

AGROECOLOGICAL KNOWLEDGE TOOLKIT FOR WINDOWS: METHODOLOGICAL GUIDELINES, COMPUTER SOFTWARE AND MANUAL FOR AKT5

2001



AKT5

Dixon, H.J., Doores, J.W., Joshi, L., and Sinclair, F.L.

School of Agricultural and Forest Sciences, University of Wales, Bangor.

This manual is based on the original Agroforestry Knowledge Toolkit manual:

Walker, D.H., Sinclair, F.L., Kendon, G., Robertson, D., Muetzelfeldt, R.I., Haggith, M. and Turner, G.S. (1994) *Agroforestry Knowledge Toolkit: methodological guidelines, computer software and manual for AKT1 and AKT2, supporting the use of knowledge-based systems approach in agroforestry research and extension*. School of Agricultural and Forest Sciences, University of Wales, Bangor.

The manual was written by Helga Dixon, Jim Doores, Laxman Joshi and Fergus Sinclair.

Cite as: Dixon, H.J., Doores, J.W., Joshi, L. and Sinclair, F.L. (2001) *Agroecological Knowledge Toolkit For Windows: Methodological Guidelines, Computer Software And Manual For AKT5*. School of Agricultural and Forest Sciences, University of Wales, Bangor, UK.

ISBN: 1 84220 025 9

Acknowledgements: This publication includes outputs from research projects wholly or partly funded by the United Kingdom Department for International Development (DFID) for the benefit of developing countries. The views expressed are not necessarily those of DFID. R6322, R7264, Forestry Research Programme; R7516, Natural Resources Systems Programme.

FOREWORD

Agroforestry practices, particularly traditional practices in the tropics, are characterised by their complexity. Information about the ecology of such practices, and the agricultural, economic and social reasons for adopting them is often qualitative, sparse and uncertain. Development professionals planning research and extension activities need to use such information as effectively as possible, but knowledge sources may be dispersed and their information, though complementary, may not be immediately compatible.

The 'Agroforestry Knowledge Toolkit' methodology and software provide an environment in which you can create knowledge bases about your chosen topics, by collating knowledge from a variety of sources. The formal approach to synthesising, evaluating and reasoning with knowledge about interdisciplinary topics will facilitate the use of that knowledge in planning agroforestry research and extension. This gives a powerful alternative to existing, less formal approaches.

The AKT software provided with this manual, provides:

- a knowledge base structure for storing statements of fact as formal representations.
- a statement interface and a diagram interface for entering knowledge into the knowledge base.
- a set of dialogs for access to the contents of the knowledge base.
- a set of tools for reasoning with the contents of the knowledge base.
- an environment for creating new tools appropriate to user-specific needs.

The methodology associated with AKT is flexible and may be adapted for application in a range of contexts. Furthermore, the AKT software can be customised to meet particular needs. Some approaches to customisation (for example, the creation of customised reasoning tools) are outlined in the manual.

This publication documents the methodology as it stands at the end of the third major cycle of development in July 2001. For details of subsequent development, or if you are interested in adapting the methodology for implementation at an institutional level, please contact:

Dr Fergus Sinclair
School of Agricultural and Forest Sciences
University of Wales,
Bangor,
Gwynedd, LL57 2UW
UK

Email: akt5@bangor.ac.uk

HOW TO USE THIS MANUAL

INSTALLATION: These are the instructions for installing the AKT program on to your computer.

METHODOLOGY: The first six chapters of this manual outline the methodology for knowledge base creation and use. It is important that you read these before beginning to construct knowledge bases for practical use.

KNOWLEDGE ACQUISITION: Chapters 7 and 8 demonstrate how to install AKT and give detailed guidelines on how to use it to create your own knowledge base using both the statement and the diagram interfaces.

USING KNOWLEDGE: Chapters 9 and 10 introduce the reasoning tools and explain how to create your own reasoning tools for exploring and evaluating the knowledge base

INCORPORATING PICTURES AND DIAGRAMS: Chapter 11 describes how to incorporate diagrams and pictures into the knowledge base

HELP: Chapter 12 answers the most common queries.

TUTORIAL: Chapter 13 and 14 provides basic tutorials for using AKT, chapter 13 gives you a quick guided tour around the knowledge base 'Atwima' and chapter 14 is an exercise in building an example knowledge base.

APPENDICES AND REFERENCES: These give you further information about knowledge base creation and references to other works on indigenous ecological knowledge

KNOWLEDGE BASES: Three knowledge bases are supplied with the manual, 'soil', 'treefodd' and 'atwima'. 'Soil' was created as a small example knowledge base, 'treefodd' and 'atwima' are genuine knowledge bases, the first capturing knowledge of farmers in the midhills of Nepal and the second capturing knowledge of farmers in the Atwima district of Ghana.

HOW TO INSTALL AKT5



IMPORTANT

AKT5 is designed for 32 bit systems only. It will run on Win95, Win98, Windows NT4 and Windows 2000

Installing AKT5 from the CD ROM

1. Insert the AKT5 CD Rom into your CD ROM drive.
2. Double click on the file AKT5 Install.exe
3. Follow the instructions displayed on the screen. If you wish to install AKT5 plus the User Manual, the example Knowledge bases and a guide to the use of the knowledge bases then choose the **Typical** installation option. If you just want to install the AKT5 application then choose the **Compact** option. If you select the **Custom** option you can make your own choice of which items to install.

Installing AKT5 from the Website

1. Create a new folder on your hard drive (usually C:) and call it **AKT 5** or something similar
2. Go to www.bangor.ac.uk/afforum
3. Select AKT followed by **Downloads**, then **Program Files**.
4. **After** pressing **Program Files** a **Save As** dialog box appears. Save the file AKT5 Install.exe to your new folder.
5. To install the program, double click on AKT5 Install.exe.

Note

When installing AKT5 from the website, the AKT5 Install.exe will not give you a choice of the installation type. It will only download the compact version to avoid excessive download time. The documentation and knowledge bases can then be downloaded later, as required.

Running AKT5

After installing AKT5 and its associated documentation, you can then run the application by selecting **Start / Programs / AKT5**

Now you are ready to begin. Go to chapter 7.

Knowledge Bases

A number of knowledge bases are supplied with the programme, including soil, treefodd and atwima mentioned in the manual. It is recommended that a separate folder be used for knowledge bases. To load a knowledge base and get started, turn to Chapter 7 of the User Manual.

For those of you who have downloaded the manual from the web, all the knowledge bases can be downloaded in the same manner – go to <http://www.bangor.ac.uk/afforum> then select **AKT**, then **Downloads**, then **Knowledge base**. Unzip the files and save them on your computer.

TABLE OF CONTENTS

FOREWORD.....	i
HOW TO USE THIS MANUAL.....	ii
HOW TO INSTALL AKT 5	iii
TABLE OF CONTENTS.....	iv
CHAPTER ONE – OVERVIEW.....	1
1.1 BACKGROUND OF OBJECTIVES.....	1
1.2 WHAT DO WE MEAN BY ‘KNOWLEDGE’.....	2
1.3 KNOWLEDGE BASE CREATION.....	2
1.4 KNOWLEDGE BASE STRUCTURE.....	3
1.4.1 STATEMENT INTERFACE.....	4
1.4.2 DIAGRAM INTERFACE.....	5
1.5 REASONING WITH KNOWLEDGE BASES.....	6
1.6 APPLICATION OF KNOWLEDGE-BASED SYSTEMS APPROACH TO AGROFORESTRY RESEARCH AND EXTENSION.....	6
1.7 THE UTILITY OF THE APPROACH.....	7
1.7.1 THE EXPRESSIVENESS OF THE AKT APPROACH AND IMPLEMENTATION.....	7
1.7.2 THE KNOWLEDGE BASE AS A RESOURCE.....	8
1.7.3 THE UTILITY OF THE REASONING MECHANISMS.....	9
1.7.3.a Property inheritance.....	10
1.7.3.b Causal diagramming.....	10
CHAPTER TWO – KNOWLEDGE ELICITATION.....	11
2.1 DESIGNING A KNOWLEDGE ELICITATION STRATEGY.....	11
2.1.1 A FRAMEWORK FOR DESIGNING A KNOWLEDGE ELICITATION STRATEGY.....	11
2.1.1.a Scoping.....	12
2.1.1.b Definition of The domain.....	12
2.1.1.c Compilation.....	12
2.1.1.d Generalisation.....	13
2.1.2 SAMPLE SIZE.....	13
2.2 CONSTRAINTS UPON KNOWLEDGE ELICITATION.....	13
2.3 INTERVIEW TECHNIQUE.....	14
2.3.1 SOME GOLDEN RULES FOR INTERVIEW TECHNIQUES.....	14
CHAPTER THREE – PREPARATION FOR KNOWLEDGE BASE CREATION.....	17
3.1 PREPARATION.....	17
3.1.1 SPECIFICATION OF OBJECTIVES.....	17
3.1.2 BOUNDARIES OF THE KNOWLEDGE BASE.....	17
CHAPTER FOUR – KNOWLEDGE REPRESENTATION.....	19
4.1 ABSTRACTING KNOWLEDGE.....	19
4.1.1 BASIC CONCEPTS.....	19
4.1.2 UNITARY STATEMENTS AS BASIC UNITS OF KNOWLEDGE.....	19
4.1.3 RECORDING THE CONTEXT OF A STATEMENT.....	19
4.1.3.a Conditions.....	20
4.1.3.b Source.....	20
4.2 FORMAL REPRESENTATION.....	21
4.2.1 INTRODUCTION.....	21
4.2.2 THE ELEMENTS OF FORMAL REPRESENTATION.....	23
4.2.2.a Reserved terms.....	24
4.2.3 FORMAL REPRESENTATION OF SINGLE UNITARY STATEMENTS..	24
4.2.3.a Re-Evaluation of the unitary statement.....	24
4.2.3.b Identification of the elements in the statement.....	25
4.2.3.c Identification of the statement structural type.....	26
4.2.3.d Creation of the formal statement.....	26
4.2.4 STATEMENT TYPES.....	26
4.2.4.a Attribute value statements.....	26
4.2.4.b Causal statements.....	27
4.2.4.c Comparison statements.....	28
4.2.4.d (User Defined) link statements.....	29
4.2.4.e Representation Of Conditions.....	29

4.3	FORMAL TERMS SPECIFICATION.....	29
4.4	DIAGRAM BASED REPRESENTATION.....	30
4.4.1	INTRODUCTION.....	30
4.4.2	NODES.....	31
4.4.2.a	Classification of nodes.....	31
4.4.2.b	Labelling of nodes.....	31
4.4.3	LINKS.....	31
4.4.3.a	Attaching information to links.....	31
4.4.3.b	Stating the meaning of links.....	32
4.4.3.c	Linguistic correspondence of linked pairs of nodes.....	32
4.4.3.d	Labelling of links.....	33
4.4.4	SUBSETS OF DIAGRAMS.....	33
4.5	CREATING KNOWLEDGE BASES THROUGH THE COMBINATION OF EXISTING KNOWLEDGE BASES.....	34
	CHAPTER FIVE – KNOWLEDGE BASE MANAGEMENT	37
5.1	INTRODUCTION.....	37
5.2	EVALUATING INDIVIDUAL UNITARY STATEMENTS.....	39
5.2.1	VALIDITY OF REPRESENTATION.....	39
5.2.2	RELEVANCE AND UTILITY.....	39
5.2.3	AMBIGUITY.....	40
5.2.3.a	Complete specification of meaning.....	40
5.2.3.b	Context of application.....	41
5.2.3.c	Precise use of terms.....	41
5.2.3.d	Intrinsic ambiguity.....	41
5.3	EVALUATING SETS OF UNITARY STATEMENTS.....	41
5.3.1	REPETITION.....	41
5.3.1.a	Strict repetition.....	42
5.3.1.b	Deducible repetition and the use of hierarchies in compacting the knowledge base.....	42
5.3.1.c	The use of hierarchies in compacting the knowledge base.....	42
5.3.2	CONTRADICTION.....	44
5.3.3	COMPLETENESS.....	45
5.3.4	CONSISTENCY AND PRECISION IN THE USE OF TERMS.....	45
5.4	EVALUATING HIERARCHICAL STRUCTURES.....	45
5.4.1	FORMAL TERMS.....	45
5.4.2	RELATIONSHIPS BETWEEN OBJECTS.....	45
5.4.3	DEFINITION OF FORMAL TERMS.....	46
	CHAPTER SIX – KNOWLEDGE BASE ANALYSIS.....	47
6.1	EVALUATION OF THE REPRESENTATIVENESS OF A KNOWLEDGE BASE....	47
6.1.1	DEFINITIONS OF REPRESENTATIVENESS.....	47
6.1.2	VALIDITY OF ABSTRACTION.....	47
6.1.3	REPRESENTATION OF THE KNOWLEDGE HELD BY THE COMMUNITY.....	47
6.2	TWO METHODS OF TESTING REPRESENTATIVENESS.....	48
	CHAPTER SEVEN – SOFTWARE MANUAL FOR AKT.....	51
7.1	INTRODUCTION.....	51
7.2	GETTING STARTED.....	51
7.3	STARTING WORK ON A KNOWLEDGE BASE.....	53
7.3.1	STARTING ON A NEW KNOWLEDGE BASE.....	53
7.3.2	OPENING AN EXISTING KNOWLEDGE BASE.....	53
7.4	ENTERING KNOWLEDGE THROUGH THE STATEMENT INTERFACE.....	54
7.4.1	ENTERING SOURCE INFORMATION.....	55
7.4.1.a	New sources.....	55
7.4.1.b	Sources already entered.....	56
7.4.2	ENTERING A NEW STATEMENT.....	57
7.4.3	EDITING A STATEMENT.....	58
7.4.3.a	Adding definitions to formal terms.....	60
7.4.3.b	Appending or detaching additional sources.....	61
7.4.3.c	Appending memos.....	62
7.4.4	DELETING A STATEMENT.....	63

7.4.5	DIAGRAMMATIC REPRESENTATION.....	63
7.4.6	SORTING STATEMENTS IN THE STATEMENT CARD	64
7.4.7	INVERSE STATEMENTS AND NUMBERING.....	64
7.5	OBJECT HIERARCHIES.....	64
7.5.1	CREATING OBJECT HIERARCHIES.....	64
7.5.2	VIEWING OBJECT HIERARCHIES.....	65
7.5.3	BUILDING / EDITING OBJECT HIERARCHIES.....	67
7.5.3.a	Adding objects.....	67
7.5.3.b	Detaching objects.....	69
7.5.3.c	Moving objects within a hierarchy.....	70
7.5.3.d	Copying objects between hierarchies.....	72
7.6	SOURCES.....	72
7.7	FORMAL TERMS.....	74
7.7.1	VIEWING FORMAL TERMS.....	74
7.7.2	ADDING FORMAL TERMS.....	75
7.7.3	DELETING FORMAL TERMS.....	76
7.8	SYNONYMS.....	76
7.9	MEMOS.....	77
7.10	SELECTING SUBSETS OF THE KNOWLEDGE BASE.....	79
7.10.1	THE BOOLEAN SEARCH.....	79
7.10.1.a	The use of 'And' and 'Or'	80
7.10.1.b	The use of brackets.....	80
7.10.1.c	Studying the 'Search Results'.....	80
7.10.2	USING THE BOOLEAN SEARCH ON OBJECT HIERARCHIES.....	83
7.11	TOPICS.....	84
7.11.1	CREATING A TOPIC.....	84
7.11.2	MANAGING TOPICS.....	84
7.11.3	USING TOPICS.....	85
7.11.4	CREATING A NEW KNOWLEDGE BASE OUT OF A TOPIC.....	86
7.12	TOPIC HIERARCHIES.....	87
7.12.1	VIEWING TOPIC HIERARCHIES.....	87
7.12.2	CREATING TOPIC HIERARCHIES.....	88
7.12.3	APPENDING TOPICS TO A TOPIC HIERARCHY.....	89
7.12.4	CREATING A NEW KNOWLEDGE BASE OUT OF A TOPIC HIERARCHY.....	90
7.13	PRINTING AND SAVING SEARCH RESULTS.....	91
7.13.1	PRINTING IN LANDSCAPE.....	92
7.14	SAVING A KNOWLEDGE BASE AND CHANGING ITS NAME.....	92
7.15	MOVING FROM ONE KNOWLEDGE BASE TO ANOTHER.....	93
	CHAPTER EIGHT – THE DIAGRAM INTERFACE.....	95
8.1	INTRODUCTION.....	95
8.2	VIEWING A DIAGRAM.....	95
8.2.1	DIFFERENT NODES.....	96
8.2.2	DIFFERENT LINKS.....	97
8.2.3	THE ZOOM.....	97
8.2.3.a	Zooming In And Out.....	97
8.2.3.b	Centre Zoom.....	98
8.2.4	LABELLING.....	98
8.2.4.a	'Causes1way', 'Causes2way' labelling.....	98
8.2.4.b	Link labelling.....	99
8.2.4.c	Showing only causal or only link statements.....	99
8.2.4.d	Statement labels.....	99
8.2.4.e	Hiding labels.....	100
8.2.5	HIDING NODES AND LINKS.....	101
8.2.6	REFRESHING A DIAGRAM.....	101
8.3	CREATING SUB-DIAGRAMS.....	102
8.3.1	SHOW PATHS.....	102
8.3.2	NAVIGATING.....	103
8.3.2.a	Navigating via the 'Statements' list and via 'Search Results'.....	106
8.3.2.b	Making a mistake when navigating.....	106

8.3.3	CAUSES / EFFECTS DIAGRAMS.....	106
8.3.3.a	Causes / Effects diagrams via the 'statements' or 'Search Results' dialog boxes.....	107
8.4	VIEWING DIAGRAM STATEMENTS.....	107
8.4.1	STATEMENTS FOR THE COMPLETE DIAGRAM.....	107
8.4.2	STATEMENTS FOR A SUB-DIAGRAM.....	109
8.5	MOVING BETWEEN DIAGRAMS.....	109
8.6	SAVING DIAGRAMS.....	109
8.6.1	LABELLING DIAGRAMS.....	110
8.6.2	MEMO FIELD FOR EACH DIAGRAM.....	110
8.6.3	COPYING DIAGRAMS.....	111
8.6.4	SAVING DIAGRAMS AS A SEPARATE KNOWLEDGE BASE.....	111
8.7	DELETING DIAGRAMS.....	111
8.8	CREATING NEW DIAGRAMS.....	111
8.8.1	CREATING A DIAGRAM.....	112
8.9	EDITING/DELETING A DIAGRAM STATEMENT.....	116
8.10	ALTERNATIVE ROUTE TO EDITING DIAGRAMS.....	117
8.11	PRINTING DIAGRAMS.....	118
	CHAPTER NINE – REASONING WITH AKT TOOLS.....	119
9.1	INTRODUCTION.....	119
9.1.1	WHAT IS A TOOL?	119
9.1.2	WHAT DO THE TOOLS DO?.....	119
9.1.2.a	What Is A Primitive.....	119
9.1.2.b	What Is A Control Structure.....	120
9.1.2.c	What Is A Tool?.....	120
9.2	WORKING WITH TOOLS.....	120
9.2.1	PRIMITIVES.....	120
9.2.1.a	Opening primitives.....	120
9.2.1.b	Different categories of primitives.....	121
9.2.1.c	The 'Details' box.....	121
9.2.2	CONTROL STRUCTURES.....	124
9.3	SYSTEMS TOOLS.....	125
9.3.1	SOME EXAMPLES OF SYSTEMS TOOLS.....	126
	CHAPTER TEN – CREATING YOUR OWN TOOLS.....	129
10.1	INTRODUCTION.....	129
10.2	TOOL FILES.....	129
10.2.1	CREATING A NEW TOOL FILE.....	129
10.2.2	OPENING A PREVIOUSLY CREATED TOOL FILE.....	129
10.2.3	KEEPING TRACK OF TOOL FILES.....	129
10.2.4	SAVING TOOL FILES.....	130
10.2.5	CLOSING TOOL FILES.....	130
10.3	AN EXAMPLE OF A USER DEFINED TOOL.....	131
10.4	CREATING YOUR OWN TOOL.....	135
10.4.1	INCORPORATING EXISTING TOOLS/PRIMITIVES WITHIN A NEW TOOL DEFINITION.....	136
10.4.2	TESTING THE SYNTAX OF A NEW TOOL.....	136
10.4.3	AN EXAMPLE OF CREATING A TOOL.....	136
10.4.4	TRACING AND TESTING A NEW TOOL.....	141
10.4.4.a	The Trace button.....	141
10.4.4.b	The Test button.....	143
10.4.5	COMPLETING THE TOOL.....	144
10.5	EDITING TOOLS.....	144
10.6	DIRECTING TOOL OUTPUT TO A FILE.....	144
	CHAPTER ELEVEN – INCORPORATING PICTURES AND DIAGRAMS INTO THE KNOWLEDGE BASE.....	145
	CHAPTER TWELVE – THE HELP FACILITY.....	147
12.1	THE FORMAL GRAMMAR.....	147
12.2	THE TOOL LIST.....	148
12.2.1	SAVING 'TOOL INFORMATION' AND PRINTING.....	148
12.3	SOME POPULAR MISTAKES AND DIFFICULTIES.....	149

12.4 ABOUT AKT.....	150
CHAPTER THIRTEEN – A QUICK SIGHTSEEING TOUR AROUND AKT5.....	151
CHAPTER FOURTEEN – TUTORIAL IN CREATING A SIMPLE KNOWLEDGE	155
BASE.....	
14.1 OPENING AKT.....	155
14.2 KNOWLEDGE FROM THE SOURCE.....	155
14.3 ENTERING KNOWLEDGE INTO THE KNOWLEDGE BASE.....	157
14.3.1 ENTERING KNOWLEDGE THROUGH THE DIAGRAM INTERFACE...	157
14.3.1.a Entering The Information Source.....	158
14.3.2 ENTERING KNOWLEDGE THROUGH THE STATEMENT CARD.....	160
14.3.3 PLAYING WITH THE KNOWLEDGE BASE.....	161
APPENDIX 1 – GLOSSARY OF TERMS	167
BIBLIOGRAPHY.....	169

CHAPTER ONE - OVERVIEW

This chapter is an introduction to using a knowledge based systems (KBS) approach to support decision making when planning agroforestry research and extension.

An environment is provided that helps the user to store and access what is known about interdisciplinary topics such as agroforestry as an appropriate starting point for planning research and extension work. The knowledge is obtained by talking to people and consulting literature. We refer to these people and documents as the sources of the knowledge. A store, called a knowledge base, of explicitly recorded knowledge statements or facts is developed. When a knowledge base has been created on computer, there is an explicit and accessible record of the knowledge that can be used later to help in making decisions for and during development of research and development programmes. Unlike, many existing expert systems, the KBS approach is not intended to provide definitive or prescriptive answers to questions but to ensure that decisions are based upon consideration of relevant information.

The approach comprises a methodology for acquiring knowledge and storing this explicitly recorded knowledge and an associated computer software for creation and use of knowledge bases. The toolkit is called AKT, which stands for **A**groecological **K**nowledge **T**oolkit and can be used to provide various levels of support to suit different needs in knowledge analysis and decision making. Although the methodology was initially developed for agroforestry domain, it can be applied equally well in other disciplines.

This first part of the manual provides an overview of the approach, and of the thinking behind it. The manual as a whole provides guidelines on how to apply this approach when planning agroforestry research and extension.

1.1 BACKGROUND AND OBJECTIVES

Agroforestry either involves farmers growing trees or shrubs in various productive or environmentally protective niches on their farms or the integration of agricultural activities in forests. As such, agroforestry practices have multiple objectives and components and are characterised by their complexity. In recent years agroforestry has received considerable attention from people working to foster sustainable and equitable land use in the developing world.

Because agroforestry practices are generally complex, effective decision making in research and extension depends upon making effective use of all available knowledge. Increasingly, development professionals recognise the value of augmenting scientific and professional understanding with knowledge held by local people (Brokensha *et al.*, 1980, Warren *et al.*, 1995, Sinclair and Walker, 1999). This knowledge is of particular interest, but is often incomplete or contentious – and different knowledge sources, though complementary, may not be immediately compatible or comparable. Moreover, much of the information about the ecology of agroforestry practices is qualitative and may include observational information (such as qualitative correlations), or be descriptive. Precise environmental data and quantitative models are rarely available.

In order to combine local, scientific and professional knowledge, effective mechanisms are needed for accessing, recording, evaluating and synthesising knowledge on specified topics from these sources. Existing mechanisms for doing this, as they are currently applied in research and development institutes, are often inadequate (Walker *et al.*, 1997).

The methodology outlined in this manual has been designed to allow the evaluation and use of complex, qualitative information about agroforestry practices. It has been developed with a particular emphasis on the local knowledge of farming communities in developing countries. It provides an approach to decision-support in planning agroforestry research and extension activities, which is appropriate to the nature of existing knowledge. The AKT software provides an environment for knowledge acquisition in order to create knowledge bases from a range of sources. Database functions and a graphical user interface allow flexible exploration, retrieval, and evaluation of the knowledge. More powerful means of retrieving information from

knowledge bases by using automated reasoning techniques (widely used in the field of artificial intelligence) are also available. A task language allows more advanced users to make their own reasoning tools appropriate to their particular decision support tasks by customising the existing tools or by writing new tools.

1.2 WHAT DO WE MEAN BY 'KNOWLEDGE'?

To define knowledge is to enter a philosophical minefield but in order to work with AKT we must state explicitly what we mean by 'knowledge' in this context. For the purposes of AKT we define knowledge as *the outcome of the interpretation of data, independent of the interpreter*. Data is a recorded set of observations (which may be quantitative or qualitative) and information is a continuum that has data and knowledge as two extremes (Figure 1.1). Information, data and knowledge are distinct from understanding. Understanding is the outcome, specific to the interpreter, of the interpretation of information, data or knowledge.

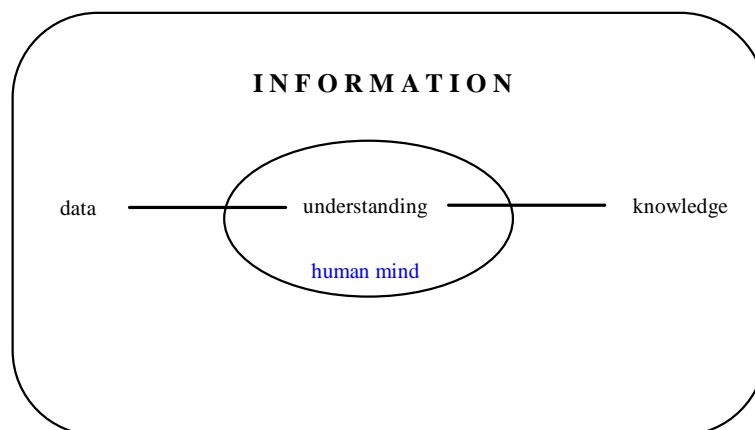


Figure 1.1. Diagrammatic representation of Information as a continuum, with Data and Knowledge as the two extremes.

