.

Fastlegging av sivilstatus er en omfattende prosess. Vi vil her begynne forklaringen med den ytterste betingelsesgrenen, den ytterste noden i treet også kalt blad-noden. Her vises en meny hvor brukeren kan velge om verdien av variabelen klientens_sivilstatus$ skal være UGIFT, GIFT, ENKE/ENKEMANN eller SKILT (4B). Når brukeren har valgt er konlusjonen "sivilstatus er oppgitt" sann (5B). Vi har ikke tatt stilling til hvordan f.eks. separerte eller samboende skal behandles. I forhold til § 7-2 trenger vi kun å fastslå om klienten har eller har hatt ektefelle med trygder. Hvis et program som dette skulle brukes i praksis vil man antagelig finne denne avklaringen av sivilstatus mangelfull i og med at programmet ikke gir hjelp til å avgjøre selv enkle tolkningsspørsmål angående behandling av f.eks. separerte eller samboende.



[18] 29900 kr var grunnbeløpet ved siste justering 1987.

```
Text
Hva er sivilstatus for [klientens_navn$          17
                    (UGIFT)

sivilstatus er oppgitt
                                                  < Rule >
  IF    :Menu   KLIENTENS_SIVILSTATUS$                  9B
```

Konlusjonen "sivilstatus er oppgitt" leder imidlertid ikke umiddelbart til at konklusjonen "sivilstatus er fastlagt". Før det skjer vil vi innhente eller gi ytterligere informasjon. Først tester programmet om klientens_sivilstatus$ er UGIFT (6B). Hvis det er sant trenger vi ikke mer informasjon. Når "sivilstatus er oppgitt" er sant OG klientens_sivilstatus$ er UGIFT er sant SÅ er "sivilstatus er fastlagt" sant og programmet kan gå videre med neste betingelse på listen. Hvis klientens_sivilstatus$ ikke er UGIFT så prøves første alternative betingelse. Det testes om klientens_sivilstatus er GIFT, og isåfall blir brukeren bedt om å fylle inn navn og fødselsdato på ektefellen (6B).

```
sivilstatus er fastlagt
                                                  < Rule >
                                                    6B
 IF    sivilstatus er oppgitt
   AND :Test    KLIENTENS_SIVILSTATUS$ = "UGIFT"

 OR    :Test    KLIENTENS_SIVILSTATUS$="GIFT"
   AND :Assign ektefellens_navn$:=""
   AND :Assign ektefelles_F_aar:=0
   AND :Display Form
   AND :Assign EKTEFELLES_ALDER:= 1988 - ektefelles_F_aar
```

```
sivilstatus er fastlagt
                                                  < Rule >
                                                    7B
 OR    :Test    KLIENTENS_SIVILSTATUS$ = "ENKE/ENKEMANN"
   AND :Menu    gjenglfte$
   AND :Test    gjengifte$="JA"
   AND :Assign KLIENTENS_SIVILSTATUS$:="GIFT"
   AND :Assign ektefellens_navn$:=""
   AND :Assign ektefelles_F_aar:=0
   AND :Display Form
   AND :Assign EKTEFELLES_ALDER:= 1988 - ektefelles_F_aar
```

Neste alternative betingelse er at klientens_sivilstatus er ENKE/ENKEMANN (7B). Fordi det i § 7-2 3.ledd siste setning står at "Denne trygdetid beholdes etter ektefellens død, *med mindre vedkommende inngår nytt ekteskap.*" har vi her spurt brukeren om klienten er gift på nytt. Hvis JA får brukeren vite at klienten da regnes som GIFT, klientens_sivilstatus$ forandres til GIFT og det spørres etter ektefellens navn og fødselsdato. På grunn av Crystals begrensning til setningslogikk er det ikke mulig samtidig å behandle det tilfelle at brukeren svarer NEI på om klienten er gift på nytt. Vi måtte lage en egen alternativ betingelse hvor det testes om klientens_sivilstatus$ er ENKE/ENKEMANN og om gjengifte$ er NEI. Her satte vi ektefellens_navn$ til "avdøde ektefelle" (8B). Det siste alternative er at klientens_sivilstatus$ er SKILT. Fordi vi bare ønsker å operere med UGIFT, GIFT og ENKE/ENKEMANN gis brukeren da beskjed om at klienten regnes som UGIFT og klientens_sivilstatus$ endres til dette (8B). For alle alternativene gjelder det at når et av alternativene har slått til og operasjonene er utført, prøves ingen flere alternativer. Konklusjonen "sivilstatus er fastlagt" blir altså oppfylt og programmet går videre til neste betingelse på listen, se (2B).

```
sivilstatus er fastlagt                                          < Rule >
                                                                   8B
  OR    :Test   KLIENTENS_SIVILSTATUS$="ENKE/ENKEMANN"
    AND :Test   gjengifte$="NEI"
    AND :Assign ektefellens_navn$:="avdøde ektefelle"

  OR    :Test   KLIENTENS_SIVILSTATUS$="SKILT"
    AND :Display Form
    AND :Assign KLIENTENS_SIVILSTATUS$:="UGIFT"
```

5.5 Del (C).

Neste betingelse er "informasjon om ektefelles pensjoner er gitt". Først testes igjen sivilstatus (9B). Dersom den er UGIFT settes variabelen ektefelle_med_pensjon$ til NEI, og vi trenger ingen flere opplysninger. Neste alternativ er at sivilstatus er ENKE/ENKEMANN. I tilfelle vil vi bare vite om den avdøde ektefellen fikk trygd eller pensjon (dvs. ektefelle_med_pensjon$ = JA/NEI?). Alternativet er at sivilstatus er GIFT og ektefellens alder er større eller lik 70. Da settes ektefelle_med_pensjon til JA og ektefellenspensjonsandel (efp-andel) til 100%.

De neste alternativene vil bare aktualiseres for gifte med ektefelle yngre enn 70 år. Brukeren kan nå gi ektefelle_med_pensjon$ verdi JA eller NEI (10B). Hvis svaret er JA settes efp-andel til 100%. I motsatt fall prøves neste alternativ hvor det spørres om ektefellen får/fikk utbetalt uførepensjon eller attføringspenger i ventetid før uførepensjon kan tilstås. Igjen blir det et spørsmål om å sette ektefelle_med_pensjon$ til JA eller NEI. Hvis svaret er JA blir brukeren bedt om å oppgi hvor stor prosentvis andel av full utbetaling som blir/ble utbetalt og brukeren fastsetter efp-andel. Det siste alternativet blir bare aktuelt dersom brukeren har svart NEI på de to siste alternativene. Da settes ektefelle_med_pensjon$ til NEI.

```
informasjon om evt. ektefelles pensjoner er gitt                    < Rule >
                                                                      9B
  IF    :Test   KLIENTENS_SIVILSTATUS$="UGIFT"
    AND :Assign EKTEF_MED_PENSJON$:="NEI"

  OR    :Test   KLIENTENS_SIVILSTATUS$="ENKE/ENKEMANN"
    AND :Menu   EKTEF_MED_PENSJON$

  OR    :Test   EKTEFELLES_ALDER >=70
    AND :Assign EKTEF_MED_PENSJON$:="JA"
    AND :Display Form
```

```
informasjon om evt. ektefelles pensjoner er gitt                    < Rule >
                                                                      10B
  OR    :Menu   EKTEF_MED_PENSJON$
    AND :Test   EKTEF_MED_PENSJON$="JA"
    AND :Assign efp_andel:=100

  OR    :Menu   EKTEF_MED_PENSJON$
    AND :Test   EKTEF_MED_PENSJON$="JA"
    AND :Assign efp_andel:=0
    AND :Display Form
```

5.6 Del (D).

Kravet til minimum trygdetid er tatt hensyn til;

HVIS 1) "trygdetid er oppgitt" OG

2) "minimum 3 års trygdetid er oppfylt" OG

3) "hvilken trygdetid det skal regnes etter er avgjort" (11B).

```
sporsmaal om minimum trygdetid er tatt hensyn til
                                                    < Rule >
IF    trygdetid er oppgitt                                11B
  AND minimum 3 ars trygdetid er oppfylit
  AND hvilken trygdetid det skal regnes etter er avgjort
```

For å oppfylle 1) blir det rett og slett spurt etter klientens trygdetid (12B). Dette vil være en mangelfull prosedyre hvis programmet skulle brukes av folk uten kjennskap til reglene om bestemmelse vedrørende trygdetid. Et program som også skulle omfatte reglene om trygdetid vil bli langt mer komplekst enn dette programmet.

Når det gjelder 2) er det sporsmål om klienten kommer inn under unntaket i § 7-2 3.ledd hvor det står at "Uten hensyn til bestemmelsen i nr. 1 legges for person som nevnt i nr. 2 bokstav b ektefellens trygdetid til grunn hvis den er lengst". Hvis ektefelle_med_pensjon$ er NEI vil vi bare prøve om trygdetid er større eller lik 3 (år) (13B). Hvis den er JA vil vi ønske å kjenne til ektefellens trygdetid, og vi spør etter denne og tester om ektefellens trygdetid er større eller lik 3 (år) (13B). Merk at ovennevnte unntak gjør det tilstrekkelig at trygdet ektefelle har minimum 3 års trygdetid for at klienten skal kunne få trygd. Om ektefellen ikke har tre år tester vi nå alternativt om klientens trygdetid er 3 år[19].

[19] At vi tester ektefelles trygdetid før klientens trygdetid skyldes Crystals sekvensielle virkemåte. Crystal vil prøve om de som oppfylte ektefelle_med_pensjon$ = NEI men ikke trygdetid >= 3 oppfyller et av de neste alternativene. Hvis vi lot ektefellestrygdetid >= 3 være et eget alternativ ville alle klienter få dette prøvet. Det ønsker vi ikke fordi ikke alle er spurt om ektefelles trygdetid. Om alle derimot blir testet på trygdetid >= 3 to ganger får ingen konsekvenser.

```
< Text >
Oppgi trygdetiden til
[klientens_navn$                    ]
i antall år:

trygdetid er oppgitt
< Rule >
12B

IF    :Assign trygdetid:=0
  AND :Display Form
```

```
< Text >
Hvor lang trygdettid har
[ektefellens_navn$                  ]?

minimum 3 års trygdetid er oppfyllt
< Rule >
13B

IF    :Test    EKTEF_MED_PENSJON$="NEI"
  AND :Test    trygdetid>=3

OR    :Test    EKTEF_MED_PENSJON$="JA"
  AND :Assign EKTEF_TRYGDETID:=0
  AND :Display Form
  AND :Test    EKTEF_TRYGDETID>=3

OR    :Test    trygdetid>=3
```

