Landbrugsministeriet
Statens Planteavlsforsøg

Agricultural applications of knowledge based systems concepts

-exemplified by a prototype on weed control in organic farming WEEDOF

Ph.D. dissertation

Ulla Dindorp
The Danish Institute of Plant and Soil Science
Department of Biometry and informatics
DK-2800 Lyngby

Tidsskrift for Planteavls Specialserie

Beretning nr. S 2201 - 1992

Agricultural applications of knowledge based systems concepts exemplified by a prototype on weed control in organic farming WEEDOF

Ph.D. dissertation

Ulla Dindorp

The Royal Veterinary and Agricultural University
Department of Mathematics and Physics
Copenhagen

The Danish Institute of Plant and Soil Science
Department of Biometry and Informatics

1992

Preface

This thesis describes the work done in a Ph.D. project funded by the Danish Research Academy and the project Decentral Database Systems funded by The Research secretariate, Ministry of Agriculture. The Ph.D. study was mainly carried out at Department of Biometry and Informatics, The Danish Institute of Plant and Soil Science, and was advised from the Department of Mathematics and Physics at The royal Veterinary and Agricultural University.

The subject of the Ph.D. study is use of expert systems as a tool for research and knowledge transfer in plant production. Some preliminary results of this work has been published in references Dindorp 1990a, Dindorp 1990b, Dindorp 1991a, Dindorp 1991b. This text aims at providing an introduction to the concepts and methods of the special field of AI - expert systems - as well as describing the work in the Ph.D. project in an accessible way to researchers in agriculture.

Chapter 2 contains an introduction to expert systems, especially to rule-based expert systems. It introduces the main concepts of the field, and describes the function of expert systems, the architecture, the technique used in them and the methods used to build these systems. The first 3 sections is an overall description of rule based systems. Section 4, 5, 6 and 7 goes deeper into the parts of the rule based system. Section 8 is a survey over existing applications in agriculture.

Chapter 3 describes the development of a prototype expert system, WEEDOF, constructed during the Ph.D. project. This is a planning system designed to help organic farmers control weeds. The knowledge collection for the system as well as the resulting system design is described. The first issue is a description of the expert system shell used for the system development - EGERIA. The next sections describes the knowledge acquisition procedures used and the result of these, and the design of WEEDOF. At last the missing parts and how to possibly complete the system is described.

As a consequence of the bad explanatory power in WEEDOF the work continued with specifying a model for inclusion in a model based expert system. Chapter 4 deals with the work done on a dynamic model for plant growth. The model has been specified in a way not usually used in model building in agriculture. The specification method is shortly described, as is the preliminary results. Due to the lack of time the model is only in the preliminary stages.

Chapter 5 is a final conclusion on the varied work done in the project.

There are two important parts of research in the project. One is in the use of a rather formal method of knowledge acquisition - literature analysis - for aiding the initial knowledge collecting for the system. This is described in chapter 3. The other is the formal method used for specifying the dynamic model for plant growth. This method stems from computer system design (the Vienna Development Method) and has not been used before as a method for developing models. It is an exciting way of working with models and has so far been very suitable for the job.

Many people and institutes have contributed to the successful accomplishment of this project and I would like to thank them all. The advisors were Mogens Flensted-Jensen, at the Department of Mathematics and Physics, The Royal Veterinary and Agricultural University who provided support and advice on general

matters, Tom Østerby from The Department of Computer Science, Technical University of Denmark who has been a dedicated and involved technical advisor, and Ove J. Hansen from Department of Biometry and Biometry and Informatics, the local advisor, who first got the idea of an expert system project and helped in all the original descriptions of the project. Kristian Kristensen, head of Department of Biometry and Informatics, The Danish Institute of Plant and Soil Sciences, provided help and advice during the project time from the time when the idea of the project emerged. Henrik Schlichtkrull, Department of Mathematics and Physics, The Royal Veterinary and Agricultural University served as a mathematics instructor in the project start. And Jesper Rasmussen and Bo Melander at Department of Weed Control, Flakkebjerg, The Danish Institute of Plant and Soil Science provided their time in the construction of WEEDOF.

Contents

1 Resumé

Arbejdet i dette Ph.D. projekt har fokuseret på to emner: Dels konstruktionen af en prototype på et ekspertsystem for planlægning af ukrudtsbekæmpelse i økologisk jordbrug. Dels på specifikation af en dynamisk model for plantevækst til brug i et model baseret ekspertsystem.

1.1 Prototype

Den normale konstruktionsmetode ved konstruktion af regelbaserede ekspertsystemer er en iterativ procedure hvor især faserne begrebsopstilling, formalisering og implementering gennemføres igen og igen. Der er ingen formel konstruktionsmetode til konstruktion af ekspertsystemer, men en del beskrivelser af metoder til vidensudtrækning (knowledge elicitation) og videnrepræsentation. Der forskes for tiden en del i vidensanalysemetoder og metoder til karakterisering af domænet til brug i den første analyse af domæne og viden (Nwana et al 1991). Indtil videre må hver enkelt systemudvikler finde sin egen metode til effektiv konstruktion af disse systemer.

I dette eksperiment var vidensingeniøren ny i vidensingeniørfaget, og den første prototype tog formentlig længere tid at konstruere end det ville have taget for en erfaren vidensingeniør, men udviklingen blev lettet ved brugen af en ny metode i starten af vidensindsamlingsfasen - litteraturanalyse. Ved denne analyseres tekster fra domænet for at finde og udtrække de vigtige begreber i domænet, og regler vedrørende begreberne så som definitioner og årsagssammenhænge. En parallel metode er blevet brugt til automatisk konstruktion af små vidensbaser (Gomez & Segami 1990).

Litteratur analysen tog netto omkring 2-3 måneder. Resultatet af analysen var et begrebshierarki, en samling regler om begreberne samt også noget mere udefinerligt - en fornemmelse af domænet, og af at kende de vigtige begreber og relationer. Når først begreberne er skrevet ned er det ofte indlysende at de hører med, og mange af dem ville være blevet nævnt i et interview med eksperten. I dette tilfælde ville en af eksperterne sikkert have kunnet udarbejde begrebshierarkiet, og brug af metoder som for eksempel repertory grid eller skaleringsteknikker kunne have været brugt til at afsløre relationer mellem begreber. Styrken i litteratur analysen er, at det er en simpel halv-formel metode, som sikrer, at alle relevante begreber - i hvert fald de begreber, som betragtes som relevante i faglitteratur - medtages sammen med de vigtige relationer mellem dem.

Til resten af videnindsamlingen blev brugt interviews. På grund af den udførte litteraturanalyse, som havde leveret en grundoversigt over domænet, var det muligt at strukturere interviewene fra begyndelsen. Alt i alt blev der gennemført seks interviews, resten af videnindsamlingen blev gennemført ved hjælp af brevveksling og telefonsamtaler.

Det valgte domæne - ukrudtsbekæmpelse i økologisk jordbrug - var karakteristisk ved en mængde usikker og manglende viden. Siden de kemiske ukrudtsbekæmpelsesmidler blev opdaget har forskning i emnet været stoppet, og er først for nylig blevet genoptaget. Domænet er biologisk og en masse faktorer påvirker vækst og udvikling af planter. Forskerne i domænet var fra starten meget usikre på mulighederne for at udvikle ekspertsystemer i deres emneområde. Testen lykkedes imidlertid. Eksperterne var tilfredse med den udviklede prototype, og følte også, at de havde udviklet ny indsigt i deres forskningsområde under processen med at udvikle ekspertsystemet. Domænet studeres så grundigt under system-

konstruktionen, at eksperterne finder huller i deres viden om domænet, huller, som resulterer i nye eksperimenter for at klarlægge de svage punkter. Foruden resultatet af et ekspertsystemprojekt i form af et system, giver udviklingsprocessen altså også en bonus til de medvirkende eksperter i form af en bedre oversigt over den nuværende såvel som den manglende viden indenfor domænet.

Det resulterende system - WEEDOF - blev programmeret i EGERIA, en ekspertsystemskal. En af de vigtige ting, der mangler i det nuværende system er forklaringer. Hovedårsagen til de manglende forklaringer ligger i en kombination af skal og system. EGERIA understøtter kun forklaringer der kan genereres, som en udskrivning af regler brugt under en baglæns kædning. Da det nuværende system bruger forlæns kædning skiftende med baglæns, forhindrer dette forklaringsmekanismen i at fungere tilfredsstillende. Selv hvis forklaringer kunne genereres fra viden i den nuværende videnbase, ville disse forklaringer være mindre gennemskuelige end forklaringer fra en ekspert. Eksperten ville indbygge sin model af domænet i forklaringerne, mens systemet kun kan genspille viden i videnbasen, viden som hovedsagelig er heuristisk. Dette er en af grundene til at arbejdet fortsatte med specifikationen af en model.

1.2 Model

En anden grund til at arbejde med en model er, at det muliggør konstruktion af et system med en videnbase, som kan genbruges i højere grad end den heuristiske videnbase. En gene ved disse modelbaserede systemer er at de er langsommere.

Modeller kan bruges forskelligt i modelbaserede ekspertsystemer. Ekspertsystemdelen kan for eksempel være en del, der kun bruges til at indsamle information til simulation ved hjælp af modellen og fortolker output fra modellen - dvs den fungerer som omgivelser til modellen og kan ikke bruge modellen til at svare på vilkårlige spørgsmål. Modellen kan være en integreret del af systemet, som for eksempel også kan indeholde databaser. Endelig kan systemet indeholde flere modeller, for eksempel modeller i flere forfiningsgrader til forklaring på forskellige niveauer.

I dette arbejde var meningen, at modellen skulle være en integreret del af et system, hvor ekspertsystem delen ikke kun samler input for modellen og fortolker output, men også udfører et (heuristisk) arbejde med at finde de relevante eller mulige bekæmpelsesmetoder før simuleringen.

Arbejdet på modellen er startet, men det modelbaserede system er kun i et forstadie. Den brugte metode til specifikation af modellen er ny i jordbrugssammenhænge. At specificere systemer ved hjælp af funktionel nedbrydning er velkendt indenfor edb, hvor det bruges i en systemudviklingsmetode - The Vienna Development Method - VDM (Bjørner & Jones 1982). Modellen er specificeret i META IV, og metoden har vist sig at være brugbar også i denne type af systembeskrivelse. Top-down specifikations metoden indebærer at nedbryde problemer, og på den måde opdele dem i mindre, simplere problemer før det er nødvendigt at løse dem.

Modellen, som er blevet specificeret, eller delvis specificeret, er en dynamisk model for den totale plantevækst på en mark. Det er meningen at modellen skal gøre rede for virkninger af bekæmpelsesmetoder, f.eks. harvning, og andre aktioner på væksten. Modellen skal medtage konkurrence mellem arter. Desuden skal modellen være generel, så det er muligt at beskrive væksten af alle planter på marken. Spørgsmålet er, om det er muligt at konstruere sådan en generel model med den eksisterende biologiske viden.

Den specificerede model grundes på en generel livscyklus for planter. Der er et generelt mønster for planteliv, hvor frø spirer til planter, som vokser, blomstrer og dør. Modellen skal være i stand til at modellere både arter, som er enårige og flerårige, og frø- såvel som rodformerede arter. I modellen er der to forskellige bidrag til plantevæksten. Det ene er den naturlige plantevækst i følge arten og begrænset af konkurrence - andre begrænsninger for eksempel næringsmæssige eller klimatiske er ikke omhandlet endnu. Livscyklen er her brugt som basis i nedbrydningen af modellen i funktioner. Det andet bidrag er indflydelsen af behandlinger udført på marken på planter og frø.

Specifikationen viser alle funktioner, som er nødvendige til at beskrive dette med tilhørende input og output. De konkrete algoritmer er ikke specificeret endnu. Enhver model er en simplifikation af den virkelige verden. Nogle eller måske alle funktionerne i denne model kunne muligvis beskrives bedre med en empirisk model. Funktionerne i den benyttede mekanistiske model er opdelt i dele på en måde, som efterligner sammenhænge i naturen. For at gøre det muligt at overskue modellen er funktionerne ret simple. Dele mangler, enten fordi de er udeladt med vilje - for eksempel fordi de anses for ret betydningsløse - eller fordi viden mangler. Grunden til at holde fast i den mekanistiske model er muligheden for at forklare og begrunde resultaterne af det færdige system på baggrund af den dybe viden i domænet.

1.3 Ekspertsystemer og jordbrug

Kan vi bruge ekspertsystemteknologien indenfor jordbrug? Der er klare emner indenfor jordbrug, hvor teknologien kan være brugbar.

For eksempel:

- Overvågning af klima i væksthuse,
- planlægning af fordeling af naturgødning på økologiske jordbrug
- diagnose af sygdomme.

Mere og mere viden kræves for at styre en jordbrugsbedrift og opnå det nødvendige dækningsbidrag. Nu da pc'ere bliver mere og mere almindelige, vil der være et marked for beslutningsstøttesystemer. Ikke nødvendigvis ekspertsystemer men disse vil være en del af de nye systemer.

Udviklingenindenforekspertsystemteknologien går i retning af en integration af ekspertsystemer med andre typer software. De originale ekspertsystemer er enkeltstående systemer i et snævert emneområde. Det bliver generelt anset for en fordel at integrere ekspertsystemerne med databaser eller modeller og lade dem arbejde sammen med andre typer software, som brugeren har adgang til. På den måde bliver ekspertsystemer en naturlig del af en større 'pakke' og bruges mere.

Konstruktion af ekspertsystemer tager generelt længere tid end konstruktion af andre edbprogrammer. Derfor er det vigtigt at være forsigtig med valg af domæne og vælge et, hvor udviklingen kan begrundes. Dette kan være enten på grund af økonomiske gevinster eller mangel på eksperttid. I Australien, hvor afstande er store og eksperterne få har den sidste grund været basis for udvikling af ekspertsystemer (Waterhouse et al 1989). Hvis man ser på betingelserne i Danmark, kan fortjenesten på udvikling af systemer til jordbrugserhvervet let blive for lille til at betale for udviklingen af danske systemer. Nogle af disse systemer kan så udvikles til det europæiske marked, eller de nordeuropæiske lande i tilfælde, hvor betingelserne er meget forskellige i sydeuropa og nordeuropa.

I fremtiden er der håb om, at udviklingsomkostningerne for ekspertsystemer vil blive mindre. Nye vidensindsamlingsværktøjer dukker op. Disse sigter på at lette videnindsam-

lingen ved for eksempel at give eksperten redskaber til at indtaste hans viden. Desuden udvikles nye metoder til formalisering af konstruktionsprocessen - litteraturanalyse kan være baggrund for en sådan mere formel metode.

Jordbrugsforskere ser ud til at have fordel i at samarbejde i ekspertsystemprojekter. Dette projekt har vist, at måden at arbejde med domænet, når man udtrækker og formaliserer viden, giver en feed-back til eksperten i form af en øget indsigt i hvilken viden der er brugbar, og svagheder i viden indenfor domænet. Arbejdet på et ekspertsystemprojekt vil ofte betyde en formalisering af viden, som gør det muligt at anvende viden også sammen med mere traditionelle programsprog, hvilket kan give mere effektive programmer.

2 Expert systems

This chapter will define the subject expert systems and describe it in terms of background, function, technique and methods used. Several books have been written on the subject. Some of the best known and often cited are Rich 1983, Hayes-Roth et al 1983, Nilsson 1982, and Waterman 1986.

This definition stems from The British Computer Society '*An expert system is regarded as the embodiment within a computer of a knowledge-based component from an expert skill in such a form that the system can offer intelligent advice or take an intelligent decision about a processing function. A desirable additional characteristic, which many would consider fundamental, is the capability of the system, on demand, to justify its own line of reasoning in a manner directly intelligible to the enquirer. The style adopted to attain these characteristics is rule-based programming.*'

Many other definitions have been made. A common point in these is the built in intelligent component, the intelligent behaviour of the system and the ability to answer questions. Other definitions do not narrow the definition to rule-based systems. Although they have been far the commonest developments have introduced systems that use semantic net representations, fuzzy systems and others. Expert systems often comprise several forms of programming, and may contain ordinary program parts as for instance models and databases. Often the architecture of the systems is also an important part of the definition, including only systems where the systems knowledge is separated from the control structure.

Occasionally questions are raised whether particular systems are 'real expert systems' or just decision tables. In response different labels (decision support system, knowledge system) are sometimes used to more explicitly define a software system. The techniques used are the same, but the knowledge may be of different levels. Maybe it is not at expert level but aim at a less ambitious support of the knowledge solving process. The goal is always to deliver the most skilful decision making systems. Sometimes rule based expert systems are the best tool for the job, sometimes other approaches are better.

In the rest of this thesis - except in chapter 4 - the rule based expert system will be concentrated on, and it refers to this when the terms expert system and knowledge based system are used. The issue in chapter 4 is a model for a model based system.

The first section of this chapter addresses the background of expert systems. Section 2.2 deals with two ways of classifying expert systems. Section 2.3 describes the architecture of expert system with the segregation into the knowledge base, inference engine and user interface. User interface is an important part of a computer system but has not been elaborated in this project and will not be discussed very much. Section 2.4, 2.5 and 2.6 are further elaborations on the knowledge base and inference engine with descriptions of techniques used. Section 2.7 describes methods and techniques for construction of expert systems. And section 2.8 gives a survey of known expert systems in agriculture.

2.1 Background

The phase of computer evolution that spawned expert systems started in the early seventies, it was a breakthrough in a field of computer science known as artificial intelligence - AI.

The goals of AI scientists have always been to develop computer programs that could in some sense think - reason using knowledge, that is, solve problems in a way that could be considered intelligent if carried out by a human being.

AI can be subdivided into relatively independent research areas. One group of AI-researchers is concerned primarily with problem solving, and it is in that area expert systems are placed. Another group of AI scientists is concerned with developing computer programs that can read, speak or understand language, commonly referred to as natural language processing. A third branch of AI research is concerned with developing robots. Especially visual and tactile programs that will allow robots to observe changes in an environment. And a fourth branch is developing programs which can expand on their own knowledge by learning.

In the sixties AI scientists tried to simulate the complicated methods of thinking by general methods for solving broad classes of problems; they used these methods in general purpose programs that could solve not only one but series of logical problems. However developing general purpose programs was ultimately fruitless. The strength of the general problem solvers was their generality, on the other hand they could only solve problems of limited complexity, so the more classes of problems a program could handle, the more poorly it seemed to do on any individual problem. The work on general problem solvers was therefore overshadowed by the new field - expert systems.

The expert system concept departs from the general problem solver concept by giving up the ambition on generality. The AI scientists realized that the problem solving power of a program comes from the knowledge it possesses, and to make a program intelligent, it must be provided with lots of knowledge from the actual problem domain (Hayes-Roth 1983, Waterman 1986). This was a breakthrough in the field and led to the development of special purpose programs, systems that were experts in some narrow problem area.

In the beginning there was great optimism about the potential power of these new computer programs. A general attitude among american AI scientists was that natural and artificial intelligence were two sides of the same question, and that eventually programs would be made that would make machines as intelligent as human beings (Waterman 1986).

In the seventies and eighties it has become clear that such prophesies will not be realized for a long time - if ever (Harder 1990). The AI scientists have been criticised for overestimating the possibilities of AI, one of the early criticists says *'In each area where there are experts with years of experience the computer can do better than the beginner and can even exhibit useful competence but it cannot rival the very experts...' (Dreyfus & Dreyfus 1986).* The critics try to establish and describe fundamental constraints in computer technology, which makes it impossible to believe that all mental processes can be imitated.

Buchanan and Smith (1989) rejects this critic. They say *'The term 'expert system' suggests a program that models a human expert's thought processes... However the designers of expert systems do not subscribe to these implications. Although high performance is a goal, a system need not equal the best performance of the best individuals to be useful... designers of expert systems build into their programs much of the knowledge that human experts have about problem solving. But they do not commit to*

building psychological models of how the expert thinks. The expert may describe how he or she would like others to solve these problems. The expert system is a model of something, but it is more a model of the experts model of the domain than of the expert.'

The discussions have not stopped the development of expert systems. The evolution has made the technology available for those other than researchers, and it is now being used in private companies for developing applications. The technology developed by the AI researchers has shown to be useful for a variety of tasks although there has been a lowering of the expectations to the intelligence that is possible to build into a computer.

2.2 General classifications of expert systems

There are several ways of classifying expert systems. The classification could be made on grounds of problem categories, on system operations or on system types.

The classification according to problem categories has been used in classical expert system literature (fig. 2.1). Interpretation systems explains observed data by assigning to them symbolic meanings describing the situation. This category includes surveillance, image analysis and signal interpreting. Prediction systems employ a model to infer consequences. This category includes weather forecasting and crop estimations. Diagnosis systems relate observed irregularities with underlying causes. This category includes diagnoses of diseases

Category	Problem addressed
Interpretation	Inferring situation descriptions from sensor data
Prediction	Inferring likely consequences of given situations
Diagnosis	Inferring system malfunctions from observables
Design	Configuring objects under constraints
Planning	Designing actions
Monitoring	Comparing observations to plan vulnerabilities
Debugging	Prescribing remedies for malfunctions
Repair	Executing a plan to administer a prescribed remedy
Instruction	Diagnosing, debugging and repairing student behavior
Control	Interpreting, predicting, repairing and monitoring system behaviors

***Figure 2.1** Generic categories of knowledge engineering applications. From Hayes-Roth et al 1983.*

among others. Design systems develop configurations that satisfy the constraints of the design problem. Such problems include building design and budgeting. Planning systems employ models to infer effects of planned actions. They include problems such as experiment planning. Monitoring systems compares observations of system behaviour to features crucial to successful plan outcomes. They could be monitoring the climate in a greenhouse. Debugging systems prescribe remedies for correcting a diagnosed problem. Such could be debugging aids for computer programs. Repair systems develop plans to administer a remedy for a diagnosed problem. This could be for instance repair of machines. Instruction systems diagnose and debug student behaviours. They diagnose weaknesses in a student's knowledge and plan a tutorial to convey the knowledge to the student. Control is also a mixture of several of the above mentioned types. Control systems interpret data, predict the future, diagnose causes of anticipated problems, formulate a repair plan and monitor the execution. Problems in this class include business management and air traffic control.

Clancey (1985) suggests classification according to system operations to improve upon the distinctions made in the above generic categories. He revises the above table and classifies according to what we can do to or with a system (fig 2.2). Operations are grouped in terms of those that construct a system and those that interpret a system corresponding to synthesis and analysis.

Interpretation systems describe a system. Interpretation systems perform identification, predictions or control. Diagnosis and monitoring systems are both a kind of identifying system. In monitoring systems behaviour are checked against a preferred model. Diagnosis identifies some faulty part of a design with respect to a preferred model.

The Construction systems synthesises new systems. They perform specifications, design and assembly.

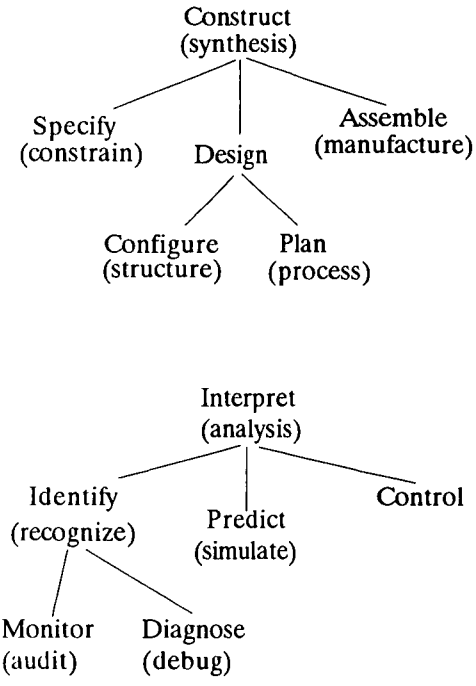

***Figure 2.2** Generic operations for synthesizing and analysing a system. Synonyms appear in parentheses. From Clancey 1985.*

Instruction is dropped because it is a composite operation.

2.3 Architecture of rule based expert systems

Rule based expert systems have three components: a knowledge base which contains the domain knowledge, an inference engine which decides how and when to use the knowledge, and a user interface (fig 2.3). During execution the system maintains a database which contains the current state of the problem.

2.3.1 Knowledge base

The part of the system which contains the domain knowledge on a symbolic form is called the knowledge base.

An expert in a domain has knowledge of several types. Part of the domain specific knowledge is simple subject knowledge, which can be found in a text book on the domain. But the expert also has knowledge not usually described in text books, this includes exceptions to general rules, how to solve problems and information on earlier problems. This latter type of knowledge is called heuristic knowledge.

The knowledge base contains knowledge of both kinds - the subject knowledge as well as heuristic knowledge to the extent that it is possible to transform this kind of knowledge into a representable form to the knowledge base.

2.3.2 Inference engine

Formalized expert knowledge is stored in the knowledge base. The inference engine contains

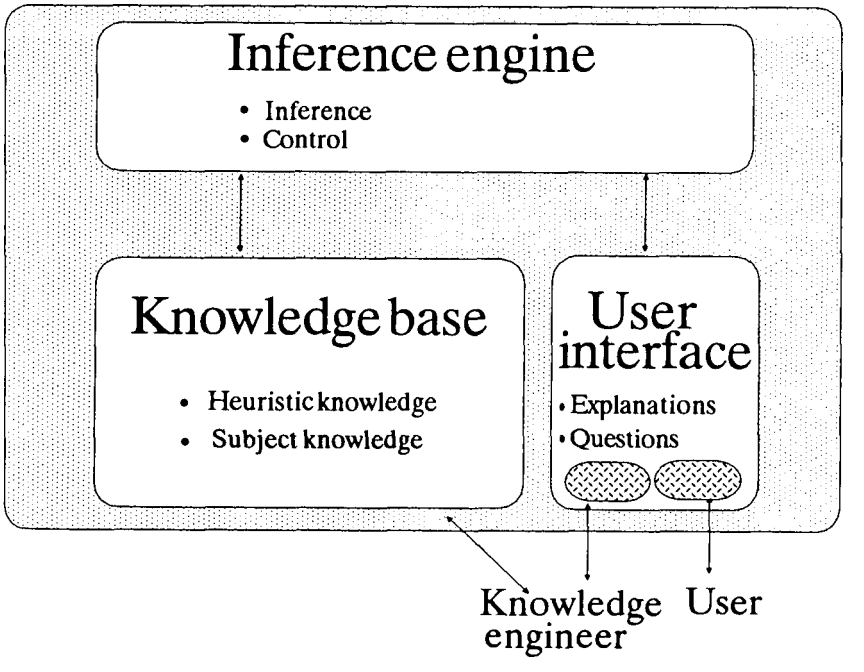

Figure 2.3 Architecture of rule based expert system.

the strategies to draw inferences and control the reasoning process. Inference and control strategies guide the expert system as it uses the facts and rules stored in its knowledge base, and the information it acquires from the user.