Knowledge is seen as a central aspect of culture, derived from education and experience, that may be used in conjunction with a certain value system and competing priorities and possibilities, to make decisions.

Local knowledge and indigenous knowledge have often been used interchangeably. However, it is possible to distinguish between the two if 'local knowledge' is used to denote locally derived understanding which is based on experience and observation and 'indigenous knowledge' is used to denote that same understanding but modified by the incorporation of cultural beliefs and values as well.

1.3 KNOWLEDGE BASE CREATION

Creating a knowledge base involves four distinct stages; knowledge elicitation from the appropriate sources, converting the knowledge elicited into simple unambiguous statements, inputting those statements into AKT using formal representation and specifying/defining the formal terms used.

Knowledge elicitation is the process whereby selected informants are encouraged to articulate their knowledge. This is normally done through repeated interviews with farmers and domain experts. Knowledge can also be abstracted from written material.

Creation of unitary statements is the process of extracting knowledge from the text or interview material, and breaking it down into simple statements each containing one 'unit' of knowledge. These 'unitary statements' form the intermediate stage between knowledge articulation and representation.

Formal representation is the process of coding knowledge for input into a computer using a restricted syntax as defined by a formal grammar developed for the purpose. Formal representation results in statements with which you can reason automatically using computer software.

Formal Term (Keyword) specification is the process of identifying and organising key components of knowledge. Formal terms in AKT are either:

objects (e.g. 'pests', 'crops', 'field'),
processes (e.g. 'erosion', 'infiltration', 'growth'),
actions (e.g. 'pruning', 'harvesting', 'planting'),
attributes (e.g. 'rate of erosion', 'pest population size', 'tree height')
values (e.g. '3 m', '10 t/ha', 'high', 'low') or
links (user defined)(e.g. 'eat' as in 'cows eat grass', 'pollinate' as in 'fruit bats pollinate *Parkia biglobosa*')
(e.g. 'eat' as in 'cows eat grass', 'pollinate' as in 'fruit bats pollinate *Parkia biglobosa*')

They are terms which may need to be defined, may have synonyms and, in the case of objects, may be organised in an object hierarchy (e.g. an oak tree is a type of tree and a tree is a type of plant).

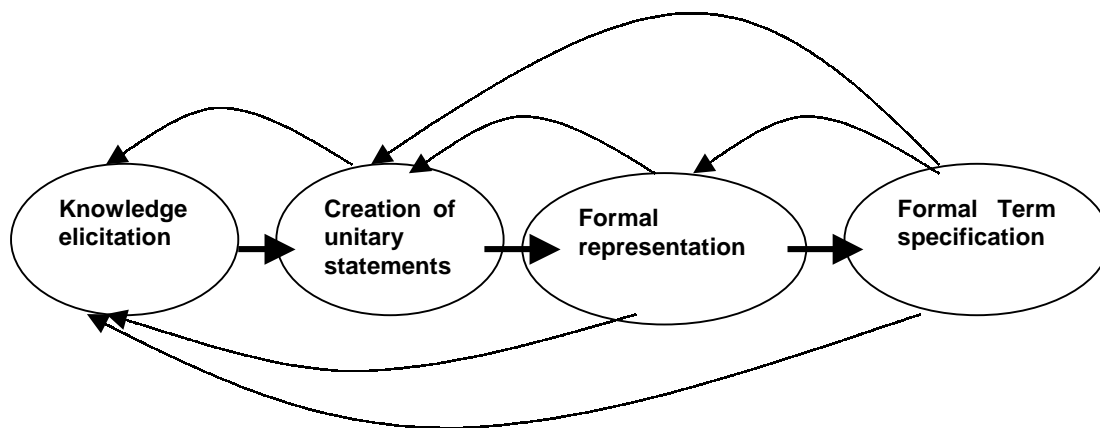


Figure 1.2 *The creation of a knowledge base. There are four principle activities in the creation of a knowledge base, as shown in the diagram. These occur in sequence (bold arrows), but evaluation during the creation of the knowledge base and consequent return to previous activities (fine arrows) means that the process is in fact a series of cycles.*

The process of creating a knowledge base is summarised in Figure 1.2. In principle, the process is linear, but in practice it is iterative in nature. It is important to emphasise that the knowledge base should be evaluated at *each* stage of development. Evaluation of the knowledge base involves assessing the relevance, utility and ambiguity of individual unitary statements. It also includes checks for repetition and contradiction amongst statements. The completeness of the knowledge base, and the consistency and precision in the use of terms, should also be evaluated continuously during the process of building a knowledge base.

1.4 KNOWLEDGE BASE STRUCTURE

The following sections summarise the main features in AKT used in developing and using a knowledge base. These features are described in more detail in relevant chapters that follow. Exploring an example knowledge base will reveal most of these features and is recommended for novice users of AKT before starting to develop their own knowledge bases.

The core content of a knowledge base created within AKT is a set of unitary statements. Unitary statements represent knowledge that is perceived to be true by the source of the knowledge, even if not scientifically verified. Unitary statements are the smallest useful unit of

knowledge, in that they contain knowledge that is useful without reference to other statements; they cannot be broken down any further into useful units of knowledge. Examples of unitary statements are:

Red soils are fertile.

Trampling by sheep increases soil erosion.

Cover crops reduce soil erosion.

Ficus auriculata has large leaves.

Water drip causes splash erosion.

Fodder from Artocarpus lakoocha is more nutritious than fodder from Ficus neriifolia.

1.4.1 STATEMENT INTERFACE

Formal representation of unitary statements in a AKT knowledge base involves using a formal grammar which is designed to allow representation of ecological knowledge. By ecological knowledge we mean information about organisms, the environment and the interactions amongst them including human actions that influence them (see Chapter 4.2 and Table 4.1). The formal grammar comprises four fundamental types of statement: attribute-value statements, causal statements, comparison statements and generic link statements. Each statement may be composed of elements from the formal terms described above: objects, processes, actions, attributes, values and links. For example a formal representation of the statement:

The germination percentage of oak seeds is high if the air temperature is between 15 and 25° C and the moisture content of the seed is between 80 and 90%.

would look like this:

att_value(process(part(oak, seeds), germination), percentage, high)

IF

att_value(atmosphere, temperature, range('15degreesC', '25degreesC')) and att_value(seed, moisture_content, range('80%', '90%'))¹

Each statement is tagged with its source(s), giving details of the literature or interview from which was derived. Most statements are only valid in certain conditions. The general format of a unitary statement is therefore:

assertion IF conditions (sources)

The formal representation of each statement is retained within the knowledge base. Formal terms in the formal statement are identified automatically by a parser², which checks the syntax of the formal statements. Each new formal term is added to the lists of terms, in the categories of object, process, action, attribute, value or (user defined) link. While developing the knowledge base, you can order related objects into object hierarchies. These capture both local and scientific classification of things and allow you to apply order-sorted logic techniques in reasoning (Robertson *et al.*, 1991). By careful management of the content of the lists of formal terms and object hierarchies, you can ensure consistent use of terminology across the knowledge base. Consistent and parsimonious use of terminology greatly improves the subsequent performance of inference mechanisms as well as improving the ease with which your knowledge base can be understood by other people.

The lists of formal terms and object hierarchies also provide a framework for use of familiar database-type functions. Sets of statements can be abstracted from the knowledge base by searching the formal statements using combinations of any of the formal terms, sources, aliases and topics. There are three possible search strategies in relation to objects – searching for just one selected object, searching for a selected object and all objects below it

¹ Formal representation of unitary statements is explained step by step in Chapter 4.

² A parser is a computer program that checks the syntax of a statement to ensure that it conforms to a defined grammar.

in an object hierarchy, or searching for objects both above and below the selected object in an object hierarchy. In this way you can abstract sub-sets of the knowledge base which can then be treated as more focused knowledge bases in their own right. Search strings (aliases) may be converted into topics which are groups of related search strings amalgamated into one topic string, using Boolean connectives, for example; 'cattle OR sheep AND grass'. The 'Welcome' memo that appears on loading a knowledge base provides direct access to the topics, so that a new user may see immediately what sort of information is in the knowledge base, and have a means of accessing it.

1.4.2 DIAGRAM INTERFACE

A diagram-based interface for knowledge representation and retrieval is also available within AKT to support the viewing of statements and the links between them. For most new users, the diagram interface also serves as the first mode of entry in recording knowledge.

A diagramming approach to representing information about agro-ecosystems is familiar to many resource managers and provides an intuitive means of synthesising and representing complex information. It has also been used successfully in cross-cultural situations to form a clear consensus about the important causes of land use problems (Lightfoot, *et al*, 1989), and in enabling articulation of local knowledge (Conway, 1989). Thus, it has been shown that diagrams in knowledge elicitation can result in a set of knowledge that is significantly more comprehensive and coherent than that which results from other approaches to elicitation.

Producing a diagram is also a powerful means of enabling the developer to synthesise available knowledge on a particular topic and, as a result, to increase his or her understanding of that topic. Diagrams can be used to make an explicit statement of what is known about the topic. Furthermore, it enables developers to assess the completeness of their current understanding or of the available knowledge, through the identification of missing linkages in the diagram. The facility for diagram generation provided within AKT, (see Chapter 8) is a powerful tool in this respect.

The diagramming interface uses a restricted syntax such that two nodes with a link between them is equivalent to a unitary statement. The two nodes and link are parsed to create both a formal representation of the unitary statement and a natural language equivalent. Figure 1.3 illustrates the correspondence between two nodes with a link and a formal unitary statement. In this way, you can add to a knowledge base without learning the formal grammar. A range of tools provides a flexible diagramming environment. The ability to develop hierarchically linked sets of diagrams overcomes the space limitations, which can be associated with a diagramming approach.

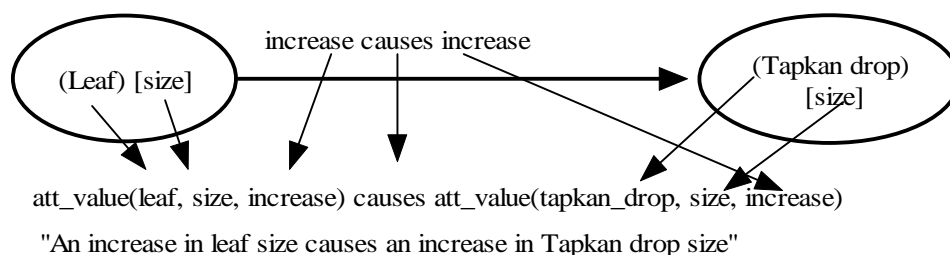


Figure 1.3 Correspondence between the content of a diagram and natural language and formal statements.

The diagram syntax does not make use of the full range of combinations possible with the formal grammar, so some types of statement (attribute value statements and comparison statements) can be entered only through the text interface. Nevertheless, while diagramming is less expressive than text, it allows creation of coherent knowledge bases by less experienced users. The diagramming and text based interfaces can be used interchangeably which allows the user greater flexibility. There is also a facility for the automatic diagramming

of selected sets of text-entered statements, to gain an overview of selected parts of the knowledge base.

1.5 REASONING WITH KNOWLEDGE BASES

A knowledge base is developed in AKT in order to create a synthesised report of the current state of knowledge on a defined topic. The knowledge may then be used for a range of reasoning tasks. Some examples of different user level reasoning tasks are listed in Table 1.1.

Table 1.1 *Some examples of user level reasoning tasks*

- Generating synthesised reports of the current state of knowledge (local, scientific or combined) on defined topics as a resource for extension and planning.
- Exploring the knowledge base in order to identify discrepancies between knowledge held by a local community and scientifically verified information.
- Correlating scientific information with local description to broaden the range of applicability of research results.
- Facilitating research planning and prioritisation, by identifying key gaps in understanding that constrain the productivity, stability and sustainability of an agroecosystem.

The formal grammar balances three competing needs:

- the need for an **expressive** grammar which allows a significant proportion of ecological knowledge about agroforestry to be represented;
- the need for a **simple** grammar which can be successfully applied by users with only limited training; and
- the need for a **flexible** grammar which can be combined with a range of appropriate inference mechanisms to automate reasoning tasks.

To maximize flexibility, a task language is provided within AKT, which is designed to allow you to customize reasoning tasks. This may be done by adapting existing tools to suit your particular needs, or by creating completely new tools, the need for which was not envisaged by the software development team.

1.6 APPLICATION OF THE KNOWLEDGE-BASED SYSTEMS APPROACH TO AGROFORESTRY RESEARCH AND EXTENSION

To date, knowledge bases have been created in conjunction with agroforestry research programmes in Nepal, Sri Lanka, Thailand, Tanzania and India, Indonesia and Kenya. Users may be researchers or extension workers, though the system is most powerful where there is an institutional investment in making better use of qualitative information. The system currently being used in a range of applications by governmental and non-governmental organisations around the world, some of which are listed below:

- i. Assessing farmers' knowledge in participatory crop improvement programmes for maize/millet (Agricultural Research Station, Pakhribas, eastern midhills of Nepal) and cassava/maize (Corpoica, Caribbean Region, Colombia) in the government sector.

- ii. Assessing farmers' knowledge of soil fertility and green manure in an NGO crop improvement programme in the Nepalese Terai (LI-BIRD).
- iii. Assessing farmers' knowledge of gap rejuvenation techniques for jungle rubber agroforests in Indonesia (ICRAF).
- iv. Assessing local knowledge of rubber intercropping practices in Sri Lanka (Rubber Research Institute).

1.7 THE UTILITY OF THE APPROACH

It is difficult to quantify the utility of output resulting from the application of the knowledge based systems approach. Many of the primary outputs are less tangible than the definitive answers produced by a more traditional decision support system which may include quantitative models or expert systems. The KBS approach is not intended to produce definitive and, therefore, testable recommendations. Nevertheless, trial applications of the approach to date have demonstrated that it can have a real and significant impact on agroforestry based research and development programmes (see Box 1).

Box 1. Researchers in Nepal design different types of research when they learn about what farmers already know

In the eastern mid-hills of Nepal it was assumed during the 1980s by the forest service that the planting of trees on farmland was constrained by lack of appropriate planting material and so nurseries were set up and seedlings offered to farmers. Take up of seedlings by farmers was low, and it was later discovered through acquiring local knowledge that farmers were, in fact, already managing abundant natural regeneration on their crop terrace risers (Thapa *et al.*, 1995). Basically there were plenty of naturally regenerating seedlings – farmers cut back those they did not want to develop into fodder trees. Furthermore they chose which species they did allow to grow on the basis of a sophisticated understanding of the seasonal feeding value of the fodder they produced (Thapa *et al.*, 1997) and the extent to which they affected crop yield and soil erosion (Thapa *et al.*, 1995) – aspects that research and extension staff had not adequately considered in choosing nursery stock. Farmers also used terminology not always understood by research and extension staff to describe tree-crop interactions. Perhaps most notably, farmers were concerned about canopy modification of rainfall drop size, because they thought that larger drops caused higher rates of soil erosion. The process of water droplets falling from leaves was locally known as *tapkan*.

Not only had research and extension staff been unaware of this farmer knowledge, but also it was actually contradicted in the scientific literature which held until recently that drop size was independent of canopy morphology (Brandt, 1989, Thornes, 1989). Scientific understanding of how leaves of different types affected drop size was revised, and brought in line with that of the Nepalese farmers in 1993 when new instrumentation allowed more reliable measurement of drop size (Hall and Calder, 1993). The key point here is that once researchers were made aware that tree-crop interactions were important to farmers and that farmers had a cogent interest in minimizing negative impacts of trees on soils and crops they could see the relevance to farmers of research in this area and had the terminology to communicate with farmers about it. In the last few years researchers at frontline agricultural research institutions serving the eastern mid-hills have done work directly on tree-crop interactions (Joshi and Devkota, 1996), and there are now plans for tree-crop interactions to form a central basis of agroforestry research in the Western Development Region (Paudel, *et al.*, 1997).

1.7.1 THE EXPRESSIVENESS OF THE AKT APPROACH AND IMPLEMENTATION

The formal grammar has proved effective in capturing a significant proportion of the description of an agro-ecosystem as given by farmers (see Box 2). It was designed as a means of capturing qualitative descriptions of components of the agro-ecosystem and the ecological relationship between these components, rather than capturing technical knowledge about the management of a practice. Technical knowledge is captured by the grammar only in relation to the impact of management actions on the ecological relationships and their

impact on management objectives. The justification for this emphasis on 'deeper' explanatory knowledge lies in the postulates that:

- actual management techniques are subject to diverse influences – available knowledge about the ecology of the system, economic considerations, personal preferences, and so on: and
- of these, explanatory ecological knowledge may often be portable between sites and across cultures while many other influences are site and culture specific (Walker *et al.*, 1991).

Box 2. Farmers' recognition of deficiencies in their knowledge about below-ground tree-crop interactions

While farmers recognised six tree attributes (leaf size, leaf texture, inclination angle, crown diameter, crown density and tree height) that affected *tapkan* (see Box 1) and shade, and described causal mechanisms for how each attribute affected them, their knowledge of below-ground competition was restricted to a rough classification of 40 out of the 90 tree species found on farms as being either *malilo* – enhancing soil fertility and less competitive with crops, or *rukho* – competitive with crops (Thapa, 1994). Causative knowledge about why trees were classified in these ways included only two elements: a gross classification of root systems as predominantly shallow or deep and some knowledge of the speed of decomposition of leaf litter (which occurred above-ground and so could be observed). As trees were regularly lopped for fodder, a number of issues pertinent to practical management arose with respect to species differences in root systems characteristics and the effects of different lopping strategies on root development and competitiveness, which farmers were hitherto unable to address.

The rigorous approaches to representation and analysis, which are used in AKT, have made it possible to explore the comparability and compatibility of knowledge from different sources. Application of the grammar across a range of agroforestry research programmes has shown that ecological knowledge can be made comparable across sites and cultures – including the divide between scientific or professional knowledge and the ecological knowledge held by farming communities (see Box 3 and Box 4).

1.7.2 THE KNOWLEDGE BASE AS A RESOURCE

Creating a knowledge base involves a significant investment of time – particularly when many people have to be interviewed. The product therefore, should be a resource that is suitable for many purposes. AKT has been designed in a way that should allow development professionals within research institutions to make routine use of a set of centrally maintained topic-specific or problem specific knowledge bases for a range of purposes, and to improve the content of the knowledge bases as appropriate (Sinclair *et al.*, 1993).

While this is possible with the current software, a clear vision of the range of tasks for which the knowledge base is intended is required from the outset. Thus an effective set of criteria can be developed to enable decision making during knowledge representation. This clarity of purpose is particularly important where more than one individual is involved in creating the knowledge base, or where the knowledge is derived from more than one group of informants. For example, comparison of the knowledge bases created to date suggests that a list of key processes might be identified which are likely to be important in any description of the ecology on an agroforestry practice. While the terminology used in the lists of processes of these different knowledge bases is not immediately comparable, it is apparent that the same fundamental processes (shading, rainfall interception and nutrient cycling for example) are being described in each knowledge base. Starting from a common knowledge base template may facilitate the creation of knowledge bases that are generic in their content and can be combined successfully. Earlier approaches to develop templates at a statement level (Haggith *et al.*, 1992) proved difficult to implement, but this more flexible, higher level approach may prove effective in facilitating the development of coherent and comparable knowledge bases.

Box 3. Comparative analysis of scientists' and farmers' knowledge about the tannin content of tree fodder and its implications for feeding farm animals

The existence of formally documented records of farmers' and researchers' knowledge about the nutritive value of tree fodder (Thapa, 1994; Thapa *et al.*, 1997) made it possible to compare the equivalence of terms used by farmers and scientists. This was done using automated reasoning (Kendon *et al.*, 1995) and it was found that there was some equivalence between the way in which farmers used the term 'leaf bitterness' and scientists used the term 'tannin content' – put simply, fodder that scientists described as having a high tannin content tended to be described by farmers as bitter. However, while scientists had some detailed knowledge about the role of tannins in protein digestion by ruminants and decomposition of leaf litter, they knew very little about the actual tannin contents of the 90 native species used by farmers and how this varied seasonally. In contrast, farmers did not possess detailed knowledge about the mechanism of action of tannins in ruminant digestion, although they did associate leaf bitterness with low palatability and nutritive value (Thapa *et al.*, 1997), and their local classification of fodder appears to encompass implicitly effects of tannins on protein supply to the duodenum in cattle (Thorne *et al.*, 1997). Farmers could, however, articulate detailed knowledge about how leaf bitterness varied in a large number of tree species throughout the season. This demonstrates complementarity between farmers' and scientists' knowledge that could be exploited in designing appropriate research (farmers' understanding of intraspecies variability has already led researchers to revise strategies for sampling tree material for analysis of nutritive value). Clearly, because of complementarity, the combination of what farmers and scientists know represents a more powerful resource than either knowledge system alone.

Box 4. Identification of leverage points by comparison of what farmers do with what they know

Continuing the unfolding example related to Nepalese hill farming used in Boxes 1-3, in addition to documenting farmers' and researchers' knowledge, a detailed tree inventory was conducted, permitting comparison of what farmers said about trees and how they actually incorporated them into their farming systems (Thapa, 1994). Statistical analysis of the location of trees that farmers classified as *malilo* and *rukho*, showed that farmers had a higher proportion of *malilo* trees growing in association with crops than *rukho* trees, consistent with *malilo* trees being considered less competitive with crops and enhancing soil fertility. In contrast, nearly half of the trees that farmers had, which they classified as causing heavy *tapkan*, were grown on crop terrace risers where, according to local knowledge, they would reduce crop yield and promote soil erosion. Thus, despite having a clear understanding that large-leaved tree species were competitive with crops and promoted soil erosion, farmers still planted them in association with crops. The explanation for this was that farmers were trading off the negative impacts of these trees on crops and soil against their high fodder value at key times in the dry winter season. Subsequent analysis of formally documented local knowledge (Joshi, 1998), indicates a positive relationship between leaf size and palatability of tree fodder among the species used by farmers (>70% of trees classified by farmers as large-leaved were also classified as having high palatability, whereas < 20% of trees classified as small-leaved were highly palatable). This represents a key constraint in the system where farmers are having to sacrifice crop yield and tolerate soil erosion in order to obtain fodder at key times in the season. Hence, this identifies a leverage point where research, to introduce or breed a smaller-leaved tree with the same fodder characteristics as the large-leaved species that farmers are currently using, for example, may represent an adoptable advance that addresses current constraints.

1.7.3 THE UTILITY OF REASONING MECHANISMS

The application of reasoning tools in knowledge base management makes the development of concise and coherent knowledge bases much less difficult and time consuming than it would otherwise be.

The application of reasoning tools has also allowed novel approaches to knowledge analysis to be applied to complex sets of information. These approaches would be untenably time consuming and complex to apply without automated reasoning.

Finally, trials have shown that reasoning tools are effective in allowing users access to the content of the knowledge base, either to learn from it via tutorials, or to use the knowledge in decision-making.

The two main automated reasoning features supplied by AKT are:

- 1) property inheritance
- 2) causal diagramming

1.7.3.a Property Inheritance

In an object hierarchy that is organised in the form of an inverted tree with the root at the top and the branches below, the lower objects inherit the properties or characteristics of the objects from which they are descended. Take for example, an object hierarchy describing plant taxonomy, in which the name of the hierarchy, or the SuperObject is 'plant' whilst somewhere further down the tree one of the members (sub-objects/descendants) of the object hierarchy is 'lettuce'. Thus the statement 'watering plant causes plant growth to increase' would mean that, by property inheritance, we could infer the statement 'watering lettuce causes lettuce growth to increase'.

1.7.3.b Causal Diagramming

In some knowledge bases there may be hundreds of statements and it is difficult to determine the overall structure. To enable the user to see what structure does exist between statements, AKT uses automated reasoning techniques to:

- (a) represent each statement by two nodes with a link between them describing their relationship;
- (b) sort the statements into related sets of nodes and links;
- (c) draw the resulting nodes and links in such a way that the user can immediately see what relationships exist between the statements.

Various facilities are also provided in AKT to allow the user to manipulate these diagrams to improve the representation and intelligibility of the knowledge within the knowledge base.

Key points of chapter one

- Agroforestry is interdisciplinary and involves complex decision making for planning and extension programmes.
- The knowledge-based systems approach (KBS) offers a practical method of capturing, storing and retrieving knowledge from diverse sources.
- Creation of a knowledge base involves knowledge elicitation and knowledge representation using AKT, a tailor made software for the purpose.
- AKT offers both a text mode and diagram interface for knowledge representation and retrieval from a knowledge base.
- The task language provides a user-friendly environment for developing and implementing 'tools' for processing and outputting knowledge from a knowledge base.
- Several successful applications of KBS approach have demonstrated the use and utility of this novel method in incorporating local and scientific knowledge effectively in planning research and development programmes.
- The process of creating a knowledge base can have a significant impact on the knowledge base developer's understanding and perception of domain under investigation.
- Formalisation of knowledge enables the comparison of knowledge from different sources (e.g. farmers and scientists).

CHAPTER TWO – KNOWLEDGE ELICITATION

In contrast to the later stages of knowledge base creation which will be described, knowledge elicitation does not need to conform to a specific set of guidelines. Indeed, approaches to knowledge elicitation that are appropriate in one context may be inappropriate in another. The contents of this section should therefore be regarded as suggestions and examples of approaches, to be adopted as appropriate to circumstances.

2.1 DESIGNING A KNOWLEDGE ELICITATION STRATEGY

Generally, it will not be possible to elicit knowledge from all appropriate sources when creating a knowledge base, particularly when eliciting information from a local community. Therefore, a sampling strategy must be designed. This should enable the efficient development of a knowledge base that is representative of the knowledge of a defined community, or set of communities. The sampling strategy should also incorporate a return to the source community to test how well the new knowledge base represents the knowledge of the community as a whole.

2.1.1 A FRAMEWORK FOR DESIGNING A KNOWLEDGE ELICITATION STRATEGY

The framework is divided into four stages:

- **Scoping**
- **Definition of the domain**
- **Compilation and**
- **Generalisation**

The important feature of this four stage strategy for knowledge acquisition, in terms of sampling, is the separation of knowledge base development (the first three stages) where a small purposive sample of people are intensively involved, and the generalisation stage, where a large randomised sample of people is drawn from the target community to explore how representative the knowledge base is.