Dersom noen enda ikke har oppfylt noen av de alternative betingelser vil de bli spurt om de er utenlandsk flyktning med henvisning til unntaket for slike i § 7-2 3.ledd 2.avsnitt (14B). Fordi regelen for hvorvidt disse skal få grunnpensjon henviser til en skjønnsmessig avgjørelse om hvorvidt "særlige grunner gjør det rimelig" henvises utenlandske flyktninger til å be om manuell behandling. Til slutt, hvis klienten ikke faller inn under noen av de nevnte alternative betingelser, får brukeren beskjed om at klienten ikke oppfyller betingelsene om trygdetid og ikke kan få utbetalt grunnpensjon (14B). Dette siste alternativet slutter med en FAIL kommando som bringer oss rett til "Programmet er da slutt" (1B).

```
< Text >
For flyktning gjelder særskilte unntaksregler.
Kontakt trygdekontoret for manuell behandling.

minimum 3 års trygdetid er oppfyllt
< Rule >
14B

OR    :Menu    flyktning$
  AND :Test    flyktning$="JA"
  AND :Display Form
  AND :Fail

OR    :Test    flyktning$="NEI"
  AND :Display Form
  AND :Fail
```

```
hvilken trygdetid det skal regnes etter er avgjort
                                                    < Rule >
  IF    :Test   EKTEF_MED_PENSJON$="NEI"                    15B
    AND :Assign TRYGDETID$:="egen trygdetid"

  OR    :Test   trygdetid>=EKTEF_TRYGDETID
    AND :Assign TRYGDETID$:="egen trygdetid"

  OR    :Assign trygdetid:=EKTEF_TRYGDETID
    AND :Assign TRYGDETID$:="ektefellens trygdetid"
```

Hvis en av betingelsene før denne siste blir oppfylt, går programmet igang med å prove å oppfylle 3) (15B). Det første alternativet er at ektefelle_med_pensjon$ er NEI, og trygdetiden det skal regnes etter settes da til "egen trygdetid". For alle andre er ektefelle_med_pensjon JA, så det trenger vi ikke teste. For disse prøver vi om klientens trygdetid er større eller lik ektefellens trygdetid. Hvis det er sant settes også trygdetiden det skal regnes etter til "egen trygdetid". De som ikke har oppfyllt noe alternativ må ha kortere trygdetid enn ektefellen. I siste alternativ lar vi dem overta ektefellens trygdetid og setter at trygdetiden det regnes etter er "ektefellens trydgdetid". Vi er nå ferdige med å behandle spørsmål om trygdetid.

5.7 Del (E).

Vi vil nå behandle betingelsene for at "grunnbeløpets relative størrelse er utregnet" (16B) [20]. Vi tester først om klienten er UGIFT. Er klienten det, holder det å vise *første del av setningen i § 7-2 2.ledd del a.* "Full grunnpensjon utgjør grunnbeløpet dersom pensjonisten er ugift ...", og sette den andel av grunnbeløpet det skal regnes ut ifra lik grunnbeløpet. Neste alternativ gjelder for ENKE/ENKEMANN. For disse kan det være at Ektefelle_med_pensjon$ er JA, men dette får bare relevans for

[20] Fordi vår versjon av Crystal hadde en lite hensiktsmessig begrensning, som gjorde at en variabel ble vist med like mange plasser på skjermen som det er bokstaver i variabelnavnet, begynner vi med å overføre grunnbeløpet til en variabel kalt grunnB. Dette var en prosedyre vi flere ganger grep til for å få både leselig program og leselige skjermbilder.

hvilken trygdetid som skal legges til grunn. Avdøde ektefelle får jo ikke lenger pensjon, og de faller derfor inn under *2.ledd del a* når det gjelder andelen av grunnbeløpet det regnes ut i fra. Dette får brukeren melding om og andelen fastsettes som for ugifte. I neste alternativ tester vi om ektefelle_med_pensjon$ er NEI. Da viser vi også *hele del a. av 2.ledd* og fastsetter andelen på samme måte som for ugifte (17B).

```
                                                                    < Test >
    § 7-2, 2. ledd, del A:

    "Full grunnpensjon UTGJØR grunnbeløpet dersom
     pensjonisten er ugift..."

grunnbeløpets relative størrelse er utregnet
                                                                    < Rule >
                                                                      16B
  IF     :Assign grunnB:=GRUNNBELØPET
    AND  :Test   KLIENTENS_SIVILSTATUS$="UGIFT"
    AND  :Display Form
    AND  :Assign GRUNNBELØPANDEL:=GRUNNBELØPET

  OR     :Test   KLIENTENS_SIVILSTATUS$="ENKE/ENKEMANN"
    AND  :Display Form
    AND  :Assign GRUNNBELØPANDEL:=GRUNNBELØPET
```

```
grunnbeløpets relative størrelse er utregnet
                                                                    < Rule >
  OR     :Test   EKTEF_MED_PENSJON$="NEI"                            17B
    AND  :Display Form
    AND  :Assign GRUNNBELØPANDEL:=GRUNNBELØPET

  OR     :Test   KLIENTENS_SIVILSTATUS$="GIFT"
    AND  :Test   EKTEF_MED_PENSJON$="JA"
    AND  :Test   efp_andel=100
    AND  :Display Form
    AND  :Assign GRUNNBELØPANDEL:=GRUNNBELØPET*0.75
```

De som ikke har oppfyllt de ovenstående alternativene må være GIFT og ha ektefelle_med_pensjon$ = JA (17B). Når vi likevel har testet dette, var det for å prøve om ikke programmet ble mer leselig ved å føye dette til. Det første av de to siste alternativene prøver om efp-andel er 100%. Om det er tilfelle vises brukeren *første setning av 2.ledd del b.* Andelen av grunnbeløpet det skal regnes fra settes til grunnbeløpet ganger 0.75. Det siste alternativet er for de hvis ektefelle har mindre enn 100% trygd (18B). Brukeren får da se *siste setning av 2.ledd del b.* Andelen av grunnbeløpet det skal regnes fra settes til (1 - (0.25 * (efp-andel/100))) * grunnbeløpet. Med dette er vi ferdige med alle alternativene for fastsettelse av grunnbeløpets relative størrelse.

```
grunnbeløpets relative størrelse er utregnet                    < Rule >
                                                                        18B
  OR    :Test    KLIENTENS_SIVILSTATUS$="GIFT"
    AND :Test    EKTEF_MED_PENSJON$="JA"
    AND :Test    (efp_andel>0) & (efp_andel<100)
    AND :Display Form
    AND :Assign GRUNNBEL@PANDEL:=1-(0.25*(efp_andel/100))
    AND :Assign GRUNNBEL@PANDEL:=GRUNNBEL@PANDEL*GRUNNBEL@PET
```

5.8 Del (F).

For å forklare brukeren hvordan grunnpensjonen blir regnet ut i forhold til trygdetid har vi laget et eget sett av betingelser som bare har til oppgave å gi brukeren en forklaring som er tilpasset klienten. Slike forklaringer som brukt i dette programmet vil antagelig ikke være så spennende for en bruker som rutinemessig skulle bruke et slikt program for å beregne grunnpensjonen. I så fall ville vi be programmet skrive ut forklaringene istedet som en forklaring til klienten.

Forst alternativ gjengir § 7-2 3.ledd forste og annen setning til de som ikke har ektefelle med pensjon, og klientens trygdetid gjengis (19B). Andre alternativ gjengir tilsvarende for de som har ektefelle med pensjon men hvor trygden regnes etter deres egen trygdetid. Tredje alternativ gjengir tilsvarende med tillegg av unntaket om at det for de som er gift og har ektefelle med pensjon og hvor ektefellen har lengst trygdetid, regnes etter ektefellens trygdetid. Fjerde alternativ likeldedes som tredje alternativ, men for ENKE/ENKEMANN og med tillegg at trygdetiden beholdes etter ektefellens død (20B).

```
                                                                < Text >
  § 7-2, 3. ledd, 1. og 2. setning:

  "FULL grunnpensjon ytes til den som har vært
   trygdet i minst 40 år.

andel av grunnpensjon som utbetales er forklart                 < Rule >
                                                                        19B
  IF    :Assign TT:=trygdetid
    AND :Test    EKTEF_MED_PENSJON$="NEI"
    AND :Display Form

  OR    :Test    EKTEF_MED_PENSJON$="JA"
    AND :Test    TRYGDETID$="egen trygdetid"
    AND :Display Form

  OR    :Test    KLIENTENS_SIVILSTATUS$="GIFT"
```

```
andel av grunnpensjon som utbetales er forklart
                                                        < Rule >
                                                          20B
    AND :Display Form

  OR    :Test    KLIENTENS_SIVILSTATUS$="ENKE/ENKEMANN"
    AND :Test    EKTEF_MED_PENSJON$="JA"
    AND :Display Form
    AND :Display Form
```

5.9 Del (G).

Den siste betingelse på listen er at "størrelsen på grunnpensjonen er oppgitt klienten". For å oppfylle dette begynner vi i bladnoden med å teste om trygdetiden er over 40 år. De som har mindre enn 40 års trygdetid får en forholdsmessig del av full grunnpensjon. Vi har derfor regnet ut grunnpensjonen ved å dele andelen det skal regnes ut i fra på 40. Derfor må vi begynne med å sette trygdetiden ned til 40 tellende år for alle de som har mer enn 40 års trygdetid for at de ikke skal få mer enn full grunnpensjon (21B). Når dette er gjort så følger det at "max tellende trygdetid 40 år er prøvet". Belopet klienten får utbetalt settes lik grunnbelopandelen ganget med trygdetiden delt på 40 (22B). Til slutt vises brukeren resultatet av utregningen.

```
Max tellende trygdetid 40 ar er prevet
                                                        < Rule >
                                                          21B
  IF    :Test    trygdetid<=40

  OR    :Test    trygdetid>40
    AND :Assign trygdetid:=40
```

```
                                                        < Text >
        [klientens_navn$                    ]
        fار utbetalt kr.[UTBET],-
        i grunnpensjon.
```

```
storrelsen pa grunnpensjonen er oppgitt klienten
                                                        < Rule >
                                                          22B
  IF    Max tellende trygdetid 40 ar er prevet
    AND :Assign UTBET:=GRUNNBELØPANDEL*trygdetid/40
    AND :Display Form
```

5.10 Tilbake til konklusjonene.

Når brukeren trykker return etter å ha sett resultatet av utregningen er den siste betingelsen for at "grunnpensjonen er fastsatt" oppfylt (5A) [21]. Programmet kommer nå til kommandoen CONCLUSION DISPLAY som viser Crystal Master Rule hvor det står "Dialogen er slutt" (4A). Programmet er da slutt.

[21] Den neste betingelsen for at dialogen er slutt er som sagt "NOT tilleggspensjonen er fastsatt". Programmet spør brukeren "tilleggspensjonen er fastsatt? YES/NO". Når brukeren svarer: NO er alle betingelsene for hovedkonklusjonen oppfylt.