The inference engine performs two tasks. It examines the rules and facts and adds new facts when possible, and it decides the order in which the inferences are made. In doing so the inference engine conducts the consultation with the user.

2.3.3 User interface

The last part of the expert system is the user interface, the part of the system which conducts the communication with the users. Here we distinguish between the interface for constructors (knowledge engineers) and the consultation interface.

The important techniques especially in the consultation interface are techniques which appeal to the users. First of all graphical presentations and natural language. Natural language is still far from reality to day. More important is a natural dialogue with the user. To ask questions and show explanations in a language understandable to the user.

Explanations

An important side of expert systems is the ability to explain the conclusions drawn from knowledge and user answers.

Explanations in expert systems are usually associated with some form of tracing of rules that are used during the course of a problem solving session. This type of explanation is not always satisfactory. Heuristics may have been used to make shortcuts. The reasoning can still be sound. But an explanation based on the heuristics does not explain the underlying reason for events.

A satisfactory explanation of how a conclusion was derived often demands an ability to connect the inference steps with fundamental domain principles as justification.

2.4 Knowledge representation

Expert system technology has been described as a new programming paradigm especially due to the use of declarative rather than procedural programming. Procedural programming is the usual programming paradigm in conventional programs. Here you provide the algorithm for solving the problem explicit in the program, as a step by step specification, and the domain specific knowledge is implicit in the algorithm. In declarative programming the knowledge is declared with no specific ordering, and the algorithm to reach the result is implicit - build into the systems way of treating the knowledge.

The question is how much declarative programming is really used. To describe knowledge processing both types can be used, also in shells, and the boundary between the two is very flexible. Generally the less declarative knowledge, the more procedural knowledge is required and vice versa. Some believe that the absence of an explicit algorithm in connection with the interactive use makes it difficult to foresee what will happen in such a program (Harder 1990).

Knowledge representation means encoding of justified true beliefs into suitable data structures. Expert systems and other AI systems must have access to domain-specific knowledge and must be able to use it to perform their task - they require the capability to represent and

manipulate sets of statements. Most of the representations used in AI derive from some type of logic.

2.4.1 Logic

Every logical system uses a language to write propositions or formulae. Statements and arguments are translated to the language to see more clearly the relationships between them. This language consists of an alphabet of symbols:

- Individual constants used to express specific objects such as 'Peter'.
- Variable symbols.
- Predicate names, usually relations (verbs) to assemble constants and variables such as 'send' or 'write'.
- Function names.
- Punctuation symbols.
- Connectives such as 'and', 'or', 'imply', to produce compound statements from simple statement.
- Quantifiers such as 'for all'.

And some syntax rules. Normally, when one writes a formula, one has some intended interpretation of this formula in mind. For example a formula may assert a property that must be true in a database. This implies that a formula has a well-defined meaning or semantics. In logic , usually the meaning of a formula is defined as its truth value. A formula can be either true or false.

Logic consists of deduction. From a set of formulas or propositions written according to the unambiguous language, and their truth values, new formulas may be deduced following rules which are valid in the formal deductive system. In simple systems for instance the only deduction rule could be modus ponens, which says from *A is true* and $A \Rightarrow B$, *B is true* is a direct consequence, where *A* and *B* are formulas in the language. By using modus ponens again and again we have a simple procedure which enables us to construct a proof or argument.

The popular logic programming language PROLOG has a background in predicate calculus, which is a special form of logic. It uses the deduction rule resolution (described later) for deduction of new knowledge.

2.4.2 Rules and facts

The traditional form of representing the knowledge is in terms of facts and rules, ie classification of and relationships between objects, and rules for manipulating objects, the control part of the expert system then will have information on when and how to apply the rules.

One way of representing the facts and rules is through the use of a predicate calculus notation; here we define relationships between objects by a relation name (a predicate) followed by a list of the objects (terms) being related in this way.

For example the fact 'the weed Galium aparine is present on the field' could be represented as

weed_present(galium_aparine)

Rules can then be used to define relationships, for instance a rule which warns that the proportion of winter cereals is too high in the field if Galium aparine is present, can be formulated in this way:

suspect(too_much_wintercereals) IF
weed_present(galium_aparine)

Other rules can then advise what to do if too high a proportion of winter cereals is suspected.

For very large knowledge bases the rules and facts representation soon becomes confusing. To add depth to the knowledge base there are several ways of structuring the knowledge.

2.4.3 Semantic networks

The most general representational scheme is called semantic network (Sowa 1984). A semantic network is a collection of objects called nodes. The nodes are connected by links - called arcs in directed graphs. Ordinarily both the arcs and the nodes are labelled. There are no constraints on how they are labelled but some typical conventions are:

1. Nodes are used to represent objects, attributes and values.

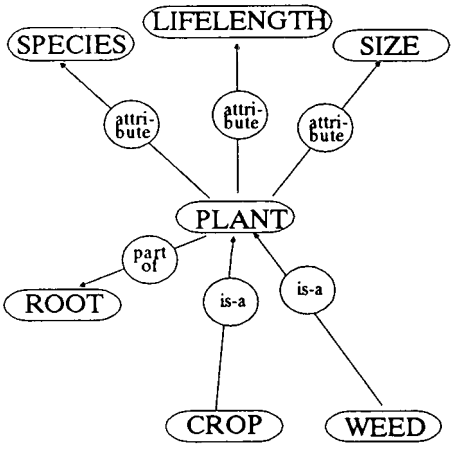

Figure 2.4 Semantic net specifying some relations about plants.

2. Arcs (links) relate objects and attributes with values. An arc may represent any unary/binary relationship. Common arcs include:

- *Is-a* arcs to represent class/instance and class/superclass relationships. In the weeds example we may say that mayflower *is a* weed. That is mayflower is an instance of the class weeds.
- A second common relationship is the *attribute* arc. *Attribute* arcs identify nodes that are properties of other nodes for instance an attribute arc could link weed with competitive ability.
- Other arcs capture causal relationships for instance 'harrowing *causes* plants to die' (fig 2.4).

2.4.4 Frames

Frames provide another method for representing facts and relationships. A frame is a description of an object that contains slots for all the information associated with the object such as attributes. Slots may store values. Slots may also contain default values, pointers to other frames, sets of rules or procedures by which values may be obtained. The inclusion of procedures in frames joins together in a single representational unit the two ways to state and

PLANT	
slots	entries
Species	Forget-me-not
Lifelength	default: 1
Size	10cm
Drymatterminim.	if needed look in table x
Propagation	if needed look in table y under species

Figure 2.5 Frame for a plant including some of the attributes

store facts: procedural and declarative representations (fig 2.5).

2.4.5 Object oriented representation

Representing knowledge with object-attribute-value triplets is a special case of semantic networks. In object oriented representation the basic unit of description is an object. Objects may be physical entities such as soil or plants, or they may be conceptual entities such as harrowing. Objects are characterized by attributes or properties where values are stored. Typical attributes for for instance physical objects are size and colour.

Objects that share properties are organized in classes. For instance *chickweed common*, *forget-me-not* and *mayweed* can be thought of as objects assigned to the class *weeds*, called instantiations of the class. A class can belong to another class as *weeds* to *plants (fig 2.6)*. This whole concept gives rise to a hierarchical representation of the world.

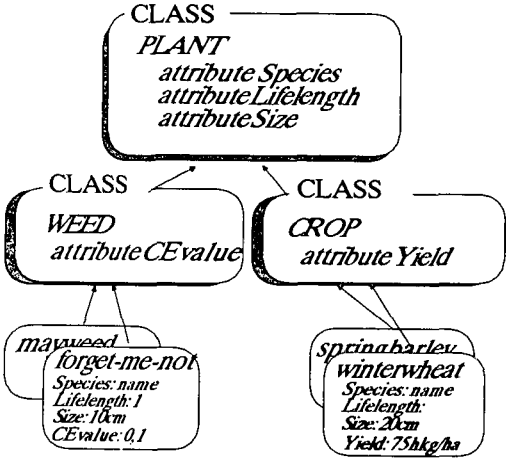

***Figure 2.6** Object oriented representation of classification.*

The class can store information relevant to all its objects and the objects are created with this information. Classes inherit information from their superclasses. One obvious advantage of classification is that it is an economical way of representing data and knowledge in areas where a hierarchical approach is used in problem solving.

2.5 Inference principles

Logical inference is the process of deriving a sentence s from a set of sentences (rules) S by applying one or more inference rules or deduction rules, usually with the purpose of showing S implies s.

2.5.1 Modus ponens

The most common inference strategy used in knowledge based systems is the application of a inference rule called modus ponens. This rule states that when A is known to be true and a rule states 'if A then B', then it is valid to conclude that B is true. Another way to say this is that if the premises of a rule is true then we are entitled to believe that the conclusions are true.

Modus ponens is very simple and the reasoning based on it is logically valid and easily understood. When this rule is the only one used certain implications which are logically valid cannot be drawn. For example the rule called modus tollens which says that if B is false and there is a rule 'if A then B', then it is valid to conclude A is false. This logical inference is seldom used in most expert systems.

2.5.2 Resolution

Resolution is a very general, and easily implemented inference rule used in logic programming. The most popular logic programming

language PROLOG uses resolution. It works on rules and facts brought on a special form called clauses (Hamilton 1988). In this form assertions are written as disjunctions of positive and negative literals. A literal being a proposition or predicate, here shown in statement calculus (1).

$$A \vee B \vee \neg C \qquad (1)$$

A rule will then be on the form '¬A or B' which is equivalent to 'if A then B' (this may be seen from the truth tables). Every sentence in first-order logic can be brought on this form. The operation needed for resolution is very simple. Resolution operates by taking two clauses containing the same literal. The literal must occur in positive form in one clause and in negative form in the other. The two clauses (above the line in the figure) can be resolved to one (beneath the line) by removing the literal in both clauses and combining the rest of the two parent clauses (2).

$$\frac{A \vee \neg B \quad B \vee C}{A \vee C} \qquad (2)$$

If resolution on the clauses in a knowledge base eventually reaches an empty clause, a contradiction exists. If a contradiction exists it will be found eventually, when resolving the clauses in a knowledge base. The example shown is for statement calculus, but for predicate calculus the mechanism is similar except that care has to be taken for quantifiers when the rewriting to clauses takes place (Hamilton 1988).

In logic programming the problem amounts to checking that a goal - for example a diagnosis - is a logical consequence of the set of facts and rules in the knowledge base. It is impossible to check whether the rule is a logical consequence, but it is possible to check whether the negated goal is inconsistent with the knowledge base. The goal is negated and resolution is made on the set of facts, rules and the goal. If the goal is a logical consequence of the knowledge base the inconsistent 'empty clause' will be deduced in a resolution with the negated goal and the knowledge base.

2.5.3 Reasoning with uncertainty

Experts sometimes make judgments when not all of the data are available or, some may be suspect, and some of the knowledge for interpreting the knowledge may be unreliable. These difficulties are normal situations in many interpretation and diagnostic tasks. The problem of drawing inferences from uncertain or incomplete data has given a variety of approaches.

One of the earliest and simplest approaches was used in one of the first expert systems, MYCIN. It uses a model of approximate implication, using numbers called certainty factors to indicate the strength of a rule. The certainty factor lies between 1 and -1, where 1 means definite certainty, -1 means definite not, and 0 means uncertain . Evidence confirming a rule is collected separately from that which disconfirms it, and the 'truth' of the hypothesis at any time is the algebraic sum of the evidence.

It is often questioned whether this solution to the handling of uncertainty is unnecessarily ad hoc. There are probabilistic methods, for example Bayes' theorem that could be used to calculate the probability of an event in light of a priori probabilities. The main difficulty with Bayes' theorem is the large amount of data and

computations needed to determine the conditional probabilities used in the formula. The amount of data is so unwieldy that independence of observations is often assumed in order to calculate the probabilities. Lately though new methods have been found to use Bayes theorem in connection with networks (Spiegelhalter & Lauritzen 1990, Spiegelhalter & Lauritzen 1988).

Another approach to inexact reasoning that diverges from classical logic is fuzzy logic. In fuzzy logic, a statement as for instance 'X is a large number' is interpreted by a fuzzy set. A fuzzy set is a set of intervals with possibility values, such that the possibility of X being in an interval is the corresponding possibility value.

2.6 Inference control

A requirement for knowledge processing is a control structure; this determines the way in which the various rules are applied. Essentially a control structure enables a decision to be taken on what rule to apply next. In most real situations the number of rules required will be very large and many different forms of control structure are possible. Rules could be taken in sequence, or some subset of rules (metarules) might be required to decide which other rules to apply. The mechanism by which a rule is chosen in situations in which there is a choice is also a control structure problem.

2.6.1 Backward and forward chaining

Many existing expert systems use a backward chaining strategy. In backward chaining the inference engine starts with a conclusion of a rule as a goal or hypothesis and works backward taking the premises of the same rule as new subgoals to be proved. If the possible outcomes are known - for instance possible diagnoses - and if they are reasonably small in number, then backward chaining is very efficient. Backward chaining systems are also called goal-directed systems.

In the case of things to be assembled or designed the possible outcomes can be astronomic. In that case it is more efficient to reason forward from the initial states, compare data with premises of rules and add conclusions to the list of facts until a state that matches the goal is reached. This type of reasoning is called forward chaining.

Sometimes it is a good idea to attempt a solution searching bidirectionally (that is, both forward and backward simultaneously). The search then starts at both the goal state and the initial state, and the control system then decides at every stage whether to apply a forward or a backward rule.

2.6.2 Search

Under the process of searching for a solution to a problem, it has to be decided which rule to apply next. Very often more than one rule will have its left side (forward chaining) or right side (backward chaining) match the current state. It is clear that how such decisions are made have influence on whether a problem is solved and how quickly.

Many problem solving systems in AI are based on a description of the problem-solving as a search through a state space. The state space is the set of problem states and the transitions between problem states. Problem solving is carried out by searching through the space for a state equals a goal.

One class of methods to do this is blind search. This type of search can be forward, backward,

or proceed both ways at the same time. Given an orientation for the search, there are several different systematic orders in which the nodes of the search space may be considered. Depth-first search is a process that considers successive nodes in the space before considering alternatives at the same level. It does not return until a failure has been obtained. A breadth-first search expands the search graph differently and considers all nodes on one level before proceeding to the next, and so descends uniformly across all possibilities. In a complete search depth-first and breadth-first approaches examine the same number of nodes, however breadth-first search needs more memory because many paths are examined at the same time.

Complete search will in principle always find a solution to a problem if there is one. Blind search methods are not practical for many problems because the search spaces have too many nodes. For each step more the number of choices multiplies the total number of combinations. This is called the combinatorial explosion. For many applications it is possible to include domain-specific information to guide the search process and to reduce the search space. This is called heuristic information, and such search procedures are called heuristic search methods. Some heuristic search methods will guarantee to find the best answer, others will only find a 'good' answer.

Several types of heuristic search algorithms have been used for expert systems. One form of heuristic search is to direct the search in a best-first order. To determine which branch to expand, a domain-dependent function is used to estimate the closeness of the path to the goal. This function is especially useful if it is monotone, so the evaluation function decreases as a goal is approached.

Another way to avoid the combinatorial explosion is to simplify the problem. If it is possible, it is often advantageous to decompose the problem to several smaller ones, and try to solve each of these.

Another way of simplifying the problem is by making abstractions of the search space, and tackle the problem using intermediate levels of abstraction, thereby transforming the problem into less complicated problems.

2.6.3 Monotonic - non monotonic reasoning

Another distinction among inference engines is whether they support monotonic or nonmonotonic reasoning. In a monotonic reasoning system, all values concluded for an attribute remain true for the duration of the consultation session. Facts that become true remain true, and the amount of true information in the system grows steadily or monotonically.

In a nonmonotonic system, facts that are true may be retracted. Planning is a good example of a problem type that demands nonmonotonic reasoning. Early in the planning process it may seem logical to go a certain way. Later, as information comes in, an early decision may turn out to be wrong, and need to be retracted. Changing the value of a single attribute is not difficult. But tracking down the implications based on this fact may show up to be difficult.

2.7 Construction of expert systems

The construction of rule based systems is very different from ordinary program construction. In the last process knowledge of the domain is better described and even sometimes formalized, and the construction of systems proceeds in a strictly sequential way through phases as

problem analysis, program specification, planning, coding and testing.

Knowledge sources for an expert system can be several kinds, knowledge may be acquisited from books, examples or an expert. These sources contribute with different kinds of knowledge. From text books and such a kind of knowledge called public knowledge can be acquisited. This is the fundamental knowledge of the domain and contains knowledge as concepts, causal relations and definitions. The expert also possesses this kind of knowledge, but has additional knowledge such as rules of thumb, how to solve problems efficiently, and exceptions to rules. This kind of knowledge (experience) is called private knowledge and is crucial in the building of expert systems.

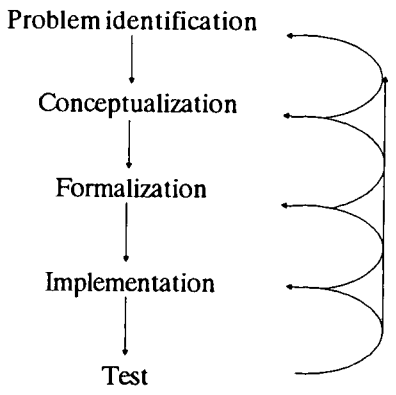

Figure 2.7 Construction model for rule based expert systems

The expert, or domain expert, is a critical factor in expert system construction. The efficiency of the system relies on the incorporation of the experts knowledge on problem solving strategies and experience. It is a well known doctrine that experts are unable to build expert systems unaccompanied. The requirements in the construction especially in the formalization phase are normally far from the requirements the expert will naturally see in the domain. Therefore another person called the knowledge engineer constructs the system based on information from the expert and sometimes other sources.

In the construction process the knowledge engineer proceeds through several stages. These stages can be characterized as problem identification, conceptualization, formalization, implementation and test as shown in fig. 2.7. The knowledge engineering process can be divided into three phases. The first phase is characterized by domain identification. The second is several iterations of knowledge acquisition and knowledge-based development including conceptualization, formalization and implementation and test. The final phase is the installation, maintenance and use of the system.

During identification, the knowledge engineer and expert work together to identify the problem area and define its scope. They also define the participants in the development process (additional experts), resources needed and the goal of building the expert systems.

During conceptualization, the expert and knowledge engineer explicate the key concepts, relations and information flow characteristics needed to describe the problem solving process in the domain.

Formalization involves mapping the key concepts and relations into a formal representation suggested by some expert system building tool or language.

During implementation, the knowledge engineer combines and reorganizes the formalized knowledge to make it compatible with the information flow characteristics of the problem. The resulting set of rules and control structure define a prototype program capable of being executed and tested.

Finally testing involves evaluating the performance of the prototype program and revising it to conform to standards defined by experts in the domain.

This process is not as neatly and well understood as it sounds. The stages are rough characterizations of the activity and are neither clearcut, well-defined or independent. The stages are traversed several times and the process will vary from one individual situation to another. The construction process is not understood well enough yet to outline a standard sequence of steps that will optimize the expert system building process. Research is going on to develop methods to improve the first phase. There has been an increased awareness that the quality of the knowledge acquisition phase can be raised by a better analysis from the start (Woodward 1990, Nwana et al 1991).

2.7.1 Knowledge acquisition

Knowledge acquisition is the collecting and formalizing of knowledge, prior to the implementing in a knowledge based systems knowledge base.

In knowledge acquisition public and private knowledge is segregated. Public knowledge is the knowledge available in text books and such. Private knowledge is attained through experience and by working with experts.

The quality of first generation knowledge based systems depends on the success of extracting and expressing the private knowledge of one or more experts in a way usable to an expert system. This part of the knowledge acquisition process is called the knowledge elicitation.

Knowledge elicitation establishes the base for the expert systems function, ie the collection of knowledge which shall exist in the knowledge base of the expert system. Knowledge elicitation is performed by the knowledge engineer in cooperation with the domain expert.

2.7.2 Knowledge elicitation

Knowledge elicitation is a problem. It is difficult and time consuming. The quality of the resulting system depends on completeness and consistency of the enclosed knowledge. Otherwise the function will be bad. The knowledge elicitation must ensure collection of knowledge of acceptable quality.

Some of the problems in knowledge elicitation are:

- The expert has difficulties describing how he solves the problems.
- If the knowledge engineer does not know the terminology, the expert may have difficulties being understood.
- Cooperation between the knowledge engineer and the expert is necessary. This means that the expert must believe in the project and trust the knowledge engineer. Otherwise he will probably lack the motivation for communicating the information.
- Experts forget to mention facts which are obvious to them, for instance assumptions in the problem solving.
- The expert says what should be done i.e gives a text book explanation which are not what the expert would really do. Even though his method of work build on the

knowledge once read, his expertise is in the development of experience. It is this knowledge which is valuable to obtain.

- The expert can only tell what he can verbalize. Something never before expressed in language is difficult to explain. And problem solving may have become a routine, making it difficult for him to explain the structure. Knowledge has been compiled. This will make the expert give 'black box' answers such as 'it is the sensible thing to do'.

Several techniques have been developed for this task. The standard method for knowledge elicitation is interviews. There are several techniques available to improve the collected information, for example structuring the interviews (Brummenæs 1990). The collected knowledge is then coded in a knowledge base editor. Other methods are protocol analysis, scaling techniques and card sorting.

Interviews

In the interview situation the knowledge engineer questions and the expert answers. The interviews in the knowledge elicitation phase vary from totally unstructured to formal structured interviews. A structured interview is planned by the knowledge engineer, who has determined specific goals and questions ahead of the interview. An unstructured interview develops more unexpected, and the questions to the expert are more spontaneous and random.

Unstructured interviews are not effective and are often only used to help the knowledge engineer to become familiar with the domain and its terminology. Structured interviews are used when the knowledge engineer has a general knowledge of the domain. They are good at concentrating the interview on a special subject.

There are different ways of structuring interviews for instance:

- Scenario simulation, where elementary problem situations are defined. The expert chooses one of the situations and talks through the reasoning towards a solution.

- 20 questions. The expert gets a problem to solve and is allowed to ask 20 questions to the knowledge engineer during the problem solving. The expert has to pick question with a high information value. The purpose is to reveal the order in which he tests the rules.

It is considered important to collect as much information as possible in the interview. Normally the interview is tape recorded, concurrent with use of notes. Another possibility is to video record the interview. Technical help should only be used if it does not disturb the expert.

Protocol analysis

Protocol analysis is a method used in psychiatry to examine how people solve problems. The analysis starts with observations of an expert solving a problem in the domain. The problem can be real or constructed. Observations of the expert and what he says is written in protocols. The purpose is to reconstruct the underlying structure in the work of the expert.

The protocols can be written during the problem solving - parallel - or afterwards in retrospect.

An advantage of this method is the possibility to directly observe the expert solving problems and the information he uses. But the protocols are often ill structured, and it may be necessary to make many protocols to cover the problem area.

Scaling techniques
In the scaling techniques the expert judges concepts from the domain in a way which gives a measure for the psychological distance between concepts. This measure reveals similarity or relationship between concepts in the experts opinion. Different scaling techniques are:

- Multi dimensional scaling (MDS), where the expert, for all pairs of concepts gives a point according to the closeness of the concepts. Low values indicates short psychological distance, which means a close relationship. MDS arranges the concepts in a multidimensional space according to the points. The distance between points in the space reflects the relation between them. An advantage with this method is that it is formal. There is some questions about how to interpret the result.

- Repertory grid is a representation of the experts view of the domain. It consists of elements, concepts and a scaling of each concept for each element. Elements are examples from the domain for instance sicknesses if it is a diagnosis system, these are the subjects whose relations are to be examined. Concepts are a bipolar attribute which all elements possess for instance friendly-unfriendly.

  First the elements and the concepts are found. Then all elements gets a character on a scale - for instance 1 to 5 - for each concept. The character shows the experts judgment of the degree to which the element possess the concept (attribute) or the opposite. The analysis shall reveal similarities between elements. The grid may be analyzed several ways. It may be reorganized so similar elements are close. Correlation coefficient can be calculated or the grid may be analyzed statistically by for instance cluster analysis.

  The method is quick, but a problem is that all elements are assumed to be included. However only a limited number can be included if the combinatorial explosion is to be avoided.

Card sorting
The purpose is to reveal the experts own classification of concepts and relations between concepts. The concepts of the domain are written on cards. The expert has to sort these cards in groups according to relationships and name the groups. The analysis gives a picture of the organization of concepts in the domain. A condition is that the domain is hierarchically organized.

Induction
Induction is the construction of a set of rules from a set of examples. The expert is asked to provide a training set of critical cases with examples of problems from the domain and the solution to them. The cases should encompass crucial and complete information. The cases are distinguished by a set of attributes, in a similar way to the repertory grid technique, and the data should not be noisy. An inductive algorithm - of which the most famous is called ID3 - is applied to the set and eventually forms a decision tree. A crucial item for the correctness of the induced rules is that all relevant information is encompassed (Hart 1986, Shaw & Woodward 1990).

The different techniques do not provide the same type of information. There is a difference between the information from the formal techniques and interviews (Burton et al 1990). Burton et al (1990) compared the relative efficiency of four methods: structured interview, protocol analysis, laddered grid and card

sorting. They measured the efficiency in information per time unit and found that protocol analysis performed least efficiently. In an evaluation of the elicited information by the expert the outcome of the protocol analysis was also rated lowest. The efficiency of the grid technique and card sorting was high and they provided complementary knowledge to the standard interview. The interview produced the highest count of true clauses.

2.7.3 Tools

When choosing a tool for expert system building a variety of options exists. General purpose languages as LISP, PASCAL and FORTRAN can be used as well as general-purpose representation systems developed specifically for knowledge engineering.

One strategy is to implement from scratch in one of the standard programming languages. LISP is chosen in many AI applications especially because it is oriented towards symbolic computation. Whatever programming language is chosen, an expert system requires two major components: An inference engine and a body of rules. A language or set of concepts to express the rules has to be built. Once the rule language has been defined, the inference engine can be built in terms of the general framework or architecture selected.