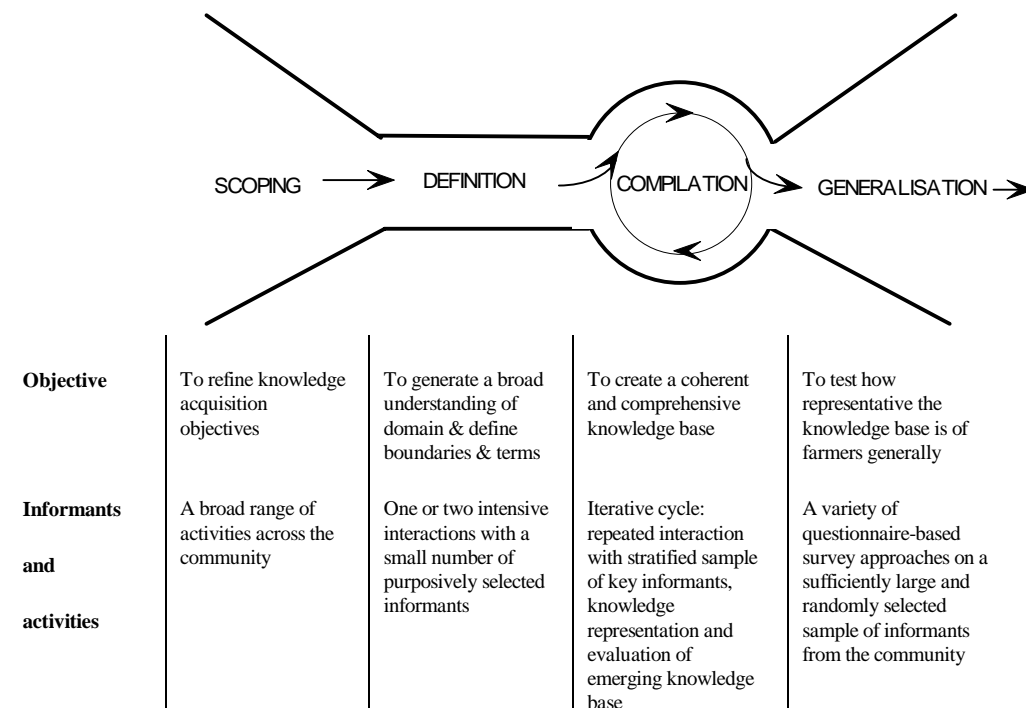


Figure 2.1 gives an overview of the four stages in the knowledge elicitation process.

2.1.1.a Scoping

The detailed design of a knowledge elicitation strategy is best undertaken during a period of introduction and establishment with the source community. Attempts to design a detailed knowledge elicitation strategy prior to fieldwork are inappropriate. The scoping period serves to:

- familiarise the researcher with the source community (when applicable), and vice versa;
- allow adjustment of the basic objectives in knowledge base creation through refinement of problem specification;
- provide a preliminary assessment of the basic and universal information held by the community on the topic in question;
- help to identify suitable informants for later stages (where applicable); and
- identify parameters within the community that might account for differences in knowledge; these parameters may then be used as variables for stratifying the local informants.

Rapid rural appraisal techniques such as household surveys, resource surveys, questionnaires and mapping have tended to be used at this introductory stage. However, scoping does not necessarily require formal surveys and in some cases a few informal conversations with local people may be sufficient to identify the key determinants of variability (i.e. the different strata) to be explored and the key informants for the definition stage.

2.1.1.b Definition of the domain

The definition stage is used to develop an overall understanding of the domain in question, defining boundaries, identifying terminologies and creating a framework.

Sources are purposely non-randomly selected from the source community. These 'key informants' are selected on the basis of interest, articulateness, depth of knowledge and willingness to participate. Key informants known to be in some way significantly unrepresentative of the community as a whole (for example, school teachers in local farming communities) have tended to be avoided, although they are often helpful in identifying other key informants.

Because the definition of the domain has a significant impact on the shape and content of the final knowledge base, an adequate spread of key informants is desirable to maximise the chances of developing a representative framework within the time available for this stage of knowledge acquisition.

2.1.1.c Compilation

The compilation stage of the knowledge elicitation strategy is used to record detailed knowledge within the framework developed in the definition stage and to indicate the variability of knowledge over the community as a whole. The focus at this stage is on talking to a few knowledgeable people in depth, rather than attempting to obtain statistically representative samples. As repeated interview of the same person is of paramount importance in obtaining deeper explanatory knowledge and resolving inconsistencies, willingness to participate must be an important criterion for selection of key informants.

For local communities, a small, stratified random sample from the source community is appropriate. The purpose of stratifying the selection of informants is to ensure coverage of knowledge about the domain where different classes of people may have different knowledge. Key determinants of variability in knowledge relative to research objectives may be gender, education, occupation, location, religion, ethnic group etc. Appropriate strata combining these

factors to allow assessment of the influence of these factors on the distribution of knowledge are identified. Within each stratum informants are then purposely selected.

A similar approach can be applied to professional communities (by discipline, for example). However, the relatively small size of professional communities or bibliographies mean that in practice the distinctions between definition and compilation may be blurred.

2.1.1.d Generalisation

Having obtained a knowledge base from a few informants, the generalisation stage involves testing the representativeness of this knowledge across the community as a whole. This requires a random sample that is statistically representative of the community as a whole, typically upwards of 100 people who have not been previously interviewed. This phase has one or more of the following objectives:

- To validate the knowledge base as representative of the knowledge held by the community as a precursor to using it to inform research and extension programmes;
- To explore the distribution of knowledge amongst people within the community; and
- To augment the knowledge base with details not recorded in the compilation stage.

Approaches to assessing representativeness are discussed in more detail in Chapter 6.

2.1.2 SAMPLE SIZE

The number of informants selected for each stage of the knowledge elicitation strategy depends on the nature of the source communities, the size and quality demanded of the knowledge base and the time available. There are no fixed rules on how many informants should be approached in the scoping stage and it depends entirely upon the methods used for scoping (see above 2.1.1.). From the definition stage onwards, the number of informants grows at every stage. So six – ten key informants in total might be adequate for the definition stage, five for each identified stratum in the compilation stage, and 100 for generalisation in a relatively homogeneous local community. In general it has proved more productive to speak to fewer people on more occasions than to cover a larger number.

2.2 CONSTRAINTS UPON KNOWLEDGE ELICITATION

Interview technique is a skill which is best developed through experience. This section cannot provide a prescription for how to interview informants, but outlines a few important considerations. Useful discussions of interview technique can also be found in Werner and Schoepfle (1987a & b).

Problems of knowledge elicitation can arise because people often fail to recognise that they have knowledge, let alone how they use it (Hart, 1986) and thus informants often find it hard to give detailed descriptions of their knowledge and how they use it. Indeed, the process of closely questioning an informant can interfere with his own perceptions of what he does. This is because much knowledge is tacit; that is, it has been learnt through observation and experience, and is understood, but is not generally expressed.

In a similar fashion, the status assumed by the researcher in the study community will also influence the elicitation process since all attempts to reduce social and intellectual barriers, and improve understanding will enhance knowledge elicitation and co-operation. If the researcher assumes the role of learner his reception by expert informants will differ markedly to the one he may encounter if he presents himself as 'scientist' or 'planner', or otherwise assumes an elevated status.

The role of communication skills, both in researcher and informant, is an essential element in knowledge elicitation, and is of particular significance in cross-cultural work (see Werner and

Schoepfle, 1987a & b). The familiarity and skill with which words are used to express concepts and procedures will affect the quality of knowledge elicited through interview. Although people identified for interview may be 'experts', it is unlikely that they have previously been required to describe their knowledge and decision-making procedures. Additionally, much knowledge learnt through experience may be used without a conscious awareness of explicit details (Hart, 1986) and even conscious knowledge may not be expressed in a way that it can be recognised as such (Breuker and Wielinga, 1987). This necessarily has implications not only for the elicitation process but also for the subsequent process of formally representing the knowledge for use by others.

2.3 INTERVIEW TECHNIQUE

Developing a representative abstraction of local explanation of the behaviour of an agroforestry practice or system is a significant undertaking and may involve interviewing 50 or more informants, four or five times each. Approaches to interviewing are well developed and documented (see for example, Cordingley and Betsy, 1989, Diaper, 1989, Cooke, 1994). Ethnographic techniques of knowledge elicitation as used by anthropologists (such as participant observation etc.) have been recognised as useful in the development of expert-systems because they capture insider knowledge, or knowledge described and explained from the informant's point of view (Benfer and Furbee, 1990). An investigation designed to access the insiders' knowledge without relying on shared assumptions and presumptions is known as an 'emic' approach (Werner and Schoepfle, 1987). This approach is important if knowledge is to be gathered in a way that reflects the structure of the indigenous knowledge system and allows the representation of that knowledge system to remain intuitively 'correct' to the informant. The alternative approach is an 'etic' one, which seeks to understand local practice using external scientific explanations that may have no internal logic for the indigenous informant (Knight, 1980).

2.3.1 SOME GOLDEN RULES FOR INTERVIEW TECHNIQUES

1. Send a formal letter to farmers requesting interviews. The time and location of the interview should be arranged to fit in with the interviewee's schedule.
2. The interviews should be conducted in the field rather than in the interviewee's home, because farmers often need to point to things that they cannot express.
3. It is important to consider whether the season in which the interviews are to be carried out is appropriate. Farmers will typically have more time when the harvest is in, than at the beginning of the growing season.
4. One interview is seldom enough, a key informant will need to be interviewed 3 – 5 times.
5. An interview should never go beyond an hour, as the interviewee will lose interest.
6. All interviews should be tape recorded to ensure that the interviewer and interviewee can concentrate on the discussion during the interview and no important points are missed while writing statements. The background and contextual information remain intact with the recording, which may be used for future reference.
7. At the knowledge compilation stage, no structured questionnaires should be used. The basic questions 'How?' and 'Why?' are asked during discussions and more questions are framed as the discussion progresses until the farmer cannot explain any further.
8. Preliminary analysis of answers from one interview should be used to set up topics and questions for the following interview with the same informant.

9. Knowledge gained from one interview may be verified through the process of knowledge elicitation with other informants to resolve conflicting information and to assess whether an item of knowledge was idiosyncratic or consistent for the group.
 10. Where there is some serious contradiction between farmers' statements which cannot be explained by differences between the selected strata, it is possible to carry out group interviews and put the contradictions to the group for elucidation.
 11. The attitude of the interviewer towards the farmer is all important, and should be one of respect. This cannot be overemphasised. The interviewer should:
 - a) Approach the farmer as a student desirous to learn from a teacher. Approached in this manner the informant will be more inclined to teach and explain all he knows.
 - b) Keep an open mind and suspend judgement during knowledge elicitation as indigenous knowledge may be tacit, and the interviewer may not know what there is to know.
 - c) Minimise his or her own influence and encourage informants to express knowledge in their own terms.
 - d) Avoid leading informants into formalising their expressions for the convenience of knowledge structuring.
 - e) Make the interview atmosphere as relaxed as possible.
-

Key points of Chapter Two:

A knowledge elicitation strategy should contain 4 parts:

Scoping
Definition
Compilation
Generalisation

For knowledge acquisition, the informant population may be stratified according to gender, age, ethnic origin, economic status etc. in so far as belonging to one or more of these strata will affect the scope of knowledge of the informants.

The sample size depends on the size of the source communities, their homogeneity, and how much time is available. In general it is better to interview a smaller number of people more often, than a larger number of people only once or twice.

Interviewing techniques are more an art than a science but can be improved with practice. The crucial issue is the attitude of the interviewer towards the informant. A student-teacher relationship is likely to be far more fruitful than a 'top down' approach, on the part of the interviewer.

CHAPTER THREE – PREPARATION FOR KNOWLEDGE BASE CREATION

The following chapters 3 – 5 detail the procedure for creating a knowledge base within AKT and should therefore be considered and applied in conjunction with one another. Knowledge base creation involves abstracting knowledge from the information collected, the formal representation of that knowledge and the specification of hierarchical relationships between terms in the knowledge base. As soon as the first statements have been entered into AKT, a knowledge base has been created. This knowledge can then be evaluated and thereby provides a basis for further knowledge elicitation and representation as well as modification of current representation. The process of knowledge base creation involves a tight cycle of knowledge elicitation, knowledge representation and knowledge base evaluation.

NOTE: The importance of evaluating the knowledge base after each new round of inputting unitary statements cannot be overemphasised.

3.1 PREPARATION

Creating a knowledge base involves specifying the objectives behind it, and defining its boundaries.

3.1.1 SPECIFICATION OF OBJECTIVES

The creation of a knowledge base involves identifying the knowledge to be included. These decisions are most effectively and consistently made if taken in relation to an explicit set of objectives. The quality (usefulness) of a knowledge base depends very much upon the objectives being both *specific* and *clear*.

When specifying the objectives it is necessary to strike a balance between the need for precise objectives to guide the knowledge acquisition process and the need to avoid compromising the knowledge acquisition process by adhering to any preconceived ideas which helped formulate the objectives in the first place. An appropriate balance may best be achieved by making the *purpose* of the knowledge base the objective, rather than the actual structure and content of the knowledge base the objective.

For example, research in Nepal was intended to:

Document explanatory ecological knowledge used in decision making by farmers in managing their farmland tree fodder resources in order to better inform national and regional research efforts.

Subsidiary objectives for the same research could be:

Comparison of the local knowledge with scientific knowledge.

These objectives for using the contents of the knowledge base(s) provided a framework for its creation.

The iterative nature of the process of knowledge base creation allows the reassessment and modification of objectives as the knowledge base develops.

3.1.2 BOUNDARIES OF THE KNOWLEDGE BASE

The second stage in the process of preparation is to define the boundaries and contents of the proposed knowledge base. The boundaries must be defined with reference to the stated objectives.

The knowledge base is an arbitrary unit. Two knowledge bases may be merged into a single knowledge base or a single knowledge base may be split into two. Furthermore, different sets of knowledge may frequently be needed for different tasks. The mechanism for developing different topics from a formal knowledge base (see Chapter 7) provides a means of dividing a knowledge base into subsets, effectively creating smaller sets of knowledge with which to reason.

It is better to create fewer knowledge bases covering a broader range of knowledge rather than many small knowledge bases. This reduces repetition of core knowledge amongst knowledge bases. However, ensuring consistent and unambiguous use of terms, coherence and completeness becomes an increasingly demanding task as the knowledge base grows in its breadth of coverage. Furthermore, too large a knowledge base is unwieldy, implementational constraints (such as computer memory and speed of processing) have an increasing impact as the knowledge base grows in size. It is necessary to define clear boundaries for a knowledge base and to represent unrelated knowledge in different knowledge bases in order to ensure the creation of tractable results.

For example, in the Nepalese case study three discrete topics were identified on the basis of the objectives given above, and in consideration of the domain that had to be investigated in order to generate a comprehensive record of the explanatory ecological knowledge used by farmers in managing their farmland tree fodder resources. These topics were:

- the propagation of tree fodder resources,
- tree-crop interactions and the selection and
- evaluation of different fodder for livestock.

As a result, three knowledge bases were created.

As with the specification of objectives, the specification of boundaries may be iterative. Initial acquisition of knowledge related to tree fodder demonstrated that while these three topics merited consideration, much basic understanding was common to all of them. The knowledge base boundaries were, therefore, modified by merging the three knowledge bases.

By identifying the boundaries of the knowledge base one automatically sets the specifications upon which the knowledge elicitation strategy can be based.

Key points of Chapter Three:

Preparation before actually developing a knowledge base involves two crucial activities:

Specifying objectives of creating a knowledge base

Defining scope and boundary of the knowledge base

The purpose of the knowledge base should define the objectives – not any preconceived specification of the structure and contents of the knowledge base

As a knowledge base develops, both the objectives and the boundaries can be redefined.

CHAPTER FOUR – KNOWLEDGE REPRESENTATION

This part of the manual describes the process of knowledge representation. Knowledge representation comprises;

- the process of abstracting knowledge
- the process of formal representation
- the process of formal term specification

4.1 ABSTRACTING KNOWLEDGE

4.1.1. BASIC CONCEPTS

The knowledge given by an informant is made up of a combination of basic units of knowledge. These units will be structured according to the context in which they are articulated. In the creation of a knowledge base we are more interested in the individual units used to construct that combination, than in the combination itself. We also need to understand the ways in which these units can be linked.

It is more important to ‘disaggregate’ and record the individual units of knowledge that make up a particular statement, than to record the original combination. The units can be recorded along with enough associated information to allow them to be re-combined with others to explain aspects of the behaviour of a system.

The first step in the process of knowledge representation therefore is to abstract these basic units from the knowledge articulated by the informant. These basic units are referred to here as ‘unitary statements’.

4.1.2 UNITARY STATEMENTS AS BASIC UNITS OF KNOWLEDGE

The term ‘unitary statements’ is used here to refer to the smallest useful units of knowledge. A unit of knowledge is useful if it can be used in combination with other knowledge in reasoning.

Unitary statements express assertions. A statement is not the same as a sentence because different sentences can be used to express the same statement. Unitary statements contain knowledge that is useful without reference to other unitary statements but which cannot be broken down into further unitary statements. So:

“The biomass production of a plant is proportional to leaf area index of that plant.”

is a unitary statement.

By contrast:

“Rainfall is low.”

is not a unitary statement, because it does not contain enough information to be used in reasoning; it does not refer to any system, any time or any place.

“Clover is eaten by sheep and goats.”

is not a unitary statement, by this definition, because it can be broken down into two unitary statements:

1. *“Clover is eaten by sheep.”*
2. *“Clover is eaten by goats.”*

Unitary statements are divided into two categories, binary statements and attribute-value statements.

A binary statement captures a relationship between two entities, e.g.:

“Rainfall causes increased soil erosion.”

Attribute-value statements describe an attribute of an entity or classes of entities, e.g.:

“Bananas are yellow.”

This distinction is important in the diagrammatic representation of knowledge. Within AKT, binary statements can be represented diagrammatically, but attribute-value statements cannot.

Unitary statements can also be divided into those which refer to classes or types of entities (termed ‘statements of class’) and those which refer to particular instances (termed ‘statements of instance’):

Statement of class *“Barley seeds germinate in seven days at 10°C.”*

Statement of instance *“Seedling 97 germinated after nine days at 10°C.”*

These two forms of statements are syntactically identical but they are significantly different in utility. Explanatory ecological knowledge can be derived from statements of instance, however, statements of instance are, in themselves, data and are unlikely to constitute a useful component of a knowledge base.

4.1.3 RECORDING THE CONTEXT OF A STATEMENT

Knowledge is inescapably contextual. Disaggregation of knowledge into a set of unitary statements will cause this context to be lost. This loss of contextual understanding demands explicit information about the circumstances under which the statements apply (i.e. conditions) and the source of each statement.

4.1.3.a Conditions

Most unitary statements will have only a limited validity. Validity relates to the circumstances or conditions under which the unitary statement is held to be true and the certainty that can be placed on the unitary statement being true. Neither are adequately captured in unitary statements themselves. Appending conditional information and certainty to a statement, results in a more complete record of the knowledge articulated by an informant. However, it is extremely difficult to elicit meaningful information about the confidence informants have in the knowledge that they articulate. Furthermore, use of statements of certainty is problematic. For this reason AKT provides facilities for recording conditions, e.g.:

“Soil erosion is severe IF

*the slope is greater than 20° AND
rainfall is over 1000 mm per annum AND
vegetation cover is thin.”*

but it does not give any indication of the confidence the informant, or others, have in this particular piece of knowledge. Although attempts have been made to capture the degree of confidence in a statement, it was found that informants were uncomfortable in providing statements of confidence, in part because it appeared to question their veracity.

4.1.3.b Source

One important piece of contextual information for the interpretation and use of knowledge is the source of that knowledge. Where a user of knowledge is familiar with or aware of the informants/references, this will inform his/her use of that knowledge. For this reason knowledge is tagged according to source. This also facilitates the assessment of the internal consistency of knowledge from a particular source and the distribution of knowledge between sources.

There is an ethical need to enter the source of knowledge, otherwise one may lay oneself open to accusations of extractivism. By recording the source of knowledge, it is also possible

to identify whose knowledge is being used at any time so that if there is any value attached to that knowledge it can be attributed to the original source. However, in general the methodology is designed to record how a community understands the resources on which its livelihood is based, and the purpose is not to seek out specialist (valuable) knowledge, but to collect the knowledge of a community, rather than an individual.

4.2 FORMAL REPRESENTATION

4.2.1 INTRODUCTION

Because of the ambiguity and complexity inherent in natural language, accurate interpretation of unitary statements in unrestricted natural language may often be difficult. Natural language is extremely robust in its use and interpretation; it can contain a great deal of ambiguity and imprecision, yet still serve a useful function in communication. Meaning in natural language is often context specific and, therefore, flexible. However, it cannot be assumed that the implicit contextual meaning of a unitary statement will still be understood by users once it has been included in a knowledge base. Furthermore, automated reasoning techniques cannot cope with flexibility of meaning according to context. As a result, the second stage in the creation of a knowledge base is to create a version of the unitary statement that conforms to a formal syntax. This process is known as formal representation.

The process of formal representation results in statements that have a fixed syntax and can therefore be combined using inference mechanisms in automated reasoning (see Chapter 9). The process also provides a means of evaluating the knowledge already elicited, rationalising terminology and identifying ambiguity, inconsistency and so on (see Chapter 9).

Formal statements comprise a set of terms linked by, and ordered in relation to, other specific terms that form part of the formal language. This linkage and ordering provides information on the way in which the elements of the statement are related and therefore enables a semantic interpretation of the syntax of the statement.

The formal grammar that provides the basis for formal representation is stated in Table 4.1. A summary explanation of the grammar is provided in this section and a more detailed exploration of its application follows.

Table 4.1 The definite clause grammar. Terms in bold are reserved terms in the grammar (i.e. words reserved for use by the system); terms starting with a capital letter are variables; \Rightarrow means 'can take the form of'.

FormalSentence \Rightarrow Statement if FormalConditions
FormalSentence \Rightarrow Statement
Statement \Rightarrow Cause Causes Effect where Causes is an element of the set:{ causes1way,causes2way }
Statement \Rightarrow AttributeStatement
Statement \Rightarrow not (AttributeStatement)
Statement \Rightarrow link (influence,Thing,Thing)
Statement \Rightarrow link (Link,Object,Object)
Statement \Rightarrow link (Link,ProcessBit,ProcessBit)
Statement \Rightarrow link (Link,ProcessBit,Object)
Statement \Rightarrow comparison (Attribute,Object,Comparison,Object)
FormalConditions \Rightarrow FormalConditions and FormalConditions
FormalConditions \Rightarrow FormalConditions or FormalConditions
FormalConditions \Rightarrow Statement
FormalConditions \Rightarrow ActionBit
FormalConditions \Rightarrow ProcessBit
AttributeStatement \Rightarrow att_value (Object,Attribute,Value)
AttributeStatement \Rightarrow att_value (ProcessBit,Attribute,Value)
AttributeStatement \Rightarrow att_value (ActionBit,Attribute,Value)
Cause \Rightarrow AttributeStatement
Cause \Rightarrow ProcessBit
Cause \Rightarrow ActionBit
Cause \Rightarrow Object
Cause \Rightarrow not (Cause)
ActionBit \Rightarrow action (Action,Object,Object)
ActionBit \Rightarrow action (Action,Object)
Effect \Rightarrow AttributeStatement
Effect \Rightarrow ProcessBit
Effect \Rightarrow ActionBit
Effect \Rightarrow not (Effect)
Process_bit \Rightarrow process (Process)
Process_bit \Rightarrow process (Object,Process)
Process_bit \Rightarrow process (Object,Process,Object)
Thing \Rightarrow Object
Thing \Rightarrow ProcessBit
Attribute \Rightarrow atom
Process \Rightarrow atom
Link \Rightarrow atom
Object \Rightarrow atom
Object \Rightarrow part (Object,Object)
Action \Rightarrow atom
Comparison \Rightarrow Atom where Atom is an element of the set:{ greater_than, less_than, same_as, different_from }
Value \Rightarrow Atom Where Atom is an element of the set:{ increase, decrease, change, no_change }
Value \Rightarrow Atom
Value \Rightarrow Number Where Number is either a floating point number or an integer
Value \Rightarrow range (Value,Value)

4.2.2 THE ELEMENTS OF FORMAL REPRESENTATION

There are three basic elements to formal representation: **objects**, **processes** and **actions**. Objects, processes and actions can be considered to be the fundamental elements around which the formal statement is structured.

Objects are normally physical items in the real world, like 'trees', and 'crops', but may be conceptual, for example, 'niche' or 'wet season'. The name of objects in a formal statement are represented by atoms, so, *cow*, *tree* and *hill* are objects represented as atoms. However, objects in formal statements are usually members of a class of atoms (for example *cows*, *trees* and *hills*). Atoms are represented in lower case only. Where an atom in a formalised statement consists of two or more words, these should be joined with an underscore (e.g. *windfall_apples*). The exception to this rule is when the object name in a formal statement begins with a capital letter (e.g. in a proper or Latin name). In this case the name should be enclosed between single inverted commas (e.g. 'Nepal').

Processes (or events) describe changes or fluxes in the real world, for example the process of soil erosion describes the loss of soil, and the process of germination describes the change in a seed from dormancy to active growth. Processes, like objects, are named and are also represented by atoms. In some circumstances a process is not associated with any particular objects in the statement;

e.g. 'rainfall';

alternatively, it may be related to one identified object

e.g. 'erosion' is a process related to the object 'soil'

or may provide linkage between two objects

e.g. 'infestation' is a process which links the object 'pests' with the object 'crop';
'uptake' is the process that links the object 'plant' with the object 'water'

An **action** is similar to a process but is initiated by man; for example, it incorporates all deliberate actions carried out by managers of an agroforestry practice and is always related to either one object or two,

e.g. 'ploughing' is the action related to the object 'field', and
'application' is the action linking the object 'pesticide' to the object 'crop'

Statements are formed by combining these three elements with **attributes**, **values**, (**user defined**) **links** and a set of special 'reserved' terms used in the formal language (see below 4.2.2.a).

An **attribute** describes an object, process or action and is generally measurable. Thus *height*, *rate*, *colour*, *frequency*, *gradient* are all attributes. An attribute is represented as an atom.

A **value** is always attached to an attribute, and describes that attribute. These values can be in units, for example, *5 kg*, *20 hectares*, *40_kg_per_ha_per_year*, *3_months_7_months*, or they can be descriptive values, such as *tall*, *rapid*, *yellow*, *regular*, *steep*. A value can be represented as an atom, a number or as a range.

A (**user defined**) **link** is a term selected to link two objects, two processes or a process and an object, when these cannot be linked using the reserved terms (see below 4.2.2.a). Thus in the statement 'cows eat grass', *eat* is the user defined link, in the statement 'fruit bats pollinate *Parkia biglobosa*' *pollinate* is the (user defined) link. A (user defined) link is represented as an atom.

These above ingredients provide the basis for formal representation. Using different combinations of these, there are four types of statement that can be formed:

- Attribute Value statements
- Causal statements
- Comparison statements

- (User defined) Link statements

(See 4.2.4 below)

4.2.2.a Reserved terms

- The term **comparison** enables the comparison of the value of an attribute for two objects¹.
- The term **causes** allows a causal relationship between an attribute-value statement, object, process or action and another attribute-value statement, process or action to be captured.
- The term **if** allows conditionality to be captured whereas the terms **and** and **or** allow multiple conditions to be specified for a unitary statement.
- The term **part** allows a particular part of a specified object (for example the roots of wheat) to be represented.
- The term **not** can be used to capture negation (e.g. pigs do not eat grass).
- The term **link** allows relationships between objects or processes other than causal or comparative links to be captured (e.g. cows eat grass).