6. EKSEMPEL PÅ DIALOG MED PROGRAMMET "ALDERSPENSJON".

Etter å ha lastet inn programmet fra (Files) (Load old file) og (ALDERSP) kan vi gå til (Run) for å starte programmet. Vi trykker linjeskift og får dette skjermbildet (1C):

```
                                                          1C
   Oppgi klientens navn:
                    PEDER AAS

   Oppgi klientens fødselsår:
                        1918
```

Vi har her fyllt inn "Peder Aas" født "1918". Programmet tester at han er over 67 år og faller inn under § 7-1 (2C):

```
                                                          2C
   § 7-1.
   "Rett til alderspensjon inntrer ved fylte 67 år.
    ****
    Alderspensjon består av grunnpensjon og
    tilleggspensjon"

    Om PEDER AAS
    fyller de vilkår som følger senere i programmet
    kan PEDER AAS
    få utbetalt grunnpensjon og tilleggspensjon.
```

Programmet prøver så å beregne grunnpensjonen. Fordi det gjøres unntak fra regler om trygdetid hvis klienten har ektefelle som får alderstrygd, uførepensjon eller attføringspenger prøver programmet først å fastlegge sivil status (3C):

```
                                                          3C
   Hva er sivilstatus for PEDER AAS

                  UGIFT

                   GIFT

                  ENKE/ENKEMANN

                  SKILT
```

Her kan vi se et eksempel på at Crystals måte å vise frem sine regler hvis vi bruker Explain funksjonen F1, ikke virker særlig oppklarende (4C):

```
4C
Hva er sivilstatus for PEDER AAS                    ?

UGIFT

I am trying to decide whether
  sivilstatus er fastlagt                                  [ - ]
    by checking whether
      sivilstatus er oppgitt                               [ - ]
  and KLIENTENS_SIVILSTATUS$ = "UGIFT"                     [ - ]
or    KLIENTENS_SIVILSTATUS$="GIFT"                        [ - ]
or    KLIENTENS_SIVILSTATUS$ = "ENKE/ENKEMANN"             [ - ]
```

Vi valgte ENKE/ENKEMANN, og vi får da spørsmål om gjengifte (5C). Unntaksregler gjelder også for klienter hvis avdøde ektefelle hadde de nevnte trygder, men ikke for slike klienter som er gift på nytt[22]. Vi svarte NEI.

```
5C
Er PEDER AAS
gift på nytt?

JA

NEI
```

I dette tilfellet prøver programmet om avdøde ektefelle fikk trygder som nevnt over (6C) [23]:

[22] Gjengifte får beskjed om å oppgi ektefellens navn og fødselsår. Dvs. de kommer da i kategorien GIFT igjen, og også den nye ektefellen kan jo være trygdet og dermed gjøre unntaket gjeldene.

[23] Dersom dette hadde dreid seg om GIFTE ville vi ønsket mer detaljert kunnskap, for mens ENKE/ENKEMANN får grunnpensjonen regnet ut fra hele grunnbeløpet så får GIFTE der ektefellen får trygd

Vi svarte JA.

```
                                                        6C
Fikk avdøde ektefelle utbetalt alderspensjon,
uførepensjon eller attføringspenger i
ventetid før uførepensjon?

                    JA

                    NEI
```

Nå spør programmet om klientens trygdetid (7C):

Vi svarte 0 år.

```
                                                        7C
Oppgi trygdetiden til
FEDER AAS
i antall år:

0
```

Bare hvis klienten har eller har hatt ektefelle med trygd vil programmet også sporre etter ektefellens trygdetid. Dersom klienten ikke har/har hatt ektefelle med trygd og ikke har vært trygdet i 3 år, blir det spurt om han er utenlandsk flyktning. Er han det blir han bedt om å kontakte trygdekontoret for manuell behandling. Er han ikke flyktning får klienten beskjed om at han ikke har krav på grunnpensjon, og programmet slutter.

Dersom klienten har/har hatt ektefelle med trygd, prøves også ektefellens trygdetid (8C). Hvis også ektefellens trygdetid er

grunnpensjonen utregnet ut fra et redusert grunnbeløp. Først spørres det etter alderspensjon fordi dette alltid er full pensjon eller ikke. Hvis vi svarer NEI, så spørres det etter uførepensjon eller er en attføringspenger. Om vi nå svarer JA må vi oppgi en prosentandel mellom 1 og 100 fordi det er særskilt utregning av grunnbeløpets størrelse for de som har ektefelle med bare en del av full pensjon.

mindre enn 3 år slutter programmet på samme måte som nevnt. Har ektefellen 3 års trygdetid eller mer går programmet videre.

Vi svarte 30 år.

```
                                                        8C
Hvor lang trygdettid har
avdøde ektefelle                                  ?

30
```

Nå regnes grunnbeløpet ut og programmet gir brukeren melding om hvordan dette skjer. For klienter uten ektefelle med trygd, dvs. UGIFTE, ENKER/ENKEMENN, SKILTE og GIFTE uten ektefelle med trygd, regnes grunnpensjonen ut fra et uavkortet grunnbeløp. For GIFTE med ektefelle med alderspensjon, full uførepensjon eller fulle attføringspenger er grunnbeløpet 75% av fullt grunnbeløp. For GIFTE med ektefelle med delvis uførepensjon eller delvise attføringspenger regnes beløpet ut etter formelen: (1-(0.25*Ektefellens-pensjonsandel))*Grunnbeløpet.

Her er forklaringen gitt Peder Aas (9C):

```
§ 7-2, 2. ledd, del A:                                  9C

"Full grunnpensjon UTGJØR grunnbeløpet dersom
 pensjonisten er ugift eller har ektefelle som
 ikke oppebærer alderspensjon, uførepensjon
 eller attføringspenger i ventetid førufø-
 pensjon kan tilstås"

For ENKE/ENKEMANN regnes grunnpensjon ut fra et
uavkortet grunnbeløp, for tiden kr.   29900,-
```

Dernest skal det utregnes hvor stor del av grunnbeløpet som skal utbetales avhengig av trygdetid. Hvis klienten har/har hatt ektefelle med trygd skal det regnes ut fra den av dem som har lengst trygdetid. Programmet sier fra hvordan trygdetid er tatt hensyn til og hvilken trygdetid det regnes ut ifra, (10C) og (11C):

```
§ 7-2, 3. ledd:                                                  10C

"FULL grunnpensjon ytes til den som har vært
 trygdet i minst 40 år.

 Har vedkommende vært trygdet i mindre enn
 40 år, utgjør grunnpensjonen
 EN FORHOLDSMESSIG DEL av full grunnpensjon."

                                                        forts.
```

```
forts. § 7-2, 3. ledd:                                           11C

"Uten hensyn til bestemmelsene i nr. 1 legges
for person som nevnt i nr. 2 bokstav D (dvs.
hvis ektefelle FAR/FIKK trygd) ektefellens
trygdetid til grunn hvis den er lengst. Denne
trygdetid beholdes etter ektefellens død, med
mindre vedkommende inngår nytt ekteskap."
                                   Utregningen skjer etter
ektefellens trygdetid   som er  30 år.
```

Fra og med 40 års trygdetid får man hele den sum (grunnbeløpet) som ble regnet ut ovenfor. Ved mindre en 40 års trygdetid får man en lik Klentens trygdetid delt på 40. Dette regner programmet ut helt til slutt og sier så hva klienten får utbetalt (12C):

```
                                                                 12C
PEDER ÅAS
får utbetalt kr.  22425,-
i grunnpensjon.

Press F1 for explain or any other key to continue
```

7. BESKRIVELSE AV TAXMAN: ET EKSPERIMENT I KUNSTIG INTELLIGENS OG RETTSLIG RESONNERING.

For å kunne sette vårt arbeid med å modellere Trygdelovens §7,1-2 med Crystal i perspektiv, har vi funnet det hensiktsmessig å gi en beskrivelse av Thorne McCartys modell av "the taxation of corporate reorganization" kalt TAXMAN[24]. Dette programmet er laget i programmeringsspråket Micro-PLANNER som er et subset av programmeringsspråket PLANNER[25].

Micro-PLANNER er konstruert ut fra paradigmet om "semantic information processing". Ideen i dette paradigmet var å lage programmeringsspråk som gjorde det mulig å utvikle modeller som ihvertfall kunne sies å være grove tilnærminger til konseptuelle strukturer slik man tenker seg de forekommer i menneskesinnet. Utgangspunktet er beskrivelser av enkle fakta. Disse kan kombineres på ulike måter til komplekse begreper. Man kan lage et eller flere hierarkier av slike begreper med økende kompleksitet. Slike strukturer kan tegnes som diagrammer, de har derfor fått navnet "semantiske nettverk".

Så langt beskrevet er dette en statisk modell. Det er også mulig å beskrive handlinger som endrer strukturen. Disse kan også bygges opp fra enkle primære handlinger til mer komplekse kombinasjoner av enkle handlinger som utgjør lengre handlingssekvenser. Slik kan man bygge opp en modell som når den kjøres i maskinen simulerer komplekse transaksjonshistorier. La oss illustrere hvordan dette skjer med McCartys egne eksempler:

McCartys utgangspunkt er USAs Internal Revenue Codes (IRC), nærmere bestemet noen paragrafer om "tax-free corporate reorgani-

[24] L. Thorne McCarty "Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning", Harward Law Review 90, pp 837-893, 1977.

[25] Micro-PLANNER er et høynivåspråk skrevet i LISP. For nærmere informasjon om dette språket henviser McCarty til G.Sussman, T.Winograd, & E.Charniak, Micro-PLANNER Reference Manual, AI Memo No. 203 (MIT 1970).

zations" og den rettspraksis som har angått disse paragrafer. IRC section 354 sier at det ikke ansees å forekomme noe tap eller gevinst hvis "stock or securities in a corporation a party to a reorganization are, in pursuance of the plan of reorganization, exchanged solely for stock or securities ... in a corporation a party to the reorganization". Begrepet "reorganization" er sentralt og det er derfor nærmere spesifisert til 6 ulike typer reorganisering. McCartys mål var å lage et program som simulerte alle de transaksjoner av "stock and securities" som hadde funnet sted i en sak, og som deretter kunne si om transaksjonshistorien gjorde det mulig å klassifisere saken som en eller flere av disse seks typene av skattefri reorganisering.

Et enkelt fakta kan i Micro-PLANNER beskrives som et predikat med konstanter:

(CORPORATION NEW-JERSEY) *dvs.*: NEW-JERSEY er en CORPORATION.
Det første abstraksjonstrinn er at man innfører variabler:

(ISSUE NEW-JERSEY S1) : NEW-JERSEY utgir (issue) s1,
(STOCK S1) : S1 er en aksje (stock),
(COMMON S1) : S1 er vanlig (av typen COMMON stock).

I kunnskapsbasen ligger nå denne listen med fakta:

(CORPORATION NEW-JERSEY)
(ISSUE NEW-JERSEY S1)
(STOCK S1)
(COMMON S1)

Vi kan så introdusere en aksjeeier:

(OWN PHELLIS P1) : Phellis eier P1
(PIECE-OF P1 S1) : P1 er en del av S1
(NSHARES P1 100) : P1 er antall aksjer 100

Micro-PLANNER har en kommando ASSERT for å legge dette inn i kunnskapsbasen, ERASE for å slette det og GOAL for å finne igjen det som er innlagt. Skriver man (GOAL (CORPORATION

NEW-JERSEY)) får man svar T for true. Hvis dette fakta ikke hadde ligget i databasen hadde vi fått svar NIL for falskt/finnes ikke. Skriver vi (GOAL (CORPORATION ?X) vil programmet søke etter mønsteret (CORPORATION X) der X kan være et hvilket som helst navn, og vi får det første navnet den støter på. Skriver vi derimot (FIND ALL (CORPORATION ?X) og vi får en liste over navnet på alle korporasjoner, for tiden bare NEW-JERSEY.