Rather than building an expert system from scratch, it is sometimes possible to borrow something from a previously build expert systems. This strategy has resulted in several new software tools for knowledge engineering, which may be described as skeletal systems for instance EMYCIN. In these systems the rules in the original systems are removed and the rest reused. Some problems may occur in using these tools:

- The old framework may be inappropriate for the new task.
- The control structure may not match the one desired.
- The rule language may be inappropriate.

If these difficulties are overcome, it is possible to build a new system much quicker than from scratch.

New software tools - shells - have been built. They are not build by borrowing from expert systems, but resemble these systems in that they contain the inference engine and the user interface which is the surroundings to the knowledge base. These software systems are more or less flexible in the possibility to affect the control and inference, and to represent the knowledge. The limitations above also count for shells. When using a shell the paradigm selected in it must ordinarily be used in the constructed expert system, so it is important to select a shell which suits the task. In exchange one gains a quicker development.

2.8 Agricultural applications of knowledge based system concepts

The first papers on agricultural applications of knowledge based systems appeared around 1985 (McKinion & Lemmon 1985, Jones 1989). At that time the recent development of fast cheap personal computers and specialized software systems made knowledge based systems a promising tool and there was considerable interest in these presentations.

Since then there has been a decline in enthusiasm among agricultural researchers. One of the reasons for the general lowering of expectations is the fact that most of the described systems never seem to get into production use.

This chapter will review several known knowledge based systems - prototypes as well as those in use - to set the state of the art. It will discuss possible explanations for the present absence of operational systems and lastly give ideas for future development and use of knowledge based systems in agriculture.

Knowledge based systems can be used for many purposes and are often classified accordingly, for instance diagnosis systems. In this chapter the systems will be classified according to the first classification mentioned in section 2.2. This is not always easy, most systems comprise parts which belong to different classes. In this situation the system is classified under one - preferably the dominant - class.

2.8.1 Interpretation

FinARS (Boggess et al 1989) is an expert system for evaluating the overall financial health of farm businesses. It was developed over an 18-month period by two experts and a knowledge engineer, each of the experts spending 150 h on the project. The system uses a minimal data set to reach three key economic indicators (liquidity, solvency and profitability) to evaluate the financial health. This evaluation is quick and easy and reveals problems, but cannot provide indepth diagnosis of the source of financial problems. The system was evaluated by a test against ten financial analysts, which showed agreement among the analysts and FinARS rankings.

COTFLEX (Stone & Toman 1989) is a system which partly interpret the economic situation of a farm, partly predict and plan pest control. It integrate expert system technology with simulation models. This system consists of three advisors, two of these use a simple rule based system to call and analyze output from a simulation model - CIRMAN - of whole farm economics to advice on economic features (Helms et al 1990). The model is called externally, making the knowledge in it a blackbox to the system. The third advisor, a pest management advisor, is a diagnostic expert system that gives advice on control of three key insects of cotton, using a cotton model and insect pest models. Besides the models COTFLEX is integrated with a database. The database stores field sampling data from scouting sheets and information provided during consultations. This forms the basis for input data to the simulation models called by the system during consultations.

The National Dairy Herd Improvement Association in the USA maintains a national database of information on cow and herd performance, including information on butterfat content, protein content and milk production. DHLES (Whittaker et al 1989) compares individual herd data against standard data from the national database by extracting important features from the two datasets. Knowledge for the program was obtained from magazines and later refined by specialists in interpretation of lactation curves. An important feature for the program is an associated communications software for automatically accessing the database and extracting the appropriate data.

2.8.2 Prediction

Probably the most widely known agricultural application of an expert system is COMAX. COMAX advises farmers on their management practices ranging from irrigation to the rate and timing of fertilizer application. COMAX is the expert system part of the model based system GOSSYM/COMAX, where GOSSYM is a dynamic simulation model of cotton growth and yield which have been under development for over 20 years. The model was too difficult to initialize and interpret for others

than the developers. But with COMAX as extension to the model it is possible to use. COMAX applies rules to create the necessary data files to run GOSSYM for a particular site and scenario. It also uses rules to interpret model results to make specific recommendations. The system was tested first time on two places in 1984 and since have been tested in 14 states. Users of the system estimated the value in 1987 to be between 100 and 350 $ per hectare (McKinion & Baker 1989).

The developers of SMARTSOY have taken a similar approach. Their software makes insect management decisions in soybeans using SOYGRO, a soybean crop model, in combination with an expert system (Batchelor & McClendon 1989). The expert system portion of the software is based on the knowledge of a soybean extension entomologist. Rules predict damage rates of different populations of the four major insect pests of soybean on foliage and seeds. The range of damage rates are used in a series of simulation runs to evaluate their effect. The differences in yield between the simulations are used with the expected crop value to determine the cost of not treating. Rules were also developed to recommend specific insecticides in different combinations of insect populations. SMARTSOY was tested on eight soybean fields in south Georgia in 1988 and the results were in agreement with the expert for about 80 % of the scenarios.

QSOY (Gold et al 1990) is an expert system for decisions on use of insecticides to protect soybeans against corn earworm. QSOY is integrated with two models, one is SOYGRO, the other is a heliothis population model. The system has not been tested yet.

2.8.3 Diagnosis

WEEDEX (Ballegaard & Haas 1990, Haas & Ballegaard 1988) is a system for identification of weed seedlings. It is written in PROLOG and contains about 240 rules. The system contains knowledge of 80 species of weeds and carries out identification in dialogue with the user. This identification is carried out partly by classifying the plant according to leaf form. This classification is a major source to errors in the present system. The classes of leaf forms are so close that it is difficult to classify correctly with only a verbal description of the leaf forms available. WEEDEX is intended to be integrated in a database system for weed control.

PEST (Pasqual & Mansfield 1988) is a prototype rulebased expert system designed to provide insect identification and control advice for farmers in Western Australia. Knowledge sources for the system were an identification key, a guide to chemical control and an entomology expert. The restrictions on crops and pests are not discussed but the system contains only 38 (PROLOG) domain rules. Though the system has not been evaluated, it is concluded that the domain is suitable for expert system development and that future developments could include integration with decision-making abilities in other areas.

Another very small diagnosis prototype system is reported by Gaultney et al (1989). It is an expert system for trouble-shooting tractor hydraulic systems. The knowledge source for the system was a diagnosis manual. The system should be able to guide a mechanic through testing and diagnosis of the hydraulic system on a tractor.

POMME (Roach et al 1987) is an expert system for apple growers to help them manage their orchards. It concentrates on two things.

On preventing losses from pests by choosing pesticide and spray time, and on weather damage recovery. The pest part of the system gives special attention to apple scab, cedar rust and San Jose scale - the most serious pests in the eastern U.S. apple belt. The system incorporates a model of the apple scab which simulates the growth of the pathogen under given conditions and the physiological reactions of the host. If the symptoms and weather conditions do not lead to a diagnosis the apple scab model is used for a prediction of the fungus' state.

FINDS (Kline et al 1988) is a front end to a linear programming model for machinery selection.

CUE (Morgan et al 1989) is a program with the goal to develop a series of knowledge bases to aid in selecting cultivars for a wide range of crops grown in Scotland. The problem of selecting the right cultivar consists in properly matching cultivar attributes with the characteristics of particular farms. The purpose of each CUE knowledge base is to read and analyze information in a specific crop database (the national database for all variety trials throughout the UK) and provide the farmer with a short list of suitable varieties from which to choose. The first application developed was for winter wheat.

PALMS is an expert database that contains information on palms (Beck et al 1989). The program contains information such as plant characteristics, care, growing requirements, pest problems and suppliers. The system was developed using a special data modelling language, CANDIDE, that provides a notation for describing objects. The system is intended to help customers choose the right palm for their environment.

2.8.4 Planning

Martinsen et al (1986) are three Danish students who have developed a prototype expert system. The system gives advice on pesticide plans in beets for the month of May, considering weed occurrence , - number, and weather. Beets as well as weeds are considered to have a standard developmental curve with a fixed count of days between developmental stages and a procentual restrictive influence from certain weather conditions. The experts cooperating in the project were satisfied with the system performance.

Wain et al (1988) reports on another prototype - an advising system for Scottish sheep breeders. Scottish hill farmers have a problem because the summer is too short for the lambs to reach their saleable body weight. The farmer therefore has to decide whether to finish the lambs on the forage crops at disposal, or to sell them to other farmers. The program implements the problem solving strategy of an expert manager. The strategy showed to be very algorithmic, with expert judgement only apparent in two phases of the problem.

Yoeli et al (1989) describes a prototype system for planning aerial operations for chemical applications. The prototype is an implementation of heuristic methods for planning the aerial operations in an area with a large number of fields to be serviced. All the aircraft depart from one base as a start, but reload and refuel from forward strips during the day. The heuristics finds an optimal way of allocating aircraft to fields and for selecting which refuelling strips should be manned.

EASY-MACS (Huber et al 1990) is a knowledge based system supporting integrated pest management in apples for the five most serious apple pests found in New York orchards. The

system was designed as a series of separate small knowledge bases, each dealing with its phenological stage. Databases serves as a record of the results of independent consultations and as communication between the knowledge bases, so that information gained in one session is available for use in a later session.

A lot of features is integrated with the last planning system. CALEX/cotton (Plant 1989) is the first system build with CALEX. CALEX is a blackboard system where a central part is a scheduler which uses the blackboard to plan activities according to their mutual influence. CALEX can be equipped with knowledge bases as well as ordinary procedural programs. All of these exchange information by access to the common blackboard.

2.8.5 Monitoring

Doluschitz (1990) describes a monitoring system for registering and recording of data in milk production. Sensors collect information such as milk yield, milk components, body temperature, liveweight development and nutrient- and water intake, and an expert system analyzes the behaviour and notifies the farmer when abnormal situations occur.

2.8.6 Control

MISTING (Jacobson et al 1989) is a real time greenhouse monitoring and control system. Misting systems are especially important for plant cuttings during propagation where they are highly vulnerable to water stress as well as over-watering. Control of misting systems in most commercial greenhouses is based on temporal setpoints. Dual timers set the interval between mistings and the duration of misting events, and are adjusted every few days during crop growth according to age, temperature and relative humidity. MISTING was based on the perceived optimal strategy of an experienced grower. The micro computer communicated via a telephone line with a monitor/controller in the greenhouse. Sensor data were provided as facts to MISTING which returned setpoints to the controller, which in turn regulated misting line solenoids. The system ran for 30 days following the growers strategy.

Another system for monitoring and control in greenhouses is described in Fynn et al 1989. It is a system for nutrition injection management in greenhouses. The nutrition consumption of plant is dependent on the growth which again is dependent on the weather. The system integrates three knowledge bases and a weather prediction to decide on the optimal formulation and application rate. Input to the system includes initial parameters (such as planting date,variety and latitude), values from sensors (such as temperature, pH, solar radiation), calculated values (such as crop physiological age, time, date) and once a day weather forecast values input by the operator. Output from the system regulates the amount of water irrigated and set the nutrient injectors. The system automatically anticipates plant requirements and adjusts equipment to optimal nutrient and water supply. The system is implemented and is being tested.

These control systems involve using values from sensors to make automatic setpoint adjustments with minimal user involvement. This type of problem has the obvious advantage of a narrow domain, the input and output is well defined which facilitates the knowledge acquisition. In addition there is no need to consider help facilities or user interface.

2.8.7 Discussion

Obviously very few of these systems are in production run. From a commercial viewpoint

only few of the systems described could be considered successful although the designers generally say that their domain is very suitable for expert system applications. FinARS seems to be a viable system, and so does GOSSYM-/COMAX which have been successfully tested. In Denmark the prototype WEEDEX is planned to be integrated with a database program. If a better classification procedure is found it might then be viable. On the other hand most of the projects are purely academic, and as such they can be considered a success though not a commercial, because they have been useful for evaluating knowledge acquisition procedures as well as provided training of people who continue to be active in the development of decision aid systems.

Expert system technology has been considered a new programming paradigm where declarative rather than procedural programming is used. Many projects have been initiated with expectations of shortcuts in system development. A recent Danish report (Harder 1990) describes five expert system projects where the conclusions are that expert system programming paradigms are not a shortcut and that traditional approaches often are better routes to success. In those five projects it showed up that most ended up using ordinary algorithmic programming to a great extent. This can raise the question whether these systems are real expert systems or not. In an article Jones (1989) discusses this problem and concludes that the goal is to deliver skilful decision making systems and that the best tools for the job should be used in delivering this.

Looking at the systems described, the conclusion about the state of the art is: many projects, few products. What are the trends in systems? It seems that stand-alone expert systems are more successful the narrower and more goal-oriented the project is. The trend in the systems development though seems to be: integrating expert systems with other kinds of software - for instance models and databases (Barrett & Jones 1989). In this way the AI techniques are used in connection with ordinary system building techniques. At all time using the best tool.

Expert system techniques may only be used in part in these hybrid expert systems. Expert systems can be built into larger systems, where the expert system is only a minor module in the whole. It may even be a question of whether they are expert systems in their strict definition. The knowledge based techniques may be used for systems where the goals are more moderate than in ordinary expert system definition. For the user integrating all tools means a great advantage. He could for instance be able to use the same databases for saving data as for running a decision aid program. The perspective in integrating expert systems with other kinds of software is that the use of traditional tools for specific purposes will improve the performance of the decision aid programs. Similarly traditional systems may have an advantage of using knowledge based techniques.

2.8.8 Future use of knowledge based systems in agriculture

In many fields the use of knowledge based techniques seems to have come to stay. Over the years evolution has taken place and now expert systems are often used in integration with other software. The same trend will probably be seen in agriculture. The systems will tend to integrate several different software types and to be useful for different tasks.

The future will bring expert systems in agriculture. The technique is especially suited for

building decision aid systems and monitoring systems so the future will probably bring us systems for for instance monitoring and controlling climates in greenhouses, systems for aiding in planning field operations and other sorts of decision aid in agriculture as well as model based systems.

3 WEEDOF, a prototype of an expert system

The construction of expert systems, as mentioned in section 2.7, is an iterative process. It starts with identification of the domain or subject for the system, and the goals for the development. When the first initial analysis of the chosen domain has been carried through, the programming tool may be chosen. Knowledge is collected in the knowledge acquisition phase and formalized to make it possible to represent it in the language of the programming tool. After a count of iterations of knowledge acquisition and implementation the system will be finished.

This chapter describes the development of a prototype - WEEDOF. The sections follows the construction process. The first section treats the choice of domain. In Section 2 choice of tool is treated. The chosen expert system shell - EGERIA - is described in section 3. Section 4 describes the knowledge acquisition phase with the methods used: Literature analysis, a new method in knowledge acquisition, and interviews. Section 5 describes the implementation in the shell. Section 6 is an assessment of what is missing to make the prototype a finished system. And section 7 is summary and conclusions.

3.1 Choice of domain

As a part of this project the techniques for construction of knowledge based systems should be used for creating a prototype. One of the proposed problem domains was non-chemical methods of weed control. Other domains were control of couch grass, and sclerotinia on rape. Non-chemical control of weeds was obviously the most complex domain. The others were both very narrow and looked straight forward to solve using traditional methods.

The complexity was seen as an advantage here, as the building of the prototype was an informative experience, and the complexity would give a chance to deal with many types of problems in the construction process. Furthermore the expert from the Institute of Weed Control assigned to the project seemed interested, and willing to spend the time needed for the development. He also had a little programming experience enabling him to better understand a computer program, and what computers can do.

Old traditional forms of weed control include preventive methods as balanced crop rotations, weed free seeds, good soil preparation and good growing conditions, as well as mechanical methods.

Today these methods are somewhat eclipsed by chemical control methods, which are much more effective. Mechanical control has, for a long time, only been used in organic farming.

Recently a growing interest has emerged for alternatives to chemical control. One of the reasons is the consumers interest in a minimal use of pesticide. Likewise recent dictation of large reductions in pesticide use by the National Agency of Environmental Protection has renewed interest in alternatives to herbicides.

Control of weeds by non-chemical methods has proven to be very difficult. A lot of factors influence the growth of crop and weeds on a field. The whole idea in growing a crop without using herbicides involves using all means to strengthen the crop and give it a high potential for competition. At the same time the weeds should have bad conditions for growth and development.

The aim for mechanical control is to damage the weeds sufficiently enough to kill them or at least make them bad competitors. If mechanical control has to be used, the crop is likely to be damaged also due to the small selectivity of the mechanical methods. The effect of mechanical control is thus influenced especially by the weather conditions. Rain can make the effect of harrowing on the weeds negligible. The harrowing tears the weed up, but rain afterwards will make it possible for it to root before drying out. Soil type, type and method of treatment, and crop and weed size also influence the effect and make the result hard to predict.

To replace part of the herbicide use by non-chemical methods requires research and development to improve the methods. This research started some years ago on the Department of Weed Control, The Danish Institute of Plant and Soil Science, at Flakkebjerg. Progress has been made in improving the methods, and establishing the relations between conditions for control and the resulting effect.

As the work in the project was purely research, and the goal merely was to produce a prototype, no calculations of economic or other benefits of the development were made. In the light of the demand to diminish the pesticide use and the following need for alternative control methods, the most recent knowledge in the area will be needed to effectively control the weeds. In the organic farming where these non-chemical methods are the only used, the consultants also reports a need for knowledge on the best use of these methods. A quick propagation by means of knowledge based advice systems will probably become of practical importance.

3.2 Choice of tool

For development of rule based systems two types of developmental languages are possible: conventional languages or expert system shells. Conventional languages like Pascal, C, Prolog or LISP give great flexibility but demand everything to be programmed from scratch. Shells contain algorithms and routines for many uses and reduce the time for prototyping, but give less possibility to control the end system.

For the development in this project a shell was chosen, partly to experience such a programming tool, partly to speed the development. The choice of the shell took place in cooperation with researchers on another expert system project for discount reasons. As the tool was going to be used for several prototypes, the shell preferred would have several knowledge representation facilities, and preferably several control capabilities. The shell chosen was EGERIA, developed and marketed by Expertech Ltd. In Denmark Axion A/S is the distributor. EGERIA is delivered as a development - and a runtime license.

3.3 The expert system shell, EGERIA

Egeria is an advanced expert system shell with many forms of knowledge representation. The language though resembles more a programming language than a rapid prototyping tool. The emphasis in EGERIA is on strong typing like in most ordinary programming languages. Knowledge bases, known as models, are developed by writing an ASCII source file and compiling it. The windowed interface is specified in the source while the actual appearance to the user is defined in a separate window editor.

EGERIA permits both declarative and procedural programming. The syntax is highly structured and consistent but it is difficult to intuitively read the knowledge. consequently the application's knowledge cannot be shown directly to the expert for comment and correction. It offers an array of knowledge representation features including simple types, classes, objects, relationships, groups, tasks, procedures and functions, collectively called items.

3.3.1 Knowledge representation

Data values are held in typed variables much like conventional programming languages. The simple types are CONDITION, STRING, REAL, INTEGER and PROBABILITY. Similar to Pascal an enumerated type can be specified. Variables are defined by declaring their type, name and a 'derivation expression'. The derivation expression describes all the possible ways of finding a value for the variable and may include a variety of test and expressions. Condition variables are similar to booleans in programming languages but can take the value UNKNOWN in addition to TRUE and FALSE. This derivation expression defines a condition variable:

```
CONDITION corn IS
          cropanswer IN (winterrye,winterbar-
          ley,  winterwheat,  springwheat,
          springbarley)
```

This is the EGERIA equivalent of a rule.

String variables hold text strings of variable length. Integer and real variables hold numeric data and probability variables hold a real number between 0 and 1. All numeric variables are stored as a pair of numbers ie a range representing the current best estimate of the value. As more information is gathered the numbers converge.

The enumerated type allows the programmer to define a new type with a set of values. Variables of this type may be single or multiple valued. There is a restriction on the allowed values of an enumerated type. Values must only appear in one enumerated type definition and must not clash with reserved words. This first condition can cause problems. For instance one may want to have menus with all possible crops to select between in some parts of a program, and in other parts one may want to restrict the choice to only wintercrops. As the same crops are included in these two enumerated types this is illegal.

EGERIA allow the use of multi enumerated variables. These contain sets of values, for instance a variable *weed_population* can contain the set of weeds present on a field. For multi enumerated variables set manipulation functions and operators are provided. For example IN to test set overlap.

Variables of the same or different types can be formed into a group and addressed as a single variable. A group variable actually holds a set of references (pointers).

EGERIA provides object oriented knowledge representation. Classes describe concepts with relating attributes, tasks, procedures and window definitions. The items becomes slots of the class. One class may inherit from any number of classes which in turn may inherit from any number of other classes. Instances of classes, called objects, can be defined statically in the source or created dynamically using the CREATE command or a relationship order. The values of class variables (defaults) may be overridden at the object level:

```
OBJECT weed couch_grass WITH count = 12
```

Defining classes within classes allows component part information to be represented, The outer class could for instance be *plant* the inner classes *root, leaf* and *stem*. This is not a hierarchical structure - the attributes of the outer class are not inherited by the inner class but can be accessed.

Variables in a class can be referenced with the OF operator. Although it should be possible, problems have been encountered in referencing variables from classes in one line of the hierarchy from another line of the hierarchy. Instead it has been necessary to make a reference from the top class COMMON which can be accessed by all other classes.

Relationships can be defined as named links between different objects. One to one, one to many, many to one, and many to many relationships may be defined. Primary relationships create objects to stand in the relationship. For example:

```
CLASS people
        MANY people offspring RELATING
        ONE parent INITIALLY numberofOff-
        spring
END CLASS
```

Each object of class people will have a relationship with a number 'numberofOffspring' of other people - called offspring. The reverse relation is named parent. Secondary relations define links between objects that already exists.

3.3.2 Control

Control of consultations is done using 'active items'. These include tasks, procedures, break items and explain items. The active items execute procedural statements and can assign values to variables, cause backward chaining, cause questions to be asked, initiate procedures and so on.

Procedures can only be initiated by other active items. Variables may be passed as parameters by the use of groups. The procedural language is similar to structured languages as Pascal and includes structures such as loops and if-then-else constructs.

Tasks contain similar statements but do not take parameters. Instead they have a condition clause that indicates when they should be fired. The condition clause is a logical expression referring to any item in scope. If it evaluates to true the task becomes eligible for activation. Tasks are used mainly to control the progress of a consultation. When an object is created any task defined within the class is created, a task with a WHEN CREATED clause will then be eligible for activation after it is created.

Builder defined functions can accept any type of parameter and return a single result. The body of a function can only include one single expression. A range of built in functions is provided for the data types including real to integer conversion, string manipulation and mathematical functions.

3.3.3 Reasoning

The major feature in EGERIA is forward chaining. It ensures that any change of a value is propagated throughout the model. During the forward chaining cycle the inference engine puts all the tasks eligible for activation onto a stack as it comes across them. When the forward chaining phase is complete the top task is popped and activated.

The backward chaining is initiated using the INVESTIGATE command in a task or procedure. Parameters for the command is a group of goal variables for which a search for values shall be performed. The values for the variables are then deduced by backward chain-

ing using the knowledge in the knowledge base, in the database and if necessary by asking the user. The chaining continues on each variable in the parameter list (depth first) until the condition specified in one of a number of UNTIL clauses is satisfied. The procedural part of the particular UNTIL clause is then executed. Each backward chaining cycle is followed by a forward chaining cycle to keep the model self-consistent. This forward chaining cannot be controlled or scoped. In my prototype I have not had problems with speed but with larger applications this action may cause significant delays.

3.3.4 User interface

The default interface is useful in the first stage of development. Predefined windows are used for asking questions (with a QUESTION command), to output text (generated with the WRITE and WRITELINE commands) and to show DOS text files.

windows

Later the window system will be used to make application windows. This means defining logical windows in the source code and mapping them to window images designed in the window editor. The window images are held in disk files.

Each window may contain a number of fields for input or output of variables. In the window definition variables are declared with the INPUT or OUTPUT keywords defining the nature of the field. The variables are declared in the order by which the fields are numbered in the window design. Different formatting options are available for output. When variable values are changed in the system the window fields are updated as well.

Menus can be generated easily: A multiline field is defined in a window, and the field is defined to contain an enumerated type variable. The values in the type then appears in the field of the window when variable values are asked, and one or several alternatives can be chosen according to the variable type. With a single-line field the alternative values can be displayed using the cursor keys.

In numeric fields the user can enter single values or a range. For all variable types the user can enter unknown input with the default string '!'.

Windows can be temporarily or permanently displayed using procedural statements. When a value for a variable is sought all variables defined in the window must be answered. Text output can be directed to any window whether it is displayed or not.

Breaks

Break items are a mechanism for trapping keystrokes and response according to the key by executing specific procedural code. Breaks may be global to the model or local to window or class definitions.

Explanation facilities are provided using the WHY statement. The command retraces the line of reasoning in the most current backward chaining, by executing any EXPLAIN clauses attached to variables on that path. The EXPLAIN clauses contains procedural statements, typically output text. That only the most current backtrack can be traced makes it of very limited use. Many shifts between forward and backward chaining or only forward chaining disrupt the line of reasoning.

For questions asked by a forward chaining procedure there will be no justifications. In WEEDOF the search changes between forward

and backward chaining fairly often and the traces produced by WHY go only one step back. This is a serious deficiency.

Others
The built-in graphics language supports CGA, EGA and VGA. A separate image grapper is provided and should be able to capture images to be displayed directly from the model, I have not used this feature.

Reports can be generated during consultations. Text contributing to the report can be output with a key number used to sort the text by the key before printing. Reports can be saved on disk or sent to printer from within the model.

3.3.5 Programming environment

The development environment has a window editor, syntax directed editor, and relative easy switching between editor, compilation and execution.

The editor is a multi-file editor. All files from a chosen model are loaded together and are available for editing. Text may be copied between files. The syntax directed editor facility works by inserting syntax templates into the model file. In practice it is of very little use. Besides this there is no help system attached to the editor. In the system parameters an external editor may be specified.

The compiler is quick and produces a listing and cross-reference output from which the editor (from here only the default editor is used) can be switched on to show the line with an error. The error messages are unhelpful, sometimes misleading, and the compiler does not always find the correct error line. Fixing syntax errors are therefore very tedious and especially in the start very time consuming.

Program execution is hard to control. A simple debugger, activated from a break, is provided which can be invoked during execution to browse variable values and class/object structure. A regular trace facility to follow the reasoning progress in the program is not included, which makes it very difficult to track down application bugs.