Some elements of formal statements may be represented by atoms that have special meanings and are used in particular contexts. For example, a value may be represented by the special terms **increase**, **decrease**, **change**, **no_change** and **range**. Comparison types can be expressed by one of the following terms; **greater_than**, **less_than**, **same_as**, or **different_from**. Finally, under some circumstances, a link may be represented by the term **influence**.

The grammar was designed with a particular emphasis on causal, comparison and attribute-value statements. However, management actions (i.e. deliberate actions carried out by the managers of an agroforestry practice) are distinguished from ecological processes and can be represented where a statement captures the impact of the management action on the ecology of the practice.

4.2.3 FORMAL REPRESENTATION OF SINGLE UNITARY STATEMENTS

A detailed description of the application of the formal grammar and justification for its structure is best achieved through example.

The following guidelines are divided into four parts:

- a) re-evaluation of the single unitary statement;
- b) identification of fundamental elements;
- c) identification of statement structural type; and
- d) creation of the formal statement.

4.2.3.a Re-evaluation of the unitary statement

The first step in the process of formal representation is to reassess the individual unitary statement. For each new unitary statement one should ask oneself the following questions:

- i. Is the statement clear?
- ii. Is it sensible?
- iii. Is it unambiguous?
- iv. Is it complete?

¹ To compare the process of two objects (such as growth, or germination), then the process must be made into an attribute of the object, e.g. `rate_of_growth` or `rate_of_germination`.

- v. Is it a genuinely unitary statement (and not a compound statement)?

If the answer to all five questions is not 'Yes' then the unitary statement should be reconsidered. In particular, statements are frequently found to be compound, so that it is necessary to break them down into individual unitary statements.

Formal representation is justified by the automated reasoning (Chapter 6) that it makes possible. However, the rigorous consideration of the meaning of each unitary statement that is demanded by the process of formal representation is of significant value in its own right.

4.2.3.b Identification of the elements in the statement

The second step in formal representation is to identify the objects (and parts of objects), processes, actions, links, attributes and values in the statement.

This step is important because knowledge must be explicitly and unambiguously stated in creating formal statements that provide a robust resource for automated reasoning. By contrast, natural language tends to contain implicit elements. A distinction can be drawn between (a) an implication resulting from inadequately abstracted statements (where domain-specific knowledge may be needed to achieve more explicit representation) and (b) an implication that the user of the system might understand but which is unacceptable for formal representation.

The fact that the statement:

Khasru causes sickness in cattle

means that cattle become sick if they eat Khasru, is an example of (a) because the user needs to know that Khasru is eaten.

By contrast, and as an example of (b), it might be assumed that any user of the system can interpret the statement:

Utis is a tall tree

as being about the height of the tree, while formal representation demands that this is explicitly stated,

i.e. the height of the Utis is tall

Attributes are often implicit. For example, in the statement:

Loam is very fertile

'loam' (a type of soil) is an object, 'very fertile' is a value but it is not immediately clear of what this value is a measure (i.e. the attribute to which it refers). However, the attribute must be identified in formal representation. 'Soil fertility' might be suitable in this case.

By contrast, objects, processes and values are implicit only occasionally, usually only where the unitary statement is inadequate (as with the implicit process in the first example above) or where representing statements that stretch the use of the grammar. For example, the statement:

Rainfall is maximum in January

does not contain an explicitly stated object. Rainfall is taken to be a process. In other circumstances, it might be viewed as an object but 'maximum in January' implies that the attribute is a rate rather than a volume. Rates can only be associated with processes and actions. So, 'maximum' is a value for the attribute 'rate' of the process 'rainfall'. 'January' is a value for an attribute time, or maybe time of year. The attribute 'time' refers to the time at which the event occurs.

The statement:

Soil erosion reduces soil fertility

contains implicit values. Fertility is an attribute of an object 'soil' while soil erosion is a process with an implicit attribute 'rate'; both attributes have implicit values, in this case 'decrease' and 'increase' respectively. Therefore the statement expressed explicitly would read:

an increase in the rate of soil erosion causes an decrease in soil fertility

'Part' relationships between objects are also identified at this stage. For example, in the statement:

Siris has a light crown

Siris (a tree species) and crown are objects, however, they are further related in that the crown is a particular part of the object Siris. The identification of 'part of' relationships is similar to the identification of 'type of' relationships between objects in the object hierarchy and can similarly be used in reasoning.

4.2.3.c Identification of the statement structural type

The third step in the process of formal representation is to identify which type of formal statement best captures the meaning of the unitary statement. The identification of structural type, i.e. binary statements or attribute value statements, was introduced in 4.1.2. However, the process can be more demanding during formal representation because some statements which cannot be represented directly in the formal grammar (for example temporal statements) can be represented using the other, existing, structural types.

4.2.3.d Creation of the formal statement

Formal statements may be of one of five types: a causal statement, a comparison statement, a link statement, an attribute statement or a negative attribute statement (see Table 4.1). Some unitary statements may be captured by more than one formal statement type. However, the different types have a differing utility in reasoning with the knowledge base. In general, causal statements are the most useful. Any type of statement may additionally have conditional information attached.

4.2.4 STATEMENT TYPES:

4.2.4.a Attribute Value statements

The most basic form of statement is the attribute-value statement. An attribute-value statement is descriptive, it describes an object, or process or action. An attribute-value statement for an object takes the form:

att_value(Object², Attribute, Value) e.g. att_value(tree, height, tall)

An attribute-value statement for a process takes the form:

att_value(Process, Attribute, Value) e.g. att_value(process(leaf, decomposition), rate, slow)

Entire formal unitary statements can be captured as attribute-value or negative attribute-value statements where the statement consists of a single object or process or action and information about the value of an attribute of that object or process:

Unitary statement

Siris has small leaves

Sirius does not have big leaves

Formal statement

att_value(part('Siris', leaf), size, small)

not(att_value(part('Siris', leaf), size, big))

These statements may also occur within causal statements. For example:

² A capital letter is used for this 'argument' of the att_value 'clause', to denote a variable

Unitary statement

The small leaf size of Siris causes a low shading effect

Formal statement

att_value(part('Siris',leaf), size,small)causes1way att_value(process(shading),effect,low)

4.2.4.b Causal statements

Given that a causal statement takes the general structure:

X causes Y

Y will usually be a change in the value of an attribute. This change can be captured by using one of the special values 'increase', 'decrease', 'change' or 'no change'. X may also be a change in the value of an attribute (again taking one of the four special values) or may be a process or an action. The grammar also allows objects to be a cause (for example, sheep cause soil creep) but this is incomplete, containing implicit information. The complete statement (for example, 'trampling by sheep causes soil creep') is always preferable, but the feature is retained in the grammar for cases in which it is not known how an object causes a change, but simply that something about its presence does.

Thus a causal statement can take one of the following forms:

Attribute statement causes attribute statement	e.g. <i>A decrease in stem thickness causes a decrease in stem strength.</i>
Process causes attribute statement	e.g. <i>Soil erosion causes a decrease in soil fertility</i>
Action ³ causes attribute statement	e.g. <i>Ploughing slopes causes an increase in soil erosion</i>
Object causes attribute statement	e.g. <i>Cows cause an increase in soil compaction</i>

Causal statements can be divided into those for which the reciprocal is also true and those for which it is not. For example, both the statements:-

- a) *An increase in atmospheric temperature causes an increase in germination rate⁴;*
and
b) *A decrease in atmospheric temperature causes an decrease in germination rate*

might be held to be true. As they are reciprocals of one another it is desirable to be able to enter just the one into the knowledge base and be able to infer the other.

However, the reciprocal is not always true. For example, while it may be true that:

An increase in intensity of rainfall causes an increase in surface moisture.

it is not the case that

A decrease in intensity of rainfall causes a decrease in surface moisture.

because it is not rainfall, but evaporation, drainage, etc. which dictate the rate of drying.

To capture this difference, two versions of causes are supported: **causes1way** and **causes2way**.

For these six examples, these result in the formal statements:

³ An action should usually be attached to at least one object, *ploughing to field, harvesting to crop* etc. Although the grammar allows an action to stand alone, this is less useful in terms of the information it imparts and should therefore be used only sparingly.

⁴ rate – percentage seed germinating within a given time

1	<i>att_value(stem, thickness, decrease) causes2way att_value(stem, strength, decrease)</i>
2	<i>process(soil, erosion) causes1way att_value(soil, fertility, decrease)</i>
3	<i>action(ploughing,slopes) causes1way att_value(process(soil,erosion),rate,increase)</i>
4	<i>Conifers causes1way att_value(process(soil,acidification),rate,increase)</i>
5	<i>att_value(atmosphere,temperature,increase) causes2way att_value(process(seed, germination), rate, increase)</i>
6	<i>att_value(process(rainfall),rate, increase) causes1way att_value(surface_ moisture, value, increase)</i>

Not all causal statements describe a change in the value of an attribute. Y may also be a process in other words, something (an attribute statement, an action, a process or an object) causes a process to take place, for example;

Object causes process *Conifers cause soil acidification*

Y may also be an action, for example;

Attribute statement causes action *An increase in pest numbers causes the application of pesticides.*

Nevertheless, we are creating ecological knowledge bases, and as actions are usually determined by a far more complex set of factors than ecological conditions alone, (a farmer's decision to apply pesticides, for example, will be dependent, not only on the increase of pest numbers, but also on economic constraints such as affordability, availability and time) and thus an action will rarely simply be an effect of an ecological state alone.

4.2.4.c Comparison statements

Comparison statements compare the relative value of a pair of objects.

The formalised comparison statement takes the following form:

comparison(Attribute, Object1, Comparison_type, Object2)

Comparison statements may be self evident, for example:

Bamboo grows faster than fruit trees.

Frequently, however, comparison is implicit, usually against an implicit 'norm'. The unitary statement:

Bans leaves decompose slowly

can be interpreted as being a comparison with the average rate of leaf decomposition. This kind of implicit comparison is, however, best captured as an attribute value statement, in this case:

att_value(process(part(bans, leaf), decomposition) rate, slow)

The only instances in which implicit comparison may best be represented as comparative statements are those in which there are clearly only two possible circumstances. So, for example, the statement:

Forests with closed canopies cast deeper shade

is a genuinely comparative statement and might be more explicitly stated as:

Unitary statement:

Forests with closed canopies cast more shade than forests with open canopies.

Formal statement:

*comparison(depth_of_shade, closed_canopy_forest, greater_than,
open_canopy_forest)*

4.2.4.d (User defined) Link statements

This link is specified as (user defined) link in order to distinguish it from certain reserved terms that act as links between two parts of a statement. For example, the reserved terms *causes1way*, *causes2way*, *greater_than*, *less_than*, *same_as* link the two halves of a formal statement together. User defined links on the other hand, are terms selected by the user.

User defined link statements take the basic form:

link(link_type, Object1, Object2)

Ecological relationships such as *cows eat grass* and *bees pollinate clover* are good examples of user defined link statements, and would be expressed in formal language thus:

*link(eat, cows, grass)
link(pollinate, bees, clover)*

User defined link statements are also used when the knowledge cannot be expressed using any of the other three types of statement, e.g. *Tithonia diversifolia is found on unfertile ground* or *oak is used for timber*. These would be formally represented as:

*link(is_found_on, 'Tithonia diversiflora', unfertile_ground
link(is_used_for, oak, timber)*

The grammar includes one type of link statement, in which the link type is 'influences'. In this instance the link may be between any combination of objects and processes. Influence relationships are very closely related to causal relationships. However, in an influence relationship there is no information on what attribute of the object or process is affected or how it is changed, e.g. *Trees influence crop yield*⁵. Where there is information on the result of the influence, this should be captured as a causal statement.

4.2.4.e Representation of conditions

Statements may be conditional. Conditions in the formal language can take the form of attribute value or negative attribute value statements, link statements, causal statements and comparison statements. All these statements are as previously described and may also occur as the main part of a formal statement.

Conditions may be linked by **and** and **or**. For example:

*Crops are prone to lodging if (there is a strong wind **and** crop roots are exposed) **or**
(there is a strong wind **and** crop stems are weak),*

or equally:

*Crops are prone to lodging if (crop stems are weak **or** crop roots are exposed) **and**
(there is a strong wind).*

4.3 FORMAL TERMS SPECIFICATION

Structural manipulation of the knowledge base involves the development of consistent and useful glossaries of terms and ensuring the consistent use of those terms within unitary statements.

⁵ In formal language this is expressed as: *link(influences, trees, crop_yield)*

The development of glossaries of attributes, processes, actions, values and link types is automated in AKT. Objects are automatically identified but the user is required to identify 'type of' relationships and 'part of' relationships through the creation of object hierarchies and application of the formal grammar respectively.

However, while the mechanisms for creating glossaries and hierarchies are straightforward, the management of hierarchies and glossaries is a demanding task. The consistent use of terms has a profound impact on the utility of the resulting knowledge base, particularly in relation to the use of automated reasoning tools. The consistent use of terms is facilitated by:

- minimising the number of object, attribute, process, action, value and link terms used;
- ensuring the consistent use of values for attributes and
- providing definitions for each term, such that their use is transparent and can be assessed by the knowledge base developer or other users.

The creation of object hierarchies makes further demands. Sets of objects can be hierarchically classified in many different ways for different purposes. Indeed it may not be possible to develop a single classification of a set of objects that is appropriate for all the intended uses of a knowledge base. For that reason it is possible for an object to appear in several hierarchies, although it is not possible for an object to appear more than once in the same hierarchy.

4.4 DIAGRAM BASED REPRESENTATION

4.4.1 INTRODUCTION

The representation of knowledge as a diagram provides a powerful means of helping to ensure that the knowledge set that is represented is comprehensive and coherent and therefore useful.

The diagramming approach to representation is not constrained by linearity. A set of unitary statements may be legitimately explored from any point in any direction. As a result, diagrams provide a more succinct representation of knowledge than textual approaches.

A further advantage of diagrammatic representation is that statements entered through the diagramming interface are automatically formalised (see Chapter 8).

The diagramming interface in AKT uses the grammar to form a precise formal statement without the user needing to understand the syntax of that formal statement. Diagrams developed on the basis of this syntax display the following features:

- every node in the diagram represents an object, a process, an action or an attribute of an object or process;
- every node in the diagram is fully labelled;
- information is attached primarily to links rather than nodes;
- the meaning of every link is explicitly stated;
- a pair of linked nodes represents information that corresponds to a unitary statement.

4.4.2 NODES

4.4.2.a Classification of nodes

Four types of node are identified:

Object nodes represent things, or more commonly, groups of things. An object is something that occurs physically e.g. an oak tree, or in an abstract sense, e.g. climate.

Process nodes represent things that happen e.g. seed germination or soil erosion. Depending on the time scale involved, a process might be called an event (it is clearer to refer to death of an organism as an event rather than a process) but an event is simply a special type of process that occurs instantaneously. From the point of view of the semantics of a diagram, they are equivalent.

Action nodes refer to management actions. An action is a type of process, and actions could be represented as processes, however, an action is distinct in that there is a human directly responsible for causing the process to occur.

Attribute nodes refer to a particular attribute of a process or an object. An attribute is something relating to an object or process that might be measured e.g. height of oak tree, rate of germination, colour of flower. Experience shows that attribute nodes are more frequently used than object, action or process nodes.

4.4.2.b Labelling of nodes

Node names that contain implicit meaning are ambiguous. For example, nodes in influence diagrams might be given the name of objects (e.g. pests) or processes (e.g. grain set), while the links mean 'influence(s)'. However, 'pest influence grain set' is incomplete information: all that it explicitly states is that some attribute of pests influences some attribute of grain set.

Where this attribute is known, it has to be stated rather than implied, e.g. 'pest population size' 'influences' 'timing of grain set' if knowledge is to be explicitly stated. Complete information about the meaning of each node is critical for explicit representation of knowledge in the diagram.

4.4.3 LINKS

As in the statement interface, there are two types of link, the reserved term links (in this case only the reserved term links *causes1way* and *causes2way* can be represented diagrammatically) and the user defined links.

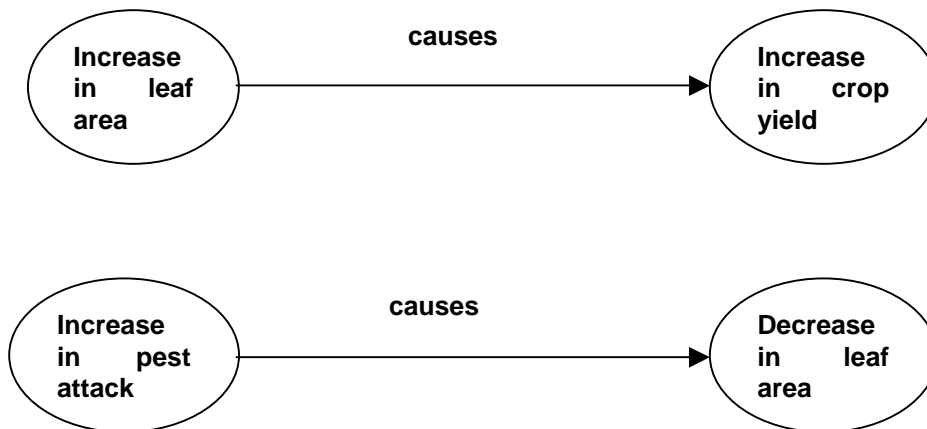
4.4.3.a Attaching information to links

It is a normal convention in creating node and link diagrams to attach the information to nodes, so that links are rarely explicitly labelled. This approach may be intuitive but imposes constraints on the diagram. When this approach is used in constructing, for example, causal diagrams the links simply means 'causes', while exactly what is caused is specified in the nodes. So, for example, the node 'increase in leaf area' might be linked by a 'causes' link to 'increased crop yield'. This severely constrains further additions to the diagram. The fact that pest attack reduces leaf area might need to be included. This would only now be possible by adding a further node to the diagram 'decreased leaf area'. It is not desirable to have two nodes referring to leaf area, one to increase and one to decrease. Changes are not necessarily only expressed in terms of changes in the quantity of an attribute. A change may also occur in the presence or absence of an object or process. Under this scheme there may be clumsily labelled nodes e.g. 'occurrence of ovulation' or 'disappearance of frost'.

These problems are overcome by attaching the information about changes occurring to the link rather than the nodes. So 'increase in leaf area' 'causes' 'increase in crop yield' becomes

'leaf area' 'increase in causes increase in' 'crop yield' (see Figure 4.1). Over a large diagram the discipline of attaching the majority of information to the link simplifies diagram construction.

a)



b)

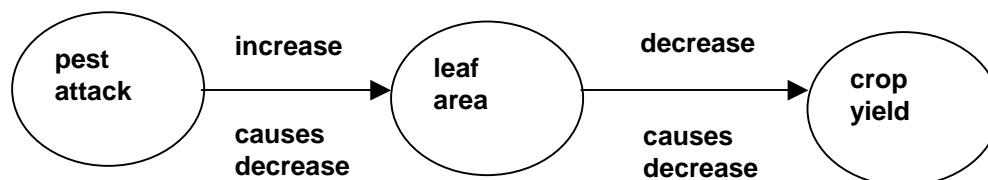


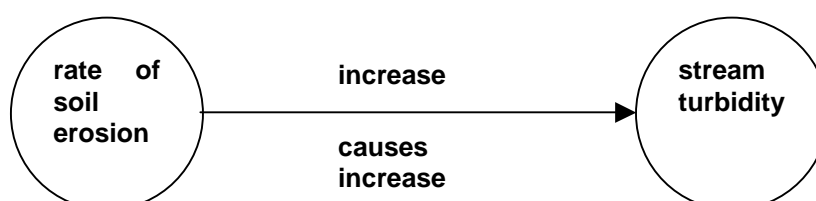
Figure 4.1 Two ways of expressing the same thing; a) shows the information attached to the nodes whilst b) shows the bulk of the information attached to the links and the advantages in reducing the number of nodes that ensues.

4.4.3.b Stating the meaning of links

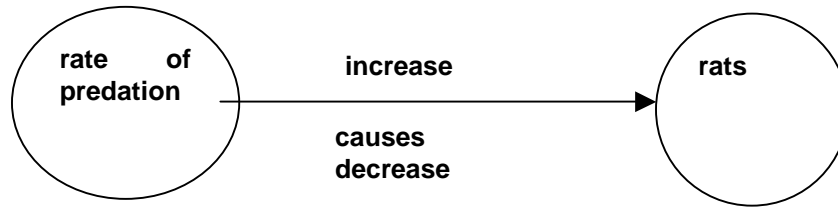
If a diagram is to represent an explicit statement of knowledge, links can only remain unlabelled if they all have exactly the same meaning and this meaning is clearly defined. This makes the mixing of different types of link in a single diagram impossible and is extremely restrictive where the links are considered to be the primary source of information. The explicit statement of the meaning of each link facilitates the flexible development of a representative diagram.

4.4.3.c Linguistic correspondence of linked pairs of nodes.

If two nodes connected by a link represent useful knowledge, there will be a correspondence to a meaningful English sentence. This sentence will, in fact, comply with the definition of a unitary statement (see Part 4.1.2). Where this correspondence does not occur, either one or both nodes or the link must be inappropriately labelled.



can be read as 'An increase in the rate of soil erosion' 'causes' 'an increase in stream turbidity' which makes sense in English.



would read as 'An increase in the rate of predation' 'causes' 'a decrease in rats'. This is understandable but incomplete. To formalize this statement one needs to be clear what it is about the rats that decreases, - in this case it is the *number* of rats or their *population size* which decreases, not the rats themselves. Thus this should be explicitly stated by re-labelling 'Rats' as 'Rat population size' or 'Rat number'.

The node 'rate of predation' is also incompletely labelled and clearly refers to the rate of predation on the rat population. It can be argued that this can be assumed from the context of the diagram, but, where a diagram is created as a means of generating a coherent knowledge base, this cannot be assumed. Predation of other species may occur elsewhere in the knowledge base which can probably not be assumed as being of the same nature as, and therefore equivalent to, the predation of rats. Therefore the first node should properly read 'Rate of predation on rat population'.

4.4.3.d Labelling of links

It will often be possible to express the meaning of a particular link in different ways. However, using different terms for the same concept reduces the comparability of the resulting diagrams, particularly given that differing terms for the same concept will rarely be exactly equivalent.

Fungal wilt causes yield loss

is similar to:

Fungal wilt results in yield loss

and:

Fungal wilt leads to yield loss

and not very different from:

Fungal wilt is followed by yield loss

which might be restated as:

Fungal wilt happens before yield loss

The above example illustrates the importance of ensuring a consistent use of terms both within a diagram and between diagrams in order to attain unambiguous comparability of links.

4.4.4 SUBSETS OF DIAGRAMS

Experience in the creation of diagrams as a means of producing coherent knowledge bases suggests that diagrams containing more than 40 nodes tend to be increasingly difficult to interpret. This is in part the result of the balance that must be achieved between the size and therefore legibility of nodes and text associated with links, and standard paper or computer screen size. However, even where large boards or paper are used, diagrams rapidly become

unsustainably complex. It is the clear representation of links rather than nodes that presents difficulties. Typically a node will be linked to between two and five other nodes. With an increasing number of nodes it becomes increasingly difficult to place nodes near to all the other nodes to which they are linked: links have to travel further across the diagram, crossing increasing numbers of nodes and links until the diagram becomes impenetrable. The placement of nodes, the technique used for representing the meaning of a link and the arrow type used to link nodes all have an impact on the number of nodes and links that can be successfully represented, nevertheless, fundamental limitations will still be reached.

In order to overcome some of these problems, sub-diagrams can be built up from the main diagram. It is possible to make a sub-diagram of the pathway between any two nodes on the diagram. It is also possible to build up a diagram by selecting any number of nodes greater than one, for which the system will then produce a sub-diagram showing all the connections.

4.5 CREATING KNOWLEDGE BASES THROUGH THE COMBINATION OF EXISTING KNOWLEDGE BASES

In parts 4.1 – 4.4 the creation of knowledge bases from scratch has been described. However, it is also possible, and may frequently be desirable, to create a knowledge base through the combination of two or more existing knowledge bases, for example, knowledge in the same domain but from different sets of sources.

In principle the combination of two knowledge bases is a straightforward task. The process of merging knowledge bases is essentially one of ensuring consistency of terminology.

Although effective software support for merging knowledge bases is available, the process is demanding for two reasons. In the first place, two knowledge bases in the same domain are unlikely to use precisely the same terminology, especially where created by different knowledge base developers. Although many equivalent terms (for example, alternative names for the same species) can be identified without difficulty, equivalence for other terms is much more problematic and, even where definitions of terms are provided, can only be resolved through reference to the original knowledge base developer. As a result, it is doubtful that consistent use of terminology across knowledge bases is practically possible unless strict protocols for the use of terminology in the creation of the knowledge bases in the first instance are developed and applied.

A further problem is encountered where the content of the knowledge bases to be merged has been previously formally represented. The process of formal representation involves many decisions about the most appropriate approach to representation. When this process has been undertaken by different individuals for different knowledge bases these decisions will not have been consistently applied to both sets of knowledge. The implications of inconsistency in formal representation for automated reasoning are likely to be serious.

Thus, although it is possible to create a large knowledge base by merging two or more smaller knowledge bases, it is much simpler and much more feasible to begin with a large knowledge base and then split it into smaller ones.

Key points of Chapter Four

- Unitary statements are the basic units of statements in a knowledge base.
- Contextual information can be stored with knowledge statements.
- The definitive clause grammar has been tailor-made for representing knowledge in statements that can be processed by the computer.
- Six elements form the basis for representing knowledge: object, process, action, attribute, link and value.
- Formal statements can be one of the following: attribute-value statement, negative attribute-value statement, causal statement, link statement, comparison statement.

Chapter Four – Knowledge representation

- Determining the elements (formal terms specification), the structure of the statement and statement type are important steps in formalising a knowledge statement.
- Knowledge statements can be entered using the diagram interface through nodes and links.
- Knowledge bases can be split and combined as and when necessary.

CHAPTER FIVE - KNOWLEDGE BASE MANAGEMENT

5.1 INTRODUCTION

Now that the relationships between stages in the process of knowledge base creation have been outlined, this section discusses iterative evaluation of the knowledge base during its development. The repeated evaluation of the growing knowledge base to drive further elicitation is distinct from the evaluation of the representativeness of the knowledge base considered to be complete in relation to the objectives for its creation (see 6.1) and the correctness and utility of its content. While these processes have tasks in common, their motivations and outputs are different.