Hvis vi ønsker å finne om Phellis er en aksjeeier med aksjer i NEW-JERSEY må vi bygge en mer kompleks foresporsel:

```
(PROG (S P)
  (GOAL (ISSUE NEW-JERSEY S1))
  (GOAL (STOCK ?S))
  (GOAL (PIECE-OF ?P ?S))
  (GOAL (OWN PHELLIS ?P)) )
```

S og P blir her brukt som variabler i beregningen på samme måte som X ble. Listen med GOAL kommandoer ansees å være en rekke operasjoner som skal utføres etter hverandre. Hvis den får oppfylt det første målet, men ikke det neste målet vil Micro-PLANNER la programmet gå tilbake (backtrack) og lete på nytt etter et nytt fakta som kan oppfylle det første målet og også passe med det neste. McCarty skriver:

> "To sum up: We want the program in this example to determine whether there are two objects satisfying the four propositions that make up the "stockholder" relation. To do this it must match the two variables and satisfy the four commands simultaneously, an essentially *parallel* task, but it accomplishes this result by a trial-and-error search, a *sequential* procedure."

Forelopig har vi bare laget en kompleks foresporsel, men Micro-PLANNER lar oss også representere det absrakte begrepet "aksjeeier". Vi beskriver da begrepsstrukturen innen en kommando kalt THEOREM. "Aksjeeier" kan beskrives:

```
(THEOREM ABSTRACT (O C S P)              :(THEOREM ABSTRACT
<variabelliste>

  (STOCKHOLDER ?O ?C)                    : <begrepsmønster>

    (GOAL (ISSUE ?C ?S))               : <sekvens av operasjoner>)
    (GOAL (STOCK ?S))
    (GOAL (PIECE-OF ?P ?S))
    (GOAL (OWN ?O ?P)) )
```

Intuitivt uttrykker dette at begrepet STOCKHOLDER er på en abstrakt måte er ekvivalent med et spesielt mønster av begrepene ISSUE, STOCK, PIECE-OF og OWN. Hvis vi heretter ønsker å finne om Phellis eier aksjer i NEW-JERSEY kan vi skrive:

```
(GOAL (STOCKHOLDER PHELLIS NEW-JERSEY) ABSTRACT)
```

Vi har beskrevet hvordan McCarty kan bygge begreper av økende kompleksitet. Han skisserer også hvordan man kan forfine et begrep med kommandoen:

```
(THEOREM EXPAND (O C ....)
  (STOCKHOLDER ?O ?C)
    (ASSERT (...))
    (ASSERT (...))
      ....
    ....          )
```

Senere kan vi da enkelt lage nye aksjeeiere ved gi kommandoen:

```
(ASSERT (STOCKHOLDER PHELLIS NEW-JERSEY) EXPAND)
```

McCarty skriver:

> "It is then possible to query the system at the higher level with a (GOAL <conceptual pattern> ABSTRACT) command, and have the lower-level searches carried out automatically: or state a fact at a higher level with an (ASSERT <conceptual pattern> EXPAND) command, and have the lower-level representations generated automatically. This process of conceptual abstraction can be extended to more than one level... Continuing in this way,

an abstraction hierarchy of any desired complexity could be developed."

Slik bygges en statisk struktur. Det neste McCarty gjør er å innføre en tidsakse. Alle fakta er nå tilfelle til en viss tid. Vi får fakta som:

(ISSUE NEW-JERSEY S1 T4) : NEW-JERSEY utstedte S1 på T(id) 4

Så beskriver han en "overføringshandling" TRANS(-fer):

```
(THEOREM EXPAND (T P O R TA TB)

  (TRANS ?T ?P ?O ?R ?TB)

  (IF          (GOAL   (OWN ?O ?P ?TA))
       THEN    (ERASE  (OWN ?O ?P ?TB))
               (ASSERT (OWN ?R ?P ?TB)) )
```

Hvis vi nå onsker å uttrykke at Phellis overfører eiendommen P1 til Delaware på tiden T4 kan vi nå skrive:

(ASSERT (TRANS PHELLIS P1 PHELLIS DELAWARE T4) EXPAND)

dermed forandres modellen fra å inneholde utsagnet (OWN PHELLIS P1) til å inneholde utsagnet (OWN DELAWARE P1).

Slik bygger Thorne McCarty opp abstrakte begreper som "å overføre en del av sine aksjer", "å distribuere aksjer", "å konvertere aksjer pr. antall og pr. kurs", "å kontrollere et selskap" og til slutt "særskilte typer av reorganisering"

McCarty lykkes ikke helt med å beskrive alle de 6 typene av reorganisering, for i noen av dem støter han på vage begreper som "in pursuance of the plan of reorganization" som ikke lett lar seg uttrykke i språket Micro-PLANNER. Han kritiserer også sin modell for å være alt for begrenset. Programmet er også for utilgjengelig for folk uten EDB-kjennskap. Likevel oppnådde han å avsløre enkelte tvetydigheter i reorganiseringsbegrepet. Disse viser seg ved at programmet finner en delvis oppfyllelse av flere typer reorganisering å kunne være tilfelle samtidig. McCarty

mener også det er mulig å utvide modellen med en mer sofistikert tidsmodell og med mulighet for å operere på partielle/ufullstendige beskrivelser. Han ville gjerne utstyrt programmet med et grensesnitt for kommunikasjon i naturlig språk, og han ville gjerne ha utvidet modellen til et større juridisk domene.

Målet er å få et system som kan:
1. Finne igjen enkelt saker.
2. Analysere ny saker som presenteres for systemet.
3. Planlegge hvordan en transaksjon må foregå for å være av en gitt type.
4. Være et verktøy for utvikling av teorier om rettslig resonnering.

I sin konklusjon skriver McCarty:

> "In general, then, the detailed analysis of the TAXMAN system has tended to support what should surely be a lawyer's intuition: that the current TAXMAN paradigm fails to capture many of the significant facts about the structure of legal concepts and the process of legal reasoning. I do not believe, however, that this is the final word on the subject. A systematic exploration of the limitations of the current TAXMAN paradigm would, I believe, lead us to a modified paradigm which could correct some of these limitations and permit us to say something about the structure and dynamics of even the more vaguely defined concepts of corpotrate reorganization law."

Vi slutter oss til siste setning som en beskrivelse av målet med forskning angående representasjon av rettslige normer.

Om vi sammenligner McCartys modell med den vi har laget av folketrygdlovens § 7,1-2 ser vi at de er svært ulike. McCartys siktemål var å konstruere en strukturert og detaljert modell av rettslige begreper som "shareholder" og "corporate reorganization". Vårt program er prosedyreorientert, det kan bare brukes til det ene formål å fastsette grunnpensjon. McCartys representasjon er derimot deklarativ, den kan anvendes til flere ulike formål ved at programmet kan gi svar til helt ulike spørsmål. Vår modell ville kunne falle inn under det McCarty kaller "computer-assisted instruction programs". Vi vil derfor gjengi hele det avsnitt der McCarty kritiserer slike programmer:

"The second approach to computer-aided legal research today is based upon the methods of computer-assisted instruction (CAI). The computer poses questions designed to elicit the essential facts of a case, and then, depending on the information received, either suggests a tentative legal analysis or poses an additional factual question. There are parallels here, of course, to the way an extended version of TAXMAN might develop the description and analysis of a case. But note the rigid structure of these CAI-based systems: the user faces a series of preprogrammed questions and usually responds in a strict multiple-choice format; the author of the program faces the tedious task of anticipating all possible responses and programming explicitly every logical branch.

In the extended version of TAXMAN, by contrast, the most important and difficult work is done at the start, when the description and analysis mechanisms are designed to capture the basic conceptual structure of the problem domain. Once these basic conceptual structures are written, the "analysis" and "planning" modes of the system can then be programmed more easily, in a uniform and systematic way. For the user of the system this permits a much greater flexibility in the interactive process, and for the author of the program it requires much less concern about the specific details of the interaction. In general, then, the TAXMAN approach appears to have much greater long-range potential than the CAI approach as a technique for computer-aided legal research, either for the law student or for the practicing attorney."

Om vi derfor er enige med de forskningsmål McCarty framsatte synes det å lage programmer i et ekspertsystemskall av typen Crystal å bringe oss et steg bakover isteden for et steg framover. Programmeringsspråket micro-PLANNER har derimot flere likheter med programmeringsspråket PROLOG[26]. Dette er interessant fordi PROLOGs egenskaper er vel kjent og språket er anerkjent og utbredt. Som illustrasjon på likheten vil vi beskrive begrepet "stockholder" i henholdsvis micro-PLANNER og PROLOG syntaks:

[26] PROLOG er akronym for PROgramming in LOGic.

micro-PLANNER:

```
(THEOREM ABSRACT (O C S P)
  (STOCKHOLDER ?O ?C)
    (GOAL (ISSUE ?C ?S))
    (GOAL (STOCK ?S))
    (GOAL (PIECE-OF ?P ?S))
    (GOAL (OWN ?O ?P)) )
```

PROLOG:

```
stockholder(O,C):-
  issue(C,S),
  stock(S),
  piece_of(P,S),
  own(O,P).
```

Om man ønsker å drive forskning ut i fra McCartys målsetning. synes det å være mer nærliggende å bruke PROLOG enn Crystal.

8. VANSKER MED Å LAGE OG OPPDATERE CRYSTAL-PROGRAMMER.

Det er verdt å gjenta et sitat fra Thorne McCarty:

> "But note the rigid structure of these CAI-based systems: the user faces a series of preprogrammed questions and usually responds in a strict multiple-choice format; the author of the program faces the tedious task of anticipating all possible responses and programming explicitly every logical branch".

Dette er en god beskrivelse av en av de største vanskelighetene med å lage Crystal-programmer. Man må til enhver tid ha full oversikt over alle de logiske betingelser og følger som impliseres av det man har gitt av kommandoer. Når man endrer en betingelse må man fort gjore endringer i alle de delene av programmet som har noen logisk sammenheng med den endrede betingelsen. Dette gjor det vanskelig å *lage* Crystal og enda vanskeligere å gå inn og *endre* på dem ved en senere anledning.

Disse vanskene forsterkes av at det er umulig å få noen oversikt på skjermen over reglene i programmet. Man er til enhver tid i et av programtreets grener, man ser aldri treet utenfra eller noen liste over reglene man har laget[27]. Hvis man ønsker å flytte betingelser eller bare sin oppmerksomhet fra en gren til en annen, må man bevege seg bakover fra der man er helt inn til punktet hvor grenene deler seg og så utover den nye grenen. Dette krever en rekke operasjoner som stadig må gjentas og er lite smidig. Et PROLOG program kan man derimot redigere på samme måte som man redigerer en tekst, og man kan flytte seg fritt i programteksten.

For å få noen oversikt over et Crystal program, må man lage en utskrift av programmet på papir. Hver regel får et nummer som

[27] I en nyere versjon har man muligheten for å få tegnet opp et grafisk bilde av treet. Men det er bare "rule lines" (det jeg har kalt konklusjoner) som tas med og ingen kommandoer. Det er lett å se at treet for "Alderstrygd" da ville bli ufullstendig tegnet fordi flere bladnoder og alternative betingelser består utelukkende av kommandoer. Men det er trolig at kommende versjoner av Crystal vil løse problemene med å få oversikt over programmet.

brukes til å referere hvor man finner den i utskriften, men i vår versjon av Crystal ble numrene ikke tildelt etter noen forståelig systematikk utover at toppen av treet alltid kommer til sist. Numrene varierte dessuten hver gang man laget en ny utskrift. Hver gang hadde man derfor et enormt puslespill med å finne igjen hvordan betingelsesgrenene hang sammen[28].