The runtime system is used from the development environment to see the running application.

The window editor has a separate menu that allows files of window images to be created, deleted, copied etc and individual windows to be edited. In the editor the window size, colour and position are specified along with the fields and texts. Fields are sequentially numbered as they are created, but may be reordered. Once windows are defined they may be copied and may also be pasted to the background while editing other windows to help positioning.

External functions can be linked into EGERIA. They can be used in the model as a built-in function. Roll-out of the model is supported to make space for called programs. I have not used these facilities.

Access to DOS files are provided through FILEIO blocks, defined in the same way as windows. A FILEIO block with no fields defines a stream file. Through the FILEIO blocks EGERIA has interface to files in DBASE III format, the fields of the record corresponds to the input/output fields of the FILEIO block. The USE command can send output to the specified file, and ASK and INVESTIGATE statements seek input from a file.

USERIO blocks are used to handle devices other than files.

3.3.6 Hardware requirements

EGERIA runs on PC's under DOS with a minimum requirement of 640KB. The software itself takes up 900 KB on the harddisk. Examples are provided with the program. But these only shows procedural code and window management, not object-oriented facilities, database interfaces or other more special things. Three manuals are provided, one 435-page book called 'Expert systems with EGERIA' (Anon. 1989a) which describes the concepts of EGERIA. A PC-DOS reference manual and a 'Technical Reference Manual' (Anon. 1989b and c), the examples in this last book are very short and sometimes lack clarity. The two first books lack an index which is very irritating.

3.3.7 Summary

EGERIA is a sophisticated tool with excellent knowledge representation facilities, and good inference control capabilities. It has complete truth maintenance through the automatic forward chaining, which cannot be controlled. Furthermore it seems very quick.

It has some weaknesses:
From a programmers standpoint the development environment is old-fashioned. For instance, it does not include graphic facilities for displaying class-hierarchies. The compiler is frustrating to use, it is often unable to find the correct error line, and the error messages are very unhelpful. There is no trace facility. The syntax is difficult, and the complex language takes time to learn well.

An advantage normally claimed for declarative programming is that the knowledge represented in the language is accessible to others than programmers. The expert are then able to read and understand the knowledge implemented, which aids in the knowledge acquisition process. This is however not the case for EGERIA - as earlier stated the syntax and language is difficult, which makes it difficult to read the knowledge.

Finally there is a problem with explanations. It is considered very important for expert systems to be able to explain conclusions and the reasons for asking specific questions. In EGERIA there is no possibility to produce justifications and explanations can only be generated when backtracking is the only regime used.

3.4 The prototype, knowledge acquisition

Knowledge acquisition for expert systems is an iterative procedure, as described in section 2.7. The procedure perhaps starts with reading some text books about the subject to become acquainted with the domain. Then the knowledge acquisition procedure goes on cycling through knowledge elicitation - usually from an expert - data analysis and knowledge representation.

In the development of WEEDOF this procedure was different. The knowledge engineer, having herself an agricultural degree was acquainted with the domain from the start. Text books were used in the beginning of the knowledge acquisition process. The knowledge in the text was analyzed and formalized. So a great deal of knowledge was collected before the knowledge elicitation from the expert took place.

As described earlier the domain for the prototype example had been chosen to be weed control in organic farming. The Department of Weed Control in Flakkebjerg was interested in cooperating in development of an expert sys-

tem. One of the experts in weed control in organic farming was willing to participate in the development of the system.

This chapter will describe the process which started with a literature analysis of selected texts. On the basis of the information extracted from the texts, interviews were planned with the expert. Each interview provided information which was formalized and build into prototypes.

3.4.1 Literature analysis

Texts are not always used in knowledge acquisition. When they are used it is only informally for the knowledge engineer to get acquainted with the domain ahead of the knowledge acquisition process. The different use of texts in this project was inspired by the work described in two papers on the knowledge acquisition for an expert system to control biological water cleaning systems (Østerby 1990, Sørensen 1987).

The knowledge acquisition procedure started with collecting possible candidates of texts for use in the literature analysis. Ideally the material for the analysis should be a text book giving a thorough description of elements, causal relations between these and methods in the domain.

The expert was asked to suggest texts, preferably text books on the subject. If such did not exist then texts which gave an introduction to the domain. The expert came up with 12 texts, most of them papers on research subjects. Three of the texts though were introductory texts - two of them chapters from a book about weed control written by researchers at Flakkebjerg (Rasmussen 1990, Rasmussen & Vester 1990), the third an examination treatise. These three were selected for the analysis with priority on the two chapters from the weed control book.

The text in the selected papers was read thoroughly. Every sentence with elements of relevance was marked. Even for someone not familiar with the field these sentences can be identified by the content of words specific for the domain - words, that do not appear in other contexts eg fiction (appendix 1). The marked sentences were collected in a document, a sort of knowledge survey.

The knowledge survey was an incoherent set of sentences containing information on concepts, causal relations, attribute values etc written out directly as it was in the text. Then the sentences were further analyzed. The purpose of the analysis was to identify concepts and attributes of the domain, build a hierarchy of these to clarify their connections, and get a collection of rules and facts as complete as possible about the concepts.

Work started by rewriting the rules. The ultimate goal was to get a collection of rules and facts in note form, where every note is short - preferably only one sentence - and contains one piece of information. This means that the rewriting implies: Splitting sentences with more than one unit of knowledge, for instance the sentence: '*Hoeing has a bad effect when conditions are moist or when weeds are big*' should be split in a 'moist conditions' rule and a 'big weeds' rule. Discovering sentences where there are hidden inferences - and making these explicit, for instance '*wet soil makes the crop a bad competitor*' - which probably contains many intermediate inferences about the effect of wet soil on the parts of the crop plants, which again effects the competitive ability. Each sentence should be made as short as possible. The goal was not obtained by just one rewriting, several rewritings took place

before the sentences had the desired form of short notes (appendix 2).

During the rewriting concepts and attributes were identified. These elements can be identified from the logical form of the sentences. A grammatical analysis of a sentence will reveal for instance an attribute being related to a concept by a 'for' as in *'Dry matter minimum* for *couch grass is 3-4 leaves'*. For a manual analysis, grammatical analysis is mostly unnecessary as these relationships are intuitively seen when reading the text.

The identified concepts were then considered candidates for inclusion in a concept hierarchy. The upper part of this hierarchy is general and domain independent (fig 3.1). The top concept is *Everything* which embraces all other concepts, ie every concept is a subconcept of *Everything*. The concepts in this top concept or concept class are divided in the concept classes *Attributes, Objects, Situations, Locations,* and *Times*. These concept classes again embrace concepts: *Objects* include *Animals, Plants* and *Things* and so on.

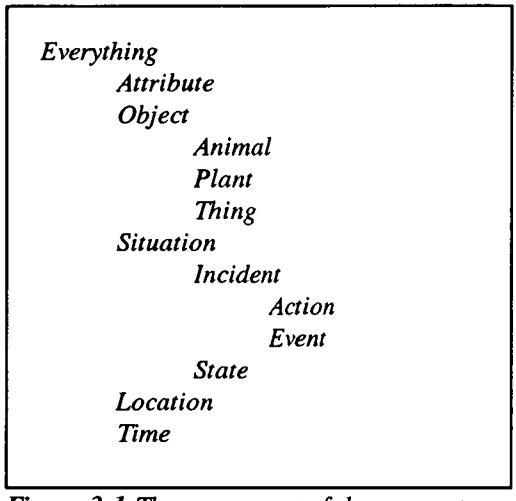

Figure 3.1 *The upper part of the concepts hierarchy which can be used in all sorts of domains (From Østerby 1990).*

Definitional notes help to build the hierarchy by building on the upper part. For instance the notes *'weeds are plants'* and *'Mayflower is a weed'* would include *weed* as a subconcept of *plant* and *mayflower* as a subconcept of *weed* in the hierarchy. Attributes have a special entry in the hierarchy and are ordered according to the concepts they relate to.

The notes were finally gathered in entries - all notes relating to one concept were collected in one entry. Notes concerning several concepts were placed in all the appropriate entries.

The full analysis included the two chapters from the weed control book, totally 28 pages giving an introduction to weeds in agriculture and non-chemical control of weeds. Analysis on the last of the selected papers - the examination treatise - was started, but was considered not to give any further information. The original text was rewritten in approximately 240 notes containing around 40 different objects and attributes. They were grouped in 29 groups or object entries (appendix 2). From the notes the objects and attributes were marked and placed in the concept hierarchy at the appropriate place (fig. 3.2) (appendix 3).

The collection of notes together with the concept hierarchy can be seen as a knowledge model for the domain. It defines the concepts, facts about the concepts and relations between them.

The model was however not complete. And the different parts of the model had different completeness, because some aspects of the subject

were treated more carefully in the text than others.

The texts were intended to give an introduction to weed and weed control, and covered the subject in a general way. The text provided examples on values for attributes as for instance dry matter minimum, but the lists of attribute values were not complete. The text also lacked relations to fully explain the dynamics in the system, and the relations between system components. Some of this knowledge would probably have been included in a book intended to teach the subject if such a book had been available.

On the other hand the domain concepts derived from the text, and the concept hierarchy were very complete and useful. For an expert system to be based on the analysis, it seemed that the concept hierarchy was immediately useful but it would be necessary to complete the models with information from the experts. The information needed was of several kinds. First of all the texts had no description of problem solving strategies. These had to be provided by the expert. Secondly the expert had to fill in the holes from the texts, such as possible values for attributes, and provide heuristics, before a complete system could be made.

The time used to make this literature analysis is hard to calculate, because it was made over a long period alternating with other activities. An estimate for a similar, rather narrow domain and a knowledge engineer with earlier experience in the technique would be a time consumption of 1-2 months.

Everything
- *Attribute*
  - *Plant attribute (12)*
  - *Crop attributes (2)*
  - *Seed attribute (4)*
  - *Harvested crop attributes (1)*
  - *Soil attributes (5)*
  - *Sowing attributes (5)*
  - *Climate attributes (2)*
- *Object*
  - *Animal (0)*
  - *Plant*
    - *Weed*
    - *Crop*
    - *Seed*
    - *Vegetative reproduction organs*
  - *Thing*
    - *Soil*
    - *Harvested crop*
    - *Reserves*
- *Situation*
  - *Incident*
    - *Action*
      - *Sowing*
      - *Soil preparation (4)*
      - *Weed control (5)*
      - *Crop rotation*
    - *Event*
      - *Climate*
  - *State*
- *Location*
- *Time*

***Figure 3.2** Concept hierarchy of concept classes for the prototype domain. The numbers are count of subclasses in the class (complete version in appendix 3).*

3.4.2 Knowledge elicitation

The next step in the system construction was to involve the expert. Normally there are only one expert involved in the construction of expert systems. Some occurrences have been described where two or several experts have cooperated (Huber et al 1990). In this project

two experts were present in all the interviews. One of them, originally assigned to the construction, was conducting research in methods for control of seed propagated weeds -primarily harrowing and hoeing. The other - a coworker of his - was researcher with expertise in control of root propagated weeds, and interested in cooperating in the construction. In the start the partner was only listening and contributed by discussing the knowledge, later he contributed with knowledge on his field.

The literature analysis had provided a background of a structured representation of concepts from the domain, and some rules about these. The usual initial series of unstructured interviews to get acquainted with the domain was therefore considered unnecessary. The important topic to start with was decision of the goal of the prototype and specification of the problem solving strategy. Structured interviews were selected as appropriate for eliciting this type of knowledge.

Interviews

In the following the interviews are treated one after another with a description of the goals for the interview and of the results.

In the first interview the goal was to define the domain and the purpose of the system and to establish the problem solving strategies used by the experts. The first two were achieved by defining the system boundaries asking about crop rotations, data types and values relevant to the system and by presenting the possibilities, and discussing with the experts the kind of system they would like to build. For the problem solving strategy the expert was asked to describe the strategy normally used, information always looked for and information only sought in special cases, to describe some concrete advice sessions and to make a decision table which defined conditions and actions. These questions throw light on the question from different sides.

However there were too many questions for one interview and the question about concrete advice sessions, and the decision table were put out.

The selection of the domain was easily done. Weed control in organic farming was selected because both the experts works in that area.

The experts were asked to make a survey of data types and values for input and advice in the domain. The literature analysis had provided the concepts of the domain, but the analysis did not include lists of variable values. The survey listed the relevant values of crops, weeds, soils etc and in that way defined the domain to work in. The list of concepts made by the experts could also be compared with the concepts list from the literature analysis, to check for missing concepts.

The purpose of the system caused more discussion. The researchers intention was to create a system to help growers better manage weed control. There are two ways to do this: Create a system to deal with acute problems during the period of growth, or a system to help planning control actions.

A planning system is possible because most experienced growers have expectations on what weed problems they will experience in certain fields and certain crops. This sort of system can include preventive actions to reduce the weed problem. A diagnosis system for acute problems on the other hand must rely primarily on mechanical control.

One of the ideas the researchers had beforehand about the purpose of the system was to help growers become more adept at preventing

weed problems by better growing practices, crop rotations etc. They felt that many of the serious weed problems stemmed from problems of bad planning. On the other hand the problems in most consultations with growers were acute ones. The discussion ended up with the decision to start the development on the planning system, and if there was enough time to connect a diagnosis system later.

It is generally known that people seldom recognizes the problem solving strategies they are using (Hayes-Roth et al 1983). The hardest and most abstract of the questions for the first interview was the strategy normally used. It took the experts well over an hour and later some changes to outline the strategy they used in advice situations - or more accurately, would be using in planning advice situations.

The experts strategy showed that when solving a problem they considered different sources of weed problems, ie different problem classes in two levels (fig. 3.3).

Two problem classes were found on the top level:

- control in cleaning crops, which are crops in the crop rotation intended to reduce the weed seed content of the soil by an intensive control of the weeds. In this group the expert was of the opinion that the only relevant control action would be mechanical (direct control) methods,
- control in other crops.

The difference between these two classes is in effect a question of: Only treating mechanically irrespective of the size of the weed population.

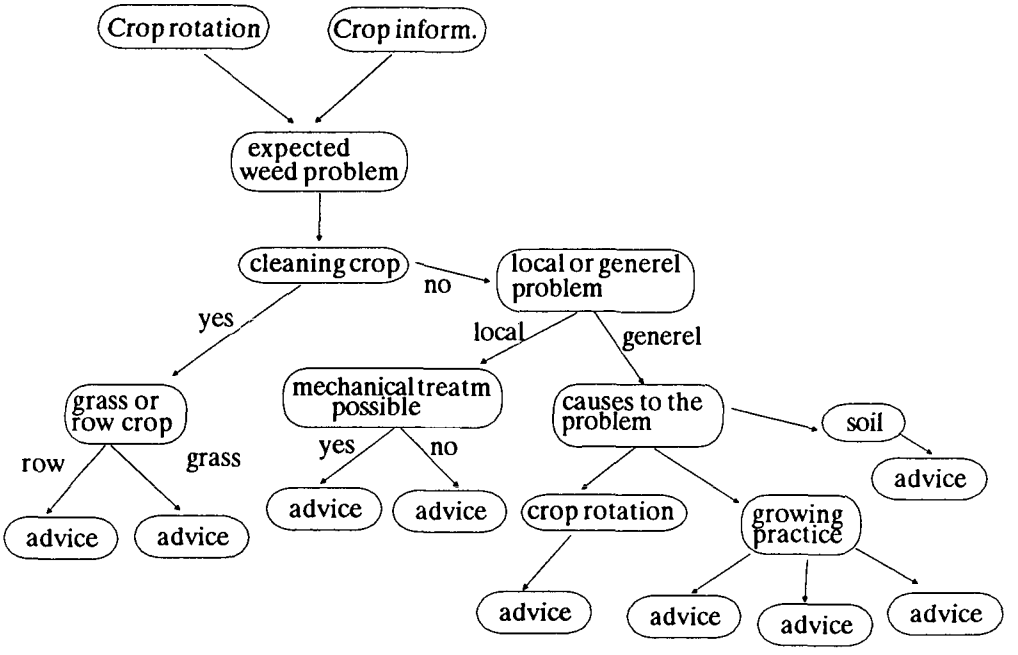

Figure 3.3 Sketch of the first problem solving strategy from the expert.

Or considering other control means and only treating when damage is above a threshold.

In the latter group there were again problem classes:

- crop rotation problems giving occasion for advice to change the crop rotation,
- growing practice problems which give rise to advice on for instance better sowing bed preparation,
- normal weed problems causing direct control advices
- soil problems, which could be for instance drainage problems.

A given weed composition problem could belong to all of the classes in this latter classification, so each of them could contribute to the advice given. A weed problem which is in part caused by an inappropriate crop rotation can still be treated by direct control, so the advice given will include two parts: ways to control the weeds mechanically and proposed corrections in the crop rotation.

In the second interview the requirements to the final system was discussed so was the decision table from the first interview, and the problem solving strategy.

In the discussion of the requirements, the decision was between two proposals which also had an effect on the problem solving strategy.

- The system could be asked to restrict the search for solutions to either preventive or direct control means. In the case of direct control only the best method should be output.
- The system gives all possible solutions with an indication of the effect for instance as percent effect on weeds or yield.

The experts favoured a system of the second type, they considered that the success of a weed control program depended on giving the growers a possibility to chose between methods but giving them a tool to make a better choice. The choice of this system made a selection of methods based on the users access to the necessary machines unnecessary.

The problem solving strategy was revised (fig 3.4). The top level of the problem classification was removed, because the experts revised the opinion on the cleaning crops, making preventive methods relevant here too. The revised strategy thereby became very simple. After initial information is collected, the different classes of problems are examined. The order of examination of the problem classes is irrelevant as none of them influence the results of the others.

The decision tables was on the agenda again. The decision tables are a survey of conditions or state descriptions, and the following appropriate actions. The conditions could be weed population, crop, soil type, crop rotation, control method etc describing the state of the biological system. The actions are the control methods proposed to reduce the weed problem. At the interview, and before the next interview, seven situation were written. These were only a very small fraction of the possible combinations of conditions, and the method was given up as a way of extracting a strategy by specifying causes and advices. It gave the knowledge engineer examples of written advice, and the discussion about them gave new information. The experts were also more comfortable with this more example-based discussion instead of more abstract talk about strategies, and felt that the system construction was really in progress now.

The third interview also tended to be more practical. The first prototype could show a possible interface but little else and was dis-

cussed. From the literature the concept hierarchy was ready, and the first interviews had given surveys of relevant values for concepts. The connection between the conditions and the advice still had to be defined. The earlier decision that all advice should contain a measure of the effect of treatments required that a method to calculate this should be found. Finally the strategy for implementing the system parts was discussed.

In the first place the conditions for considering problem classes were discussed. It showed up that the three special problem types crop rotation problems, growing practice problems and soil problems could be indicated by specific weeds in the weed population. The result of this is that the classes direct control in seed - and root propagated weeds are always relevant, but the rest are only considered if there are weeds in the populations which indicates these kinds of problems.

For the measure of effect the experts proposed to use CE (crop equivalents). A CE value is the count of crop plant one weed plant can oust. The count of weeds was earlier decided to be part of the initial questions asked in the system. For every weed species the experts estimated one value for CE for spring crops and one for winter crops. By multiplying the CE value for each weed by the count, and summing for all the weeds, the total CE value can be calculated (3.1)

$$CE_{total} = \sum_{i=1}^{n} count_i \times CE_i \qquad (3.1)$$

This value has to be adjusted by a factor according to the specific crop (appendix 4). The general competition ability for the crops differs depending on for instance growth patterns. The reduction in yield can now be calculated by the formula 3.2

$$Percent\ reduction\ in\ yield = \frac{100 \times CE_{total}}{CE_{total} + CP} \qquad (3.2)$$

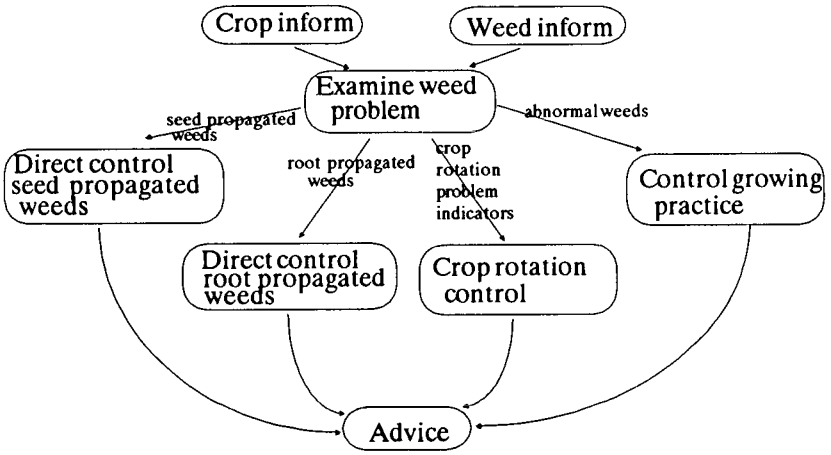

Figure 3.4 Final problem solving strategy. The problem is decomposed to smaller problems, which are solved separately.

Where *CP* is the crop population pr m^2. As an approximation the crop population has been set to 400. This method of using CE to calculate the reduction in yield for a given weed population has been used earlier, for instance in a herbicide selection system for winter wheat (Cussans & Rolph 1990). The underlying model is a Michaelis-Mentzen curve. With small weed populations the curves are steeper than with large populations. With several species the effect is not additive.

The model is valid only at moderate weed levels. Use of the crop equivalent system relies on some assumptions which are very crude. The system assumes:

- that the total biomass of a culture is constant whether it is a clean culture or a mixture of weeds and crop,
- that the harvest index is unchanged by weed competition,
- in the formula used in this study a constant crop population of $400/m^2$ is assumed.

These assumptions can all be questioned. Weed competition from some species is probably not by replacement. Total biomass yield varies between clean and weedy crops and so does the harvest index. Additionally the background for the CE values is not too well established for all weeds. At the moment, however, this model is the best that can be achieved, but it is a part where a better system can be derived by research in a new model (chapter 4).

The last decision in this interview was the strategy for the implementation. The dividing into different sorts of problem classes made it natural to work on one class at a time. So it was decided to start with the direct control of seed propagated weeds and then proceed with crop rotation problems.

From now on the knowledge needed was specific information regarding the part of the system in the present study. This made it possible to clear many questions via phone calls, and for the experts to work on specific parts at home.

The next interview was concentrated on the direct control problems of seed propagated weeds. The expert on seed propagated weeds had the CE values ready. The missing part of this problem class was to estimate effects of different control methods on the weed population and ultimately on the yield. The expert was able to construct the frames for the two methods harrowing and hoeing, a third method - flame treatment - needed consultancy with another researcher and was postponed. The expert now had to gather information by examining trials to estimate the effect of the two treatments on the weeds of importance chosen earlier.

Two more interviews were carried out with longer intervals. In the mean time the communication was by mail and telephone. The prototype was sent to the experts at intervals, and was discussed by telephone. Adjustments to the prototypes were proposed in the interviews also. Because the prototype system became running, the experts really showed excitement and eagerness to complete the system.

The two modules for preventive control, the crop rotation module and growing practice module were made as simple as possible as a start. These problem classes are only considered if the weed flora indicates problems, and the modules are called only when indicative weed species are present. The crop rotation part checks the crop composition in the rotation compared to heuristic rules about the procentual amount of crop types that are

allowed in a rotation. The growing practice module so far only contains a warning of problems, based on the indications of certain weed species.

The last module commenced in this study was the direct control of root propagated weeds module. The control of these weeds is different from control of the seed-propagated weeds in the methods used. They have to be controlled outside the growth period by preventive precautions. By treating the soil regularly in the autumn, when the crops have been harvested, the growers try to diminish the population. The soil treatment kills the weeds by cutting the leaves and roots in bits and burying them underground. The more often the soil is treated, the better the effect on the weeds.

As this module is implemented now, it is actually a cross between planning and diagnosing . The reason for this is that the expert in root propagated weeds was so used to give advice in acute situations, that it was hard for him to abstract from this. The result is a planning module, which needs fresh counts of weeds instead of expected counts as in the pure planning part. If the project is to be continued this module will have to be reviewed.

Now the module for direct control of seed-propagated weeds, for crop rotation and part of the direct control of root propagated weeds was finished. The prototype was in a phase where there was little new to learn of the methods, and the hard work was in the programming part. The work in this project had to continue in another direction. This left the cooperating researchers with a prototype and a lot of good ideas for proceeding the work and visions for the system.

3.5 The prototype, implementation

During the knowledge acquisition phase the concepts and rules of the domain have been collected. This phase is followed by a phase of formalization of the knowledge. The final representation of the knowledge should reveal the patterns in the domain in a way consistent with the chosen language.

In knowledge system building the tool is normally chosen, when the domain choice and some initial knowledge acquisition has been carried through, establishing the scope and goal for the system. In this project the tool - EGERIA - was chosen before the knowledge acquisition took place. This was an early choice based on cooperation and funding considerations, making it impossible to take special domain and system requirements into consideration. Instead a hybrid shell was selected with possibilities to produce several different sorts of system. The shell has several forms of knowledge representation, and is able to do both backward and forward chaining.

By choosing EGERIA the notation for the language to represent the knowledge, the semantics, the procedural and declarative schemes, the mechanisms for organizing knowledge and the inference schemes were determined. Selecting a shell also means favouring the paradigm the shell is built upon. A knowledge engineering tool reflects an AI viewpoint and a specific methodology for building knowledge systems and may for instance have a built in preference for building systems with causal models as the ground for solutions to diagnostic problems or conversely reflect preference for using experts empirical symptom-problem associations. This does not mean it is impossible to use them differently, only that this can create problems. In the formalizing process the early choice of a shell means that the facilities in the shell can

be the foundation of the representational schemes used.

3.5.1 Representation

One of the results of the literature analysis was the concept hierarchy. There are several ways this structure could be represented. Semantic networks or frames are two of the obvious choices.

In their program for automatic construction of small knowledge bases from texts Gomez and Segami (1990) used semantic nets to represent the knowledge.

EGERIA supports object oriented programming (chapter 3.3.1), where concepts are represented as objects and object classes which are frame like structures. Therefore an object oriented approach was chosen.

Classes describe concepts, their connected attributes and procedures for deriving values for attributes. For every concept in the hierarchy an object class is constructed. As a start the class contains the attributes in the attributes entry of the concept hierarchy which are connected to the concept - the crop concept, for instance, contains the crop attributes.