Iterative evaluation of the knowledge base occurs throughout the formulation of unitary statements, formal representation and the development of lists of formal terms and hierarchies. The process demands consideration of both individual unitary statements and sets of unitary statements as well as the specification of formal terms and the relationships between formal terms.

So, for example, the statement:

Orange caterpillars can cause sickness in livestock

is incomplete and probably contains implicit meaning. Reference to the knowledge source shows that it can be more completely represented as:

Feeding orange caterpillars to livestock can cause sickness

This still does not really capture the meaning of the source knowledge, which is that:

Feeding leaves to livestock can cause sickness to those livestock if there are orange caterpillars on the leaves.

Whilst this is now explicit representation of a piece of knowledge, the word 'can' along with a lack of any information about the circumstances under which the statement is held to be correct makes it of limited utility. This suggests a need for further knowledge elicitation, possibly revealing that this always occurs, i.e.

Feeding leaves to livestock causes sickness to those livestock if there are orange caterpillars on the leaves.

This statement can now be meaningfully formalised. Leaves, livestock and orange caterpillars are objects. 'Feeding' is an action. Sickness might be viewed as a process or to be a change (decrease) in the value of an attribute (health). This statement might be formalised in many different ways, all capturing essentially the same meaning but with different emphases, for example:

action(feeding, leaves, livestock) causes att_value(livestock, health, decrease) if att_value(orange_caterpillars, location, on_leaves¹)

or:

action(feeding, leaves, livestock) causes att_value(livestock, health, decrease) if att_value(caterpillars, location, on_leaves) and att_value(caterpillars, colour, orange)

In other contexts a more precise definition of the results of consuming orange caterpillars may be necessary - sickness may be considered to be a particular type of decline in health that has particular consequences, the difference between the sickness caused by orange caterpillars and other sicknesses might be significant. All these factors will affect the formal

¹ Note that in this case 'on_leaves' is seen as a value of the attribute 'location'

representation of the knowledge and depend on the consideration of this unitary statement with other unitary statements in the knowledge base.

Considering a unitary statement in the context of the other unitary statements in the knowledge base may reveal repetition, contradiction, incomplete sets of unitary statements, or inconsistent use of the terms. In this example, there are statements that:

Green caterpillars cause sickness in livestock

Black caterpillars cause sickness in livestock

There is also information in the object hierarchies that there are three types of caterpillars; orange, black and green. If it is assumed that this is the finite set of caterpillars in this context then the three can be replaced with the single statement that:

Feeding leaves to livestock causes sickness if there are caterpillars on the leaves.

If it is suspected or found that there are other types of caterpillars that may not, or do not, cause sickness if fed to livestock, the three statements may be replaced by the single statement:

Feeding leaves to livestock causes sickness if (there are caterpillar on the leaves) and (those caterpillars are green, or those caterpillars are black or those caterpillars are orange).

Alternatively, green, orange and black caterpillars might be classified as being poisonous caterpillars such that:

Feeding leaves to livestock causes sickness if there are poisonous caterpillars on the leaves.

Considering this unitary statement in relation to other statements may also demand consideration of linkages - for example are there statements, which describe the consequences of a decline in the health of livestock? Apparently contradictory statements may be identified and resolved.

For example, does the statement that:

Leaves of fodder trees that are attacked by caterpillars do not cause sickness in livestock

mean that caterpillars do not cause sickness or does it mean that the leaves of those trees that are attacked by caterpillars do not cause sickness, provided that there are no caterpillars on them?

At this juncture, consistency of use of terms can be ensured. The term 'livestock' is, in the current context, defined in the object hierarchy as cows, goats and buffalo. Do all the statements that use the term 'livestock' refer to all three of these, or should some be replaced, for example, with 'cattle' or 'milking livestock' (i.e., cows, goats or buffalo that are currently lactating)? Precise use must be considered in terms of precise definition. The term 'feeding' is used in this statement. It may be that this term is defined as meaning the action of providing fodder for livestock that are stalled. This might be distinguished from the process of 'browsing' where livestock actively seek out and select fodder on trees. This process of selective browsing may mean that fodder effects relating to stall-fed livestock do not occur for free-range livestock.

This example illustrates something of the range of activities that can be involved in iterative evaluation and improvement of the knowledge base. The following sections anatomise some of the key processes occurring in iterative evaluation.

5.2 EVALUATING INDIVIDUAL UNITARY STATEMENTS

Individual unitary statements may be evaluated in terms of validity of representation, relevance, utility and ambiguity.

5.2.1 VALIDITY OF REPRESENTATION

Some evaluation of the validity of representation (to check whether the knowledge statement has been correctly encoded) is built into the process of creating formal statements directly or through diagramming. This is due to the restricted structure of the formal grammar, the use of a parser and the stylised English generation in formal representation by AKT.

It cannot be overemphasised how important it is that when formulating unitary statements prior to creating a formal statement, these unitary statements should be rigorously checked to ensure that they conform to the definition of a genuine unitary statement.

For example, the statement:

Nutrient uptake by fodder trees planted on crop land causes a decrease in soil fertility.

Can be broken down into two statements:

Planting fodder trees causes an increase in soil nutrient uptake

An increase in soil nutrient uptake causes a decrease in soil fertility.

Statements which do conform to the fundamental definition of a unitary statement may often be improved through use of condition. For example:

Thorny leaves are unpalatable.

may be stated as:

Leaves are unpalatable if they are thorny.

This is important because of its subsequent impact on formal representation. Formal representation of the former would demand that 'thorny leaves' be treated as an object. This is less flexible than a formal statement in which leaves are the object and their palatability and thorniness are two attributes of that object.

5.2.2 RELEVANCE AND UTILITY

Evaluation of articulated knowledge against objectives provides an important means of maintaining the quality (fitness for purpose) of the knowledge base. Effective filtering of articulated knowledge irrelevant to the knowledge base during knowledge elicitation and formal representation greatly reduces the need for subsequent rationalisation. Unitary statements may be relevant, in as far as they are concerned with the domain under consideration, but irrelevant in that they are not useful (in the context of the knowledge base as a whole). So, for example, the statements:

Manure production influences crop production

and

Some species of tree fodder have a beneficial effect on milk production.

Might be deleted from a knowledge base in which both relationships were expanded in much greater detail in other parts of the knowledge base.

5.2.3 AMBIGUITY

Unambiguous (precise and complete) articulation and representation of a unitary statement is required if a useful knowledge base is to be created. Most ambiguity encountered in a knowledge base is resolvable, being an artefact of articulation and representation. However, knowledge may be inherently ambiguous, so some intrinsic ambiguity will remain in any knowledge base.

The level of resolvable ambiguity in the knowledge base is influenced by:

- the nature of the knowledge source;
- the nature of the knowledge elicitation process;
- the type of interface used for entering the knowledge (i.e. the statement dialog or the diagram dialog);
- the experience of the knowledge base developer; and
- the clarity of objectives for creating the knowledge base.

Resolvable ambiguity is most frequently exposed during the processes of formal representation and the creation of the glossaries and hierarchies. Both processes demand a precise use of terms. Where ambiguity is the result of inadequate unitary statements, it may be resolved through reference to interview material. However, where ambiguity is a result of incomplete or muddled articulation by the knowledge source, further knowledge elicitation is demanded.

Unambiguous representation depends on:

- **complete specification of the meaning** of a unitary statement;
- complete representation of the **context of application** of the unitary statement; and
- **precise use of terms** within that unitary statement.

5.2.3.a Complete specification of meaning

Unitary statements may frequently contain an implied meaning. Even where it can be assumed that this is understood by potential users of the knowledge base, it may significantly constrain automated reasoning.

For instance, while it might be valid to assume that users understand that the statement, in a knowledge base about tree fodder,

Celtis australis leaves can cause sickness

means that feeding the leaves to livestock may cause sickness in livestock, automated reasoning will not pick up this implication. Even simple reasoning such as answering the question:

What is the consequence of feeding Celtis australis leaves to goats?

cannot, therefore, be automated. As a result, formal representation must state explicitly any implied meaning, in this case maybe as:

*action(feeding, part('Celtis australis', leaves), livestock) causes1way
att_value(livestock, health, decrease)*

5.2.3.b Context of application

Knowledge is inescapably contextual at some level, even though the AKT approach to knowledge base creation is based on the disaggregation of sets of unitary statements. Recording conditional information associated with a unitary statement provides important contextual information about the application of that unitary statement. However, exhaustive recording of the context of application is impractical: some level of understanding from common knowledge for the evaluation of the knowledge base as a whole must be assumed. Judgement of the completeness of conditional information is subjective and dependent upon context.

It is important that where a statement is considered to be unconditional this is explicitly recorded.

5.2.3.c Precise use of terms

Another source of ambiguity is the imprecise use of terminology. The definition of terms and the consistent use of terms are discussed later (see 5.3.4). Here, the concern is with the imprecise use of terms within a statement. A widespread example in existing knowledge bases is the use of the term 'shade'. A unitary statement often refers to one aspect of shade (for example a change in light levels, light quality or temperature) rather than a composite of all shade effects.

5.2.3.d Intrinsic ambiguity

In a fundamental sense, virtually all knowledge is ambiguous at some level. In the current context, however, intrinsic ambiguity is of interest where it has an impact on the practical utility of knowledge.

Ambiguity is intrinsic where the source of an ambiguous statement cannot resolve the ambiguity in that statement. This is generally because the ambiguity is not clear to the informant, frequently because the issue in question is at the margins of his or her comprehension. Intrinsic ambiguity often occurs in the meaning of terms within a statement. Clearly, distinguishing intrinsic ambiguity from the inability of the knowledge base developer to understand the concepts under consideration is a matter of judgement.

5.3 EVALUATING SETS OF UNITARY STATEMENTS

The content of a knowledge base is greater than the sum of the content of the individual unitary statements. As a result, iterative evaluation must include the consideration of sets of unitary statements as well as individual statements. Sets of unitary statements should be evaluated in terms of:

- repetition
- contradiction
- completeness, and
- consistency in use of terms.

5.3.1 REPETITION

A compact knowledge base is significantly more tractable, and, therefore, more useful, than a less compact version. Compaction is achieved by identifying and removing repetition in the knowledge base. Two types of repetition can be identified, strict repetition and deducible repetition. Strict repetition is where a piece of information is stated more than once. Deducible repetition is where a statement in the knowledge base can be deduced from other statements in the same knowledge base and is therefore superfluous.

5.3.1.a Strict repetition

AKT will not allow exactly the same statements from various sources to be added to the knowledge base. The user can add extra sources for a single statement if it is repeated by different informants. While some repetition may be self evident, the identification of repetition will best be achieved by reference to the objectives of the use of the knowledge base. So, two different unitary statements using different terms but capturing very similar information may or may not be defined as repetition depending on the potential impact of the different formulations on the use of the knowledge base.

Selection facilities to identify all statements concerned with a particular formal term or combination of formal terms reveal much repetition. Comparing sets of formal statements also reveals repetition, as formal representation often causes convergence of apparently distinct statements. Such comparisons are most effective when care has been taken to minimise the use of equivalent terms. Common causes of repetition are inconsistent use of terms, or spelling mistakes. For example, the following three statements would all be accepted by the knowledge base, but would all be identical, were it not for inconsistency and carelessness:

Trampling by goats causes soil creep
Trampling by goat causes soil creep
Trampling by gaots causes soil creep

It is particularly important to decide early on whether objects should be entered into the knowledge base in the singular or plural - in the above example, whether statements should refer to 'goat' or 'goats'.

5.3.1.b Deducible repetition and the use of hierarchies in compacting the knowledge base

The identification of hierarchical relationships between objects provides a means of

- capturing the hierarchical nature of knowledge;
- enabling the inheritance of properties by objects up and down the hierarchy, which facilitate the development of a compact knowledge base without loss of information; and
- facilitating hierarchically structured exploration of the knowledge base.

5.3.1.c The use of hierarchies in compacting the knowledge base

Hierarchies provide a means of compacting the knowledge base. They allow knowledge to be recorded at its most general level of application, yet be used to consider more specific instances. This is achieved by considering hierarchical relationship between terms. If, for example, 'wheat', 'barley', 'maize' and 'fava beans' are identified as being annual crops within a hierarchy, the information that annual crops only live for one year is best recorded as generic information about annual crops, rather than for each type of annual crop individually². This influences the size and tractability of the knowledge base. Compare Tables 5.1 and 5.2. Each captures the same information; in Table 5.1 the information is explicitly stated, in Table 5.2 it is more implicitly captured.

² This is known as 'property inheritance' and is discussed more fully in Chapter 1.7.

Table 5.1 A complete set of 43 unitary statements

Crops are economically useful	Legumes photosynthesise
Legumes are economically useful	Chick peas photosynthesise
Root crops are economically useful	Pigeon peas photosynthesise
Cereals are economically useful	Cowpeas photosynthesise
Chick peas are economically useful	Legumes have roots
Pigeon peas are economically useful	Chick peas have roots
Cow peas are economically useful	Pigeon peas have roots
Crops are deliberately cultivated	Cowpeas have roots
Legumes are deliberately cultivated	Legumes have leaves
Cereals are deliberately cultivated	Chick peas have leaves
Root crops are deliberately cultivated	Pigeon peas have leaves
Chick peas are deliberately cultivated	Cowpeas have leaves
Pigeon peas are deliberately cultivated	Legumes transpire
Cowpeas are deliberately cultivated	Chick peas transpire
Crops are plants	Pigeon peas transpire
Legumes are plants	Cowpeas transpire
Cereals are plants	Legumes are crops
Root crops are plants	Roots crops are crops
Chick peas are plants	Cereals are crops
Pigeon peas are plants	Chick peas are legumes
Cowpeas are plants	Pigeon peas are legumes
	Cowpeas are legumes

Table 5.2 A compacted statement of knowledge based on a hierarchical structuring of the objects, reducing the number of statements to 13

1 Crops are economically useful	8 Legumes are crops
2 Crops are deliberately cultivated	9 Root crops are crops
3 Crops are plants	10 Cereals are crops
4 Legumes photosynthesise	11 Chickpeas are legumes
5 Legumes have roots	12 Pigeon peas are legumes
6 Legumes have leaves	13 Cowpeas are legumes
7 Legumes transpire	

A mechanism exists for capturing the hierarchical nature of taxonomic statements, therefore all the statements not explicitly stated can be deduced by applying the general rules to lower orders of the hierarchy.

Suppose for example, that in a particular knowledge base:

- four crop species are classified as annual crops;
- these four are the only annual crops represented in the knowledge base; and
- all are recorded as only living one year.

Ensuring that knowledge is recorded at the highest level can greatly reduce the size of the knowledge base.

Deducible repetition can also occur where the implications of linked sets of unitary statements are explicitly stated in the knowledge base. Where a statement contains the assertions that:

- Fertiliser application increases soil fertility if.....*
- An increase in soil fertility causes an increase in crop yield if*

The statement that:

- Fertiliser application increases crop yield.*

is a deducible repetition.

Similarly it can be logically deduced from b) that:

A decrease in soil fertility causes a decrease in crop yield if....

Systematic identification and removal of repetition in a knowledge base may often halve the number of unitary statements in the knowledge base.

5.3.2 CONTRADICTION

Contradictory unitary statements may be of two types - conflicting and inconsistent. Conflicting knowledge is knowledge from two different sources which is contradictory. Inconsistent knowledge is knowledge from a single source which is contradictory. Until contradiction is resolved, either by rejecting one unitary statement in favour of another or by demonstrating that apparent contradiction does not represent actual contradiction, the two contradictory unitary statements can be viewed as being competitive.

Where contradictory knowledge in the knowledge base is identified, it may be resolved through further knowledge elicitation or flagged within the knowledge base. Contradictory unitary statements may become apparent at any stage during the creation of the knowledge base. Equally, contradictions may only become apparent once the implications of the knowledge have been illuminated through reasoning with that knowledge.

Once contradiction has been identified it may be resolved by one of two means.

- Apparent contradiction may be resolved through clarification of the meaning of contradictory statements: in particular, two apparently contradictory statements may be distinguished by specification of the conditions under which they are held to be true.
- Contradictions may be resolved by demonstrating that there is a significant difference between the reliability of two statements such that the less reliable statement can safely be rejected in favour of the more reliable. (However, it is important to remember that judging 'reliability' of statements subjectively by the knowledge base developer is difficult. It is best to opt for the rule of thumb: consider true unless proven wrong.)

Box 1. An example of apparently inconsistent unitary statements

In an investigation into forest gardens in the Kandy district of Sri Lanka (Southern, 1994) the following statements were given by the same informant on different days:

Vegetable diseases are reduced in the shade (Abeyasinghe, 21.4.92)
Disease problems are not influenced by the degree of shade
(Abeyasinghe, 26.4.92)

However, after returning to the source to clarify this apparent contradiction, it became clear that the second statement related to a conversation about a part of the garden where vegetables were not grown. Thus by appending the conditions in which the statements were relevant, they were no longer contradictory.

Inconsistent knowledge should, in principle, be resolved by one of the above mechanisms. Conflicting statements may frequently not be resolved: under such circumstances the two unitary statements may be flagged by attaching a Memo to both statements, stating that these are alternative views. Automated reasoning tools are useful for identifying some types of inconsistency, for example the 'inconsistent att_value statements' and the 'inconsistent causal statements' tools provided in the macros.

5.3.3 COMPLETENESS

The incompleteness of a set of unitary statements (as opposed to an individual statement: see 5.2.3) is assessed through the identification of apparent gaps in the knowledge in relation to objectives. Gaps may occur in a knowledge base because the knowledge needed to fill those gaps is unknown or because the relevant knowledge has not been elicited. Therefore, the identification of gaps demands further knowledge elicitation and, if the necessary knowledge is found to be available, addition to the knowledge base. Completeness can only be defined in relation to objectives in the creation of the knowledge base. Even then, iterative evaluation of completeness tends to be subjective.

Generating diagrams provides a powerful means of facilitating the identification of gaps in sets of knowledge. For example, a diagram constructed as a result of one of a series of interviews with farmers in Nepal provided a set of questions for further interviews (such as ‘what are the stem strength properties of the different crop species grown and the different varieties of each species?’; ‘what influence does crop head size have on crop yield?’; ‘What are the consequences of an increase in straw height?’; and ‘does an increase in shade always result in an increase in pest incidence?’). This helped to identify topics for further discussion with the informant.

5.3.4 CONSISTENCY AND PRECISION IN THE USE OF TERMS

Consistent and precise use of terminology is important if the knowledge base developer is to make effective use of the knowledge base. Comparison of the use of the same term in different statements ensures the consistency of meaning. Comparison of apparently similar terms identifies overlaps in terminology.

Terminology in knowledge systems about agroforestry is rarely consistent or precise. A demand for an exacting consistency when creating knowledge bases may lead to a proliferation of terms. There is then a danger of increasingly unrepresentative use of those terms by source communities. Here again the strategy to be taken depends on objectives in the creation of the knowledge base. They may be primarily to study the current state of knowledge of a target community, or to develop a knowledge base for use in providing decision support.

5.4 EVALUATING HIERARCHICAL STRUCTURES

As well as evaluating and modifying unitary statements, iterative evaluation of the knowledge base demands evaluation of the formal terms specified, the relationships between those formal terms and their definitions.

5.4.1 FORMAL TERMS

Identifying and removing repetitious terminology can significantly increase the utility of the knowledge base. The lists of formal terms available in AKT encourage consistent use of terminology, particularly through formal representation. Nevertheless, regular comparison of the various formal terms may reveal overlapping terminology. The ability to identify equivalent terms for the same object (i.e. synonyms), provides a useful means of allowing a natural articulation of knowledge, while still identifying equivalent meaning. Synonyms can also be applied to processes, attributes, actions, values and links.

5.4.2 RELATIONSHIPS BETWEEN OBJECTS

The development of object hierarchies is one of the most demanding tasks in the creation of the knowledge base (see 4.3). Regular evaluation of object hierarchies, including testing those hierarchies on source communities has proved to be valuable.

Equally, assessment of the set of attributes that are related to a particular object or process, and the set of values that are linked to an attribute, is important in ensuring consistent terminology.

In one knowledge base for example, two separate objects, 'leaves' and 'manure' have an attribute 'texture'. However, the set of values that this attribute can take differs for the two objects ('soft' and 'hard' for leaves, 'loose' and 'firm' for manure). As a result, one or both of the attributes must be renamed to avoid this apparent repetition. By contrast, both 'leaves' and 'manure' also have the attribute 'water content', but the values that this attribute can take ('high' or 'low') are the same for both.

5.4.3. DEFINITION OF FORMAL TERMS

The definition of a formal term may be improved iteratively during the creation of a knowledge base. After each refinement of the definition of a term, all existing uses of that term in unitary statements must be checked to ensure a valid use of terminology.

Key points of Chapter 5:

- Formalised unitary statements must be unambiguous and a valid representation of the original statement.
- Iterative evaluation of sets of statements for avoiding repetition and contradiction and for ensuring completeness and consistent use of terms leads to a concise knowledge base which optimises its utility. Hierarchical arrangement of objects in the knowledge base is a powerful feature which enables valid representation of object classification and this helps compact a knowledge base significantly.

CHAPTER SIX - KNOWLEDGE BASE ANALYSIS

6.1 EVALUATION OF THE REPRESENTATIVENESS OF A KNOWLEDGE BASE

The four-stage knowledge elicitation strategy outlined in Chapter 2 provides a means for eliciting knowledge, which relies initially on particularly knowledgeable, experienced and co-operative informants (or key texts). Users of the resulting knowledge bases will assume that the contents are representative of the knowledge of a defined source community or communities. Therefore any use of the knowledge base must be informed by an evaluation of the representativeness of the knowledge base. This chapter illustrates some of the approaches taken for testing representativeness.

6.1.1 DEFINITIONS OF REPRESENTATIVENESS

Definitions of, and requirements for, 'representativeness' depend on the objectives for the creation and use of a knowledge base. A knowledge base may be evaluated in terms of, for example:

- the extent to which the knowledge in the knowledge base represents a valid abstraction of the knowledge of the sources from which it was elicited; or
- the extent to which the knowledge elicited from a sample of a community and represented in a knowledge base is representative of the knowledge held by all that community.

6.1.2 VALIDITY OF ABSTRACTION

An objective measure may be required, of the extent to which the knowledge base represents a valid abstraction of the knowledge held by the sources. Such a measure may be needed if, for example, the knowledge base is to be used in planning extension activities targeted at the group of informants.

6.1.3 REPRESENTATION OF THE KNOWLEDGE HELD BY THE COMMUNITY

Knowledge bases may be required, which represent the knowledge of a defined community. Usually, these will be created through knowledge elicitation from a sample of the community's members. Consequently, further knowledge elicitation is needed to ensure that the knowledge base is also representative of the broader community. The content of the knowledge base will be compared with knowledge held by members of the community not interviewed initially.

The extent to which a knowledge base that is assumed or has been shown to be a valid abstraction of the knowledge of the original consultants is also representative of the knowledge of a wider community will depend on:

- the heterogeneity of the knowledge held by members of the community on the domain in question; and
- the impact of the sampling bias when key informants were purposively selected.

6.2 TWO METHODS OF TESTING REPRESENTATIVENESS

There are two accepted methods for testing the representativeness of a knowledge base,

For the first method a questionnaire is drawn up with a list of statements derived from the knowledge base, of which half of these are inverted statements giving the inverse of what the informants actually said, e.g. 'Goats do not eat clover'; 'Soil erosion increases soil fertility' 'Feeding orange caterpillars to cattle improves their health'. The farmers are then asked to agree or disagree with the statement. In general, if 75% or more farmers agree with a statement, then that statement can be regarded as common knowledge.

The second method is to interview a large random sample of people not included in the original knowledge base creation and to ask them open questions, of the type posed to the original informants, taking care not to ask leading questions. The information is then analysed and compared with the original response.

The disadvantage of this second method is that it is time consuming and it makes the analysis of representativeness more difficult. However, this is the more rigorous approach, and it often brings new information to the attention of the knowledge base developer.

The appropriate method to use depends on the circumstances. If the original informants come from the same community as those on whom the representativeness of the knowledge base is being tested then the first approach is sufficient. If, however, the representativeness of a knowledge base created from the information of one community is being tested in other communities, then the second approach is recommended.

Below are sample questions from the two approaches, the first taken from research into tea gardens in northern Thailand (Preechapanya, 1996, Thapa, 1994) the second from research into tree fodder resources in the mid-hills of eastern Nepal.

Method 1:

1. Cattle do not eat the young leaves of *Imperata cylindrica*

- ☐ Agree
- ☐ Agree with conditions
- ☐ Disagree
- ☐ Don't know

Conditions specified:

2. An increase in shading intensity causes a decrease in the softness of tea leaves

- ☐ Agree
- ☐ Agree with conditions
- ☐ Disagree
- ☐ Don't know

Conditions specified:

3. The rate of nutrient transfer to tea roots of *Castanopsis armata* roots is high, if *Castanopsis armata* roots entwines tea roots.

- ☐ Agree
- ☐ Agree with conditions
- ☐ Disagree
- ☐ Don't know

Conditions specified:

Method 2:

Which of the following fodder tree species causes light or heavy tapkan effects on crops?

Fodder tree species	Heavy tapkan	Light tapkan	Don't know
---------------------	--------------	--------------	------------

Nebharo (<i>Ficus roxburghii</i>)			
Badahara (<i>Artocarpus lakoocha</i>)			
Gogun (<i>Saurauia nepaulensis</i>)			
Utis (<i>Alnus nepalensis</i>)			
Amalla (<i>Embilica officinalis</i>)			
Rato siris (<i>Albizia julibrissin</i>)			

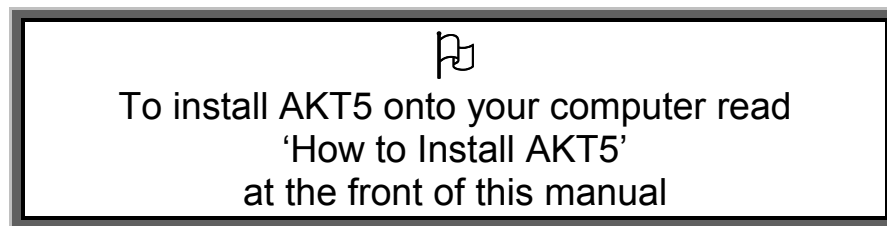
What is it about these fodder trees that causes tapkan effects on crops to be light or heavy?

Fodder tree species	Reasons for causing heavy tapkan effect	Reasons for causing light tapkan effect	Reasons for not knowing
Nebharo (<i>Ficus roxburghii</i>)			
Badahara (<i>Artocarpus lakoocha</i>)			
Gogun (<i>Saurauia nepaulensis</i>)			
Utis (<i>Alnus nepalensis</i>)			
Amalla (<i>Embilica officinalis</i>)			
Rato siris (<i>Albizia julibrissin</i>)			

Key points of Chapter 6:

- Knowledge from a purposively selected sample of informants needs to be checked to see whether this is a true representation of the knowledge of a wider community before this knowledge can be used in research and extension programmes.
- Two approaches are available for testing representativeness of knowledge in a knowledge base:
 - 1. A questionnaire listing a selection of the statements with the options 'Agree/Disagree/Don't Know';
 - 2. Open ended questions given to a selection of new informants to compare their answers with those already given and used in the knowledge base.
- Stratification of the source community for testing the representativeness of the knowledge base can reveal differences in the source community regarding knowledge held by different groups of community members.