Av dette kan det sluttes at man antagelig raskt vil miste oversikten når regeltreet begynner å bli omfangsrikt. Når man mister oversikten vil man lage logiske feil slik at programmet ikke lenger opererer som forutsatt. Om man har ambisjoner om mer enn å lage et lite demonstrasjonsprogram kan man fort måtte operere med flere hundre eller endog flere tusen regler. Da vil nok problemene med manglende oversikt melde seg.

For å unngå slike problemer når man skal lage større programmer er det ønskelig å kunne dele programmet opp i uavhengige moduler som har klart spesifiserte grensesnitt for datautveksling med hverandre. Om programmet trenger å endres skal man da kunne endre de aktuelle modulene uten at det er nødvendig å endre hele den øvrige programstrukturen. Dette er svært viktig hvis man skal kunne holde programmet oppdatert. Oppdatering er et alvorlig problem innen det juridiske domenet fordi de prosesser programmene automatiserer bygger på et rettslig regelverk som stadig endres. Modularisering vil forutsette kompilering, dvs. at hele moduler oversettes til kjørbar kode og lenkes sammen. Crystal er derimot interpreterende, dvs. den oversetter linje for linje av programmet.

Større ekspertsystemskall har likevel muliggjort *en form for* inndeling gjennom bruk av rammer og klassehierarkier, dvs. at man samler bolker av regler og spesifierer forbindelsen til andre slike bolker. Vår versjon av Crystal hadde ingen slike konstruksjoner. I Crystal var det bare to muligheter for noen slags inndeling. Det ene var muligheten til å utveksle data med andre programmer som

[28] I en nyere versjon av Crystal er det mulig å få utskrift av enkeltregler eller enkelte hele grener av treet. Dette gjør det kanskje litt lettere å få oversikt. Vi har også fått vite av forhandleren at senere versjoner av Crystal gir mer systematikk i nummereringen.

databaser og regneark. Den andre muligheten var en kommando som gjorde at Crystal fra programnivå selv kunne aktivisere andre Crystalprogrammer. Men ved å bli aktivisert fortrengte det nye programmet fullstendig det gamle fra maskinens minne. Den eneste måten å overføre data på ville vært å legge disse ut på en fil under eksekvering som det nye programmet så kunne lese dataene fra. Ingen av disse mekanismene oppveier det faktum at det ikke er mulig å inndele programmet i adskilte deler eller i uavhengige moduler.

Denne kritikken gjelder også for PC Easy. PC Plus, som PC Easy er en forenklet versjon av, ble brukt av Berg/Theisen i arbeidet med å lage et ekspertsystem for skatteregler[29]. PC Plus har "enkle mekanismer for strukturering av kunnskapsbasen". Berg/Theisen kritiserer likevel programmet:

> "Selv om det finnes muligheter for oppdeling av regelbasen i mindre deler, har disse felles navnerom.... Hvert eneste navn (og dem blir det mange av) må altså velges forskjellig for å unngå uønskede bivirkninger. Den versjonen av PC Plus som vi brukte manglet også muligheten til å strukturere data ved å spesifisere sammenheng mellom ulike navn.
>
> Den utsagnsstyrken som reglene i PC Plus gir mulighet for, er omtrent som i utsagnslogikk, dvs. at det ikke kan brukes variabler i utsagnene. Dette medfører en del dobbeltutforming av regler fordi vi i stedet for en regel
>
> Hvis over_70(x) så skal_ha_særfradrag(x)
>
> som gjelder enten x er skattyteren eller ektefellen, må ha to regler
>
> Hvis skattyter_er_over_70 så skattyter_skal_ha_særfradrag
> Hvis skattyter_er_over_70 så ektefelle_skal_ha_særfradrag"

Begge disse vanskene med å lage programmer gjelder også for PC Easy og Crystal. Dette gir også grunn til å foretrekke PROLOG hvor man har noe av utsagnskraften til 1. ordens logikk, blant annet kan man bruke variabler i utsagnene.

[29] Berg/Theisen, NR rapport no 811, jan 1988.

9. BRUKERGRENSESNITT.

Vår versjon av Crystal hadde noen enkle virkemidler til raskt å lage skjermbilder der brukeren kan skrive inn sine svar, velge fra menyer eller angi posisjon på en skala. Et Crystal program framtrer da for brukeren som en serie med slike skjermbilder. Hvilken serie som vises kommer an på hvilken vei gjennom programtreet brukersvarene leder en. Dette var den eneste mulige interaksjon med Crystal, og den ble fort svært trettende[30]. Hvis dette var et program f.eks. en saksbehandler skulle bruke et antall ganger hver dag ville vedkommende snart huske alle programmets forklaringer og ønske seg programmet uten dem. Men i vår versjon av Crystal var det ingen opsjoner for å ta med mer eller mindre tekst i rutene. Vi tror en saksbehandler ville ønske å kunne fylle alle data inn i en form for skjema framfor å traversere en slik sekvens med et spørsmål for hvert skjermbilde på nytt og på nytt. En lignende kritikk gir Berg/Theisen av PC Plus[31]:

> "PC Plus er utstyrt med et standard grensesnitt, som er svært lite fleksibelt. Grensesnittet er utformet for å ta seg av spørsmål-svar dialoger og åpner ikke for f.eks. skjemabaserte dialoger. Den eneste måten å utforme alternative grensesnitt på, er å skrive egne Lisp-rutiner for all inn- og utmating av data, noe som krever en god del arbeid."

Denne kritikken gjelder i ennå større grad for PC-Easy hvor det ikke er adgang til å lage egne Lisp-rutiner.

[30] Crystal versjon 2.10 hadde også den irriterende mangel at vi ikke fritt kunne definere størrelsen på skjermbildene. Til hvert bilde laget programmet en fast definert rute, og den eneste måten å få med lange tekster på var å bruke flere ruter etter hverandre eller redigere teksten som en hjelpetekst som programmet hentet inn fra en ekstern fil. Dette er blitt endret i den påfølgende versjon av Crystal. Det er nå større fleksibilitet til å lage de skjermbilder man ønsker.

[31] Berg/Theisen, NR rapport no 811, jan 1988.

En oppgave man tiltenker et slikt system er å kunne simulere resultatet av en beslutningsprosess ut i fra ulike input-verdier. Framfor å kjøre gjennom hele sekvensen flere ganger og fylle inn mange av de samme verdiene igjen og igjen, ville det være lettere å sette inn ulike verdier i et skjema eller regnearklignende grensesnitt og se hvordan de andre verdiene endret seg samtidig. Et slikt fast skjema vil gi brukeren en langt bedre oversikt over beslutningsprosessen fordi tidligere svar og innvirkningen på andre felter er synlig gjennom hele interaksjonen.

Dette innebærer også en kritikk av sider ved utformingen av konseptet om *dynamiske skjema*[32]. I diskusjoner av dette konseptet på Institutt for Rettsinformatikk er dynamiske skjema blitt framstilt som et skjema på skjermen der det avhengig av hvilke informasjoner som blir gitt, dukker opp vinduer med nye skjemaer som må fylles ut. F.eks. hvis man har selvangivelsen på skjermen, så ville ingen spørsmål om ektefelle være synlig før brukeren har angitt sin sivilstatus som gift. Tanken er at det er unødvendig å bry alle ugifte med spørsmål om ektefellen. Når dagens skjemaer med faste rubrikker for alle kanskje er å foretrekke er det fordi de gir en mer fullstendig oversikt. Det er ingen spørsmål som dukker opp og forsvinner igjen. Man kan hele tiden se hva som før er fyllt ut, og man kan gå tilbake og korrigere[33].

Under arbeidet med å lage programmet "Alderstrygd" var det likevel ikke utfyllingen av et fast skjema vi tenkte på som det mest nærliggende alternativ til interaksjonen med Crystal. Inspirert

[32] Om dynamiske skjema, se Jon Bing (ed), F*KUS, Knowledge based systems for public administration, CompLex 8/87.

Opphavsmann til kritikken er Dr. Trevor Bench-Capon, Liverpool University.

[33] Vi må få tilføye at den siden ved dynamiske skjema som her er kritisert ikke var med i det konsept som er beskrevet i CompLex 8/87. Ideene som der er tatt opp om gjensidig binding av feltverdier, automatisk supplering av informasjon fra eksterne databaser og ulike hjelpefunksjoner for brukeren vil fortsatt ha sin berettigelse.

av Jon Bings ideer om "konseptuell soking"[34] der rettsreglene er tenkt representert som et flytskjema, dvs. som en trestruktur eller et nettverk av noder, så vi for oss spørsmålene til brukeren hengt på disse nodene. Hvis man tenker seg skjermbildet som et brettspill ville det innebære at målet for brukeren var å komme fram til noden for "Pensjon blir utbetalt (kr:)".

Brukeren står på første node (symbolisert av markøren eller et passende symbol). Denne har teksten "Rett til alderspensjon inntrer ved fylte 67 år." og spørsmålet "Er klienten fylt 67 år?" eller evt. "Oppgi navn og fødselsdato:". Foran ham ligger et nett av åpne veier gjennom regelstrukturen markert ved at alle linjer er grønne. Hvis klienten ikke er 67 år blir alle linjer røde og brukeren får beskjed om at klienten ikke får noen grunnpensjon. Hvis derimot klienten er over 70 år flytter symbolet seg til neste node og linjen den har flyttet seg blir blå. Samtidig blir alle delene av strukturen som omhandler adgang for personer mellom 67 og 70 år til å ta ut deler av alderspensjonen røde, men det er mange andre grønne veier til "Pensjon blir utbetalt". Det bør være anledning til å gå tilbake så langt man måtte ønske, og la linjene skifte farve som tegn på at de lukkede veier åpner seg på nytt. Man kan også tenke seg at informasjon fra andre kilder enn brukeren er med på å bestemme hvilke veier gjennom strukturen som er åpne.

Et slikt grensesnitt kan tenkes kombinert med et grensesnitt for konseptuell soking. Man kan tenke seg at det er en felles representasjon av strukturen av rettsregler og at systemet kan brukes i to forskjellige modi. Først kan man bruke systemet som beskrevet over. Hvis man så ønsker å søke i rettskildene settes søkemodus på. Da forsvinner sporsmålene og brukeren kan fritt bevege seg i strukturen med piltaster eller mus. Farger kan da brukes til å markere hvilke noder brukeren ønsker å forbinde i søket, f.eks grønt for konjunksjon, gult for disjunkjson og rødt for negasjon.

Et skjemabasert grensesnitt og et regelspill som beskrevet må kanskje sees som to alternative utforminger av grensesnittet.

[34] Jon Bing, "Designing Text Retrieval Systems for Conceptual Searching", IJCAIL proceedings, ACM press, New York 1987.

Felles for dem er at de gir oversikt og lar brukeren bevege seg relativt fritt fram og tilbake mellom nodene eller postene. Slik vi ser det er et skjema mest hensiktsmessig for rutinemessig saksbehandling. Regelspill virker mer egnet som pedagogiske hjelpemidler til på en levende måte å illustrere hvilke regler som gjelder, hvordan de henger sammen og hvordan de virker. Vi kan kanskje tenke oss også disse to grensesnittene som ulike modi, og at man fritt kan velge den ene eller den andre presentasjonsform etter ønske.