The hierarchical concept structure is represented in a corresponding hierarchical class structure. Subclasses inherit properties, procedures and rules from super classes ie the subclasses are specializations of the superclasses. This corresponds to the relation 'is-a' for a semantic network described in section 2.4.3. For instance the concept plant could be represented by this class (in pseudo language):

```
CLASS plant
   single speciestypes species
   single lifelengthtypes lifelength
   single propagationtypes propagation
   integer size
   real CE
   integer dry_matter_minimum
END CLASS plant
```

Where speciestypes, lifelenghttypes and propagationtypes are lists of possible string values for the corresponding variable. Single means that only a single value can be chosen for the attribute value.

The crop concept could then be represented by this

```
CLASS crop
   inherits plant
   string sort
   single usagetypes used_as
   single methodtypes method_of_cultivation
END CLASS crop
```

The 'crop' class inherits the attributes from the 'plant' class (and everything else in the class). Besides it has the attributes *sort*, *used_as* and *method_of_cultivation*.

Besides these attributes the classes contain rules and procedures to calculate the value of attributes. For instance the class 'crop' contains rules to find out if the crop is a winter crop:

```
CONDITION wintercrop IS species IN (winter-
rye, winterwheat, winterbarley)
```

These examples are based on the material from the literature analysis. The knowledge elicited in the analysis contained partly knowledge on control of weeds and partly knowledge about the biological relations in a field. The last part could be used to form a model of the biological system as a basis for a model based expert system - if it was complete. Unfortunately it was not and the system was therefore built as a heuristic expert system. Later work commenced on a model which could be the kernel of a model based system (chapter 4).

The concepts from the analysis have been included in the system when needed. Some of the concepts were not needed, or have not yet been used. There are several reasons for this

- First, the concept can be irrelevant when talking about weed control, as for instance the climate. It has influence on the growth of weeds and crop, and on the effect of some of the control methods. But as the climate cannot be controlled, the effect of it is included implicit as part of a variance on the effects of control methods.
- Secondly, some concepts are not considered when choosing control methods for a weed population. For instance weeds are always controlled when emerged, no control method is directly aimed at killing seeds. The seeds are important in the biological system, but are only considered implicit in weed control. For instance by trying to prevent weeds from producing seeds.
- Finally, some concepts have not been included because the system is not finished yet. Examples are sowing and soil preparation. Using suitable methods here has a preventive effect on weeds. Unsuitable methods may create weed problems in crops, where those weeds are normally of minimal importance.

The concepts included in the prototype are now weed, crop, soil, weed control, split in the two types for seed-propagated and root-propagated weeds, crop rotation and abnormal occurring weeds (fig. 3.5).

Some of the attributes from the analysis are not relevant in the system. For instance methods of cultivation are normally of relevance in the choice of control methods - row cleaning is only possible in row cultures. In the system it is irrelevant because in a planning situation the cultivation method can be changed if the change gives opportunity to use better control methods.

The interviews also showed that attributes and concepts may be irrelevant for other reasons. The information needed for crop and weeds in a system for control is different subparts of the plant attributes. For instance all plants have a dry matter minimum - the stage of growth where the dry matter content is at a minimum. When we are talking about weed control, the attribute is only used in the control of root propagated weeds on bare land. They should be cut at the dry matter minimum, when there are minimum reserves to start new growth. The attribute are of no relevance in the crop.

Ultimately all the attributes for a plant could be divided in three parts: Those irrelevant, those relevant for crops and those relevant for weeds. The hierarchical structure could have been retained, because it provided an excellent connection to the natural model of the domain. For efficiency reasons the plant class was removed and the attributes removed or moved to the relevant subclass.

3.5.2 Inference and control

Because a shell was chosen for the development of the prototype the inference engine and the way of inferring new knowledge was given beforehand. The course of the session could be controlled by active elements in the program. The overall control in WEEDOF is performed by two classes - a main and a dialogue class. Instantiations - objects - of these classes perform the control necessary to reach the goals of the system. The main object successively starts a session, by creating a dialoque object, and clears all values from memory until the user chooses to stop. The dialogue object creates the relevant domain objects in a fixed sequence.

The basic feature in EGERIA is forward chaining, and this is also true for WEEDOF. The goal is to produce a list of possible plans for weed control from many different combinations and this is done in a rather procedural way. In parts of the program backward chaining has been used especially in collecting values for attributes in the objects. There are two reasons to use backward chaining here. Firstly, it is often easier to assign values by

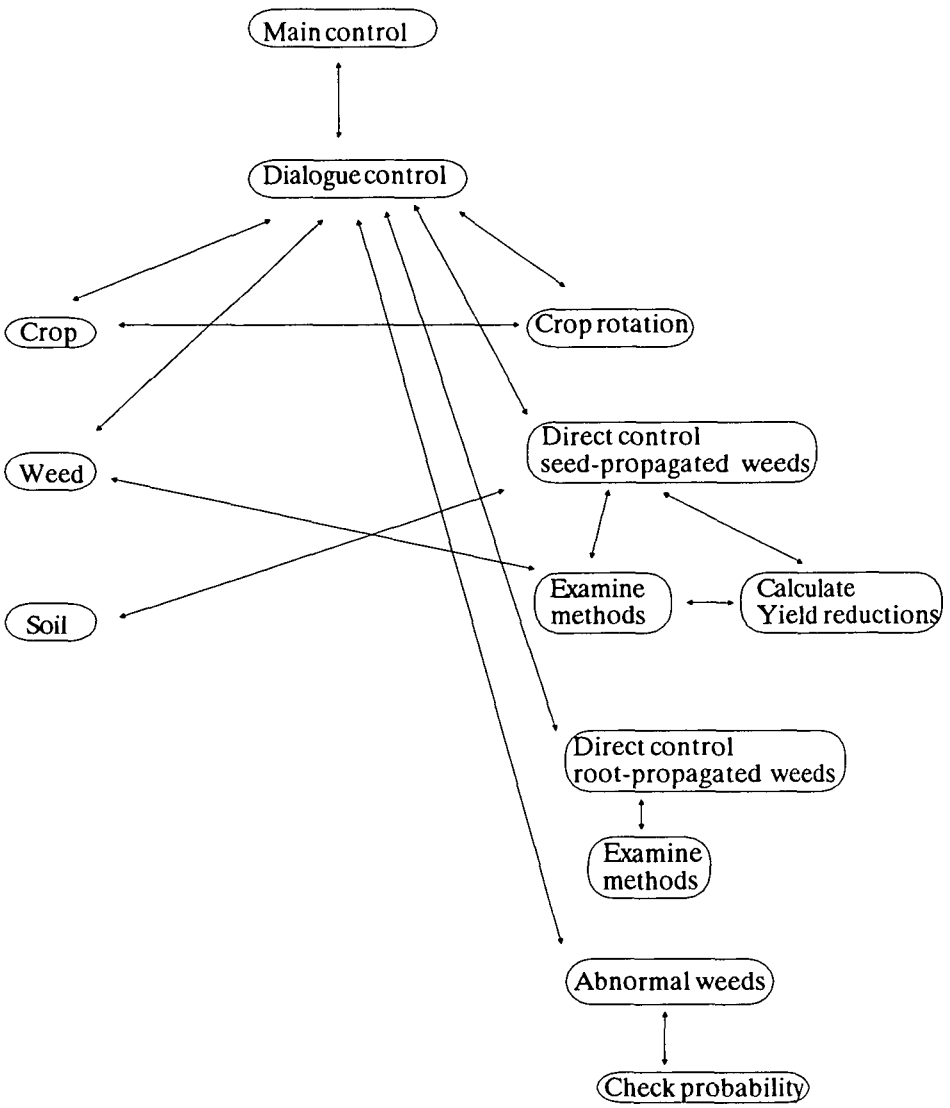

Figure 3.5 Objects in WEEDOF and their relations.

telling the variable where a value should be used and leave it to the system to find the variable values which are prerequisites. Secondly for explain reasons - refer to the section 3.5.3.

In the interviews a decision was taken that all possible solutions to problems should be given. So the inference engine has to search for all solutions not only the best. This means that solutions do not have to be compared for effectivity.

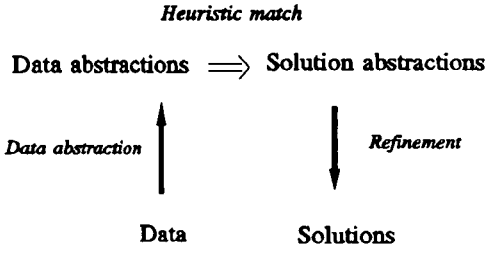

Figure 3.6 Heuristic classification from Clancey 1985.

The problem solving follows the problem solving strategy from the expert. After collecting the initial information - the data - the problem is classified in one of the problem categories. This is parallel to Clanceys (1985) data abstraction in heuristic classification (fig. 3.6). These categories corresponds to different control possibilities which is then explored further. This corresponding is equivalent to Clanceys heuristic match. So the problem solving being used can be referred to as heuristic classification. The match deviates in the direct control classes from Clancey's match in that the match directly gives the possible solutions and the remaining task is to calculate a measure of the effectivity of the control actions.

3.5.3 Explanations

An important part of an expert system is the explain facility. EGERIA has a built in explain facility. The WHY statement retraces the most current line of backward chaining and executes any EXPLAIN clauses attached to variables on that path. Of course this facility should be used in explaining the results in WEEDOF. This proved to be of very limited use. As mentioned earlier the basic overall feature in WEEDOF is forward chaining. In parts backward chaining has been used for assigning attribute values. It quickly showed up that most lines of backward reasoning in WEEDOF were only one step and then the explanations goes also one step back. Without changing the structure the reasoning was changed to backward chaining where ever it was possible. This changed very little in the performance of the explanations - there are too many shifts to forward chaining for the explain mechanism of EGERIA to be of any use in a program like WEEDOF.

3.6 From prototype to final system

WEEDOF has not been finished. It is still in a state of prototype. In virtually all parts of the program knowledge is missing.

The module for direct control of seed propagated weeds are the one which have been elaborated the most. It is containing knowledge on the subjects hoeing and harrowing. Where as knowledge acquisition and formalization on the field of flame treatment initially was postponed and never was done. This type of direct control of course should be included.

In the module for root propagated weeds only knowledge about couch grass (Elymus repens) has been included in the present prototype, of course the rest of the root propagated weeds treated in the prototype should be included.

The module furthermore departs from the rest in showing more of a diagnostic character than of a planning one. Maybe this is indeed the only way to handle the problems of root propagated weeds. The problem has to be reviewed if the prototype shall be finished. A total finished system will demand a coherent presentation.

The 'abnormal weeds' module is the part in the most preliminary state. For the moment it only warns about problems if there are weeds that 'should not be there'. The intension with this module is to include reasoning about growing practice problems. The knowledge has not been acquisited and formalized in the project. The field includes several reasons for problems for instance sowing bed preparation or missing cleaning in row crops, so the acquisition of knowledge for this module could take time.

The last 'problem solving' module - crop rotation - could need a more thorough analysis of the crop rotation. For the moment it resides on heuristic rules saying that if certain weeds are present certain kinds of crop rotation problems could be the reason. This of course only covers the experiences when people stick to known crop rotations. If growers departs greatly from these, for instance if they start to grow a single crop for a big part of the rotation, this would demand for the system to possess deep knowledge to deduce the reasons.

These were all extensions needed in the modules. To finish the system some work also have to be put into a more user-friendly interface.

All of the modules uses a reasoning based on heuristics. As will be mentioned in chapter 4 it could be possible to improve on the explanations by using a model based reasoning. This of course is a totally new concept which means throwing the prototype out and beginning a new.

One appropriate way of finishing the prototype system would be to leave the prototype to professional programmers in a company with experience of decision aid systems, and let the programmers finish the knowledge acquisition, reprogram and finish the system. Now that the basis is ready this could be done in an ordinary programming language.

Before the system can be considered finished a test is necessary. So far the prototype has only been tested by the knowledge engineer and the experts. A thorough test will imply testing the system on real application problems at users, as well as testing it on other experts.

3.7 Summary and conclusion

For the development in this project an expert system shell - EGERIA - was chosen, partly to experience such a tool, partly to speed up the development. EGERIA has several weaknesses. As an expert system building tool EGERIA should support generation of explanations. EGERIA's explain system only works in systems which uses backward chaining all the time. Systems with forward chaining or shifts between backward and forward chaining cannot use the automatic generated explanations. Furthermore EGERIA has deficiencies in the compiler, it lacks a trace facility to use in the debugging, and the language is complex with a difficult syntax. On the other hand the tool is very sophisticated and also provide features which speeds up programming, for instance easy windows' definition, easy way of initiating backward and forward chaining and advanced ways of representing knowledge.

The start on the knowledge acquisition process in this project differed from other expert system development projects. The use of written sources in a systematic way had not been common earlier. Now researchers have realized that a preliminary knowledge analysis and domain characterization will facilitate the development of knowledge bases (Nwana et al 1991). Texts could be one knowledge source to this domain characterization.

Gomez and Segami (1990) describe a program with a model for comprehending scientific texts used for automatic construction of small knowledge bases from text. This model also uses the idea of building concept hierarchies and concept structures from the logical form of the sentences, using a parser to parse the sentences identifying concepts and relations. Their model apparently goes one step further than this approach and represents the concept graphs directly in the knowledge base.

The use of written material as an initial source of knowledge proved to be a good start on the knowledge acquisition process. In this project the knowledge engineer had superficial domain expertise. It seems though that domain expertise is not necessary to make the literature analysis: In a text from a domain anyone is capable of recognizing special uncommon words likely to be part of and characteristic for the domain, ie the first central part of the literature analysis. The rest of the analysis is a way of reformulating the knowledge in a short and concise manner, and extracting the concepts in the domain, making the concept hierarchy.

The analysis provided a concept hierarchy which showed up to be very complete and useful in the end. It also provided knowledge on relations in the domain, in the form of notes. The collection of notes can be viewed as models of the domain. The hope was originally to include these models directly in the expert system and to use these to simulate effects of actions on the state of the biological system, using these simulations to give advice on the best actions to take. The models derived were unfortunately not complete enough to be directly included in the prototype. The relations on the effects of control methods on the biological system needed completion. Instead a basis was made for an easier perception for the knowledge engineer of the knowledge delivered from the experts in the following phase of interviewing.

Two factors seem to be of great importance in the outcome of the literature analysis:

- The type of the domain with respect to the kind of knowledge about it. Domains with a deep and thorough understanding of the processes happening to the concepts are much easier to describe, and will give better and more complete models in the analysis, than domains where knowledge about causal relations is poor. So very technical domains will be described to a better degree, and may leave the knowledge needed from the expert to be search strategies.
- The quality of the textbook available. If the text gives a thorough reading of the domain, the results are likely to be much better and more detailed than a text which is an introduction to the domain as in this case.

Even when the results are less optimal the analysis can be a great help in providing the concept hierarchy and an understanding of the domain ahead of interviews.

Another part where the procedures in this knowledge acquisition process differed was in the use of two experts. The reason for having two experts to cooperate in the work was in this case simple interest from the researchers

side. This interest of course also made the work easier, both were very motivated to do a good job and create a system. Experts are able to question each other much more thoroughly than a knowledge engineer would be able to, thus freeing him to listen to the discussions and to control the interview.

The literature analysis has made it possible to structure the interviews providing a systematic way of constructing a knowledge based system in a top-down way, specifying the problem solving strategies early in the knowledge acquisition phase. The analysis has also reduced the time needed from the experts.

The concept hierarchy from the literature analysis can be represented using different structures. As EGERIA - with an object oriented representation - was chosen as the programming tool, objects were chosen to represent the concepts. Many of the concepts from the literature analysis has not been represented in the prototype. Concepts have been removed either because they were unnecessary or because the attributes could be moved to make the prototype more efficient. Consequently the prototype class hierarchy is flat compared to the whole hierarchy from the analysis.

The programming started by representing some of the immediately useful concepts in the language, this was the weed and crop concepts, and the associated methods for collecting initial informations. Also the overall session control was programmed early. Early in the interviews it was decided to start the development with the direct control of seed-propagated weeds, and that the development would proceed with the problem classes direct control of root-propagated weeds, crop rotation problems and abnormal occurring weeds. Because all possible solutions should be given there was no need to compare solutions. The program executes all modules which are relevant with a set of initial information of weeds and crop.

The prototype follow a problem solving strategy - heuristic classification - described by Clancey (1985). The initial information is matched to four problem categories. These categories corresponds to the different control possibilities which are then explored further. Eventually each of the control possibilities can contribute with advice, if the initial information matches to all the problem categories.

The program is still a prototype. Several things are missing before it can be called a finished system. Some of these are knowledge parts, for instance knowledge about flame treatment, others are system parts of which one important from an expert system point of view is the generation of explanations, another is a good user interface. One way the system could be finished is to leave the prototype to experienced programmers who could reprogram and finish the system - maybe in an ordinary programming language.

4 A model based system

In connection with expert systems one distinguishes between deep and surface knowledge.

Deep knowledge makes explicit the models of a domain and the inference calculus that operates over these models. A domain model for diagnosis could be a causal model linking properties of components through cause-effect relations. Surface knowledge contains selected portions of the deep knowledge, in particular those portions that are relevant for the class of problems that are likely to be encountered. It also contains additional heuristics and optimizations, for example decisions based on the most probable situation.

Traditional (first generation) expert systems only code the surface knowledge. They contain just enough knowledge to make the required inferences, but none of the underlying domain knowledge (deep knowledge), such as causal relations between symptoms and causes. The fact that only surface knowledge is represented explains why traditional expert systems are efficient and effective in problem solving. However, because first-generation expert systems only code surface knowledge, they have important drawbacks, such as brittleness and weak explanations. Model based expert systems or second generation systems intend to overcome these limitations by including deep knowledge.

Model based expert systems contains two components: One implements the deep knowledge of the domain, that is, the domain model, the other implements the surface knowledge (Jones et al 1989). Although deep problem solving is, in principle, less efficient, it typically covers a wider class of problem. Model based expert systems are therefore less brittle.

Because deep knowledge is supposed to be less biased towards use, it is hoped that knowledge becomes, to some extent, reusable. For example the same causal network can be used in both design and diagnosis.

Deep knowledge can be the source of better explanations. The explanations given by first generation expert systems are somewhat unsatisfactory because they are a simple replay of the rules that are used to arrive at a conclusion. With deep knowledge in the system, the domain knowledge that went into an inference step can be reported making richer explanations and justifications.

The prototype expert system constructed in this project can be characterized as a first generation expert system. When the literature analysis started it was hoped that the deep knowledge elicited could be included in the knowledge base. Unfortunately knowledge was missing in the models from the literature analysis. When the work on the prototype stopped it was decided to work on a domain model.

The model should be able to simulate growth of crop and weeds on a field and the effect of control actions. This model could be the basis of a new prototype where the model is used to simulate actions and predict effects, short-term as well as long-term, on the development of weeds and crop and the seed content in the soil.

In this system the model furthermore should be used to generate better causal explanations to the user, by replaying the deep knowledge that was used.

The model is not finished yet. The specification has set the framework of the model, but

the specific functions which represents the subpart of the model are still missing.

This chapter will outline the work done on the model and discuss the possible way of using it in a model based expert system. Section 4.1 shortly discusses existing work on plant population models. Section 4.2 introduces the specification language. The model has been specified using a formal method not previously used for agricultural modelling. The language used is META IV - an edp specification language. Section 4.3 presents a survey of the model structure, while section 4.4 describes the component functions. Section 4.5 presents a possible way of using the model in a planning system for weed control. At last section 4.6 presents the summary and conclusions.

4.1 Plant population models

Work on plant population models is not uncommon among agricultural researchers. Mechanistic models have in particular been used to model all parts of biological systems. Most models narrow the task by concentrating on one species (Doyle et al 1986, Moss 1985, Moss 1990), on parts of population models (Barralis et al 1988, Zwerger & Hurle 1986, Zwerger & Hurle 1989, Moss 1983, Cousens & Moss 1990), others narrow the task by omitting for instance competition and diseases, such as the Dutch model SUCROS87 of the potential growth of a crop in disease and weed free environments under the prevailing weather conditions (Spitters et al 1989), and the Danish model DAISY for simulation of crop production, soil water dynamics, and nitrogen dynamics (Hansen et al 1990).

All these mechanistic models are too limited in scope to be of use in a model based expert system for planning weed control. We need a general plant population model. General in the sense that it should be applicable to all plants. Furthermore the model should take competition between species into account. A model of the kind an expert uses when explaining the effects of different kinds of action on the population. Empirical models often give results closer to reality than mechanistic models. But the use of an empirical model is ruled out because the model is intended to be used in explanations. Empirical models do not contain the causal relations which are needed when explaining results.

The model should simulate the development of the population on a field. The input to the model is the flora, weeds and crop, the counts of plants per squaremeter, the actions performed on the field and the time period. The output is the consequential growth, counts of plants, seed shedding, germination of plants and seed content in the soil at the end of the period. Because the purpose of the model is to predict events, it will be adequate to consider the situation under standard conditions, and not include for instance climatic parameters.

If the goal is a model that gives a good description of reality, such a general model is very difficult to construct. As earlier described one of the primary goals for the modelling for expert systems is to make better explanations of the causal relations. The predictive ability of such mechanistic models does not always live up to expectations, however, a poorer description may be accepted in exchange for a simple, easy to understand model.

A population model can be constructed at different levels. For instance the plant growth can be explained as a simple curve relating to time and time of emergence. Or it can be explained on the basis of the underlying pro-

cesses, such as photosynthesis and respiration, and how these processes are influenced by environmental conditions. Building a model based expert system does not automatically imply choosing just one model, and using this for both simulation and explanations. A system can encompass several models. The models can be series of refinements, where the finest model are the one used for the simulations and the rest are simplifications making it possible to explain on different levels of difficulty.

4.2 Specification language

The model is specified in appendix 5 using META IV (Rischel 1987). META IV is a system specification language, which is used in The Vienna Development Method (VDM). The specification is performed top-down. The system to be specified is described by an overall top function. The inputs and outputs to the top function are specified, and each of the input and output parameters is declared. The overall function is then broken into subfunctions by degradating the top function in sub functions. The degradation continues until easily described functions are obtained. This degradation amounts to dividing the model in submodels each describing a separate part of the total system.

Functions are described by a head declaring input and output domains, and a definition part describing the function algorithmically. It is the definition parts which is lacking in the current work.

4.3 Model structure

The general yearly cyclus of plant growth is in short (fig. 4.1): Seeds in the soil seedbank germinate to small plants. A number of the small plants die. The rest grow to be mature plants. During growth an eventual vegetative

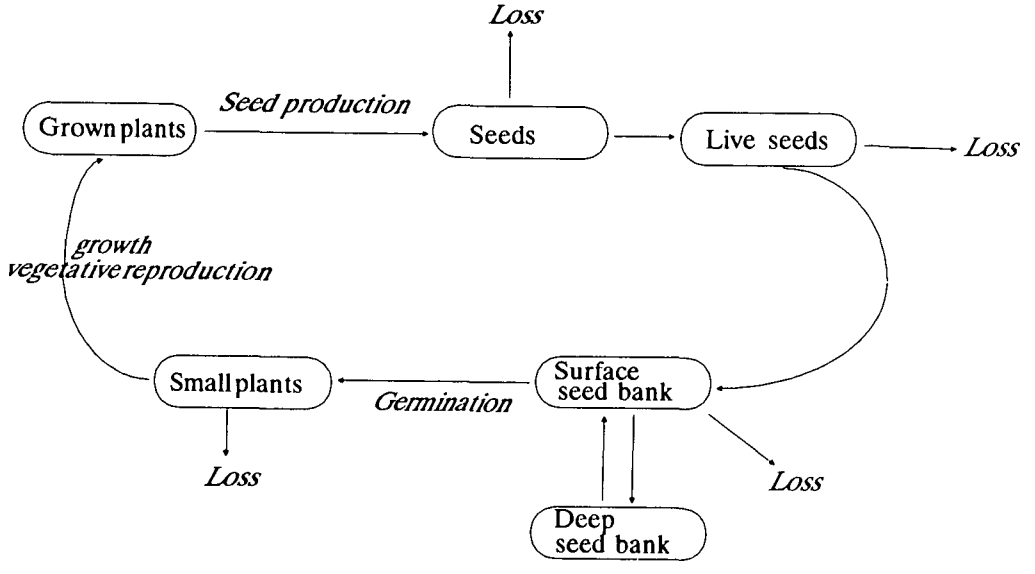

Figure 4.1 Yearly cyclus of a plant.

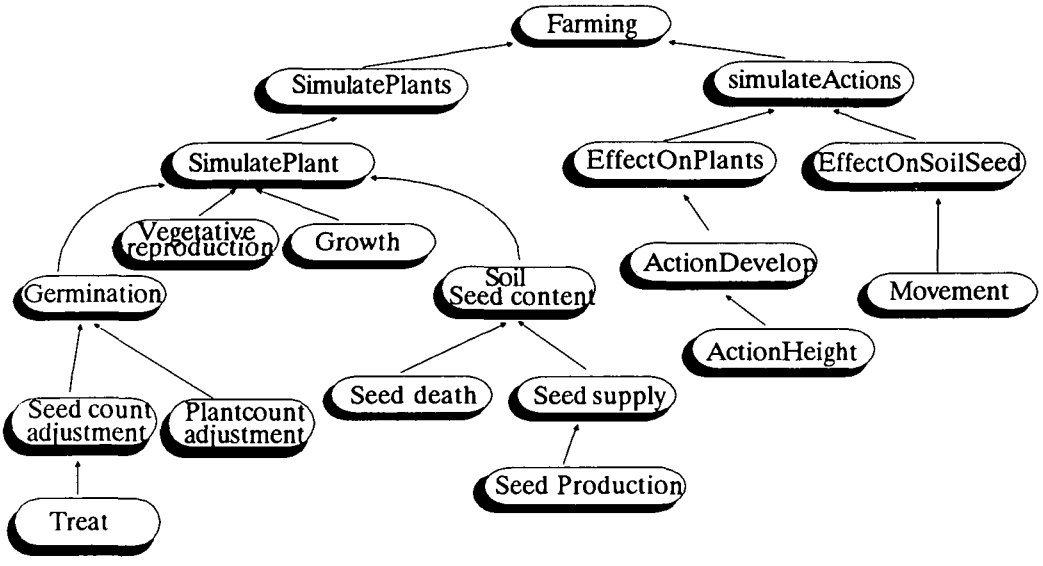

Figure 4.2 Functions in the present model structure. The top function farming has as sub functions the functions simulatePlant and simulateActions and so on.

reproduction also occurs. Competition between plants influences the growth. The mature plants bloom and sets seeds. Of the seeds some die or disappear for different reasons. The rest are incorporated in the soil surface. Seeds in the soil are spread in the soil profile. Only seeds in the top 5 cm can emerge. The rest is dormant. Actions performed on the field such as ploughing or harrowing influence the stages in the system. This is the general system we would like to model.