CHAPTER SEVEN - SOFTWARE MANUAL FOR AKT



7.1 INTRODUCTION

AKT provides an environment for the creation of a knowledge base, by allowing you to enter, view and edit knowledge, as well as enabling automated reasoning with the contents of the knowledge bases created.

Each knowledge base created with AKT includes the following information:

- the unitary statements
- source details for each statement
- lists of formal terms, their definitions and synonyms
- memos on individual sources, individual statements and the knowledge base as a whole
- records of any object hierarchies developed
- records of any topics and topic hierarchies developed
- full information about the structure and content of any diagram sets in the knowledge base

Each knowledge base created, is stored as a set of Prolog facts and is managed via the **KB** menu (Figure 7.3 below).

7.2 GETTING STARTED

On opening AKT the first image will be the title page below (Figure 7.1).

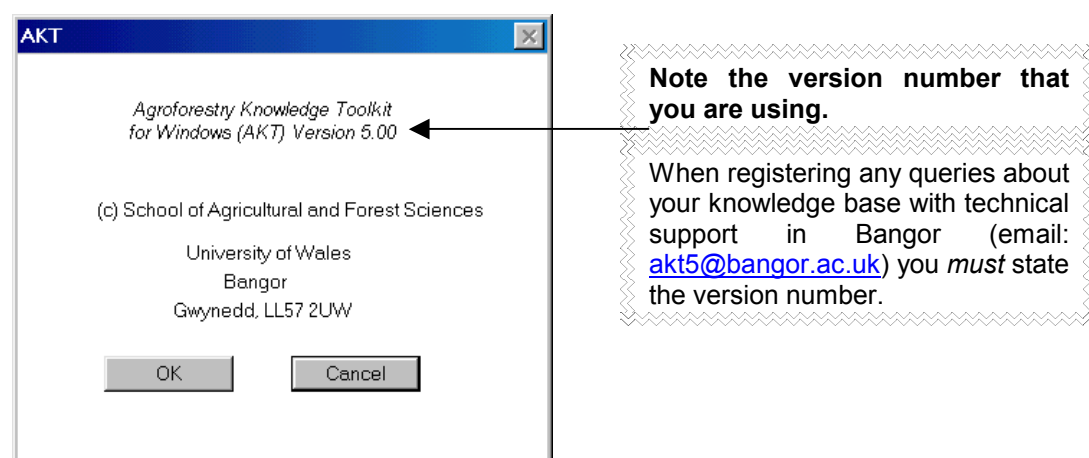


Figure 7.1. Title page of AKT

Click on **OK**. A large blank screen appears with the main menu in the top left-hand corner, as in Figure 7.2.

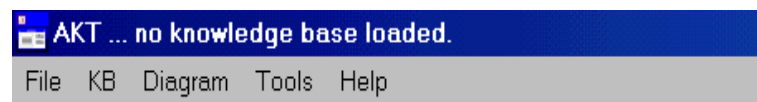


Figure 7.2 *The main menu*

Move the cursor arrow to KB and press. The following menu appears (Figure 7.3)¹;

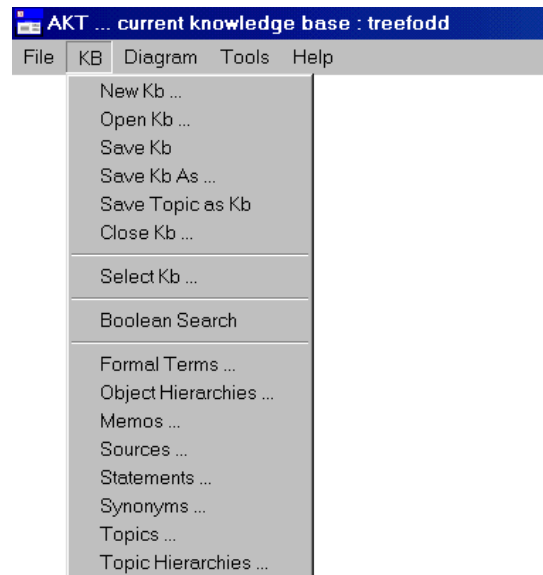



Figure 7.3 *The main KB (Knowledge Base) menu*

A knowledge base file must be opened before work can begin, either by creating a new file (**New KB**) or by opening an existing file (**Open KB**). Other file management options allow the current file to be closed (**Close KB**), the file to be saved (**Save KB**) or the file to be saved under another name or in another location (**Save KB As**). There is also the option of saving all the information about one topic as a separate knowledge base (**Save Topic as KB**).



WARNING

When adding new statements or diagrams to the knowledge base, the programme does not automatically save them. Therefore it is necessary to select **Save KB** from the **KB** menu before closing. It is good practice to save every few minutes to avoid losing work due to power failures etc.

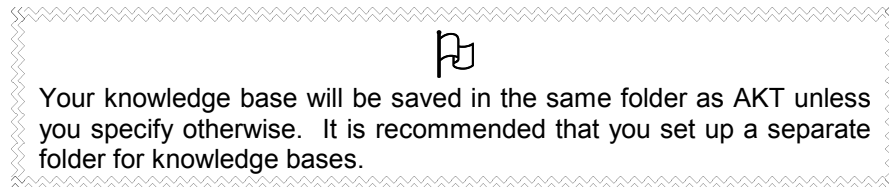
¹ There are two distinct sets of interfaces in AKT for entering, viewing and editing knowledge. These are the **KB** (for Knowledge Base) (Figure 7.3) and the **Diagram** interfaces (Chapter 8). We will begin with the **KB** interface.

7.3 STARTING WORK ON A KNOWLEDGE BASE

When opening the AKT program you will either want to work on a new knowledge base or continue working on an existing one.

7.3.1. STARTING ON A NEW KNOWLEDGE BASE

Select **New KB** from the **KB** menu. You must select the directory in which you wish it to be stored and give the knowledge base file a name.



Type the name and press **Save**. You will then return to the main menu as in figure 7.2 only now the name of the new knowledge base will appear in the top left-hand corner of the screen, in this case 'trees' (Figure 7.4).

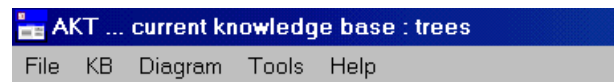


Figure 7.4 Opening menu of a newly named knowledge base

7.3.2. OPENING AN EXISTING KNOWLEDGE BASE

If you are opening an existing knowledge base, select **Open KB**. Once you have selected the knowledge base file and pressed **Open** you will return to the AKT main menu with the name of the knowledge base in brackets next to the AKT heading in the top left hand corner of the screen. Simultaneously, a memo dialog box appears on the screen, designed to give the new user a brief introduction into the knowledge base they have selected (see 7.9 below for a full description on the Memo dialog box). Once you have read the information, press **Close** to continue.

7.4. ENTERING KNOWLEDGE THROUGH THE STATEMENT INTERFACE

Formal statements are entered through the **Statements** interface in the **KB** menu (see Figure 7.3). If you are starting a new knowledge base the 'Statements' dialog box will be empty as in Figure 7.5 below. If you are adding statements to an already existing knowledge base, the statements previously entered will be listed in the 'All Statements' box as in Figure 7.6 showing the 'Statements' dialog box in soil.kb².

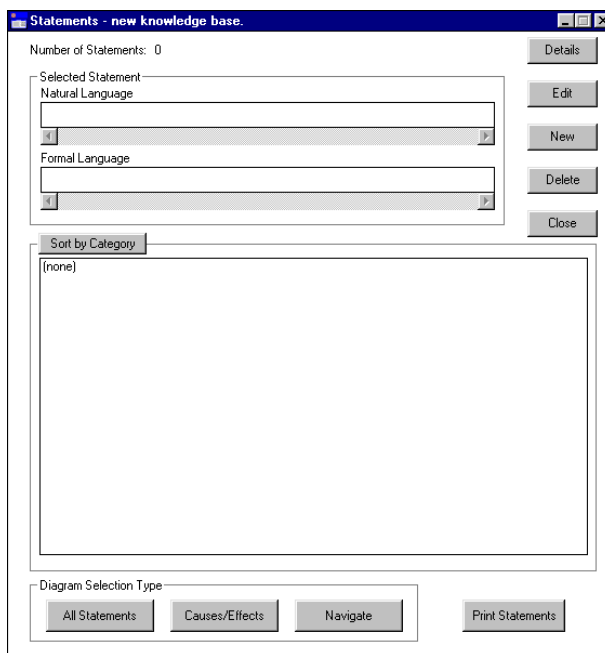


Figure 7.5 The Statements dialog box in a new knowledge base

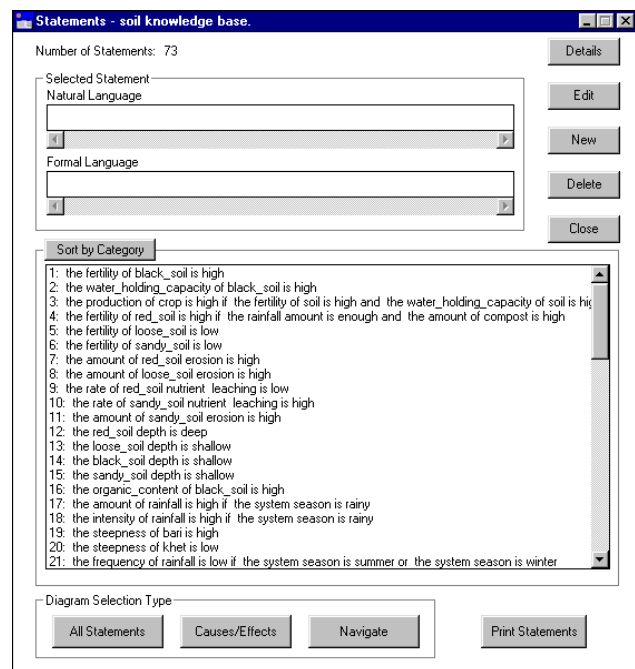


Figure 7.6 The 'Statements' dialog box in an existing knowledge base (soil)

In order to enter a new statement select **New**. The following screen will appear (Figure 7.7), requesting information on the 'Information Source'.

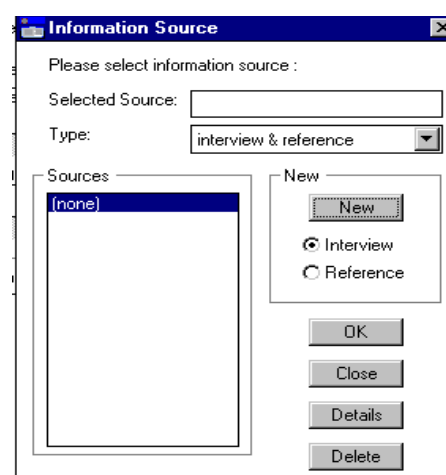


Figure 7.7 Information Sources dialog box

² The examples used throughout the user guide section of this manual are taken from the two knowledge bases provided with the AKT software, 'soil' and 'treefodd'.

7.4.1 ENTERING SOURCE INFORMATION

7.4.1.a New sources

It is not possible to enter a new statement via the statement interface without entering the source first, because it is most important to identify the source of each piece of knowledge entered into the knowledge base. A source may be an informant or it may be a reference from a book or journal or other printed matter. .

- If you are starting a new knowledge base, or using a new source, select **New**.
- If you are updating a previous knowledge base in which the source has already been quoted, highlight the relevant source from the list under 'Source'.

If entering a *new* source, select the relevant type (interview or reference) using the radio buttons and either the interview dialog box (Figure 7.8a) or the reference dialog box (Figure 7.8b) will appear.

To create a new 'interview' source, the informant's name, the date of the interview and the informant's gender is recorded. If the sample of farmers interviewed is to be stratified in any way other than by location and gender, it is possible to introduce other criteria into the 'USER defined fields'. In the 'User identifier 1' box, you put the type of stratification, for example, religion, or caste, or wealth ranking, etc. In the box beside it you put the group to which informant belongs e.g. Muslim, or Brahmin, or wealth rank A.

To create a new 'reference' source, the name(s) of the author(s), date of publication, title and other information are recorded³

Figure 7.8a The interview dialog box

Figure 7.8b The reference dialog box

The year suffix is to enable you to distinguish between multiple interviews (or multiple articles) by the same person within the same year.

If it is felt that more detail is needed either about the informant (for example, age, family background, ethnic group) or about the reference, then this information can be added by selecting **Memo**, and typing the information into the 'Memo attached to source' dialog box (Figure 7.9).

³ In the case of an article, the name of the journal should go in the 'Name' box.

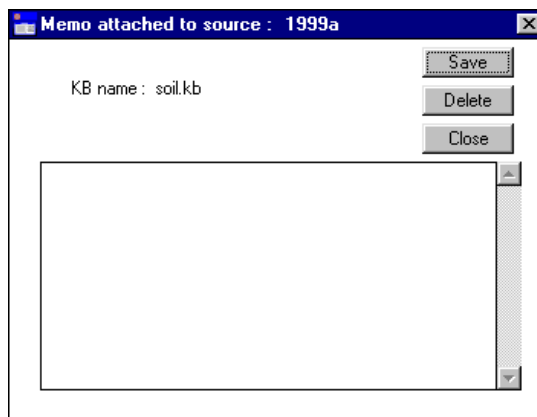


Figure 7.9 The 'Memo attached to source' dialog box, in which extra information about the informant or the reference can be added.

7.4.1.b Sources already entered

If entering new statements but from a source already in the data base, check that the correct source is highlighted and appears in the 'Selected source' box of the 'Information Source' dialog box as in Figure 7.10 below. Then press **OK**.

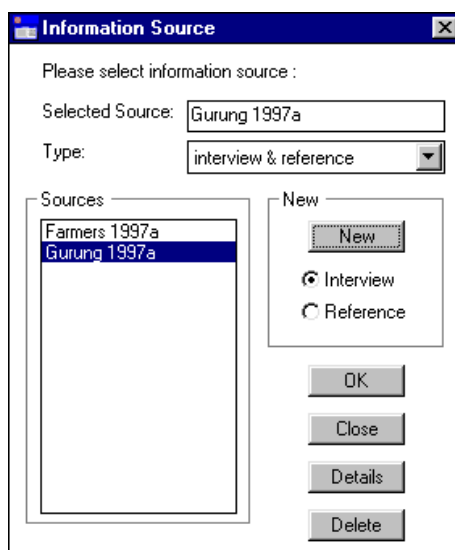


Figure 7.10 Highlighting a source already entered

7.4.2 ENTERING A NEW STATEMENT

Figure 7.11 is an example of a newly entered statement.

Figure 7.11 The 'New Statement' dialog box

All unitary statements have to be entered using the formal language (Chapter 4.2), in the 'Formal Language Statement' box, any conditions are written in the second box, the 'IF' box. After formulating a statement, press **Syntax Check**. If the syntax is correct, a box will appear with 'Syntax Correct', if the syntax is incorrect, a box will appear as in Figure 7.12 or Figure 7.13 depending on where the error lies.

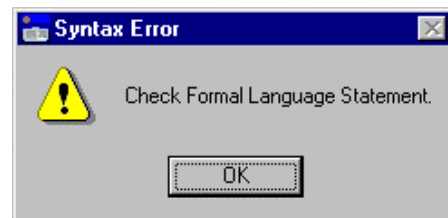


Figure 7.12 Error in the Formal Language Statement

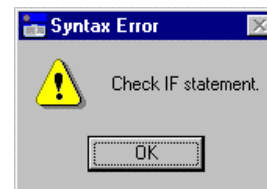


Figure 7.13 Error in IF statement

The formal statement must conform to the grammar described in chapter 4.2. If you receive the error messages, then either the structure of the statement does not conform to the formal grammar, or one of the reserved terms (e.g. att_value) is misspelt. Once the formal language statement has been correctly entered, select **Translate** from the 'New Statement' dialog box for a natural language translation of the formalized statement. The natural language statement generated (Figure 7.14) will be stylised and will not always read smoothly or be grammatically correct, it should, however, make sense and express the meaning of the unitary statement unambiguously.

Figure 7.14 Translation of the formal statement into natural language

When you are satisfied with the statement, press **Save**. If you save a statement without a condition a dialog box will appear as in Figure 7.14a asking you to confirm that you wish to save a statement without a condition. If you select **No** you will return to the new statement dialog box.

If your statement includes any formal terms new to the knowledge base, before the statement is saved a dialog box will appear, asking you whether you want to create the new formal term (Figure 7.14b). A new dialog box will appear for each new formal term. The purpose of this is to make you aware that you are creating a new formal term, to cut down on the number of synonymous terms used and the number of spelling mistakes.

Figure 7.14a Saving a statement without a condition

Figure 7.14b Creating a new formal term within a statement

If you press **Yes** (Figure 7.14b) then a new dialog box appears, announcing 'Statement saved'. If you press **No** then the statement will not be saved and you will be returned to the New Statement dialog box.

WARNING
Save in the 'New Statement' dialog box, only saves the statement in memory temporarily. To save it in the knowledge base permanently you must select **Save** under the **KB menu**.

7.4.3 EDITING A STATEMENT

Once a new statement has been saved, the programme automatically reverts back to the updated 'Statements' dialog box (Figure 7.15) from which all existing statements can be accessed. For large knowledge bases, accessing particular statements may be facilitated by doing a Boolean Search, restricting the statements to a set containing the formal terms of interest (see section 7.9 below).

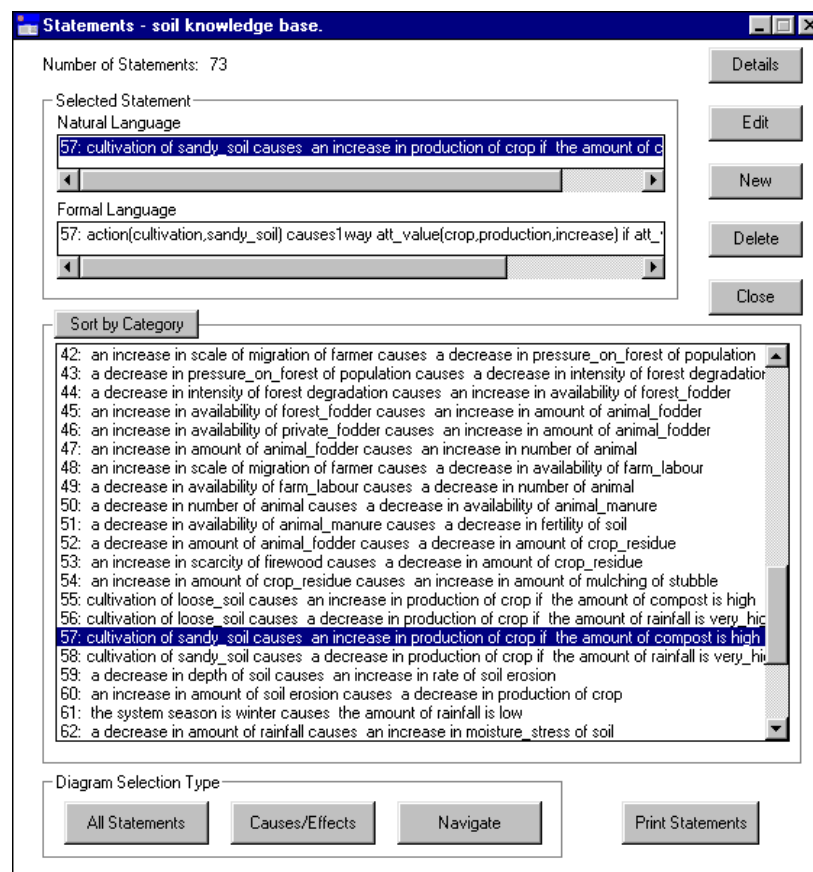


Figure 7.15 'Statements' dialog box

From the 'Statements' dialog box it is possible to check all the details of a statement (the sources, the formal terms, any definition of those formal terms, any synonyms, and any memos) and to edit that statement.

Highlighting the relevant statement and selecting **Details** will allow you to check all these details without the danger of making any inadvertent alterations.

Highlighting the relevant statement and selecting **Edit** permits you to edit the statement and all its appended details (Figure 7.16).

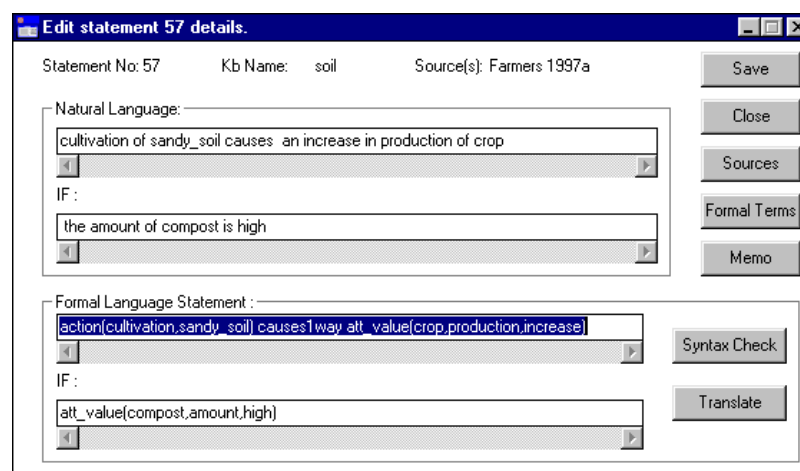


Figure 7.16 The 'Edit statement' dialog box

7.4.3.a Adding definitions to formal terms

It is important that all formal terms remain as unambiguous as possible. However, different users may have a different understanding or concept of the same term. For example 'alley cropping' may suggest to some a system where the tree component is made up of a single or multiple row of trees, whilst others envisage a system of dense hedgerows. For this reason it is useful to be able to define exactly what is meant by the term in the context of a specific knowledge base.

In order to add definitions select **Formal Terms** in the 'Edit statement' dialog box (Figure 7.16 above) and the following screen appears (Figure 7.17);

The 'Statement Formal Terms' dialog box displays two lists, the first, headed 'Statement Formal Terms' gives a list of all the formal terms within that particular statement. By highlighting one of the terms, that term's type (i.e. object, process, action, attribute, value, link or comparison) will be displayed in the 'Type' box. The second list, headed 'Formal Terms' gives a list of all the formal terms within the knowledge base. This serves as a reference and should be consulted so as to avoid unnecessary use of synonymous terms. In order to define any of the terms, highlight the required term, and select **Details**. The 'Formal Term Details' dialog box will appear (Figure 7.18). The 'Formal Term Details'⁴ dialog box enables you to:

- define the term by filling in the 'Definition' box;
- check whether the term has any synonyms;
- check in which statements the term has already been used by selecting **Show use in statements**; and
- check whether the term already appears in an object hierarchy by selecting **Show use in hierarchies**.

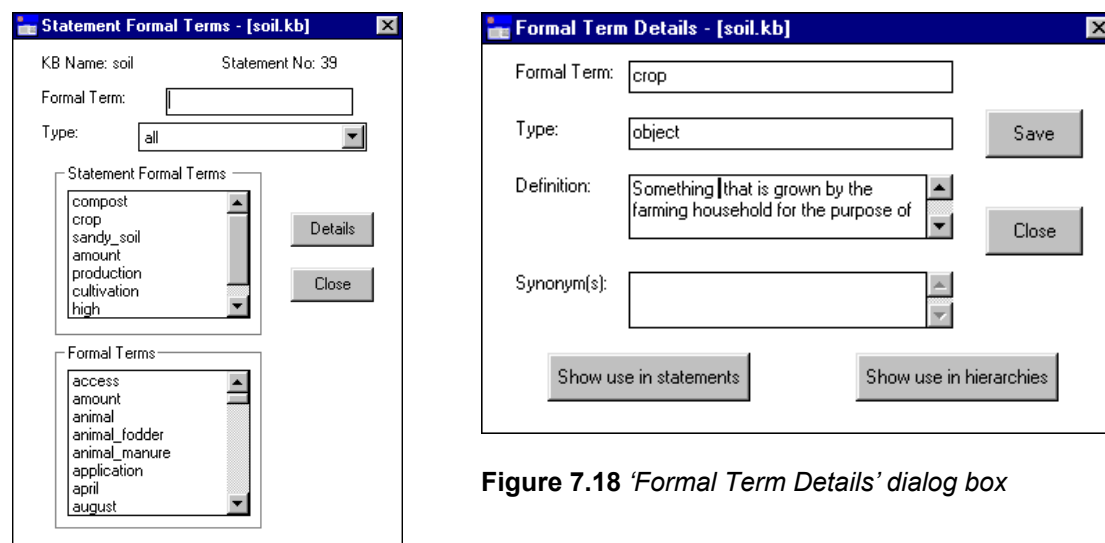


Figure 7.18 'Formal Term Details' dialog box

Figure 7.17 The 'Statement Formal Terms' dialog box



After editing a statement it is advisable to save the whole knowledge base through the main **KB** menu.

⁴ For a more detailed description of 'Formal Term Details' see section 7.7.

7.4.3.b Appending or detaching additional sources

It may be that there is more than one source for a particular statement. In that case only the first source is given when the new statement is entered into the knowledge base. If the other sources have already been entered into the knowledge base, either attached to other statements or via **Sources** in the main **KB** menu, then they can be appended to the statement at this stage.

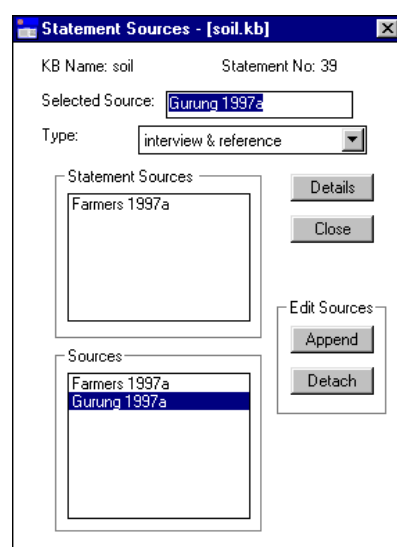
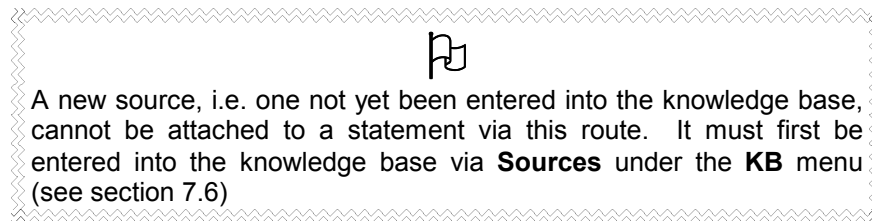


Figure 7.19 'Statement sources' dialog box

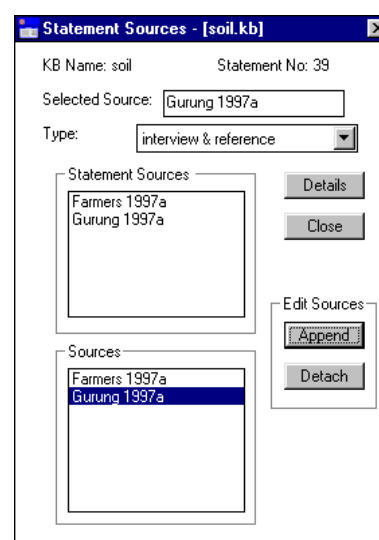


Figure 7.20 A new source appended to 'Statement Sources' list

In order to append a source to the statement in the 'Edit Statement' dialog box select **Sources**. The 'Statement Sources' dialog box appears (Figure 7.19). Highlight the source you wish to append to the statement from the 'Sources' list and then select **Append**. A message will appear, confirming that the source has been appended to the statement (Figure 7.21) and the newly appended source will appear in the upper list under 'Statement Sources' (see above, Figure 7.20).