10. FORKLARINGER.

En av de egenskaper ved kunnskapsbaserte systemer som har gjort dem populære er muligheten for å spørre programmet HVORFOR det stiller et gitt spørsmål og HVORDAN det har kommet fram til en gitt konklusjon. Ved et HVORFORspørsmål vil programmet vise de regler det prøver å få oppfyllt og ved HVORDANspørsmål hvilke regler det har fått oppfyllt. Dette gjør prosessen programmet utfører mer forståelig, det viser brukeren hva programmet prøver å gjøre og de logiske slutningene, resonnementet, som ledet fram til konklusjonen. Konklusjoner er lettere å akseptere når de er gitt med en forståelig begrunnelse. En forståelse av prosessene som leder fram til en konklusjon er også svært ofte viktig å bevare selv om prosessen er automatisert. Man sier at kunnskapsbaserte systemer er transparente med en eksplisitt representasjon av domenet. Dette i motsetning til konvensjonelle algoritmer hvor brukeren bare får se informasjonen som går til og fra programmet. Prosessene som behandler informasjonen er her bare implisitt representert i programinstruksjonene som er skjulte for brukeren.

I PC-easy er det programmet maskinen utfører skrevet i et BASIC lignende språk kalt ARL (Abbreviated Rule Language). Dette er leselig nok til programkode å være, men gir ikke de brukervennlige forklaringene man gjerne ønsker. Derfor blir man i PC-easy bedt om å henge på hver regel en oversettelse og en spørsmålstekst (prompt). PC-easy har også noen enkle lingvistiske regler innebygd slik at enkelte oversettelser kan gjøres om til spørsmål hvis oversettelsen skrives på engelsk. PC-easy legger også til ganske lange tekststrenger som "It is true (100%) that..." som gjør at norske forklaringer framtrer som et usannsynlig kaudervelsk av engelsk-norsk.

I Crystal skriver man konklusjoner rett inn i programmet med inntil en linje tekst i naturlig språk eller man bruker kommandoer fra kommandolisten til å lage tester, sette variabelverdier etc. Når man bruker forklaringsfunksjonen (F1 tasten) viser Crystal disse linjene fra der du er i programtreet. Hvis det er en samling heldige linjer/konklusjoner kan det bli en tilforlaterlig forklaring, men som vist i (4C) er dette ikke alltid tilfelle. Forklaringsfunksjonen er m.a.o. alt for enkel til å være til særlig hjelp. Et alternativ er å lage en egen fil med forklaringer. Man kunne også brukt en

grafisk bilde som en forklarende illustrasjon. Et annet alternativ som vi brukte mye i "Alderstrygd", er å bruke Display Form til å putte forklarende skjermbilder inn i dialogen med brukeren.

Spørsmålet her er hva som er hensiktsmessige måter å gi brukeren hjelp på. Det er blitt formulert i spissformuleringen "Naturlig språk er ikke brukervennlig". Med dette mener man å si at mange og lange tekster som brukeren må lese for å komme videre i programmet, gjør et program tungt å bruke. Å putte forklarende tekster inn i dialogen skal man derfor være forsiktig med. Det er ikke alltid ønskelig med hjelp man ikke har bedt om. At brukeren selv kan be om hjelp og få fram en forklarende tekst er derimot et velprøvd prinsipp. Slik våre enkle programmer virker kunne denne funksjonen antagelig forklart det meste brukeren trenger å vite om programmet.

Målet er å gi brukeren mest mulig relevante forklaringer. I siste instans innebærer dette langt mer enn hjelpetekster som forklarer hvilke taster man kan trykke eller hva som er syntaktisk riktig input. For å gi gode forklaringer behøver man, i tillegg til kjennskap til hvordan programmet virker, ha kjennskap til brukeren. En god forklaring forutsetter innsikt i hva brukeren tror og vet og hvilke intensjoner han handler ut i fra når han bruker programmet[35]. Selv uten ambisjoner om å modellere intensjoner og trostilstander i programmet, er det ønskelig med mest mulig kontekstsensitive forklaringer. Dette har man søkt å oppnå ved å la programmet vise fram reglene slik vi tidligere har beskrevet. Hvor forklarende dette er avhenger likevel av at reglene blir presentert på en leselig måte. I Crystal er presentasjonen ikke tilfredsstillende.

[35] Professor Andrew Jones, Inst. for filosofi, og Dr. Rolf Nossum, Oslo Ingeniørhøyskole har fremmet forslag forskningsprosjekt der de vil prøve å lage et grensesnitt som inkorporerer modeller av brukerens kunnskaper, trostilstander og planer.

11. ER CRYSTAL-PROGRAMMER KUNNSKAPSBASERTE SYSTEMER ELLER EKSPERTSYSTEMER?

En viktig stimulans til ideen om Kunstig Intelligens var tanken på å kunne konversere med datamaskinene slik man konverserer med et menneske. Ideen fikk i praksis den utforming at maskinen ble utstyrt med fakta, logiske regler for å kombinere fakta og en slutningsmekanisme for å kunne utlede nye fakta fra de logiske reglene. Med et slikt system skulle det være mulig fritt å spørre maskinen spørsmål som maskinen prøver å besvare ved å kombinere fakta på logisk mulige måter. Programmet har en *deklarativ* kunnskapsrepresentasjon som kan brukes til forskjellige formål. Thorne McCartys TAXMAN program er en god illustrasjon på et slikt system. Så lenge man bruker korrekt syntaks er det ingen grense for hva man kan spørre om, selv om det ganske sikkert er begrenset hva maskinen kan svare på. Man kan også godt tenke seg at maskinen må spørre brukeren underveis om fakta som er relevante for å kunne svare. Kanskje er den mest kjente implementasjonen av disse ideene er program-meringsspråket PROLOG.

En helt annen og mer strukturert situasjon er den vi har i Crystal, i PC-easy og i andre ekspertsystem-skall vi kjenner til. Det er her programmets enerett å stille spørsmålene, brukeren taster bare inn data akkurat som i tradisjonelle programmer. Dette er langt enklere å håndtere for programmet. Det har på forhånd fått oppgitt et (eller i f.eks. PC-easy, noen få) mål det prøver å få oppfyllt, og alle reglene er laget med tanke på å nå nettopp denne (disse få) konklusjonen(e). Det er ulike veier fram til konklusjonen, og det kan hende man bare når en konklusjon som sier at målkonklusjonen ikke kan nås. Det hele er likevel en strukturert, prosedyral prosess som ligger nær opp til vanlige algoritmer. Det kunstig intelligente man forbandt med en fri dialog er blitt ofret til fordel for effektivitet og kontroll. Konseptet bak disse systemene avviker derfor lite fra konseptet bak tradisjonell databehandling.

Forsker Tore Amble, RUNIT, har framsatt en distingsjon mellom "Ekspertsystemer" og "Kunnskapsbaserte systemer". Han ser Ekspertsystemer som en betegnelse på et system som yter det samme som en ekspert uavhengig av hvordan programmet er laget, og på Kunnskapsbaserte systemer som en betegnelse på systemer

som har en slutningsmekanisme samt en mekanisme for å representere kunnskap som regler og fakta. Tradisjonelt tenker man på ekspertsystemer som en sammensmeltning av disse egenskapene, men det har oppstått mye uklarhet etter at mange har laget ting de har kalt ekspertsystemer med tradisjonelle programmeringsteknikker eller med helt trivielle ytelser. Selve ordet "ekspertsystem" synes å peke på at systemet yter det samme som en ekspert. Begrepet lider under vanskeligheten ved å spesifisere hva som er en ekspertytelse, og det tynges av alt for mye ukritisk bruk i salgsfremmende virksomhet.

Hvorvidt man kan lage ekspertsystemer med Crystal kommer i Ambles forståelse an på hva slags ytelse man kan oppnå av programmet. Man kan tenke seg at man for å kalle det et ekspertsystem må kunne lage et stort omfattende program som har representert mye kunnskap om et domene. Vi har argumentert for at dette er Crystal svært lite egnet til. Vi tror at man med Crystal fort vil miste oversikten hvis man prøver å lage store programmer. Det skal derfor mye til å få noen ekspertytelse ut av et Crystal program.

Kan man likevel si at et Crystalprogram er et "kunnskapsbasert system"? I et Crystalprogram (og dette gjelder også for PC-easy) er all kunnskapen representert som regler. Det har ingen faktabase det henter kunnskap fra. Det eneste det henter inn fra andre programmer (f.eks. dBASE programmer), eller filer, eller fra bruker er data. Dataene brukes til å sette verdier til variablene i reglene. I så måte fungerer disse programmene akkurat som tradisjonelle programmer. For ordens skyld; et datum er ikke et faktum. Et datum f.eks. tallet 72, sier ingenting før det knyttes til en variabel f.eks. "alder", men "alderen er 72" er et faktum. Crystal har m.a.o. ingen kunnskapsbase i den vanlige betydning av ordet. Det har bare regler for å gå ut å spørre etter data og for å styre hva det skal gjøre med dem. PC-easy kan i tillegg ha data lagret som en default-verdi som variabelen skal få om den ikke kan få satt noen verdi på annen måte.

Det er en viktig forskjell mellom Crystal og PC-easy. PC-easy har alle reglene lagret som en liste. Det får oppgitt at en eller fler av dem er målregler det skal prøve å oppfylle. Det søker å få sine mål oppfyllt gjennom flere strategier. Først ved å se om det er andre regler som kan gjøre betingelsene for målet sanne.

Det kan godt ligge regler i basen som sjelden eller aldri kommer til anvendelse. Når den er kommet til regler som har betingelser ingen andre regler kan gjøre sanne søker den å få oppfylt betingelsene i denne regelen ved å hente inn data, fra en database hvis det er mulig, fra bruker eller ved default. Det programmet som styrer denne prosessen kan kalles slutningsmekanismen i PC-easy. Dette er en meget enkel slutningsmekanisme. Mer avanserte kunnskapsbaserte systemer har gjerne mer sofistikerte slutningsmekanismer.

I Crystal er det derimot ingen selvstendig mekanisme som velger strategi for å nå målet. I Crystal må programmereren selv spesifisere alle de regler programmet skal prøve for å nå dette målet og programmereren må spesifisere hvordan data skal hentes inn. Når man lager et Crystalprogram lager man et regeltre der alt har sin faste plass og alle mulige veier gjennom treet hele tiden spesifiseres. Veiene spesifiseres enten eksplisitt ved å angi betingelser som sammen eller alternativt må oppfylles for å komme videre, eller implisitt ved å sette betingelsene i en gitt rekkefølge ut fra vissheten om at Crystal prosesserer treet sekvensielt ovenfra og nedover. Dette gir muligheten for en effektiv prosessering og stor kontroll over prosessens gang. Men Crystal har likevel ingen selvstendig slutningsmekanisme, bare en sekvensiell regelprosessering som ligner svært meget på den man finner i tradisjonelle programmer.

Konklusjonen blir derfor at et Crystalprogram hverken har en kunnskapsbase eller en inferensmekanisme i vanlig forstand og heller ikke er egnet til å lage programmer som gir ytelser på ekspertnivå. Et Crystalprogram kan derfor ikke med rette kalles hverken et ekspertsystem eller et kunnskapsbasert system. I beste fall kan man si at Crystal lar en lage programmer med en regelbasert representasjonsform og at dette er en ide' fra ekspertsystem teknologien.