Fig. 4.2 is a schematic outline of the connections between functions. The boxes indicate functions. The overall top function in the model is called *farming*. *Farming* encompasses the whole system. The arrow indicates that the function at the end of the arrow calls the function at the beginning. *Farming* then calls the functions *simulatePlants* and *simulateActions*. *SimulatePlants* calls one function *simulatePlant* and so on.

4.4 Functions in model

In the META IV specification the functions are first described by the domains for the input and output variables - the valuespaces. Table 4.3 is a list of domains for variables in the functions. Generally all the functions operate on the state of the system. The state is a compound variabletype consisting of a *seed situation* and a *plant situation*. The state or part of the state are changed by the functions to picture the changes during time or caused by actions.

The *plant situation* is specified (table 4.3) as a function of species (*species*) to a series of developmental stages (*population_Development*). *Population_Development* is specified as a function from the developmental stage (*develop*

Table 4.3 Domains (datatypes) for the variables in the model. | = separate possibilities for variable values, × = compound values, → = picture and {} around values = list of values.*

Domain	*Definition*
Year	*INTEGER.*
Month	*INTEGER.*
Day	*INTEGER.*
Time	*year × month × day.*
Height	*INTEGER × 'cm'.*
Depth	*'>5 cm' | '<5 cm'.*
Species	*'Winterrye' | 'Winterbarley' | 'Winterwheat' | 'Rye' | 'Barley' | 'Wheat' | '...' | 'Forget-me-not'| 'Chickweed common'|.....*
Soiltype	*'Hard clay' | 'Clay' | 'Sandy loam' | 'Sand' | 'Humus'.*
Stones	*'Few stones' | 'Many stones'.*
FieldInformation	*soiltype × stones.*
Developmental stage	*'Germ' | 'Flowering' | 'Seed bearing'.*
Count	*INTEGER.*
Germinated plants	*INTEGER.*
Actions	*'Sowing' | 'Harrowing' | 'Hoeing' | 'Flame treatment' | 'Plowing' | 'Stubble treatment' | 'Harvest'.*
Actionlist	*{action × time}*.*
SeedContent	*depth → count.*
Heighttable	*height → count.*
Population_Development	*developmental_Stage → heighttable.*
Seed situation	*species → seedContent.*
Plant situation	*species → population_Development.*
State	*seed situation × plant situation.*

mental_stage) onto *heighttable*, for each developmental stage then there can be several different heights. The domain *heighttable* pictures each height (*height*, which is an *INTEGER* with a 'cm'after) onto a count (*count - INTEGER*). This count is then the count of plants of the actual height of the actual developmental stage of the species. For some species this seems very complicated. The crop, for instance, will probably be uniform, all the plants having the same developmental stage and height. Some of the weeds though will germinate for longer periods and can therefore be present in different sizes and developmental stages. *Height* from the table *heighttable* could be given in larger intervals than 1 cm for instance the heights could be given in 0-5 cm, 5-10 cm and so on.

Seed situation is a function of *species* to the seed content table (*seedContent*). *SeedContent*-pictures depth (*depth*), which is either '<5cm' or '>5 cm' onto count (*count*) of seeds in the depth. The two depths are chosen because the seeds which germinate often come from the top 5 cm.

Now the functions in the model will be examined. For each function the input, output and the intended content in the function will be described. Sometimes the domains for the variables are mentioned, in that case they are written as *domain*, sometimes they are not. In all cases the variables are named according to their domain type - a variable named population_Development are consequently of the type *population_Development*. Often a function has two input variables of the same type, then the variables have additional descriptions to clarify the difference, for example start time and stop time. The domains and the functions are specified in the META IV specification appendix 5.

Farming

Farming is the top function (appendix 5). The inputs are the start state (*state*), information about the field *fieldInformation*, time for start of simulation and a list of planned actions with connected times (*actionlist*). The output is the final state at the end of the simulation period, which is the time of the last action, and the stop time.

Farming performs the simulation of the system. This simulation consists partly of a simulation of the plants and seeds on the field - growth, development, seed shedding, changes in soil seed content in the simulation time, partly of a simulation of the effect of actions.

Farming is specified as a recursive function which calls the subfunctions *simulatePlants* and *simulateActions*. For each time of action in the list, *farming* simulates the development in the field from the last state: First the function *simulatePlants* performs a simulation for each species in the population. Then *simulateActions* simulates the effect of the action. At last *farming* calls itself to simulate the period to the next action.

SimulatePlants

SimulatePlants performs the simulation by calling the function *simulatePlant* for each species in the population.

SimulatePlant

SimulatePlant will call the four functions *germination, vegetativeReproduction, growth,* and *soilSeedContent* in that order. Competition between species must be included in some of these subfunctions.

Germination

Germination has the input variables species, state, field information, time for start and time for stop. The output is seedContent in soil after

germination and the new plant situation (*population_Development*) for the species.

The function uses two help functions - *seedCountAdjustment* and *plantCountAdjustment* to calculate the change in the seed situation and plant situation.

VegetativeReproduction
VegetativeReproduction has the input variables species, the original or start value of population_Development, the population_Development after germination, the start and stop time and possibly also fieldInformation. The output will be an updated population_Development table for the species.

The function to implement shall calculate the new count of plants of the species after vegetative reproduction in the period from start to stop time. The vegetative reproduction will of course depend on the species. But also on the crop, and possibly the other weeds present.

Growth
Growth has the input variables species, plant situation, fieldInformation and time of start and stop of simulation. The output will consist in a new plant situation. The function is a growth curve. For the moment the function to include has not been specified.

Natural functions to describe the growth could be either the cumulative normal distribution or the logistic equation. The logistic equation has been applied in many biological areas. It was first proposed by a Belgian mathematician for describing the cumulative growth of populations.

The logistic curve describes the population or individual growth under normal limitations in space and food. In integrated form the equation relating yield (y) and time (t) may be written as

$$y = \frac{y_0}{1 + e^{-(a+bt)}}$$

where e is the base of the natural logarithm, a and b are constants and y_0 *is* the upper asymptote. We assume that y_0 and y are measured from a known lower asymptote as zero. The response y then plots against t as a symmetrical sigmoid curve (fig. 4.4). According to species the parameters a, b and y_0 will change resulting in different maximum value and slope

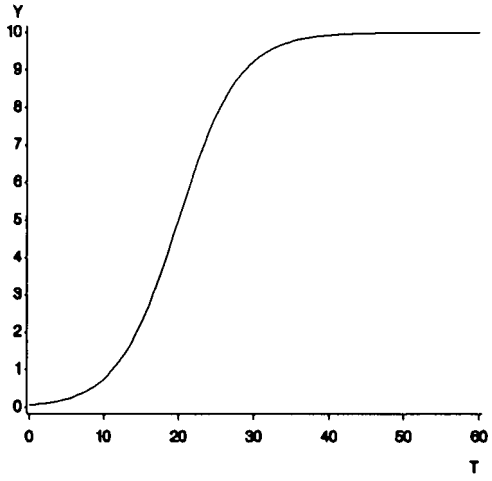

***Figure 4.4** Logistic curve for description of growth.*

of the curve. The function can then be calibrated with the actual start values in plant situation.

Influence of competition on the growth should be built into the *growth* function. How to do this is complicated. Most investigations on effect of competition on the crop has concentrated on depression in yield at a given weed

infestation at a given moment. (Wilson 1986, Wilson & Wright 1990). Investigations of competition uses weed dry weight as an indicator for competitiveness, often in the form of crop equivalents (as used in WEEDOF). A high dry weight compared to the total dry weight of the field indicates a high competitive effect. Investigations of the competitive effects during the growing season (Courtney & Johnson 1988, Wilson & Wright 1990) have shown that the crop equivalents change throughout the development of the crop. The average of the crop equivalents can often be used as an indicator of competitive effects on the yield, but sometimes the plant weight in a special time of the growing season has a better correlation (Wilson & Wright 1990).

The competition from weeds also has an effect on the harvest index - the grain/straw part of the yield. The harvest index falls as the count of weeds rises (Wilson & Wright 1990).

There has not been much research on the influence of competition on the growth of the crop and weed before harvest. Conolly et al 1990 and Håkanson 1991 has performed some research of the competition in the time-course of the growing season. They too are interested in the biomass production at harvest. Instead of static models correlating the observations to one single final harvest, Conolly et al 1990 harvested the plants several times during the season. Then they could study the development in growth periods during the growing season and the correlation to the final yield. Conolly et al (1990) showed that species interactions varies during time, and with the way they are mixed. Their results also suggests that the effect of earlier germination on the interaction is considerably greater than the species competitive ability. Håkanson 1991 concluded that biomass assessment of a weed at some time during the growing season is not a sufficient basis for predicting yield reductions.

How to incorporate this information in a function is difficult. The reason for the change in harvest index could be a reduced ability to produce seeds due to a reduced size, or a slower development because of competition (fig 4.5). So more investigations are needed to explain the effects of competition on growth.

SoilSeedContent

SoilSeedContent is the last help function used by *simulatePlant*. The function has the input variables species, the original state, the value of seedContent after germination, start time and stop time and possibly fieldInformation. It will return an updated seedContent table.

SoilSeedContent uses the two help functions *seedDeath* and *seedSupply* to calculate the seed content in soil at the end of the period.

SeedCountAdjustment

SeedCountAdjustment has the input parameters species, seed content in soil for the species *seedContent* , start and stop time, and possibly fieldInformation. The function will return an updated seedContent table for the species and the number of germinated plants (*germinated plants*). Seed content in soil for the species is a table which has depth (*depth*) as key and count (*count*) as value. For each depth in the table the helpfunction treat will calculate the germination and return the new count and the number of germinated plants. A new *seedContent* table is constructed with the results for each depth, and the sum of this and the count of germinated plants is calculated.

Treat

Treat has the input parameters species, depth, seedContent and stop time and possibly fieldInformation. The output is the new seedCon-

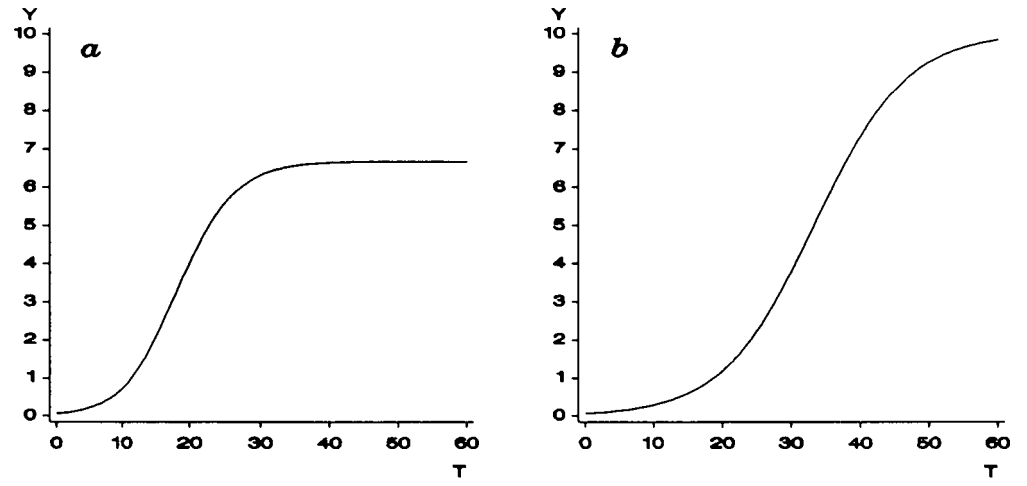

Figure 4.5 *Two possible effects of competition on the growth. In a) competition affects the asymptote ie the maximum yield. In b) the slope or growth rate is affected.*

tent and the count of germinated plants.

The function uses a table of normal germination, where species, depth and time of the year are pictured on a normal germination fraction. The count of seeds in the depth is multiplied with the germination fraction. The new count in soil - after the germination part of the seeds has been subtracted - , and the germination part is returned.

The table this function uses may be difficult to construct. The yearly germination of different species are often known, as is the germination curve over the year. But here we need information on the germination for the top and bottom layer in soil, not the soil in general. A possible approximation is to assume that all germination is from the top layer.

PlantCountAdjustment

PlantCountAdjustment has as input variables species, population_Development and the count of germinated plant from the function *seedCountAdjustment*. The output is a new population development table for the species. The function shall put the newly germinated plants into the population development table. The table is a picture of the developmental stage to a new table picturing height into count. The new germinated plants are to be put in the table with the developmental stage 'germ' as a key. If 'germ' is not a key in the table then a new key should be made and the table value pictures the normal height of germs of the species to the count of germ plants. If 'germ' is already a key in the table then the count, which is the table value of the normal germ height for the species, should be increased with the count of germinated plants. The normal germ height is looked up in a table where species is pictured into height.

SeedDeath

SeedDeath has the input parameters species, seed content in soil for the species at the start of the simulation period, seed content after germination in the simulation period, the start

and stop time, and possibly *fieldInformation*. Output is the updated seed content table.

The function uses a table of the normal seed mortality for the species and time of year, which is assumed here to be independent of depth. The data for this table could be difficult to obtain. It is known that annual species have a higher mortality rate than perennial. But the mortality of seeds in the soil also depends on the soil and the conditions for the seeds. There are examples of seeds surviving extremely long periods of burial under exceptional conditions.

The function calculates the count of dead seeds by multiplying the normal mortality rate with the initial count of seeds. and produces a new seed content table by subtracting this number from the seed content after germination.

SeedSupply
SeedSupply has the input parameters species, populationdevelopment, seedContent, start and stop time. The output is an updated *seedContent* table.

The function shall calculate the new seed content after seed production from the plants on the field. The help function *seedProduction* calculates the production of seeds. The sum of the seed production and the count of seeds in the top layer of the soil is calculated as the new soil seed content in the top layer. There is two problems here to deal with: firstly, a part of the total production dies or disappears before being incorporated in the soil due to wind or predators. This can be solved two ways: Either the seed production function should calculate a seed production which is smaller than reality to account for the missing seeds. Or a part of the seed production should be subtracted before calculating the new amount in soil. Secondly, a similar problem exists with the crop, only that here most of the seeds disappears - they are harvested - before incorporation in the soil. The last solution, subtracting a part of the seed production before calculating the new amount is the best solution here, in that way an estimate of the harvest is found at the same time.

SeedProduction
SeedProduction has the input parameters species, populationDevelopment and start and stop time. The output is the count *count* of produced seeds.

The function uses a table which pictures species and height on a count. This table relies on the assumption that the competition affects the height. Then the height could indicate the potential seed production for the species taking competition implicitly into account. The data for the table is probably not available. The assumption also may be wrong or the correlation poor between height and seed production. This question needs further investigation. In a model for the effect of cultivation on the vertical distribution of weed seeds in soil, Cousens and Moss (1990) used a different approach. They used a figure for seed production under no intra-specific competition, and then modified with a parameter for density dependent seed production. Intuitively the other approach is better because the relation between the competition and the reduced seed amount due to influence on development is explicitly stated.

SimulateAction
SimulateAction is the function which simulates the effect of actions on the seeds and plants. The function has the input parameters state, a list of the present species, fieldInformation and time of the action. The output is a new state. The function uses the two help functions *effectOnSoilSeeds* and *effectOnPlants*.

EffectOnSoilSeeds
EffectOnSoilSeeds describes what has happend to the seeds in the soil after performing an action such as harrowing. Input parameters are seed situation, fieldInformation, a list of present species, action and time of action. The output is a new seed situation.

We assume that an action only influences the seeds position in the soil. An action will move a portion of the seeds downward from the upper layer, and a portion upward from the lower layer. The help function *movement* calculates the up and downward movements, and returns the variables *UP* and *DOWN* to *EffectOnSoilSeeds*. In the upper layer then the number of seeds moving down *(DOWN)* is subtracted from the original seed content and the number of seeds moving up *(UP)* from the lower soil layer is added. In the lower layer *UP* is subtracted and *DOWN* is added.

Movement
Movement calculates the count of seeds moved upward and downward. The input parameters are seedContent and action. Maybe species has to be included if the movement depends on for instance the seed size. At moment it is assumed that the fraction of seeds to be moved are independent of species. This assumption is not always valid. It is valid for plowing, but apparently not for rigid tine cultivation (Cousens & Moss 1990).

The function uses a table which pictures action and depth onto a fraction of seeds which are moved when performing the action. The seed content in a depth is multiplied by the fraction which is moved. Down is the movement from the upper layer down, up is the movement from the lower layer. The calculation is quite simple here as there are only two layers. Cousens and Moss (1990) modelled the distribution in a model with 4 soil layers. Their approach is very similar to the one used here.

EffectOnPlants
EffectOnPlants calculates the effect on the plants from performing the action. Input parameters are plant situation, fieldInformation, a list of present species, action and time of action. The output is a new value of plant situation.

For every species in the list of present species the help function *actionDevelop* produces a new population development table which is added to the old one.

ActionDevelop
ActionDevelop has as input parameters species, populationDevelopment, action and time, and possibly fieldInformation. Output is a new table of populationDevelopment.

For all the developmental stages which are keys in the population development table the help function *actionHeight* produces a new value of heighttable, which is added to the old one, and thereby replaces old values with new.

ActionHeight
ActionHeight has the input parameters species, heighttable, action, time and possibly fieldInformation. The output is a new value of heighttable.

To look up the effect of the actions in terms of the fraction which survives the action a table is used. The table pictures species, height of species and action on the fraction of plants which are alive after the procedure. For all the developmental stages and heights of the species at the time of action the count is multiplied with the fraction from the table. The new count for the height is then included in the height table.

The fractions left after performing an action are not so straight forward to predict as this solution states. The effect of harrowing a field for instance is very dependent on the weather after the harrowing. If the weather is moist after harrowing more plants will survive than in dry weather. The present model does not consider weather conditions so a standard figure must be used. The question is then whether this should be the minimum survival, a medium count or possibly an interval. In WEEDOF an interval is used to calculate the effect from the worst to the best case.

4.5 The model as part of a model based system

The model described is to be used in a planning expert system to simulate the growth and development of crop and weeds, the seed content in soil and the effect of control actions, and explain the conclusions. The model simulates the whole plant and seed situation on the field and can cope with several actions. This differs from the present system. WEEDOF can calculate the changes in count of weeds due to the effect of an action. The seed rotation is examined when certain weeds are present and the effect on the seed content in soil is not handled explicitly.

The model must be implemented in a language and built into an expert system. For the implementation of the model any ordinary programming language, for instance PASCAL, is usable. The present framework of a model has been implemented in PASCAL. The way of specifying the model, resulting in a model consisting of functions, makes it extremely easy to implement. The functions are simply represented as procedures and functions in the chosen language. For instance the domains State, SeedSituation and SeedContent may be represented like this in PASCAL:

```
state = record
     seed : seedsituationp;
     plant: plantsituationp;
   end ;

seedcontentp = ^seedcontent;

seedcontent = record
     d: depth;
     c: count;
   end;

seedsituationp = ^seedsituation;

seedsituation = record
     sp: speciestype;
     sc:  array[1..2] of seedcontent;
     next: seedsituationp
   end;
```

Then a function like seedSupply could be implemented like this:

```
procedure seedsupply(var sp: speciestype;
     popdev:  populationdevelopmentp;  sc:
     seedcontentp; tid, tidh: tidtype);
  var
     list:seedcontentp;
  begin
     list := sc;
     list^[1].c:= list^[1].c + seedproduc-
     tion(sp, popdev, tid, tidh);
  end;
```

A possible structure for a system with the implemented model is seen in figure 4.6.

The system has a part which collects initial information just like WEEDOF. The initial information is the crop, the use of the crop, the weeds normally present on the field, a size factor for the populations, the time of the year, and the period of the simulation.

The heuristic knowledge base suggests, on the basis of the initial information, a list of actions for the first crop. As is the case of WEEDOF, the choice of mechanical actions is constrained by the soil type and stones. Other rules in the heuristic knowledge base will constrain the

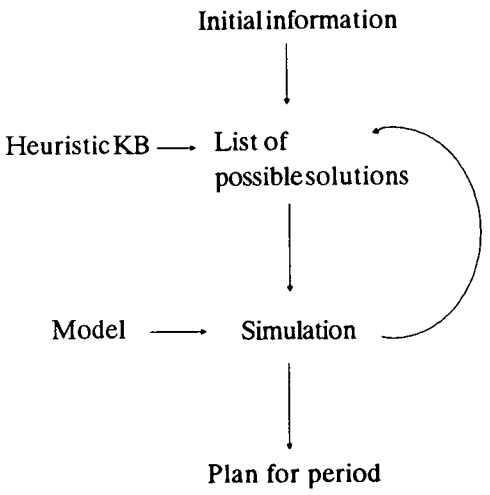

Figure 4.6 Possible structure for a model based expert system.

possible actions according to crop type, to sensible combinations according to the expert and actions which can be performed during the period.

The result from the heuristic knowledge base is then a list of possible plans for weed control in the present crop (lists of actions with times).

The model is now used for simulating the plans. The result of the simulation is a state for each plan, describing the predicted seed situation and plant situation after performing the actions. If the period is longer than one growing season another crop has to be included before the system starts over again creating a new list of plans and simulating them. A choice between the plans has to be made at some time in long periods, otherwise a combinatorial explosion will occur sooner or later. The choice should be made before a new crop is included and could be left to the user, or the system could choose the plan with the best controlling effect on weeds.

When the system comes to the end of the total period for simulation, the results shall be written to the user. The results could be several plans for controlling the weeds and the resulting predicted plant and seed situation afterwards. In case the user asks for information on how the end state was calculated the explain module will replay the model with the plan and explain each step in the function.

4.6 Summary and conclusion

This chapter describes the specification of a model which could be used in an expert system. The model is not finished yet. A skeleton has been made where the functions can be placed. For some of these functions there has been a research for suitable mathematical expressions. This has been the case for the growth function where especially the effect of competition has been examined. But other parts have never been modelled the way it is needed here. The decisions about the missing functions remains. In the skeleton abstract domains for variables have been defined. Care has been taken to define these domains in the best possible way for usage, but the decision of the exact values for for instance *Developmental stage* has been postponed until later.

Another part which has been postponed is the decision of the time steps for the model. For the moment the time steps is decided by the time between succeeding actions. It will probably be necessary to have shorter time steps. A possible way to implement this in the present structure, is to invent a new action 'no-action' to put into the list of actions when the time steps are too long.

Models can be used in different ways in model based expert systems. The expert system part may be used as simply a wrapping for the model to collect the input parameters and interpret the output parameters. Or the model could be a part of the whole system which may contain other parts as for instance data bases as in Jones et al (1987).

In this system the model is intended to be used the latter way. The model shall partly simulate the growth and developments of plants, partly explain the results. With such a general population dynamic model the predictions will probably not be very good. But we need a model with explanatory power, which can reveal the trends in the system and explain them. The explanations are extremely important - more important than the closer fit that could be obtained with an empirical model for instance.

To specify the model a new method in agricultural connections has been used. The method of specifying systems by functional decomposition is well known in computer science, but has not been used in agricultural modelling. The method has shown - not unexpected - to be very good also in this type of system description. The top-down method of specification gives the possibility to decompose problems and in that way push problems ahead, to be solved in another help function at a later stage when the problem has been split and changed to a smaller and more manageable one.

5 Summary and conclusion

The work in this Ph.D. project focused on two different subjects: Building a prototype for planning weed control in organic farming and specifying a dynamic model for plant growth.

5.1 Prototype

The standard construction method of rule based expert systems is an iterative procedure where the knowledge engineer proceeds through the phases of conceptualization, formalization, and implementation over and over again. There is no formal method for construction of expert systems, but a number of descriptions of methods for knowledge elicitation and knowledge representation. Research results on preliminary knowledge analysis methods and domain characterization methods are underway (Nwana et al 1991), but so far each new system builder has to find the best way to construct these systems.

In this experiment the knowledge engineer was new in the field of knowledge engineering, and the first prototype probably took longer to construct than it would have taken for an experienced knowledge engineer, but the development was facilitated by the use of a new method for the initial knowledge acquisition - literature analysis. In literature analysis texts from the domain are analyzed to extract the important concepts of the domain, and the rules about the concepts such as definitions and causal relations. A parallel method has been used for automatic construction of small knowledge bases (Gomez & Segami 1990).

In net time the analysis probably took about 2-3 months. The result was a concept hierarchy and a survey of rules, as well as something more indefinable - a feeling of understanding the domain, knowing the important concepts, the relations and so on. Concepts often seem very obvious when they are written down, and many of them would have been mentioned as important subjects in an interview with the expert. In this case one of the experts would probably have been able to work out the concept hierarchy and additional methods as for instance repertory grid or scaling techniques could have helped to reveal relations between them. But the strength of the literature analysis is that it is a simple semi-formal method which ensures that all relevant concepts - at least the concepts which are considered relevant in teaching the subject - are included with the important relations to them.

The rest of the knowledge acquisition was done using interviews. The interviews could be structured from the beginning because of the initial literature analysis which had provided the basis material. All in all only about six interviews were executed, the rest of the knowledge elicitation was done by letters and telephone calls.

The domain chosen - control of weeds in organic farming - was characterized by uncertain and missing knowledge. Research in the subject has been stopped for many years since the discovery of chemical methods, and was only recently restarted. It is a biologic domain and a lot of factors effects the growth and development of plants. The researchers in the domain were very doubtful about the possibility of developing expert systems in their domain. However the test succeeded. The experts were satisfied with the prototype. The experts also felt they had developed a new insight in their domain during the process of developing the expert system. The domain is studied so thoroughly that the experts discover weaknesses in the knowledge about the domain which result in new experiments. In addition to

the outcome from an expert system project in the form of a system, the project also gives a bonus for the experts involved in the form of a better survey of the present as well as the missing knowledge of the domain.