If you wish to detach a source from a statement, highlight the source to be detached from the statement in the 'Statement Sources' list and select **Detach**. The source will be removed from the 'Statement Sources' list, although it will remain in the 'Sources' list. A message will appear, confirming that the source has been detached from the statement (Figure 7.22).

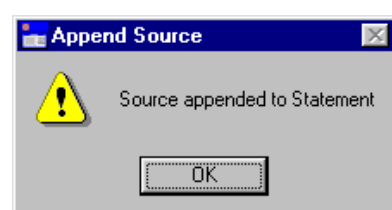


Figure 7.21 Message confirming that a source has been appended to a statement

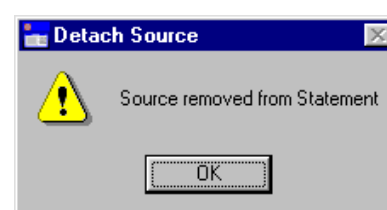
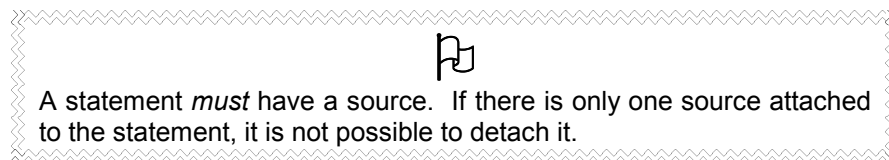


Figure 7.22 Message confirming that a source has been detached from a statement



7.4.3.c Appending Memos

Statements may need certain clarifications which cannot be expressed in the formal language and therefore have to be appended to the statement in the form of a memo. For instance, in the example given here of a new statement:

Cultivation of sandy_soil causes an increase in production of crop if the amount of compost is high (Figure 7.14, above)

it may be appropriate to list in the memo field what the compost is made of. It is better to put the explanation here, rather than in the 'Formal Terms' dialog box, because the type of compost referred to may differ from statement to statement, and the Memo dialog box is attached to the individual statement. To add a memo, select **Memo** from the 'Edit Statement' dialog box, and the following dialog box appears (Figure 7.23);

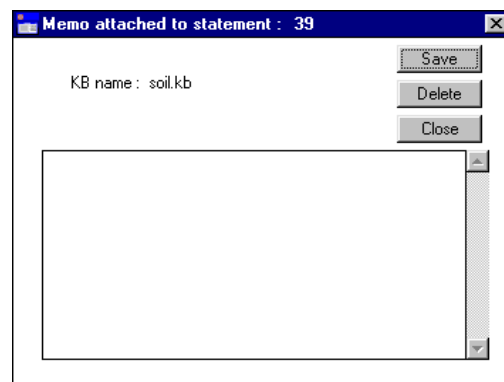
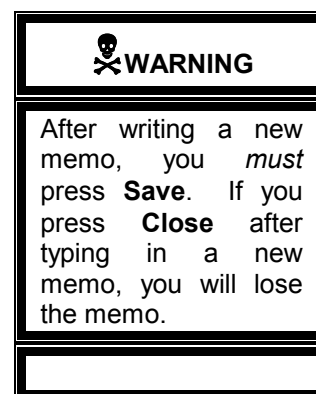


Figure 7.23 The 'Memo' dialog box, attached to a specific statement



It is now possible to enter any information about the statement that has not been adequately expressed by formal language. There is ample room to write the memo in some detail, the scroll bar on the right enabling you to read all the passage. Once the memo is complete, press **Save**. A message will appear, 'Memo saved O.K.' If you choose to delete a memo, select **Delete**. A message will appear 'Memo deleted.'. If you just want to read a previously written memo, then select **Close** when finished.

7.4.4 DELETING A STATEMENT

In order to delete a statement, enter **Statements** via the main **KB** menu and highlight the statement you wish to delete. The statement will appear in the 'Selected Statement' box. Then press **Delete**. A message will appear, giving the natural language translation of the statement to confirm whether you really want to delete that statement (Figure 7.24).

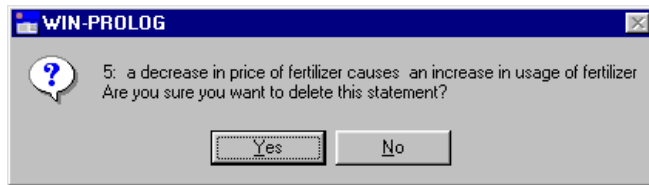


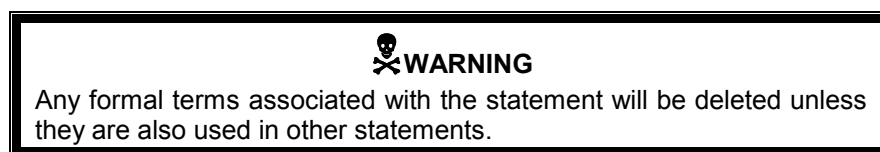
Figure 7.24 Message requesting you to confirm whether you wish to delete the highlighted statement



Figure 7.25 Message confirming that the statement has been deleted

If **Yes** is selected, another message appears (Figure 7.25) confirming that the statement has been deleted.

If you wish the statement to be permanently deleted from the knowledge base, you must save the knowledge base after deleting the statement.



7.4.5 DIAGRAMMATIC REPRESENTATION

At the bottom of the Statement card are various options for diagrammatic representation (Figure 7.26).

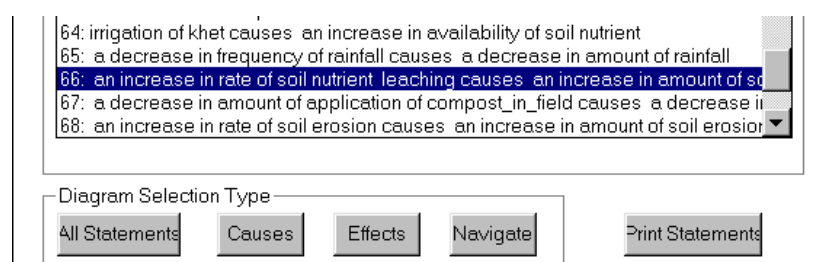


Figure 7.26 The diagram options at the bottom of the Statements card

If you select **All Statements** it will give you a diagram of all the causal and link statements in the knowledge base.

If you highlight a statement and press **Causes** it will give you a diagram showing all the connections (i.e. causes) flowing *into* the selected statement. If you highlight a statement and press **Effects** it will give you a diagram showing all connections (i.e. effects) flowing *out of* the selected statement. If there are no causes or no effects of a particular statement a message will appear to inform you of the fact.

If you highlight a statement and press **Navigate**, it will give you a diagram of the immediate causes and effects of the selected statement, from which you can build up the diagram further from within the diagram interface. Diagrammatic representation is dealt with in detail in Chapter 8.

7.4.6 SORTING STATEMENTS IN THE STATEMENT CARD

The statements in the statement card can be sorted either numerically, or by type (attribute, causal, comparison or link statements). When you open a knowledge base, or create a new one, the statements will be listed numerically. To list them by statement type press the **Sort by Category** button at the top of the statement card (see above, Figure 7.6). If the statements are sorted by category the button changes to **Sort Numerically**.

7.4.7 INVERSE STATEMENTS AND NUMBERING

Each two way causal statement (i.e. those using 'causes2way') generates an invisible inverse statement, a statement that is used in reasoning but is not displayed on the screen (see above Chapter 4.2.4.a). Each time a knowledge base is loaded all statements are renumbered from 1 to the total number of statements. The inverse statements are then generated and added to the knowledge base with numbers following on from the visible statements. For this reason, when you begin adding new statements to a knowledge base, the statement number will not be the next number available in the list of visible statements but the next number available after the *invisible* inverse statements. This is the reason for any gap in the sequence of numbers between the last visible statement on the statement card and the number given to the first new statement. Figure 7.26a below demonstrates this, the last number in the original knowledge base was 756, the number allocated to a new statement is 918. That means 162 inverse statements have been created after loading the original knowledge base.

753:	posilo_fodder_tree tree_fodder mainly_fed_to large_animal
754:	kam_posilo_fodder_tree tree_fodder not_fed_to ox
755:	painyu tree_fodder not_fed_to cattle
756:	cattle_fodder_tree tree_fodder fed_to large_animal
918:	the nutrient_value of silage is high

Figure 7.26a *Demonstrating the gap between the last number of the visible statements and the first number of any new statements. The difference (162) is taken up by a set of invisible inverse statements.*

7.5 OBJECT HIERARCHIES

AKT allows you to generate a hierarchy of objects. This provides an indexing system for the otherwise unsorted unitary statements that constitute the knowledge base. A single knowledge base may contain a number of discrete hierarchies, and a single object may occur in more than one hierarchy.

The creation and manipulation of object hierarchies are controlled by the **Object Hierarchies** menu under the main **KB** menu.

7.5.1 CREATING OBJECT HIERARCHIES

A new object hierarchy is created by selecting **New** (Figure 7.27). In the new dialog box that appears, 'New Object Hierarchy' (Figure 7.28) a new hierarchy name has to be given. The 'New Hierarchy Name' can be either one of the objects which occurs in the unitary statements or it can be a completely new name, not directly related to any of the unitary statements but

which provides a means of identifying the relationship between the other objects appearing in the hierarchy. Enter the name and press **Save**.

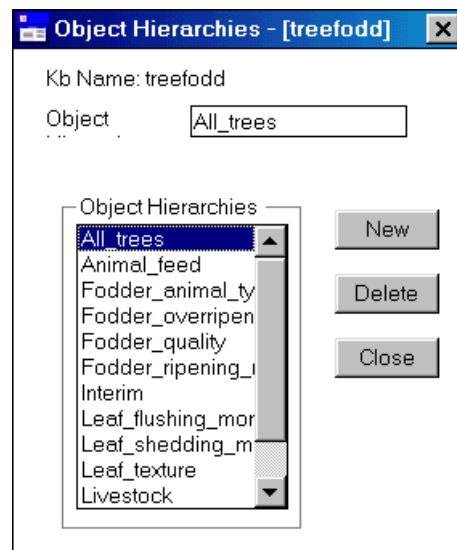


Figure 7.27 Dialog box for object hierarchies

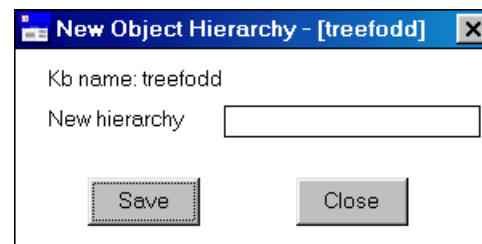


Figure 7.28 Dialog box for New Object Hierarchy

Once a hierarchy name has been entered and saved, the dialog box will revert to the 'Object Hierarchies' dialog box with the new hierarchy name included in 'Object Hierarchy' list.

7.5.2. VIEWING OBJECT HIERARCHIES

To view an object hierarchy, highlight the relevant hierarchy from the 'Object Hierarchies' list in the 'Object Hierarchies' dialog box and the object hierarchy details will appear automatically.

The screen below (Figure 7.29) is the 'Object Hierarchy' dialog box for the object hierarchy 'Livestock' in the 'treefodd' knowledge base. The left-hand side of the dialog box shows the 'Objects in Hierarchy' list, which is simply an alphabetical list of all the objects in the hierarchy. The central panel headed 'Hierarchy Structure' gives the structure of the hierarchy immediately above and below the selected object, with the selected object highlighted in the middle box under 'Object' and also in the 'Selected Object' box at the top, under the hierarchy name.

In Figure 7.29 the object in the 'Selected Object' box is Livestock. As that is also the hierarchy name, there are no objects listed in the SuperObjects box. In the 'Immediate SubObjects' box are listed 'animal' and 'bird'. If we highlight another object in the 'Objects in Hierarchy' list, for example, 'large_animal', the objects in the 'Hierarchy Structure' section will automatically change to Figure 7.30. Now 'animal' appears as the superobject and 'buffalo' and 'cattle' as the immediate subobjects.

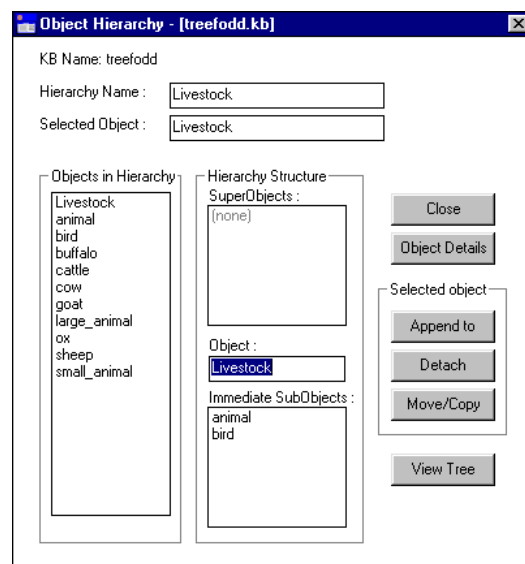


Figure 7.29 'Object Hierarchy' dialog box

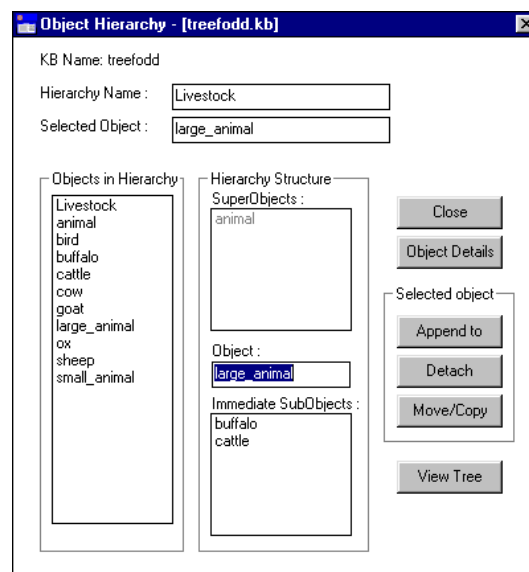


Figure 7.30 Highlighting *large_animal* in the 'Object' box of the hierarchy structure.

To get a comprehensive view of the 'Livestock' hierarchy structure, select **View Tree**. The following diagram appears (Figure 7.31), the scroll bars enabling you to view the whole diagram. It is also possible to copy the hierarchy tree and paste it to another package, by pressing the **Copy to Clipboard** button. Open the package and paste the hierarchy tree immediately by selecting **Edit** and then **Paste** or by pressing **Ctrl v**.

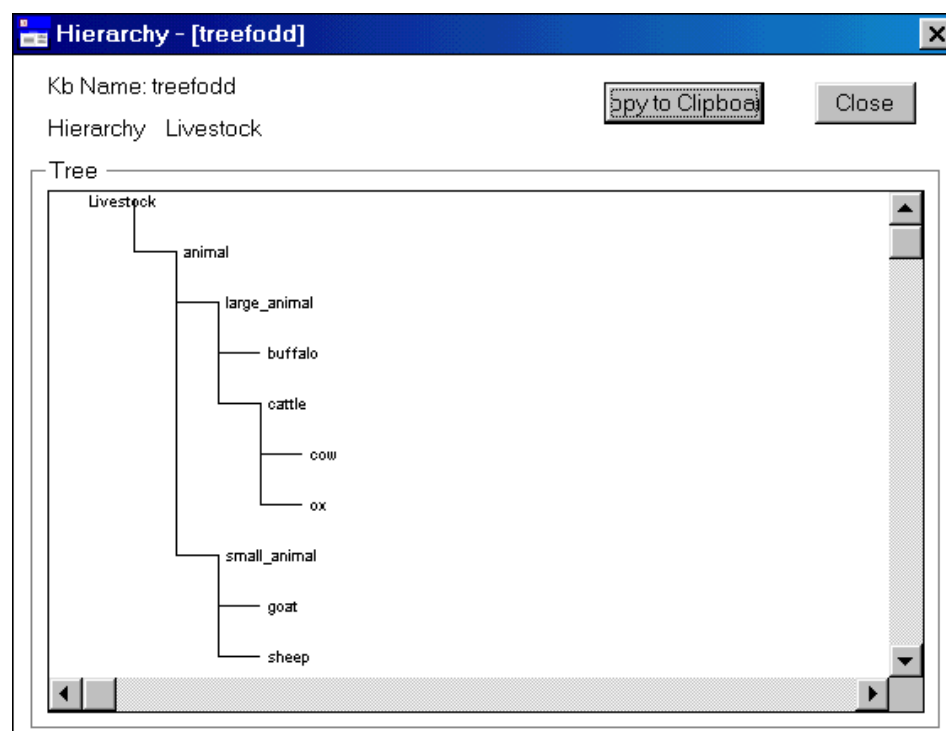


Figure 7.31 A diagram of the 'Livestock' hierarchy, generated by selecting **View Tree**

7.5.3 BUILDING / EDITING OBJECT HIERARCHIES

7.5.3.a Adding objects

To build up an object hierarchy, or to add to one already extant, highlight the relevant hierarchy from the 'Object Hierarchies' list and the 'Object Hierarchy' dialog box will automatically appear. Figure 7.32 is of the 'Object Hierarchy' dialog box for a newly created hierarchy with no objects attached.

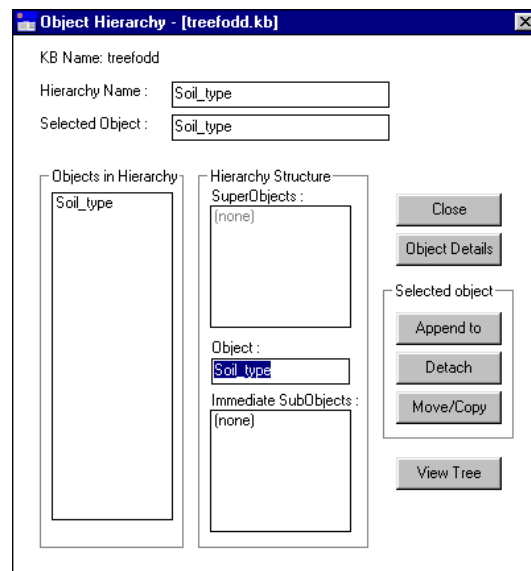


Figure 7.32 The 'Object Hierarchy' dialog box of a newly created object hierarchy

To add objects to a hierarchy, select **Append to**. The resulting screen (Figure 7.33) will offer you a list of all the objects in the knowledge base. Highlight the desired object under the 'Objects' list and select **Append**. The message 'Object appended to Hierarchy' will then appear (Figure 7.34) and the object will appear as an Immediate SubObject of the hierarchy root.

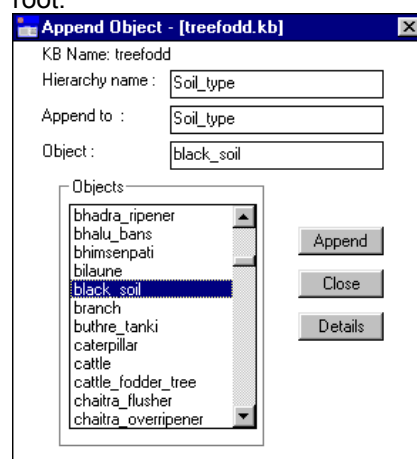


Figure 7.33 Selecting an object to append to an object hierarchy

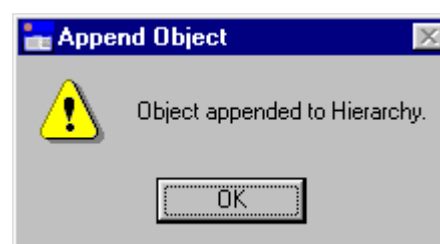


Figure 7.34 Message confirming that the object has been appended to the object hierarchy

Any number of objects can be appended to the object hierarchy in this way, appending them one at a time.

The objects that appear in the object list (Figure 7.33 above) are taken from the statements entered onto the knowledge base. However it may be useful to add objects to the object

hierarchy that do not appear in any of the statements but are implied by a more generic term. For instance, farmers may refer frequently to 'livestock', without mentioning the species. For the comprehensiveness and the power of the knowledge base it is nevertheless useful if the researcher can list the species kept, under a 'Livestock' hierarchy. To add objects that do not appear in the statements, you must first add them to the knowledge base as new formal terms (see 7.7.2) and then add them to the object hierarchy.

If you wish to create a hierarchic structure within the object hierarchy, then first select the root, or parent object from the list of objects. This object will then appear in the 'Object' box in the 'Hierarchy Structure' and objects can be appended to it as before. In Figure 7.35, *black_soil* is the head of the sub-hierarchy, and *loose_soil* is the new subobject.

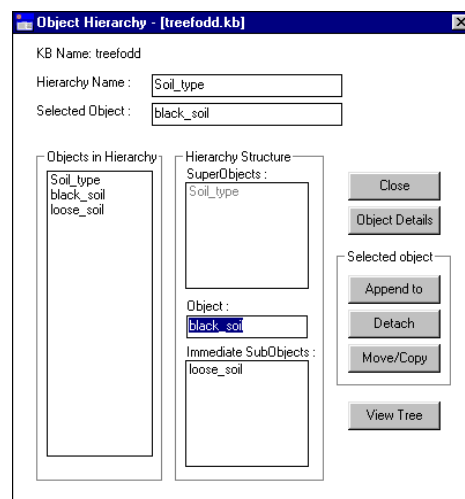
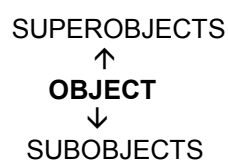


Figure 7.35 Demonstrating an object hierarchy, in which '*black_soil*' is the selected object, '*loose_soil*' its sub-object and '*Soil_type*' its parent.

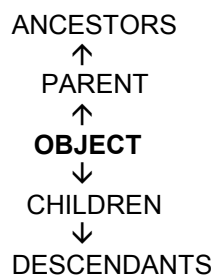
There are two ways of describing the position of an object in a hierarchy structure. Diagram A describes the position in its simplest form. All objects higher up the hierarchy than the object in question are superobjects, all those below are subobjects.

A.



But this structure can be defined more precisely in Diagram B.

B.



An object can only have one parent, or immediate superobject, but as many ancestors as there are rungs on the object hierarchy ladder above its own position on that ladder. On the other hand, an object can have many children, or immediate subobjects, and many descendants, on each rung below it on the hierarchy ladder.

7.5.3.b Detaching objects

To detach an object from an object hierarchy, enter the relevant 'Object Hierarchy' dialog box and highlight the object to be detached in the 'Objects in Hierarchy' list. The object will then appear in the 'Selected Object' box and in the 'Object' box within the 'Hierarchy Structure'. In Figure 7.36 this is the object 'goat'. Then press **Detach**.

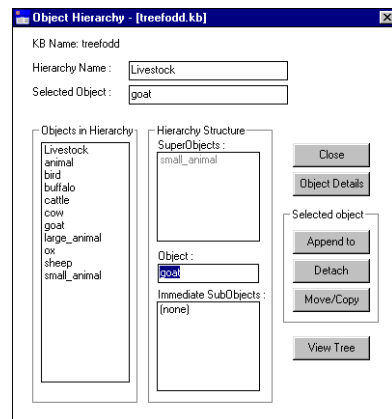


Figure 7.36 Detaching 'goat' from an object hierarchy

The following message appears (Figure 3.37) asking whether to detach the selected object. Press **Yes** and another message appears, confirming that the object 'goat' has been removed (Figure 7.38).

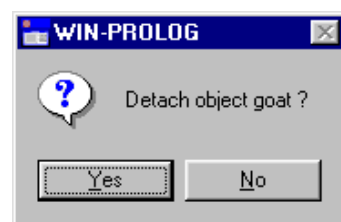


Figure 7.37 Message to confirm detaching object



Figure 7.38 Message confirming that the object has been detached

The object 'goat' is now removed from the object hierarchy 'Livestock', although it still exists as a formal term in the knowledge base.

If you wish to detach an object which is the head of a sub-hierarchy, for instance 'large animal' in Figure 7.39, which has 'buffalo' and 'cattle' as sub-objects, once you have pressed **Detach**, and confirmed that the object is to be detached (as in Figure 7.37 above), a new message appears (Figure 3.40).

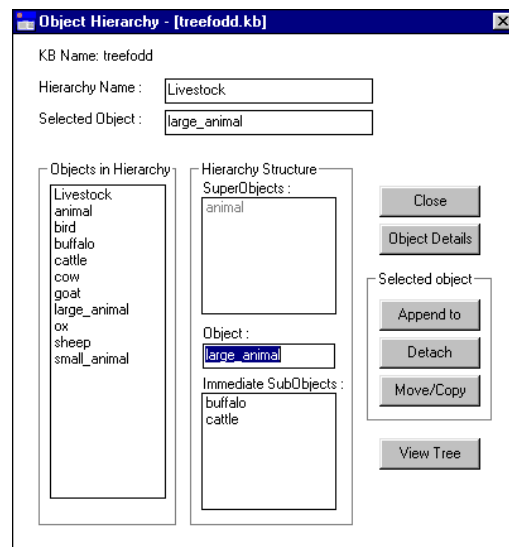


Figure 7.39 Detaching an object with sub-objects attached

The program asks whether or not you wish to detach all the sub-objects together with the selected object. If you press **Yes** then in this case 'buffalo' and 'cattle' will be removed from the object hierarchy as well. If you press **No** then another message appears (Figure 7.41), which confirms that the object has been detached and its sub-objects raised up a level within the hierarchy.