Å lære nye brukere å bruke Crystal kan være en enkel måte å lære dem å lage mindre programmer på, men det gir ikke brukeren en introduksjon til kunnskapsbaserte systemer. Hvis man ønsker at folk uten programmeringskunnskaper skal lære seg kunnskapsteknologiske metoder for å kunne lage modeller av det juridiske domene, må man utvikle et spesialverktøy for dette formålet. Det samme gjelder om man vil utvikle ekspertsystemer i

egentlig forstand, til bruk i beslutningsstøtte eller opplæring. De kommersielt tilgjengelige ES-skall vi har prøvet var ikke gode nok for våre behov.

Vi mener det er behov for å lage spesialsydde programmer for anvendelser som skal omfatte rettslige normer.

12. TAKK TIL ...

Dag Wiese Schartum, som ga seg tid til grundig gjennomgang av utkastene til denne rapporten. Dag Syvert Mæsel som alltid var rede til å diskutere problemene. Jon Bing, Andreas Galtung og Hans Christian Aakre som alle har bidratt til en diskusjon av rettslige ekspertsystemer, og Ina Herrestad som har lyttet til utlegninger om denne diskusjonen gang etter gang.

13. REFERANSER.

Hans Fredrik Berg & Mette Theisen *Ekspertsystem for skatteregler*, Norsk Regnesentral rapport nr 811, Oslo januar 1988.

Jon Bing (ed) *F*KUS - Knowledge based systems for public administration*, CompLex 8/87, Universitetsforlaget Oslo.

Jon Bing "Designing Text Retrieval Systems for "Conceptual Searching", *ICAIL'87 Proceedings*, ACM press, New York 1987.

Crystal Manual til versjon 1986 og versjon 1987, Intelligent Environment Limited, Richmond Surrey.

Roar Fjellheim, "Ekspertsystemer 1987: Status og tendenser". *DataTid*, Nr. 4 (April) 1987.

Henning Herrestad "Some Problems Concerning the Representation of Legal Norms", *Proceedings of the first Scandinavian Conference on Artificial Intelligence*, Tromsø, 1988.

Asbjørn Kjønstad "Vi får en ny trygdelov som mange flere kan forstå", *Sosial Trygd* nr 7, 1982.

L. Thorne McCarty "Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning", *Harvard Law Review 90*, pp 837-893, 1977.

Rolf Nossum, "Ekspertsystem-skall for PC og Macintosh", *Nordisk Datanytt*, vol. 16 no. 1, 1986.

Personal Consultant Easy Reference Guide, Texas Instruments Inc., Austin Texas 1986.

Paul Rynning, *"En fremstilling og vurdering av reglene for tildeling av bostøtte"*, Skriftserien JUS og EDB nr.17/1976,

Dag Wiese Schartum *"The introduction of computers in Norwegian local insurance offices"*, CompLex 9/87.

14. VEDLEGG: LOV OM FOLKETRYGD Kap. 7, §§ 7-1 og 7-2.

§ 7-1.[1] Rett til alderspensjon inntrer ved fylte 67 år.[2] For trygdet som har fylt 67, men ikke 70 år, begrenses adgangen til å ta ut alderspensjon etter bestemmelsene i § 7-10, og uttatt pensjon beregnes etter bestemmelsene i nevnte paragraf.

Alderspensjon består av grunnpensjon og tilleggspensjon.

1 Endret ved lov 16 juni 1972 nr. 60 (se dens III).
2 Jfr. § 19-4.

§ 7-2.[1] 1. Rett til grunnpensjon har den som har vært trygdet sammenlagt i minst tre år etter fylte 16 og før fylte 67 år.[2] Kalenderår hvori vedkommende er blitt godskrevet pensjonspoeng,[3] regnes i denne forbindelse som et helt år.

Som trygdetid for rett til grunnpensjon medregnes også kalenderår hvori trygdet fyller 67, 68 eller 69 år og har opptjent pensjonspoeng.

2. Full grunnpensjon utgjør:
a. grunnbeløpet, jfr. § 6-2, dersom pensjonisten er ugift eller har ektefelle som ikke oppebærer alderspensjon, uførepensjon eller attføringspenger i ventetid før uførepensjon kan tilstås,
b. 75 prosent av grunnbeløpet, dersom pensjonisten har ektefelle[4] som oppebærer alderspensjon, full uførepensjon eller fulle attføringspenger i ventetid før uførepensjon kan tilstås.
Oppebærer pensjonistens ektefelle uførepensjon eller attføringspenger som nevnt som er nedsatt etter reglene i § 8-5, jfr. § 5-5 første ledd, skal grunnpensjonen være lik grunnbeløpet med fradrag av et beløp som skal utgjøre 25 prosent av grunnbeløpet multiplisert med det tall som angir ektefellens uførepensjon eller attføringspenger i forhold til full ytelse.

3. Full grunnpensjon ytes til den som har vært trygdet i minst 40 år.[5] Har vedkommende vært trygdet i mindre enn 40 år, utgjør grunnpensjonen en forholdsmessig del av full grunnpensjon. Uten hensyn til bestemmelsene i nr. 1 legges for person som nevnt i nr. 2 bokstav b ektefellens trygdetid til grunn hvis den er lengst. Denne trygdetid beholdes etter ektefellens død, med mindre vedkommende inngår nytt ekteskap.

Hvis særlige grunner gjør det rimelig, kan utenlandsk flyktning som er bosatt i riket, tilstås inntil full grunnpensjon uten hensyn til bestemmelsene om trygdetid i første ledd og i nr. 1. Departementet gir nærmere forskrifter og kan herunder bestemme at det samme skal gjelde norsk statsborger som kan anses å stå i lignende stilling som utenlandsk flyktning.

4. Tidsrom hvori vedkommende under opphold utenfor riket har oppebåret grunnpensjon etter kapittel 8 og 10 på grunnlag av trygdetid fram til fylte 67 år, kan ved overgang til alderspensjon medregnes som trygdetid etter nærmere forskrifter som fastsettes av departementet.

1 Endret ved lover 15 des. 1967 nr. 6, 16 juni 1971 nr. 71 (se dens III), 16 juni 1972 nr. 60, 14 juni 1974 nr. 45, 9 des. 1983 nr. 71 (se dens VII).
2 Jfr. §§ 1-2, 1-3, 1-4, 19-3 og 19-4.
3 Se § 6-5.
4 Jfr. § 7-6.
5 Jfr. §§ 1-2, 1-3, 1-4, 19-3 og 19-4.

Tidligere utgitt i skriftserien CompLex

Alle heftene i skriftserien CompLex kan kjøpes hos, eventuelt bestilles fra Akedemika (se kupongen bakerst). De kan også kjøpes fra Norsk forening for jus og edb, Postboks 7557 Skillebekk, 0205 OSLO 2. Her kan man også tegne abonnement eller abonnere på «vaskesedler» som gir en kort omtale av hver rapport etterhvert som de kommer ut, og som inneholder bestillingsseddel. Abonnement på «vaskesedler» er gratis.

CompLex 1/81
Johs Hansen
Et edb-system for analyse av rettslige avgjørelser — *64 kr*

CompLex 2/81
Johs Hansen (red)
Notater om deontiske systemer — *47 kr*

CompLex 3/81
Vidar Sørensen
Informasjonssystem mm for Norges Geotekniske Institutt — *83 kr*

CompLex 4/81
Kjetil Johnsen
Systemtekniske konsekvenser av persondatalovgivningen — *111 kr*

CompLex 5/81
Marit Thorvaldsen
Teledata og rettsinformasjon — *65 kr*

CompLex 6/81
Anne Kirsti Brække
Postmonopolet — *27 kr*

CompLex 7/81
Knut S Selmer (ed)
The LAWDATA Papers — *28 kr*

CompLex 1/82
Norsk forening for jus og edb & Svenska föreningen för adb och juridik
Nordiske personregisterlover,
Council of Europe Convention og OECD Guidelines *31,50 kr*

CompLex 2/82
Harald Brænd og Vidar Sørensen
Oljevern: Brukerbehov og kildemateriale *55 kr*

CompLex 3/82
Jon Bing og Dag Frøystad
Rettskildebruk *90 kr*

CompLex 4/82
Thomas Prebensen Steen
Post- og televerkets regulerte ansvar *77 kr*

CompLex 5/82
Datatilsynet
Årsmelding 1981 *44 kr*

CompLex 6/82
Birger Eckhoff
in cooperation with Jon Bing, Dag Frøystad and Anja Oskamp
CATTS: Computerized program for teaching text retrieval systems *61 kr*

CompLex 7/82
Jon Bing
Informasjonssystemer for Trygderettens kjennelser *126 kr*

CompLex 8/82
de Mulder, Oskamp, van der Heyden and Gubby
Sentencing by Computer: An Experiment *123 kr*

CompLex 9/82
Colin Tapper
An Experiment in the Use of Citation Vectors in the
Area of Legal Data *56 kr*

CompLex 1/83
Arve Føyen
Utredning om endringer i pensjonregisterloven *70 kr*

CompLex 2/83
Stein Schjølberg
Computers and Penal Legislation *140 kr*

CompLex 3/83
John S Gulbrandsen og Terje Hoffmann
Rettigheter i idrettsarrangementer *64 kr*

CompLex 4/83
Sally Moon and Anja Oskamp
The Law of Legal Information Systems: Two Essays *95 kr*

CompLex 5/83
Datatilsynet
Årsmelding 1982 *utsolgt*

CompLex 6/83
Olav Torvund
Betalingsformidling *146 kr*

CompLex 7/83
Else Ryen
Lov og lovmottaker *103 kr*

CompLex 8/83
Daniel Stripinis
Probability Theory and Circumsantial Evidence *23 kr*

CompLex 9/83
Frede Cappelen
Edb-basert informasjonssystem for forvaltningens praksis *36 kr*

CompLex 10/83
NORDIPRO
Legal Acceptance of international Trade Data *utsolgt*

CompLex 11/83
Jørgen Hafstad og Thomas Prebensen Steen
Teleksrett og merverdiavgift på programvare *42 kr*

CompLex 12/83
Kristin Kjelland-Mørdre
Om forenkling av regler *utsolgt*

CompLex 13/83
Jon Bing
Edb: Mulighet og problem ved forenkling av regelverk *92 kr*

CompLex 14/83
Tarjei Stensaasen
Utvalgte emner i jus og edb (3. utgave) *190 kr*

CompLex 15/83
Anette Klafstad og Ulf Alex Samer
Plan for et forsikringsrettslig informasjonssystem *60 kr*

CompLex 1/84
Jon Erling Skjørshammer
Kabelnett: Bygnings- og ekspropriasjonslov *utsolgt*

CompLex 2/84
Tore Andreas Hauglie og Dag Wiese Schartum
Forslag til et helserettslig informasjonssystem *120 kr*

CompLex 3/84
Justisdepartementet
Den elektroniske grunnbok *utsolgt*

CompLex 4/84
Datatilsynet
Årsmelding 1983 *utsolgt*

CompLex 5/84
Tove Fjeldvig
Tekstsøking: Teori, metoder og systemer *90 kr*

CompLex 6/84
Jon Bing
Offentlighetsloven og edb *94 kr*

CompLex 7/84
Gunnar Bach, Beate Jacobsen and Vidar Stensland
The National Social Insurance System of Norway *51 kr*

CompLex 8/84
Dag Frøystad
Data Protection in Practice I: Identifying and Matching Elements *80 kr*