The resulting system - WEEDOF - was coded in EGERIA, an expert system shell. One of the important things missing from the present system is the explanations. First of all the explanations are very poor because of the combination of the shell and the system. The shell only supports explanations as a trace of the rules used during backtracking. As the present system uses forward chaining alternating with backward, this prevents the mechanism from functioning satisfactorily. Even if explanations could be formed from the knowledge in the present knowledge base, these explanations would be poor compared to the explanations from an expert. The expert will incorporate his models of the domain in the explanations, whereas the system can only replay the knowledge in the knowledge base which is largely heuristic. This is one of the reasons why the work proceeded by specifying a model.

5.2 Model

Another reason for working on the model is to make a system with a knowledge base which is more reusable than the heuristic knowledge base. A disadvantage of these model based systems is that they are less efficient.

Models can be used in different ways in model based expert systems. The expert system part may be used simply for collecting information for the simulation and for interpreting the output. The model could be an integrated part of the system as could for instance databases. The system could also embrace several models, as for instance refinements to be able to explain on different levels.

In this work the model was intended to be an integrated part of the system where the expert system not only collects input for the model and interprets its output, but also does a heuristic job finding the relevant or possible control actions before simulating.

The work on the model has been started but the model based system itself is only in the preliminary stage. The method used in specification of the model is new in agricultural connections. The method of specifying systems by functional decompositions is well known in computer science, where it is used in the Vienna Development method - VDM (Bjørner & Jones 1982) - for computer systems. The model has been specified in META IV, and the method has shown to be useful also in this type of system description. The top-down method of specification implies decomposing problems, and in that way trying to simplify them before they have to be solved.

The model which has been specified, or partly specified, is a dynamic model for the total plant growth on a field. The model is intended to account for effects on the growth of different actions, as for instance harrowing. The model should also incorporate competition between species. The model should be general, making it possible to describe the growth of all the plants on a field. The question is if it is possible to make such a general model at present.

The model specified here relies on the life cycle of plants which are fairly common. There is a general pattern of life in plants where seeds germinate to plants which grow, set flowers and seed. The model then has to be able to model both those plants which are

annual and those which are perennial, seed - as well as root propagating species. In the model there are two different contributions to the plant growth. One is the natural plant growth according to the species and constrained by competition - other constraints for instance nutritive and climatic have not been considered yet. The life cycle was used as basis in the decomposition of the model into functions. The other contribution is the impact on plants and seeds of the actions performed on the field.

The specifications show all the functions which are necessary to describe this, with the input and output to them. The concrete algorithmic specifications have not been made. Every model of course is a simplification of the real world. Some or maybe all of the functions in this model could be described better for instance with an empirical model. The functions in the mechanistic model are made of parts in a way that try to imitate the construction in nature. To make it possible to survey the model the functions are kept rather simple. Parts are missing, either because they are deliberately omitted - for instance because they are considered of minor importance - or because the knowledge is missing. However the reason for retaining the mechanistic model is the ability to explain and justify the functioning of the resulting system in terms of the deep knowledge of the domain.

5.3 Expert systems and agriculture

Can expert system technology be used in agriculture? There are obvious possibilities in agriculture where the technology will be usable. Examples are:

- surveillance for instance of climate in greenhouses,
- planning in farming - for instance planning the distribution of the available manure in organic farming,
- diagnosis of for instance sicknesses.

During time more and more knowledge has to be included in decisions in agriculture to ensure the necessary profit. Now that pc's are getting used in the farmers production, there will be a marked for decision support systems. Not necessarily expert systems but they will be part of the new systems.

The trend of expert systems usage is to integrate them with other types of software. The original expert systems are stand-alone systems on a narrow domain. It is generally considered to be an advantage to integrate the expert systems with databases or models and let them work in cooperation with other software the user is attending. In that way the expert systems becomes a natural part of a larger package and is used more.

Construction of expert systems generally takes longer time than construction of ordinary computer programs. Therefore it is important to be careful in choosing domains where the development can be justified. This could be on basis of for instance profit or lack of available expert time. The last reason has been the basis for instance in Australia where the distances are enormous and the experts few (Waterhouse et al 1989). Looking at the conditions in Denmark, the income on systems in agriculture could easily be too small to pay for the development of Danish expert systems. Some systems could in stead be developed for the larger EEC marked, or North European countries in cases where South Europe is very different from Denmark.

In the future there are hopes that the developmental costs of expert systems will become smaller. New knowledge acquisition tools are coming up which aims at easing the knowledge

collection, for instance by giving the expert tools to codify his knowledge, and new methodologies are developed to formalize the development process - literature analysis could be the background of a more formal approach.

The agricultural researchers seems to have advantages from cooperating in expert system projects. The different way of working with the domain when eliciting and formalizing the knowledge gives a feed back to the expert in the form of a better insight of usable knowledge and weaknesses in the knowledge of the domain, as the present project has shown. The work on an expert system project will often mean a formalization of knowledge making it possible to rewrite the knowledge in traditional program languages which will give more efficient programs.

6 References

Anonymous 1989a. Expertech. Egeria. Expert systems with Egeria. Version 1.1.

Anonymous 1989b. Expertech. Egeria. Technical reference manual. Version 1.1.

Anonymous 1989c. Expertech. Egeria. PC-DOS reference manual. Version 1.1.

Ballegaard, T. & Haas, H. 1990. WEEDEX - an expert system for identification of weed seedlings. Papers presented at a workshop on expert systems in agricultural research Ebeltoft 4-5 december 1990. Fællesberetning nr SF1, 27-31.

Barralis, G., Chadocuf, R. & J. P. Lonchamp 1988. Longevité des semences de mauvaises herbes annuelles dans une sol cultivé. Weed Research 28, 407-418.

Barrett, J.R. & Jones, D.D. 1989. Knowledge engineering in agriculture. American Society of Agricultural Engineers.

Batchelor, W.D. & McClendon, R.W. 1989. Evaluation of SMARTSOY: An expert simulation system for insect pest management. Agricultural Systems 31, 67-81.

Beck, H., Jones, P., Watson, D. & Zazueta, F. 1989. An expert database system for ornamental plants. Agricultural Systems 31, 111-126.

Bjørner, D. & Jones, C.B. 1982. Formal specification & software development. Prentice-Hall.

Bliss, C.I. 1970. Statistics in Biology. Statistical Methods for Research in the Natural Sciences. McGraw-Hill.

Boggess, W.G., Blokland, P.J. & Moss, S.D. 1989. FinARS: A financial analysis review expert systems. Agricultural Systems 31, 19-34.

Buchanan, B.G. & Smith, R.G. 1989. Fundamentals of Expert Systems. In Barr, A., Cohen, P.R. & Feigenbaum, E.A. (eds): The Handbook of Artificial Intelligence vol IV. Addison-Wesley.

Burton, A.M., Shadbolt, N.R., Rugg, G., Hedgecock, A.P. 1990. The efficacy of knowledge elicitation techniques: a comparison across domains and levels of expertise. Knowledge acquisition 2, 167-178.

Brummenæs, N. 1990. Teknikker for tapping av kunnskap. Department of Computer Science. Technical University of Denmark.

Clancey, W.J. 1985. Heuristic Classification. Artificial Intelligence 27, 289-350.

Conolly, J., Wayne, P. & Murray, R. 1990. Time course of plant-plant interactions in experimental mixtures of annuals: density, frequency, and nutrient effects. Oecologia 82, 513-526.

Courtney, A.D. & Johnson, R.T. 1988. Crop Equivalents of Weeds in Spring Barley. Aspects of applied biology 18, 57-62.

Cousens, R. & Moss, S.R. 1990. A model of the effects of cultivation on the vertical distribution of weed seeds within the soil. Weed Research 30, 61-70.

Cussans, G.W. & Rolph, J. 1990. HERB-MAST - a herbicide selection system for winterwheat. Proc. EWRS Symposium 1990,Integrated Weed Management in Cereals, 451-457.

Dindorp, U. 1990a. Ekspertsystemer og deres brug i jordbrugssektoren. Temadag om Biometri og Informatik. Tidsskrift for Planteavls Specialserie. Beretning nr s2053. 77-82.

Dindorp, U. 1990b. Development of an expert system for non-chemical weed control. Papers presented at a workshop on Expert Systems in Agricultural Research. Fællesberetning nr SF1, 23-26.

Dindorp, U. 1991a. Developing WEEDOF, a prototype expert system for planning weed control in organic farming. Temadag om Biometri og informatik. Tidsskrift for Planteavls specialserie. Beretning nr s2129, 7-14.

Dindorp, U. 1991b. Literature analysis for

knowledge acquisition. Proceedings of the SCAI '91, Roskilde. Denmark 21-24 may 1991, 77-82.

Doluschitz, R. 1990. Expert systems for management in dairy operations. Comput. Electron. Agric. 5, 17-30.

Doyle, C.J., Cousens, R. & Moss, S,R, 1986. A model of the economics of controlling Alopecurus myosuroides Huds. in winterwheat. Crop protection 5,143-150

Dreyfus, H. & Dreyfus, L. 1986. Mind over Machine. Basil Blackwell Ltd.

Fynn, R.P., Roller, W.L. & Keener, H.M. 1989. A decision model for nutrition management in controlled environment agriculture. Agricultural Systems 31, 35-53.

Gaultney, L.D., Harlow, S.D. & Ooms, W. 1989. An expert system for troubleshooting tractor hydraulic systems. Comput. Electron. Agric. 3, 177-187.

Gold, H.J., Wilkerson, G.G., Yu, Y. & Stinner, R.E. 1990. Decision analysis as a tool for integrating simulation with expert systems when risk and uncertainty are important. Comput. Electron. Agric. 4, 343-360.

Gomez, F. & Segami, C. 1990. Knowledge acquisition from natural language for expert systems based on classification problem-solving methods. Knowledge acquisition 2, 107-128.

Haas, H & Ballegaard, T. 1988. Identifikation af ukrudtsarter på unge udviklingsstadier ved hjælp af et edb-ekspertsystem. Danske Planteværnskonference / Ukrudt ,102-112.

Hamilton, A.G. 1988. Logic for Mathematicians. Revised edition. Cambridge University Press.

Hansen, S., Jensen. S.E., Nielsen, N.E. & Svendsen, H. 1990. DAISY - Soil Plant Atmosphere System Model. NPO-Research Report No. A 10. The National Agency of Environmental Protection. Copenhagen, Denmark.

Harder, B. 1990. Konstruktion af videnbaserede systemer, EC-rapport 240.

Hart,A. 1986. Knowledge acquisition for expert systems.Kogan Page Ltd.

Hayes-Roth, F., Waterman, D.A., Lenat, D.B. 1983. Building expert systems. Addison-Wesley.

Huber, B., Nyrop, J.P., Wolf, W. , Reissig, H., Agnello, A. & Kovach, J. 1990. Development of a knowledge based system supporting IPM decision making in apples. Comput. Electron. Agric. 4, 315-331.

Håkansson, S. 1991. Growth and competition in plant stands. Crop Production Science 12. SLU/Repro, Uppsala.

Jacobson, B.K. , Jones, P.H. , Jones, J.W. & Paramore, J.A. 1989. Real time greenhouse monitoring and control with an expert system. Comput. Electron. Agric. 3, 273-285.

Jensen, P.K. & Kudsk, P. 1988. Prediction of herbicide activity. Weed Research 28, 473-478.

Jones, J.W., Jones, P. & Everett, P.A. 1987. Combining Expert Systems and Agricultural Models: A Case Study. Transactions of the ASAE 30,1308-1314.

Jones, P. 1989. Agricultural applications of expert systems concepts. Agricultural Systems 31, 3-18.

Kline, D.E., Bender, D.A., McCarl, B.A. & Van Donge, C.E. 1988. Machinery selection using expert systems and linear programming. Comput. Electron. Agric. 3, 45-61.

Martinsen, I.-M., Stausgaard, J.P., Weibel, S. 1986. Ekspertsystemer i landbrugssektoren. Specialerapport, Aarhus Universitet.

McKinion, J.N. & Baker, D.N. 1989. Application of the GOSSYM/COMAX system to cotton crop management. Agricultural Systems 31, 55-65.

McKinion, J.M. & Lemmon, H.E. 1985. Expert systems for agriculture. Comput.

Electron. Agric. 1, 31-40.

Morgan, O.W., McGregor, M.J., Richards, M. & Oskoui, K.E. 1989. SELECT: An expert system shell for selecting amongst decision or management alternatives. Agricultural Systems 31, 97-110.

Moss, S R. 1983. The production and shedding of Alopecurus myosuroides Huds. seeds in winter cereals crops. Weed Research 23, 45-51.

Moss, S.R. 1985. The survival of Alopecurus myosoroides Huds. seeds in soils. Weed research 25, 201-211.

Moss, S.R. 1990. The seed cycle of Alopecurus myosuroides in winter cereals: A quantitative analysis. Proc. EWRS symposium 1990,integrated weed management in cereals, 27-36.

Nilsson, N.J. 1982. Principles of Artificial Intelligence. Reprint. Springer-Verlag.

Nwana et al 1991. Facilitating development of KB's. AICom 4, 60-73.

Pasqual, G.M. & Mansfield, J. 1988. Development of a prototype expert system for identification and control of insect pests. Comput. Electron. Agric. 4, 263-276.

Plant, R.E. 1989. An artificial intelligence based method for scheduling crop management actions.Agricultural Systems 31, 127-155.

Rasmussen, J. 1990. Ukrudt i agerlandet In Ukrudtsbekæmpelse i landbruget. Statens Planteavlsforsøg, Planteværnscentret.

Rasmussen, J. & Vester, J. 1990. Ikke-kemisk ukrudtsbekæmpelse In Ukrudtsbekæmpelse i landbruget. Statens Planteavlsforsøg, Planteværnscentret.

Rich, E. 1983. Artificial Intelligence. McGraw-Hill.

Rischel, H. 1987. Programdesign sat på formler. Department of computer science. Technical University of Denmark.

Roach, J., Virkar, R., Drake, C. & Weaver, M. 1987. An Expert System for helping Apple Growers. Comput. Electron. Agric. 4, 97-108.

Shaw, M.L.G. & Woodward, J.B. 1990. Modeling expert knowledge. Knowledge acquisition 2, 179-206.

Sowa, J. F. 1984. Conceptual structures : Information processing in mind and machine. Addison-Wesley.

Spiegelhalter, D.J. & Lauritzen, S.L. 1988. Local computations with propabilities on graphical structures and their application to expert systems. J.R.Statist.Soc.B 50(2)-,157-224.

Spiegelhalter, D.J. & Lauritzen, S.L. 1990. Techniques for Bayesian analysis in expert systems. Annals of Mathematics and Artificial Intelligence 2, 353-366.

Spitters, C.J.T., Keulen, H. van & Kraalingen, D.W.G. van 1989. A simple and universal crop growth simulator: SUCROS87 In R.Rabbinge, S.A.Ward & Laar, H.H-.van (eds) Simulation and Systems management in Crop protection. Wageningen, 32,147-181.

Stone, N.D. & Toman, T.W. 1989. A dynamically linked Expert-Database system for decision support in Texas Cotton Production. Comput. Electron. Agric. 4, 139-148.

Sørensen, P.S. 1987. Ekspertsystem til biologiske renseanlæg. Exam.proj. Technical university of Denmark.

Wain, N. , Miller, C.D.F. & Davis, R.H. 1988. A rule-based inference system for animal production management. Comput. Electron. Agric. 2, 272-300.

Waterhouse, D.B., Lodge, G.M. & Bishop, A.L. 1989. LUPEST personal communication. New South Wales Department of Agriculture and Fisheries. Agricultural Research Centre.

Waterman, D.A. 1986. A Guide to Expert Systems. Addison-Wesley.

Whittaker, A.D. , Tomaszewski, M.A., Taylor, J.F., Fourdraine, R. & van Overveld, C. J.

1989. Dairy herd nutritional analysis using knowledge systems techniques. Agricultural Systems 31, 83-96.

Wilson, B.J. 1986. Yield responses of winter cereals to the control of broad-leaved weeds. Proc. EWRS Symposium 1986, Economic Weed Control, 75-82.

Wilson, B.J. & Wright, K.J. 1990. Predicting the growth and competitive effects of annual weeds in wheat. Weed Research 30, 201-211.

Woodward, B. 1990. Knowledge acquisition at the front end: defining the domain. Knowledge acquisition 2, 73-94.

Yoeli, R., Manor, G. & Gill, A. 1989. Modeling human intuition in an expert system for planning aerial application operations. Comput. Electron. Agric. 4, 13-22.

Zwerger, P. & Hurle, K. 1986. Veranderung der lebens- und keimfahigkeit von unkrautsamen im boden. Med. Fac. Landbouww. Rijksuniv., Gent 51,325-332.

Zwerger, P. & Hurle, K. 1989. Untersuchungen zur relativen Bedeutung einzelner populationdynamischer Parameter fur die entwicklung der Verunkrautung. Zeitschrift fur Pflanzen krankheiten und Pflanzenschutz 96, 346-352.

Østerby, T. 1988. Knowledge acquisition and specification in Information modelling and knowledge bases, IOS Press Amsterdam.

For the literature analysis a couple of articles from a book on weed control (Rasmussen 1990, Rasmussen & Vester 1990) was used. In this appendix a part of one of the articles is reproduced in an english translation. The underlined sentences were the ones considered to contain important information.

Non-chemical weed control

Jesper Rasmussen and Jacob Vester (translated from danish)

.....

4.1 Preventive methods

Crop rotation

Formerly there used to be some constraints on the crop rotations which alone was due to weed considerations. Such constraints are still known in organic farming, where no plant protection chemicals are used. Without effective control methods, the key to clean fields are in a balanced crop rotation where crops of different life-length are alternating. Many weeds are propagated in a certain crop type and this diminishes the possibility for single weeds to multiplicate.

The effect of crop rotation on the weeds are often difficult to predict. This is because the crop rotation deals with both crops in different orders, and with the methods of cultivation for the crops. In the following there will only be given some general guiding lines for the influence of crop rotation on weeds.

For weeds which propagates in special crops, there are good possibilities to use crop rotation in weed control. This counts especially for weeds with a short durability in soil, for instance Avena fatua, Galium aparine and Apera spica-venti. If these species only have opportunity to seed in one or two crops in a balanced crop rotation, they will have difficulties to survive, because a large part of their seeds are destroyed before the right propagation conditions are available again.

For weeds which can propagate in a variety of crops, for instance Stellaria media and Poa annua the crop rotation will have a minor influence on the control. This also counts for species with a long durability in soil. They will not have difficulties surviving as seeds and emerge when the right conditions are available (for instance Chenopodium album).

Crop rotation the is not a cure for weeds. Crop rotations which are especially suited for some weeds will some times favour other weeds. This is seen in figure 4.1.1: Avena fatua and Alopecurus myosuroides propagate in different crop rotations. Avena fatua propagate in spring cereals, and Alopecurus myosuroides in winter cereals. If crop rotation has to be used for weed control, one has to know which weeds one wishes to control. For the root propagated weeds the rotation must give possibility to control mechanically or chemically.

Deep soil treatment

Plowing has especially an effect on root propagated weeds but also influences the annual weeds.

Often problems are encountered with annual grass weeds and root propagated weeds when plowing is omitted (table 4.1.1).

The root propagated weeds propagate when plowing is omitted. Plowing weakens the vegetative reproductive organs by burying them, so they have to use energy to regrow. This is important, when the weed grows with a strongly competitive crop as for instance corn (fig 4.1.2).

The reaction of the annual species are more complicated but still understandable.

An ordinary winter plowing will bury approximately 95% of the seeds from the soil surface in more than 5 cm depth. ie deeper than most weeds will be able to germinate from. At the plowing next year many seeds will be plowed up again. When plowing, the main part of the germ plants will stem from seeds more than a year old (figure 4.2.3).

In plowing free cultivation the main part of the germinated weeds will stem from seeds less than one year old, because the last produced seeds still is near the soil surface. This is an important cause of the different reactions from the weeds to deep soil treatment. Species with a short seed durability has a handicap to species with long durability, when the majority of the germination is from seeds more 1 1/2 year old, as is the occasion when plowing.

A species as Galium aparine will propagate itself immensely when cultivating without plowing. Its seeds has a very short durability in soil. The germination percent falls about 60% per year compared to about 30% normal for several other species.

The different weed species ability to survive as seeds in soil is treated in chapter 2.

It is no law of nature that species with a short seed durability will give problems with plowing free cultivation, even if their possibility for propagation will increase. These species will often be removed faster from the soil seed reserves if, at the same time, an effective control is carried through, than by plowing. If the soil is plowed a much larger volume of soil has to be depleted for living seeds than by a more superficial treatment, therefore it will take a longer time. In table 4.1.2 and 4.1.3. is a couple of examples of adapting the deep treatment to make the weeds germinate in the crop, where it is desirable. This could be in a crop where the particular species is easy to control.

Sowing bed preparation

The sowing bed preparation has also an influence on the weeds. In the fall an early sowing will generally give the biggest weed problems (figure 4.1.4.),conversely late sowing gives the biggest weed problems in the spring for corn crops. In the warmth demanding crops several harrowings before sowing can reduce the germination in the crop.

Here the weed species also differs. For instance it has shown, in Swedish and English trials, that 10 days delayment in the sowing of spring crops can reduce Avena fatua problems considerably. It is important to sow in the right depth, in order for the crop to germinate quickly and uniform. That will give the greatest possible competitive ability towards the weeds.

Notes from the literature analysis

The literature analysis produced two things as described in chapter 3. A concept hierarchy which is reproduced in appendix A3 and a set of notes concerning information on the concepts in the hierarchy. This appendix contains some of the notes from the analysis.

Plants
Plants = 'green part' + root.
Vegetative reproductive organs is a part of root.
Vegetative reproduction = above soil rep., in soil rep., on place rep.
Perennial plants normally has vegetative reproduction.
Plants with vegetative reproduction has vegetative reproduction organs.
The 'green part' of plants with vegetative reproduction produces reserves.
Plants with vegetative reproduction collects reserves in soil organs.
Soil organs is a part of root.
Winter removes the 'green part' of some plants.
Weed control removes the 'green part' of plants.
When the 'green part' of plants is missing they will use reserves to grow again.
Shadowed plants grows poorly or die.
Covering plants causes them to die.
Competitive ability = ability to make other plants grow poorly or die.
Good growing conditions causes good competitive ability.
Plant sensitivity to harrowing decreases with increasing size.
Plant size increases through summer.
Intensity of harrowing can increase when plant size is increasing.

Weeds
Weeds are plants.
Weeds depreciate the crop qualitatively or quantitatively.
Many weeds only grows on cultivated soil.
Most seed propagated weeds stems from locally produced seed.
Some seed propagated weeds are sown with the crop because the seed is mixed with the crop seeds.
Annual grass weeds are seed propagated.
Annual grass weeds has a very low seed durability.
Root weeds are weeds with vegetative reproduction.
Reproduction of root weeds depends on the reserves in the organs under soil.
Weeds causes harvest troubles, drying costs, vaste by seed cleaning.
Some weeds are poisonous.
There is most weeds on humus, less on clay soil.
Plowing is important to control perennial weeds.
If the crop germination time is earlier than the weeds, then the crop competitive ability is the

largest.
Agricultural crop plants often have a lower germination temperature than weed seeds.
A varied crop rotation reduces the weed count.
Most weeds are annual.
Dry matter minimum = time where a plant have used reserves to shoot and are about to start producing dry matter.
Dry matter minimum:
Elymus repens = 3-4 leaves.
Sonchus arvensis = 5-7 leaves.
Cirsiun arvense = early bud.
Ten days later germination in the spring diminish the amount of Avena fatua germinating.
The amount of Elymus repens doubles from corn harvest (july-august) to time of winter plowing (oct-nov).
Elymus repens grows in the summer.
Elymus repens rests from late fall, and during the winter.
Elymus repens germinates from runners.
Do not spread Elymus repens runners.
Elymus repens amount is largest at borders, around stones and in perennial plants.
Hoeing: Weeds with large competitive ability and early stretching growth gives problems in the row.
Effect of hoeing on large weeds increase with speed (from 4-6 km/h to 10-12 km/h).

Crop
Make sure the crop has a good competitive ability.
Crop with a high germination temperature has a late sowing time.
Several harrowings can control weeds if the crop has a late sowing time.
Hoeing can be used in Zea mays against all seed reproductive weed.
Hoeing can be used in Solanum tuberosum against Elymus repens.
Hoeing can be used against Elymus repens.
Yield will decrease if more than 20% of the crop leaves are covered
Yield = the harvested part of crop.
Row crops = crops, which are planted or sowed in a manner so the distance between rows are larger than the distance between plants.
Wet soil gives a bad competitive ability.
Sour soil gives a bad competitive ability.
Secale cereale shades the best.
Avena sativa shades better than Triticum aestivum.
There is a substantial difference between varieties in the competitive ability.
Agricultural crop plants often have a lower germination temperature than weed seeds.

Soil
Soiltype:
There is most weeds on humus, less on clay soil.
Harrowing: Sandy soils are the easiest to harrow.

Concept hierarchy

An immediate usefull result from the literature analysis (chapter 3) is the concept hierarchy. It gives a good overview over the important concepts in the subject weed control. And it was used in the implementation in the object oriented shell.

The top part of the hierarchy (fig 3.1) is universal and can be used in all domains, This is the left most part of this table. The attributes is the first part of the table. These are ordered after the concepts to which they are connected.

Attributes	Plant attributes	Species Lifelength Propagation Competition ability Size Dry matter minimum Indicator of lime deficiency Indicator of poor drainage Harrow toleration
Attributes	Weed attributes	Count
Attributes	Crop attributes	Sort Used as Method of cultivation
Attributes	Seed attributes	Germination time Temperature for germination Germination capacity Germination ability
Attributes	Harvested crop attributes	Yield

Attributes	Soil attributes	Type Seed content Manure content Water content Acidity		
Attributes	Sowing attributes	Seed quantity Time Depth Quality of sowing bed Row space		
Attributes	Climate attributes	Temperature Humidity		
Object	Plant	Grown plant	Weed	
Object	Plant	Grown plant	Crop	
Object	Plant	Seed		
Object	Plant	Vegetative reproduction organs	Underground reproduction organs	
Thing	Soil			
Thing	Harvested crop			
Thing	Reserves			
Situation	Incident	Action	Sowing	
Situation	Incident	Action	Soil preparation	Manuring Plowing Draining Lime
Situation	Incident	Action	Weed control	Harrowing Hoeing Stubble treating Row cleaning Flame treatment
Situation	Incident	Action	Crop rotation	
Situation	Event	Climate		

Calculations of reduction in yield

In WEEDOF CE values is used for calculating predicted reductions in yield. CE values expresses the count of crop plants one weed plants can replace. There is a CE value for winter crops and one for spring crops. According to the crop species the CE values are adjusted with a factor.

CE values for weeds

Weed	Winter crops	Spring crops
Sinapis/Brassica	0.65	0.7
Viola arvensis	0.10	0.06
Stellaria media	0.50	0.10
Chenopodium album	0.10	0.15
Lamium spp.	0.30	0.01
Galeopsis spp.	0.30	0.30
Polygonum spp.	0.08	0.12
Galium aparine	0.90	0.11
Tripleurospermum inodorum	0.50	0.11
Chrysanthemum segetum	0.31	0.30
Papaver rhoeas	0.70	0.10
Poa annua	0.01	0.01
Veronica persica	0.5	0.05
Myosotis arvensis	0.20	0.09
Apera spica-venti	0.35	-
Alopecurus myosuroides	0.35	-
Elymus repens	0.80	0.60
Cirsium arvense	5.00	3.00
Sonchus spp.	5.00	3.00
Avena fatua	0.8	-

Factors for multiplying with CE for cultures

Winter crops	factor
Triticum aestivum	1
Secale cereale	0.8
Hordeum vulgare	0.9

Spring crops	factor
Hordeum vulgare	1
Triticum aestivum	1.25
Avena sativa	0.9
Pisum sativum	1.5
Vicia faba	1.5

Example
We have an expected count of Stellaria media of 10/m^2 and of Viola arvensis of 15/m^2 in winter

wheat - Triticum aestivum. The CE value is 0.5 for Stellaria media in winter crops, and 0.1 for Viola arvensis. The multiplying factor for winter wheat is 1.

The total CE value is then $1 \times ((10 \times 0.5) + (15 \times 0.1)) = 6.5$
The predicted reduction in yield is then $(100 \times 6.5) / (6.5 + 400) = 1.6\%$

Appendix A5

Model

This appendix contains a META IV specification of a model for crop and weeds on a field. The model should describe growth and development in a specified period. Including the effects of actions as for instance weed control.

The overall function is called *farming* and pictures system state, field information, time and a list of actions on a new state and time:

```
1.0  type farming: State × FieldInformation × Time × Actionlist → State × Time
.1     farming (st, field, t, acl)
.2     ≝
.3          if acl=<> then (st,t)
.4          else
.5               let a1=hd(acl) in
.6               (let st2= simulatePlants(st, field, t, s-time(a1)) in
.7               (let st3 = simulateAction(st2, s-action(a1), field, s-time(a1)) in
.8               farming(st3, field, s-time(a1), tl(acl))
1.9              ))
```

Comments:

3. If the list of actions is empty, the function returns the start state and time.
5. otherwise a1 is the first element in the action list.
6. The help function simulatePlants simulates plant growth and - development from start time to time of a1.
7. simulateAction simulates the effect of the action a1.
8. farming is called again with the rest of the action list as parameter.

The model shall be used to simulate a plan - the actionlist - generated by heuristic rules .

The data in the function are represented by the following domains:

2.0	State	=	seedSituation × plantsituation.
.1	SeedSituation	=	species $\overrightarrow{m}$ seedcontent.
.2	Seedcontent	=	depth $\overrightarrow{m}$ count.
.3	PlantSituation	=	species $\overrightarrow{m}$ populationDevelopment.
.4	PopulationDevelopment	=	developmentalStage $\overrightarrow{m}$ heighttable.
.5	Heighttable	=	height $\overrightarrow{m}$ count.
.6	Species	=	'Vinterrug' | 'Vinterhvede '| ... |'Kløvergræs'|'korsblomstrede'| ... | 'agersennep'.
.7	Count	=	INTG.
.8	DevelopmentalStage	=	'kim' | 'blomstring' | 'frøbærende' ...
.9	Height	=	INTG 'cm'.
.10	Depth	=	'<5 cm'| '>5 cm'.
.11	FieldInformation	=	soilType × stone .
.12	SoilType	=	'svær ler' | 'ler' | 'sandblandet ler' | 'sand' | 'humus'.
.13	Stone	=	'få sten' | 'mange sten'.
.14	Time	=	year × month × day.
.15	Year	=	INTG.
.16	Month	=	INTG.
.17	Day	=	INTG.
.18	Actionlist	=	{action × time } *.
2.19	Action	=	'Såning' | 'Harvning' | 'Radrensning' | 'Flammebehandling' | 'Pløjning' | ... | 'Høst' .

Comments:

0. The state in the system is represented by the combined information in the seed situation and the plantsituation.
1. The seed situation pictures the species in the soil seed content.
2. The soil seed content is a depiction of soil depth onto count.
3. Plant situation is a function of species onto population development, which is
4. a height table for each developmental stage present.
5. The height table is a count of plants for each height.
6. Species are any of the plants relevant in a model to be used in a weed control system. All relevant species has to be filled in.
7. Count is an integer.
8. The stage of development can be small plant, flowering, seed bearing.
9. Height is an integer with the measure cm.
10. Depth is soil depth. It can be under 5 cm or over 5 cm. this is the important zones in soil. Only seeds in the top 5 cm can germinate.
11. Field information is the combine information of soiltype and stone amount.
12. Soiltype is one of the mentioned types.
13. The classification of stone amount is: many or few stones.
14. Time is the combination of year, month and day.

15-17. Year, month and day are all integers.

18. Actionlist is a list of actions with a connected time to each action.
19. Action are any of the possible treatments in growing crops, except chemical.

The model does, for the moment, not include factors which are unaffectable, for instance climate, or soil water, but considers the situation from standard conditions.

Another restriction in the current specification is the time steps for the simulation. The specification gives the time steps as the time between actions in the actionlist. It is very likely that the periods will be to long. A possible way to solve the problem later is to make a new kind of action, no_action, and make sure that simulation periods have a maximum time by inserting no_action in the actionlist.

Specieslist

For the simulation of growth and development we need a list of the species present:

3.0 Specieslist = Species *.

Invariant for specieslist:

4.0 *type inv-Specieslist*: Specieslist × SeedSituation × PlantSituation → BOOL
.1 inv-specieslist (sl, ss, ps)
.2 $\stackrel{def}{=}$
.3 ∀ a ∈ elems(sl): a ∈ dom(ss) ∨ a ∈ dom(ps)

Comments:
3. Expresses that each element in the species list must be a member of the domain of species in the seed situation or member of the domain of species in the plant situation.

Function: simulatePlants

The function simulatePlants should simulate growth and development of the entire population on the field.

5.0 *type simulatePlants*: State × Specieslist × FieldInformation × Time × Time → State
.1 simulatePlants (st, sl, field, t, ta)
.2 $\stackrel{def}{=}$
.3 if sl = <> then st
.4 else
.5 let (seeds1,plants1) = simulatePlant(hd(sl), st, field, t, ta),
(seedsr, plantsr)= simulatePlants(st, tl(sl), field, t, ta)
in
(seeds1 ∪ seedsr , plants1 ∪ plantsr)

Comments:
3. If the specieslist is empty the state is returned.
5. Otherwise growth and development for the first species in the specieslist is simulated by the help function simulatePlant, and simulatePlants is called recursively with the rest of the list.

SimulatePlant simulates growth and development of one species.

Function: simulatePlant

6.0 *type simulatePlant*: Species × State × FieldInformation × Time × Time → State

```
.1      simulatePlant(sp, st, field, t, ta)
.2      def
.3          let (ps,ss) = st in
.4          ( let (ss2(sp), ps2(sp)) = germination( sp, st, field, t, ta) in
.5          (let ps3(sp)= vegetativeReproduction( sp, ps(sp), ps2(sp), field, t, ta) in
.6          (let ps4(sp)=growth( sp, ps, ps3(sp), field, t, ta) in
.7          (let ss3(sp)=soilSeedContent( sp, st, ss2(sp), field, t, ta) in
.8          (ss3, ps4)
.9          ))))
```

Comments:
SimulatePlant uses 4 helpfunctions to simulate growth and development: germination, vegetativeReproduction, growth and soilSeedContent.

4. germination is used to calculate the germination on the basis of the start state.
5. vegetativeReproduction calculates the reproduction on basis of the start state and modifies the plant situation output in 4.
6. growth calculates growth from the start state and modifies plant situation output from 5.
7. soilSeedContent calculates the seed content at time ta, calculating the amount of seed shedding and death in the period. Modifies the seedsituation output from 4.
8. The final output is the seed and plant situation for species sp.

Function germination

```
7.0  type germination: Species × State × FieldInformation × Time × Time →
         SeedContent × PopulationDevelopment
.1       germination ( sp, st, field, t, ta)
.2       def
.3           let (ss, ps) = st in
.4           (let (ss2(sp), germ) = seedCountAdjustment ( sp, ss(sp), field, t, ta) in
.5           (let ps2(sp) = plantCountAdjustment ( sp, ps(sp), germ)) in
.6           (ss2(sp), ps2(sp)) )).
```

Comments:

4. The help function seedCountAdjustment calculates the change in seedsituation for sp after germination, the result is a new seed content for sp and a count of germinated plants.
5. The help function plantCountAdjustment calculates the new count of plants of species sp adjusting populationDevelopment.

Function vegetativeReproduction

8.0 *type VegetativeReproduction*: Species × PopulationDevelopment × Population-Development × FieldInformation × Time × Time → PopulationDevelopment
.1 vegetativeReproduction (sp, pd, pd2, field, t, ta)
.2 $\stackrel{def}{=}$
.3 ?

Comments:
The function shall calculate the vegetative reproduction from start time, time of the year, and possibly also field information and competition between species and plants. The output will be a new populationDevelopment(sp)

Function growth

9.0 *type growth*: Species × PlantSituation × PopulationDevelopment × FieldInformation × Time × Time → PopulationDevelopment
.1 growth (sp, ps, pd, field, t, ta)
.2 $\stackrel{def}{=}$
.3 let p1 = ps(sp) in
.4 ?

Comments:

1. The function shall calculate the growth for the species sp in the period from t to ta. ps is the start state at time t, and ps2 is the state which should be modified and output. The function will probably be realized as a growth curve, for instance a logistic curve. In the function should also be an effect of competition.

Function soilSeedContent

```
10.0 type soilSeedContent: Species × State × SeedContent ×
        FieldInformation × Time × Time → SeedContent
.1      soilSeedContent ( sp, st, sc, pd, field, t, ta)
.2      def=
.3          let s0=(s-seedSituation(st))(sp),
.4          p0=(s-plantSituation(st))(sp) in
.5          ( let ssd= seedDeath ( sp, s0, sc, field, t, ta) in
.6          seedSupply( sp, p0, ssd, t, ta)
.7          )
```

Comments:

1. sc and pd are the seedcontent and population development for the species.
5. The help function seedDeath calculates the death of seeds in soil during the period from t to ta and delivers a new seed content, ssd.
6. seedSupply calculates supply of seeds by seed shedding and calculates the final seed content.

Function seedDeath

```
11.0 type seedDeath: Species × SeedContent × SeedContent × FieldInformation ×
        Time × Time → SeedContent
.1      seedDeath ( sp, s0, sc, field, t, ta)
.2      def=
.3          let d ∈ dom(s0) in
.4          [d → sc(d)-(normseeddeath(sp, t) * s0(d))]
```

Comments:

4. Calculates the count of seeds of species sp which dies in the period on the basis of the start amount in soil. The product of a seed death rate from a table and the original count is subtracted from the count in the end seed content.

Domain normseeddeath

SeedDeath uses a domain normseeddeath:

12.0 Normseeddeath = Species × Time → REAL.

Comments:

Normseeddeath is the normal death rate for the species. This specification probably has to be extended with depth. The seed death rate will the be a function of species, time and soil depth.

Function seedSupply

13.0 *type seedSupply*: Species × PopulationDevelopment × SeedContent
× Time × Time → SeedContent
.1 seedSupply (sp, pd, sc, field, t, ta)
.2 $\stackrel{def}{=}$
.3 ['5 cm' → sc('<5 cm') + seedProduction(sp,pd,t,ta)]

Comments:

3. The count of seeds in the top 5 cm is summed with the seed production from the plants.

Function seedProduction

14.0 *type seedProduction* : Species × PopulationDevelopment × Time × Time → Count
.1 seedProduction(sp, pd, t, ta)
.2 $\stackrel{def}{=}$
.3 if 'frøbærende' ∈ dom(pd) then
.4 SUM (∀ h ∈ dom(populationDevelopment('frøbærende'))
.5 heighttable(h) * seedprodtable(sp))

Comments:

3. If seedbearing plants are present in the start population development, pd,
4. then the count of all seedbearing plants are
5. multiplied with the normal seedproduction of a plant. The normal seed production are in a table which are a new domain.
4. The seed counts from the multiplication are summed.

This function should maybe contain a modifier for seed production in relation to competition

Domain SeedProdTable

The function seedProduction uses a table of normal seed production for a plant, represented with the following domain:

15.0 Seedprodtable = Species $\overrightarrow{m}$ Count.

Function seedCountAdjustment

```
16.0  type seedCountAdjustment: Species × SeedContent × FieldInformation × Time
          × Time → SeedContent × GerminatedPlants
.1        seedCountAdjustment(sp, sc, field, t, ta)
.2        def=
.3             let sd=dom(sc) in
.4             (if sd={} then ([],0)
.5             else
.6             let a∈ sd in,
.7             (let (sc1, g1) = treat( a, sc, field, t, ta),
.8             (scr, gr) = seedCountAdjustment( sd\a, sc, field, t, ta)
.9             in (sc1 ∪ scr, g1+gr) )
```

Comments:

4. if there are no keys in seedcontent the the result is an empty list and 0 for germinated plants.
6. if the set is not empty, one key is chosen,
7. the count of germinated seeds and the adjustment are calculated by the help function treat.
8. SeedCountAdjustment are called with the rest of the set.

Function treat

17.0 *treat*: Species × SeedContent × FieldInformation × Time × Time →
SeedContent × GerminatedPlants
.1 treat(sc, field, t, ta)
.2 $\stackrel{def}{=}$
.3 let [d → c])in
.4 (let g= germin(sp,d,t) * sc(d) in
.5 ([d → sc(d)-g], g))

Comments:

4. Treat uses the table germin to find the germination
   The product of the table value and the start soil content is the germinated amount.
5. subtraction of germinated plants from the seed content gives the new seed content.

Domain Germin

18.0 Germin = (species × depth × time) → REAL.

Comments:
Germin is a table of the amount of germination on basis of species, soil depth and time of the year.

Domain GerminatedPlants

19.0 GerminatedPlants = INTG.

Function plantCountAdjustment

```
20.0 type plantCountAdjustment : Species × PopulationDevelopment
        × GerminatedPlants → PopulationDevelopment
.1      plantCountAdjustment(sp, pd, germ)
.2      ≝
.3      cases 'kim'
.4      ( ∉ dom(pd) →
.5            pd'= ['kim' → [spgermheight(sp) → germ]] in
.6            pd+pd'
.7      ∈ dom(pd)→
.8            let heightt=pd('kim') in
.9            ((if spgermheight(sp) ∈ dom(heightt) then
.10           heightt'(spgermheight(sp)) = heightt(spgermheight(sp)) + germ
.11           else heightt'(spgermheight(sp))= germ) in
.12           Pd'= ['kim' → heightt + heightt'] in
.13           pd+pd') )
```

Comments:

3. There are two cases
4. If 'kim' is not a key in the population development table,
5. the key is created.
6. The new entrance is summed with the prior development table.
7. If 'kim' is already a key:
9. The normal height for new germinated plants are looked up in the table spgermheight. If the height is a key in the heighttable for 'kim'
10. then the count in the heighttable are summed with the count of germinated plants.
11. else a new entrance are created.
13. The new entrance is summed with the prior table.

Domain Spgermheight

21.0 Spgermheight = Species → Height.

Comments:
Spgermheight is a table which pictures species in a height for newly germinated plants.

Function simulateAction

22.0 *type simulateAction*: State × Action × Specieslist × FieldInformation × Time → State
.1 simulateAction(St, ac, sl, field, ta)
.2 $\stackrel{def}{=}$
.3 let (ss , ps) = st in
.4 (let ss' = effectOnSoilSeeds (ss, field, sl, ac, ta),
.5 ps'= effectOnPlants (ps, field, sl, ac, ta) in
.6 (ss', ps'))

Comments:
.4 The help function effectOnSoilSeeds calculates the effect on the soil seed distribution.
.5 The help function effectOnPlants calculates the effect on the plants.

Function effectOnSoilSeeds

23.0 *type effectOnSoilSeeds*: SeedSituation × FieldInformation × Specieslist ×
Action × Time → SeedSituation
.1 effectOnSoilSeeds (ss, field, sl, ac, ta)
.2 $\stackrel{def}{=}$
.3 ∀ sp ∈ sl :
.4 (let sc=ss(sp) in
.5 ((down, up) = movement(sc,ac,sp) in
.6 ∀ d in dom(sc):
.7 case d:
.8 '<5 cm' →
.9 sc'=[d → sc(d) - down + up] in
.10 ss + [sp → sc+sc']
.11 '>5 cm' →
.12 sc'=[d → sc(d) + down - up] in
.13 ss + [sp → sc+sc']))

Comments:

3. For all the plants in the species list:
4. The helpfunction movement calculates the movement of seeds from the toplayer down, and from the bottom layer up.
9. From the count in the toplayer, the movement down is subtracted, up is summed .
12. From the bottomlayer count, the movement down is summed, up is subtracted.
10,13. A new seed situation is created by summing the two seedcontents, overwriting the old values.

Function movement

```
24.0 type movement: SeedContent × Action × Species → Count × Count
.1      movement( sc, ac, sp)
.2      ≝
.3          ∀ d in dom(sc)
.4          (if d='<5 c,' then down=sc(d) * actionMovement( ac, d)
.5          else up=sc(d) * actionMovement( ac, d) in
.6          (down,up)
```

Comments:

3. For the depths in the seed content table, '<5 cm' and '>5 cm'
4. The downward movement is calculated as the product of the seed content in the toplayer and a table value.
5. The upward movement is calculated as the seed content in the bottom layer and a table value.

Function effectOnPlants

```
25.0 type effectOnPlants: PlantSituation × FieldInformation × Specieslist
        × Action × Time → PlantSituation
.1      effectOnPlants ( ps, field, sl, ac, ta)
.2      ≝
.3          ∀ p ∈ sl:
.4       let ds= actionDevelop ( p, ps(p), field, ac, ta) in
.5          ps + [p → ds]
```

Comments:

3. For all the species in the specieslist
4. The action effect are calculated by the help function actiondevelop
5. and the plant situations are summed.

Function actionDevelop

26.0 *actionDevelop*: Species × PopulationDevelopment × Fieldinformation × Action × Time → PopulationDevelopment

```
.1    actionDevelop (sp, pd, field, ac, t)
.2    ≝
.3        ∀ d ∈ dom(pd):
.4            let ht = actionheight(sp, pd(d), field, ac,t) in
.5            pd + [d → ht]
```

Comments:

3. For all the development stages in the population development
4. the action effect are calculated by actionheight,
5. and the populationDevelopments summed.

Function actionHeight

26.0 *actionHeight*: Species × Heighttable × Fieldinformation × Action × Time → Heighttable

```
.1    actionHeight(sp, ht, field, ac, t)
.2    ≝
.3    ∀ h ∈ dom(ht):
.4        ht'= [h → ht(h) * actioneffect(h, ac, sp)] in
.5        ht + ht'
```

Comments:

3. For all the heights in the heighttable
4. the effect of the action are calculated by multiplying a percent effect from the table actioneffect with the count.
5. The new heighttable entrance is summed with the old writing over the old values.

Domain ActionEffect

27.0 ActionEffect = Height × Action × Species → REAL.

Comments:
Actioneffect is a table which pictures plant heigt, action, and species on a procentual effect.

Glossary

AI Abbreviation of artificial intelligence - a research field in computer science. The researchers examine possibilities to construct systems which show properties normally connected to intelligence. For instance ability to solve problems, communicating in natural language, ability to learn and interpreting visual information.

Backward chaining The opposite of forward chaining. In backward chaining the inference engine finds a rule whose conclusion is the same as the hypothesis. Then it checks if the premise of the rule is true. Either by checking the knowledge base, asking the user or by finding other rules which conclusions are the same as the premise proposition. The search continues until all premises for the hypothesis are true or there is no more rules.

Compound variable domain Domain which are put together from two domains. This could be for instance a number from the Integers and a string 'cm'.

Conceptualization Finding the concepts, relations and information-flow characteristivs needed to solve problems in the domain.

Concept A descriptive schema for a class of things.

Deep knowledge All the knowledge about components or concepts in a domain and the relations that links them together, including cause-effect relations.

Domain The portion of a humans problem situation that are chosen to be studied and included in the computer system.

Domain for variables
In programming languages data types specifies a collection of data objects with a certain structure. In META IV, which is an abstract notation used on design level, the word domain is used instead of data type.

Dry matter minimum The time during germination where the plant has used dry matter from the reserves to germinate and now are big enough to start producing dry matter reserves.

EGERIA A software tool designed for the construction of expert systems

Expert system shell Tool for developing expert systems. An expert system shell contains an inference engine and a user interface, The knowledge engineer then only has to program the knowledge base.

Formalization Mapping concepts and relations found during conceptualization into a formal representation suggested by some expert system building tool or language.

Forward chaining The opposite of backward chaining. The inference engine finds rules which premises are true according to data, and can then infer the conclusions. Then other rules may have premises which are true. In that way the rules are chained in a forward direction. The search stops when there is no more rules or a hypothesis has been confirmed.

Harvest index The grain/straw part of the yield.

Inference The process of deriving a conclusion from a set of rules by applying some deduction rules.

Inference engine The part of an expert system which draws inferences and controls the reasoning process.

Knowledge acquisition This is the extraction and transformation of knowledge from some knowledge source, especially experts, to a program.

Knowledge base The part of an expert system where the knowledge of the domain is expressed.

Knowledge elicitation Knowledge collection

from the expert. Knowledge elicitation is a part of the Knowledge acquisition.

Knowledge representation Writing the knowledge in a way which can be understood by a machine.

Object In object oriented programming concepts in the domain is represented by objects. Objects is described by attributes and operations, and can inherit attributes and operations from each other.

Surface knowledge A selected part of the deep knowledge needed for problem solving in a domain - also including heuristics about the domain and the problem solving.

Windowed interface An interface to a computer program where the information and the questions all go through windows.

Afdelinger under Statens Planteavlsforsøg

Direktionen

Direktionssekretariatet, Skovbrynet 18, 2800 Lyngby	45 93 09 99
Afdeling for Biometri og Informatik, Lottenborgvej 24, 2800 Lyngby	45 93 09 99

Landbrugscentret

Centerledelse, Fagligt Sekretariat, Forskningscenter Foulum, Postbox 23, 8830 Tjele	86 65 25 00
Afdeling for Grovfoder og Kartofler, Forskningscenter Foulum, Postbox 21, 8830 Tjele	86 65 25 00
Afdeling for Industriplanter og Frøavl, Ledreborg Allé 100, 4000 Roskilde	42 36 18 11
Afdeling for Sortsafprøvning, Teglværksvej 10, Tystofte, 4230 Skælskør	53 59 61 41
Afdeling for Kulturteknik, Flensborgvej 22, Jyndevad, 6360 Tinglev	74 64 83 16
Afdeling for Jordbiologi og -kemi, Lottenborgvej 24, 2800 Lyngby	45 93 09 99
Afdeling for Planteernæring og -fysiologi, Vejenvej 55, Askov, 6600 Vejen	75 36 02 77
Afdeling for Jordbrugsmeteorologi, Forskningscenter Foulum, Postbox 25, 8830 Tjele	86 65 25 00
Afdeling for Arealdata og Kortlægning, Enghavevej 2, 7100 Vejle	75 83 23 44
Borris Forsøgsstation, Vestergade 46, 6900 Skjern	97 36 62 33
Lundgård Forsøgsstation, Kongeåvej 90, 6600 Vejen	75 36 01 33
Rønhave Forsøgsstation, Hestehave 20, 6400 Sønderborg	74 42 38 97
Silstrup Forsøgsstation, Oddesundvej 65, 7700 Thisted	97 92 15 88
Tylstrup Forsøgsstation, Forsøgsvej 30, 9382 Tylstrup	98 26 13 99
Ødum Forsøgsstation, Amdrupvej 22, 8370 Hadsten	86 98 92 44
Laboratoriet for Biavl, Lyngby, Skovbrynet 18, 2800 Lyngby	45 93 09 99
Laboratoriet for Biavl, Roskilde, Ledreborg Allé 100, 4000 Roskilde	42 36 18 11

Havebrugscentret

Centerledelse, Fagligt Sekretariat, Kirstinebjergvej 10, 5792 Årslev	65 99 17 66
Afdeling for Grønsager, Kirstinebjergvej 6, 5792 Årslev	65 99 17 66
Afdeling for Blomsterdyrkning, Kirstinebjergvej 10, 5792 Årslev	65 99 17 66
Afdeling for Frugt og Bær, Kirstinebjergvej 12, 5792 Årslev	65 99 17 66
Afdeling for Planteskoleplanter, Kirstinebjergvej 10, 5792 Årslev	65 99 17 66
Laboratoriet for Forædling og Formering, Kirstinebjergvej 10, 5792 Årslev	65 99 17 66
Laboratoriet for Gartneriteknik, Kirstinebjergvej 10, 5792 Årslev	65 99 17 66
Laboratoriet for Levnedsmiddelforskning, Kirstinebjergvej 12, 5792 Årslev	65 99 17 66

Planteværnscentret

Centerledelse, Fagligt Sekretariat, Lottenborgvej 2, 2800 Lyngby	45 87 25 10
Afdeling for Plantepatologi, Lottenborgvej 2, 2800 Lyngby	45 87 25 10
Afdeling for Jordbrugszoologi, Lottenborgvej 2, 2800 Lyngby	45 87 25 10
Afdeling for Ukrudtsbekæmpelse, Flakkebjerg, 4200 Slagelse	53 58 63 00
Afdeling for Pesticidanalyser og Økotoksikologi, Flakkebjerg, 4200 Slagelse	53 58 63 00
Bioteknologigruppen, Lottenborgvej 2, 2800 Lyngby	45 87 25 10

Centrallaboratoriet

Centrallaboratoriet, Forskningscenter Foulum, Postbox 22, 8830 Tjele	86 65 25 00

Frederiksberg Bogtrykkeri a-s 920631