CompLex 9/84
Tove Fjeldvig og Anne Golden
Automatisk rotlemmatisering - et lingvistisk hjelpemiddel for tekstsøking *90 kr*

CompLex 10/84
Elling Øyehaug Ose
Retstidene som informasjonssystem *120 kr*

CompLex 1/85
Jon Bing
Data Protection in Practise II: International Service Bueraux and Transnational Data Flow *47 kr*

CompLex 2/85
Jon Bing
Opphavsrett og edb *176 kr*

CompLex 3/85
Dag Wiese Schartum
Codex, Calculation and Computers *utsolgt*

CompLex 4/85
Olav Torvund
To informasjonsrettslige arbeider *52 kr*

CompLex 5/85
Datatilsynet
Årsmelding 1984 *utsolgt*

CompLex 6/85
Hans Chr Aakre (red)
Utvalgte artikler i rettsinformatikk *utsolgt*

CompLex 7/85
Johannes Hansen
SARA: Brukerveiledning og programdokumentasjon *150 kr*

CompLex 8/85
Johannes Hansen
Modelling Knowledge, Action, Logic and Norms *90 kr*

CompLex 9/85
Thomas Prebensen Steen (red)
Kompendium i informasjonsrett *utsolgt*

CompLex 10/85
Tarjei Stensaasen
Opphavsrettslovens § 43 («katalogregelen») *97 kr*

CompLex 11/85
Jon Bing
Straffelovens definisjon av «trykt skrift» anvendt på datamaskinbaserte informasjonssystemer *65 kr*

CompLex 12/85
Magnus Stray Vyrje
Vanhjemmel, opphavsrett og datamaskinprogrammer *161 kr*

CompLex 1/86
Anne Kirsti Brække
Formidlingsplikt for kabeleier *80 kr*

CompLex 2/86
Dag Wiese Schartum
Stans av edb-tjenester i krigs- og krisesituasjoner *80 kr*

CompLex 3/86
Ingvild Mestad
«Elektroniske spor» - nye perspektiv på personvernet *100 kr*

CompLex 4/86
Jon Skjørshammer
Opphavsrett, databaser og datamaskinprogrammer: Kontraktrettslige aspekter *205 kr*

CompLex 5/86
Datatilsynet
Årsmelding 1985 *73 kr*

CompLex 6/86
Johannes Hansen
Simulation and automation of legal decisions *132 kr*

CompLex 7/86
Stein Schjølberg
Datakriminalitet og forsikring *54 kr*

CompLex 8/86
Thomas Prebensen Steen (red)
Kompendium i informasjonsrett (2.utg) *70 kr*

CompLex 9/86
Peter Blume
Edb-retlige foredrag *77 kr*

CompLex 1/87
Joseph A Cannataci
Privacy and data protection law:
International Development and Maltese Perspectives *170 kr*

CompLex 2/87
Jon Bing og Jon Bonnevie Høyer
Publisering av rettsavgjørelser *170 kr*

CompLex 3/87
Hagen Kuehn
The Social Security System in the
Federal Republic of Germany *77 kr*

CompLex 4/87
Hanne Plathe Maartmann
Personvern i sykepengerutinen *70 kr*

CompLex 6/87
Jon Bing
Electronic Publishing:
Data Bases and Computer Programs *55 kr*

CompLex 7/87
Ánde Somby
Selektiv gjenfinning av bestemmelser i
bygningslovgivningen *119 kr*

CompLex 8/87
Jon Bing
FOKUS:
Knowledge based systems for public administration *125 kr*

CompLex 9/87
Dag Wiese Schartum
The introduction of computers in the
Norwegian local insurance offices *230 kr*

CompLex 10/87
Knut Kaspersen
Kredittopplysning *75 kr*

CompLex 11/87
Ernst Buchberger, Bo Göranzon and Kristen Nygaard (ed)
Artificial Intelligence:
Perspectives and Implications *120 kr*

CompLex 12/87
Jon Bing, Kristine M Madsen og Kjell Myrland
Strafferettslig vern av materielle goder *118 kr*

CompLex 13/87
Tove Fjeldvig
Effektivisering av tekstsøkesystemer:
Utvikling av språkbaserte metoder *175 kr*

CompLex 14/87
Jon Bing
Journalister, aviser og databaser *66 kr*

CompLex 15/87
Andreas Galtung
Skatterettslig ekspertsystem: Et eksempel basert på skattelovens § 77 *82 kr*

CompLex 1/88
Robin Thrap-Meyer
Utvalgte emner i rettsinformatikk (4. utgave) *228 kr*

CompLex 2/88
Ernst Buchberger, Bo Göranzon and Kristen Nygaard (ed)
Artificial Intelligence: Perspectives of AI as a social technology *178 kr*

CompLex 3/88
Jon Bing og Anne Kirsti Brække
Satellittfjernsyn *128 kr*

CompLex 4/88
Jon Bing
Journalists, Databases and Newspapers *68 kr*

CompLex 5/88
Joseph Cannataci
Liability and responsibility for expert systems *108 kr*

CompLex 6/88
Datatilsynet
Årsmelding 1987 *58 kr*

CompLex 7/88
Henrik Sinding-Larsen (ed)
Artificial Intelligence and Language
Old questions in a new key 295 kr

CompLex 8/88
Jon Bing
Database for offentlige publikasjoner (DOP):
Fremtidig organisering
NORIS (87) 178 kr

CompLex 9/88
Jon Bing
Conceptual Text Retrieval
NORIS (77) 118 kr

Fremtidige utgivelser i skriftserien CompLex

Nedenfor følger omtale av CompLex-utgivelser som vil foreligge i nærmeste fremtid. De kjøpes/bestilles på samme måte som for tidligere utgitte nummer i serien – se foran og kupongen bakerst.

CompLex 11/88

Ingvild Mestad

L'INFORMATIQUE, LA LIBERTE INDIVIDUELLE ET LA RECHERCHE EPIDEMIOLOGIQUE
Data Protection and Epidemiologic Research

The conflict between the needs of medical research and the data protection of individuals is one of the more acute problems with respect to individual privacy.

In this report, a comparative study of this issue is offered. The conflict is, of course, especially visible with respect to research relevant to the control of epidemics - an issue of international concern and emphasized by the current efforts to contain the spreading of the HIV-virus and the AIDS disease.

The study is divided into two parts. The first part examines the French data protection legislation of 1978, supplementing this with a discussion of confidentility within the health care system in France. The second part offers a similar discussion of the data protection legislation i Norway and Sweden. In this way, this the difficult balancing of different interests is illuminated by the law and practice from three different countries, offering a valuable contribution to the literature on data protection, and an addition to the all to few comparative studies within the area.

The study will be of interest to lawyers and medical doctors alike. The language of the reports is French, and the research basic to the report has been conducted at IRETIJ of the University of Montpellier.

108 sider. Pris kr 118,-.

CompLex 12/88

Johs Hansen

SAFE-P:

SIKRING AV FORETAK, EDB-ANLEGG OG PERSONVERNINTERESSER ETTER PERSONREGISTERLOVEN TERESA (50)

Personvern karakteriseres gjerne som den interesse den enkelte har i å kontrollere bruk og lagring av opplysninger som angår vedkommende selv. Et ledd i en slik kontroll, vil være sikkerhet mot uautorisert aksess, tilfeldig sletting eller forvanskning av data, ukontrollert spredning av data osv - elementer som gjerne samles under betegnelsen «datasikkerhet».

Personregisterloven har i § 8b en generell hjemmel for utferdigelse av «regler for sikring av personopplysninger». I arpil 1986 nedsatte Justisdepartemetnet en arbeidsgruppe for å utforme sikkerhetsforskrifter. Forut for dette arbeidet hadde imidlertid Institutt for rettsinformatikk - under ledelse av Datatilsynets spesialrådgiver i edb, Johs Hansen - trukket i gang et prosjekt som tok sikte på å utforme prinsipper for sikring av personverninteresser, foretak og edb-anlegg. Den foreløpige rapporten ble lagt til grunn for arbeidsgruppens vurderinger, og danner på mange måter et forarbeid til det utkast til forskrifter som nå foreligger. Rapporten har en analytisk innfallsvinkel. Den omfatter bl a en diskusjon av risikoanalyse av anlegg og foretak som behandler personregistre, en sikkerhetsmodell bygd på ressursbegrepet, en vurdering av sensitivitet og sårbarhet som fører frem til en sensivitetsmodell, og en diskusjon av hva som ligger i «informasjonssikkerhet». I oppsummeringen drøftes sentrale spørsmål for utforming og administrasjon av sikkerhetsforskriften.
Rapporten fra Justisdepartementets arbeidsgruppe med utkast til forskrift for sikring av personregistre følger som vedlegg.

176 sider. Pris kr 178,-.

CompLex 13/88

Henning Herrestad & Dag Syverts Mæsel (eds)

FIVE ARTICLES ON ARTIFICAL INTELLIGENCE AND LEGAL EXPERT SYSTEMS NORIS (83) F KUS

This compendium comprises five articles on legal expert systems and artificial intelligence. The first three articles present three different legal expert systems, demonstrating the wide range of different motives, research interests and approaches to making legal expert systems. The fourth article focuses on the question of how to make formal representations of law. The fifth article presents more visionary ideas concerning how artificial intelligence research could be used as an intrument in reducing international tension.

The compendium originates from a roundtrip made by the editors to six different institutes situated in the Netherlands, West-Germany and Austria. Three of the institutes are doing research especially on legal expert systems, the other three on various aspects within the field of artificial intelligence. In their introduction the editors have summed up their impressions and thoughts after these visits.

100 sider. Pris kr 108,-.

COMPLEX

Bestillingskupong

Ja takk, vi bestiller herved følgende eks.
av **COMPLEX:**

.....eks. av nr..... à kr .....

.....eks. av nr..... à kr .....

.....eks. av nr..... à kr .....

.....eks. av nr..... à kr .....

Av fremtidige **COMPLEX** utgivelser (se omtale foregående sider)
bestilles herved:

.....eks. av nr..... à kr .....

.....eks. av nr..... à kr .....

Navn:.....

Firma:.....

Adresse:.....

Postnr./sted:.....

Kupongen sendes til:

Denne rapporten er utviklet i forbindelse med Institutt for rettsinformatikks prosjekter knyttet til forvaltningsorienterte, kunnskapsbaserte systemer font=4> F*KUS). På markedet finnes det i dag flere såkalte «skall» som tillater konstruksjon av ekspertsystemer, dvs systemer hvor brukeren beskriver sitt problem gjennom en dialog med systemet, og hvor systemet til slutt gir råd om hvilken avgjørelse som bør eller kan treffes.
Rapporten beskriver et forsøk på å beskrive reglene i folketrygdlovens §§ 7-1 og 7-2 og grunnpensjon ved hjelp av et slikt program, Crystal. Dette sammenlignes med Thorne McCartys arbeid innenfor rammen av det kjente TAXMAN-prosjektet. Sammenligningen gir grunnlag for en kritikk av hvor egnet slike skall er til å lage ekspertsystemer eller kunnskapsbaserte systemer generelt innenfor det juridiske domenet. Man tar opp spørsmål omkring utforming og vedlikehold av slike programmer, brukergrensesnitt og hjelpefunksjoner, og om hvilket formål slike systemer kan tjene i juridisk arbeid.

ISBN 82-518-2594-6

TANO

9 788251 825948