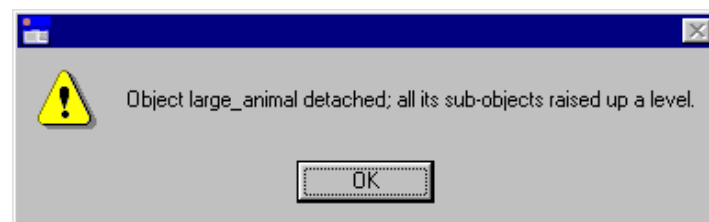


Figure 7.41 Message appearing if an object is detached but its sub-objects are left in place

7.5.3.c Moving objects within a hierarchy

It may be necessary to rearrange an object hierarchy, moving objects to different levels within the hierarchy. In the below example, Figure 7.42, 'chickens', 'ducks' and 'geese' have been added to the Livestock hierarchy, and are directly below 'Livestock' on the hierarchy tree. However, it would be more appropriate to put them under 'bird'. In order to do this highlight the first object, 'chickens', in the 'Objects in Hierarchy' list and press **Move/Copy**. The 'Move Object' dialog box appears (Figure 7.43).

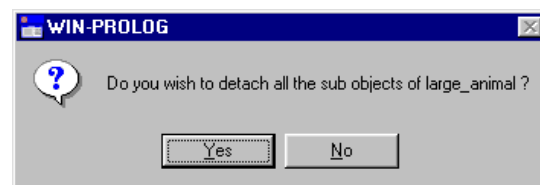


Figure 7.40 Message displayed when detaching an object with sub-objects attached

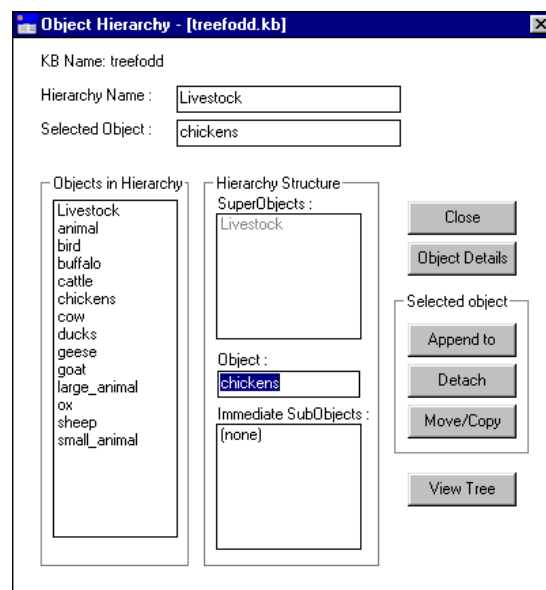


Figure 7.42 Highlighting an object before moving it within an object hierarchy

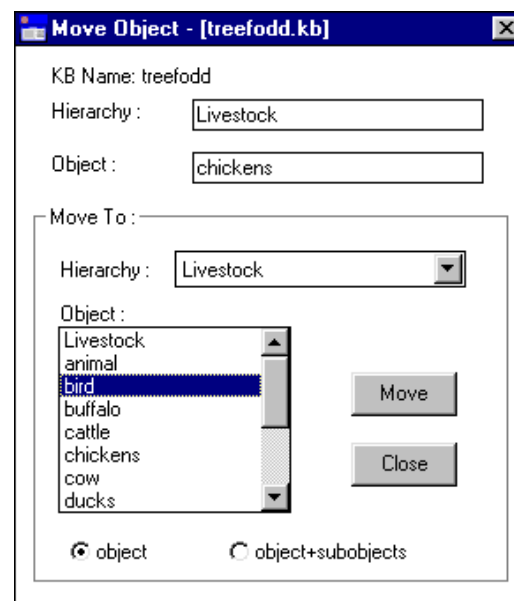


Figure 7.43 Moving an object to another part of the object hierarchy

The 'Move Object' dialog box gives the name of the hierarchy in use ('Livestock'), and the object to be moved ('chickens'). The 'Move To:' section allows you to select the hierarchy under which the object is to be moved, and under 'Object' lists the objects within that hierarchy under which the object can be placed. At the bottom of the dialog box is the option to move, either just the object, or the object plus its sub-objects. In this case the object 'chickens' has no subobject, so the 'object' bullet is highlighted. Then press **Move**. The following message appears (Figure 7.44);



Figure 7.44 The message confirming an object has been successfully moved within an object hierarchy

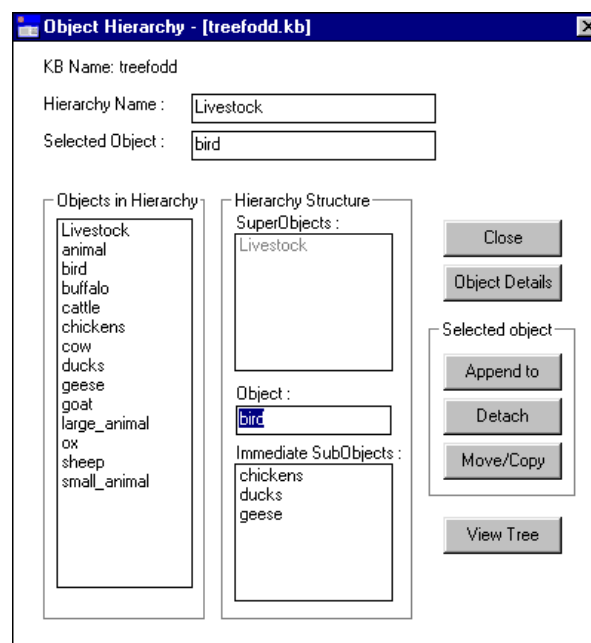


Figure 7.45 An object ('bird') and its three immediate subobjects ('chickens', 'ducks' and 'geese'), after they have been moved within the object hierarchy.

Exactly the same procedure is carried out for 'ducks' and 'geese', so that all three are now subobjects of 'bird' as in the 'Object Hierarchy' dialog box in Figure 7.45.

7.5.3.d Copying objects between hierarchies

It is possible to copy objects with or without their immediate sub-objects between different object hierarchies. To do this, follow the same procedure as above, highlighting the object to be copied in the 'Objects in Hierarchy' list in the 'Object Hierarchy' dialog box and press **Move/Copy**. In the example below, Figure 7.46, we are copying the sub-hierarchy 'bird' to the object hierarchy 'Plant_pest'. In the 'Move Object' dialog box the present hierarchy name 'Livestock' is in the top 'Hierarchy' box. In the 'Copy To:' section the hierarchy 'Plant_pest' has been selected from the drop down menu. The objects in the 'Object' list are now all objects from the 'Plant_pest' hierarchy. Highlight the object under which the new object should be copied. In this case it is 'crop_pest' (Figure 7.47) Then select whether the object alone should be copied or the object plus its sub-objects by selecting the relevant radio button. In this case the subobjects of 'bird', 'chickens', 'ducks' and 'geese' cannot be described as crop pests, therefore the object 'bird' is moved alone, without its subobjects.

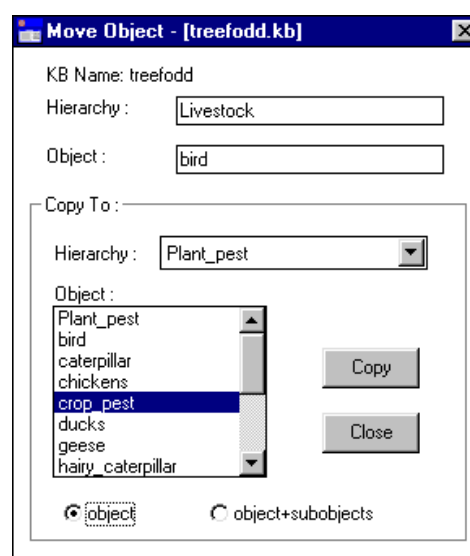
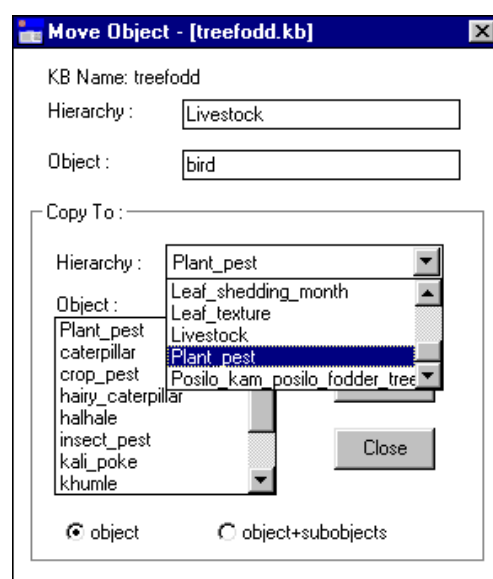


Figure 7.46 Selecting the new object hierarchy to which the object is to be copied

Figure 7.47 Selecting the object under which the object to be copied is to appear

Then press **Copy**. The object is then automatically copied to the new object hierarchy.



It is only possible to *copy* objects and objects + subobjects from one hierarchy to another, not to *move* them. If you want to move them, copy the objects to the new object hierarchy and then detach them from the original object hierarchy as described in 7.5.3

7.6 SOURCES

Sources are usually entered into the knowledge base when statements are being entered either via the Statements dialog or via the Diagram interface (see chapter 8) as a statement cannot be entered without a source (see above, 7.4.2). However, only one source per statement can be entered via the **Statements** menu (see above, 7.4.1), whereas there may be several sources for one statement. Additional sources can therefore be entered via **Sources** in the main **KB** menu.

The 'Information Source' dialog box is exactly the same as that entered via the Statements dialog (see 7.4.2). Figure 7.48 below is the 'Information Source' dialog box of the knowledge base, 'treefodd'.

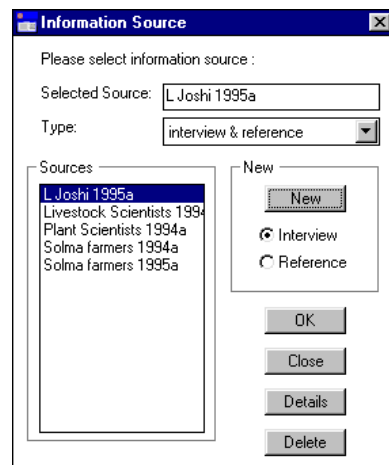


Figure 7.48 The 'Information Source' dialog box, entered directly via **Sources** in the main **KB** menu

The details of existing sources can be accessed via **Details**. New sources can be entered by highlighting the relevant radio button and pressing **New**. The two dialog boxes 'Create a new 'interview' source' and 'Create a new 'reference' source' function in exactly the same way as described in 7.4.2.

Sources can be deleted by selecting **Delete**. However, sources which are attached to statements can *only* be deleted if there is more than one source for that particularly statement.

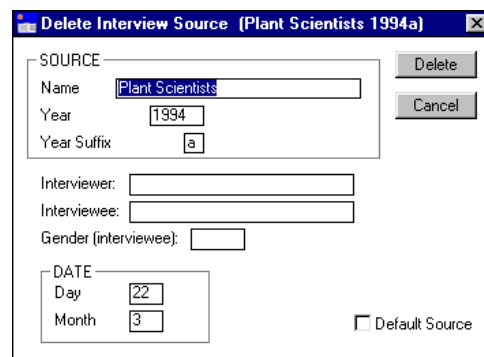


Figure 7.49 The first step in deleting a source

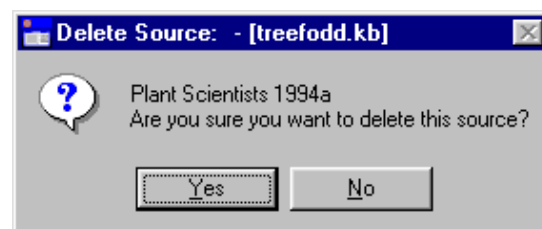


Figure 7.50 The second step in deleting a source

Figures 7.49 to 7.51 describe the steps taken to delete the source 'Plant Scientists 1994a' from the knowledge base 'treefodd'. 'Plant Scientists 1994a' is the sole source for some of the statements within the knowledge base. For this reason the message appears saying that it cannot be deleted (Figure 7.51).

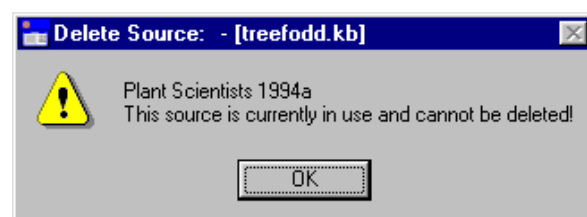


Figure 7.51 The message that appears if a source is the sole source for a statement and therefore cannot be deleted.

7.7 FORMAL TERMS

7.7.1 VIEWING FORMAL TERMS

Formal Terms, under the main **KB** menu brings up a glossary for all the formal terms in the knowledge base. Selecting a formal term type (object, process, action, attribute, value, link and comparison) from the 'Type' box will bring up the glossary for that type (Figure 7.52). Unlike object hierarchies, these glossaries can only be flat - simply an alphabetical list of terms. Items are added to glossaries automatically during the process of formal representation. The details of each term can be accessed by selecting **Details**. The 'Formal Term Details' dialog box (Figure 7.53) gives the formal term type, its definition and any synonyms there may be. If you select **Show use in statements** a dialog box 'Search Results' appears, giving all the statements in which the formal term appears (Figure 7.54). For objects it is also possible to select **Show use in hierarchies**. If the object appears in any hierarchies the relevant hierarchy names will appear (Figure 7.55). In the example below, the formal term 'cattle' appears in three formal statements and in the object hierarchy, 'Livestock'.

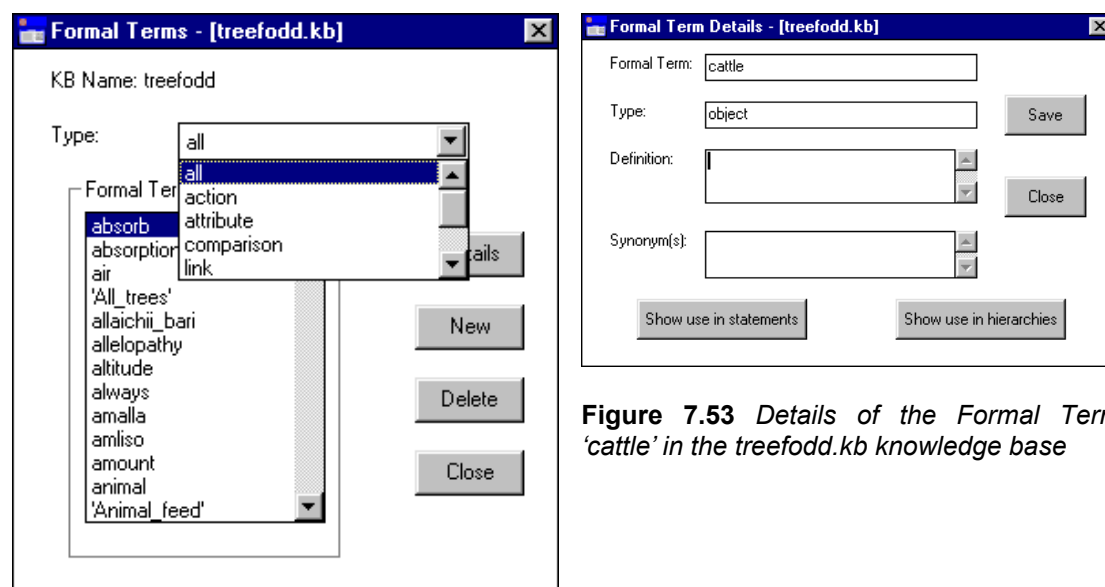


Figure 7.53 Details of the Formal Term 'cattle' in the treefodd.kb knowledge base

Figure 7.52 Selecting the type of formal term in the Formal Terms dialog box

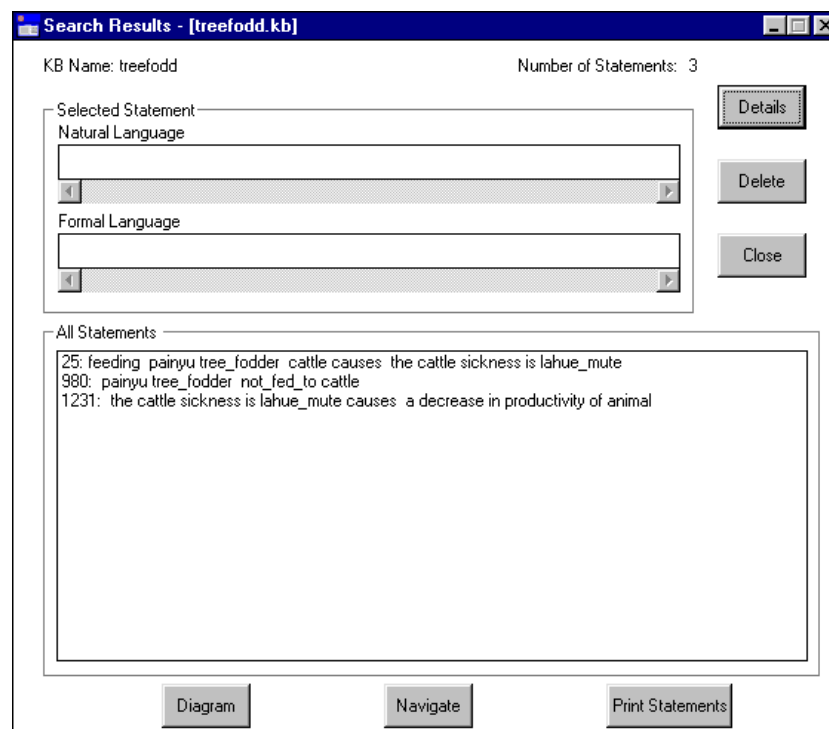


Figure 7.54 'Search Results' for the formal term 'cattle' – screen display after selecting **Show use in statements** in Formal Terms dialog box



Figure 7.55 The message that appears having selected **Show use in hierarchies** in the Formal Term dialog box, with 'cattle' as the formal term

7.7.2 ADDING FORMAL TERMS

It is also possible to add new formal terms, unattached to statements, to the glossaries via **Formal Terms**. This is of particular use for objects and object hierarchies (see 7.5.3).

In order to add new formal terms, press **New**. A 'New Formal Term' dialog box appears (Figure 7.56).

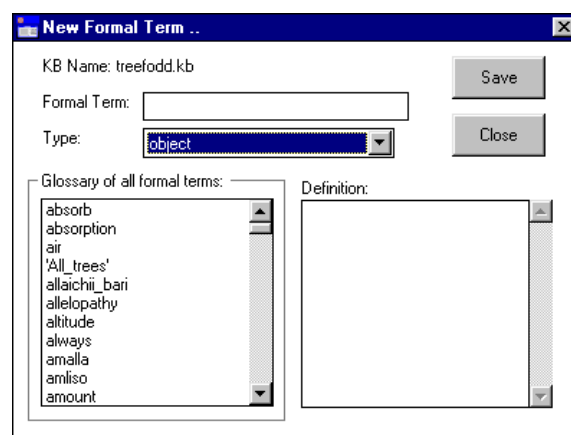
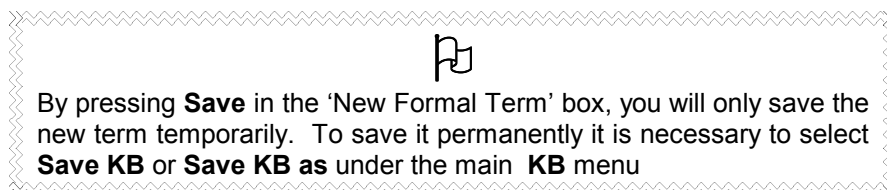


Figure 7.56 'New Formal Term' dialog box

The new 'Formal Term' is typed into the top box. The 'Glossary of all formal terms' exists to enable you to check that the term, or something similar, is not already in the knowledge base. You select the formal term type from the drop-down menu in the 'Type' box. The 'Definition' box enables you to add a precise definition of the term as it is to be understood in the context of the present knowledge base.

When finished, press **Save**. A message will then appear, 'Do you wish to add the formal term to the Kb?' (as it does whenever you introduce a new formal term when you create a new statement). Press **Yes**, whereupon a message appears to confirm that the new term has been saved.

If you attempt to add a term which already exists in the knowledge base, a message appears 'A term with the same name already exists in the kb'.



7.7.3. DELETING FORMAL TERMS

These added terms may also be deleted by highlighting them in the 'Formal Terms' list and selecting **Delete**. However, it is not possible to delete a formal term that still appears in a formal statement. In order to remove a formal term which appears in a formal statement, the term must either be replaced, using the edit function in the **Statements** menu (see above, 7.4.4) or by deleting the statement altogether, using the **Delete** function in the **Statements** menu (see above 7.4.5).

7.8 SYNONYMS

The 'synonyms' option allows you to specify a synonym for a formal term. An example of this might be the local name for a plant beside the common English name and the scientific Latin name. For example, the tree with the common English name 'shea butter tree' has the scientific name *Vitellaria paradoxa*, and is also commonly known in English by its French name 'karité' whilst in the Hausa language in Nigeria it is known as 'ka'danya', in the Bambouk language in Senegal it is known as 'toulou'. Thus a knowledge base developed on the shea butter tree in West Africa would need to record all the common synonyms for the tree.

The synonym could also be for a value such as 'decrease' for 'decline' or a process such as 'furlowing' for 'ploughing'.

The reason it is important to have the synonyms option is that it permits you to draw on all available statements using the formal term or any of its synonyms and prevents accidental exclusion in a topic knowledge base because different terms have been used for the same object/process/value etc.

In order to enter a synonym, select the main **KB** menu and then select **Synonyms**. In the 'Synonyms' dialog box that appears, select **New**. A 'New Synonym' dialog box appears. A list of all the formal terms in the knowledge base appears in the box marked 'Choose Formal Term'. Once you select the formal term type from the drop-down menu, only formal terms of the specified type appear in the 'Choose Formal Term' list. From this list highlight the word you wish to give a synonym and then enter the synonym in the 'New synonym name' box (Figure 7.57). Press **Save**. A message will appear confirming that the new synonym has been saved (Figure 7.58).

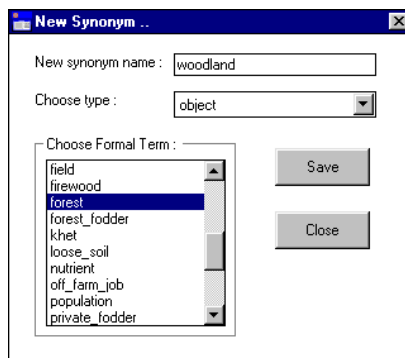


Figure 7.57. Entering a new synonym into the New Synonym dialog box

The screen will then revert back to the 'Synonyms' dialog box, where the new synonym is now listed under 'Synonyms' (Figure 7.59).

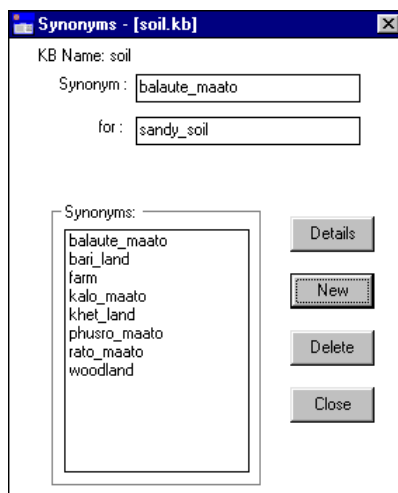
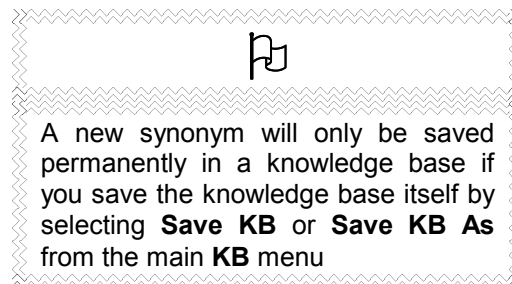


Figure 7.59 The new synonym 'woodland' listed under the list of synonyms



Figure 7.58 Message confirming that a new synonym has been saved



If you wish to add a definition to the synonym, press **Details**. This will bring you directly to the 'Formal Term Details' dialog box (see above, 7.7.1 and Figure 7.53) where a definition can be added.

If a synonym is no longer appropriate or desired it can be deleted simply by highlighting it in the 'Synonyms' list box and pressing **Delete**. Note however, that if a synonym is already in use as a formal term in the knowledge base (i.e. appears in one of the formal statements), then it is not possible to delete it, and to remove it you must follow the same procedure as removing an unwanted formal term still in use (see above 7.7.3).

7.9 MEMOS

When opening an existing knowledge base, the first dialog box to appear is the Memo dialog box. This Memo dialog box can also be accessed by selecting **Memos** in the main **KB** menu.

The Memo dialog box should be used as an introduction to the knowledge base. The reason it appears as you open the knowledge base, is to give users, other than the knowledge base developer, an immediate general understanding of the knowledge base, thus facilitating access to and manipulation of that knowledge.

Figure 7.60a The Memo dialog box

Figure 7.60b The Further Details dialog box

Each section of the memo dialog box (Figure 7.60a) is self explanatory:

- **Title of this knowledge base:** – Here the full title of the knowledge base should appear, for example, the full title of the ‘treefodd’ knowledge base is ‘Tree Fodder and Indigenous Knowledge in Nepal’.
- **What the knowledge base is about:** – Here the area of local knowledge covered by the elicitation process should be described.
- **This knowledge base was developed by:**– Here the name of the knowledge base developer(s) is(are) recorded.
- **Acknowledgements:** here all acknowledgements are listed – to those, other than the developers, who assisted in the design and creation of the knowledge base and to any funding bodies that supported the project.

If you press on **Further Details** another dialog box appears (Figure 7.60b) giving further details of the knowledge base;

- **Purpose** - If the knowledge base was built as part of a particular project, or designed to supplement work in a certain field, this can be recorded here.
- **Methods** – Here information about the methods used for selecting the interviewees and collecting the knowledge may be described.
- **Location** – Here information about the exact location of the communities visited can be recorded.
- **Timing of Knowledge Collection** – The time of year, and the season that the knowledge collection was undertaken, may influence the knowledge elicitation process and should therefore be noted.
- **General Comments** – Here any other pertinent pieces of information may be recorded.

The Memo dialog box also contains a **Topics** button, which, if selected, takes the user directly to the Topic Hierarchies dialog box (7.12 below). This enables the user to have immediate access to the major subjects of interest in the knowledge base.



When you open a knowledge base, the Memo dialog box will be in Read Only format, with the **Save** button deactivated. This is to prevent you inadvertently making changes to the Memo field. To reactivate the Save button, and to make amendments to the Memo field, you must enter it via the **Memos** option in the main **KB** menu.

7.10 SELECTING SUBSETS OF THE KNOWLEDGE BASE

AKT enables the user to extract subsets of a knowledge base according to user-specified search criteria. Often, this selection of subsets of the knowledge base may be a part of the process of evaluating and using the knowledge base.

7.10.1 THE BOOLEAN SEARCH

The **Boolean Search** offers a search criterium for creating subsets of knowledge. You can search for individual keywords, formal terms or sources. It is also possible to link any of these together using **and** and/or **or**, or even whole hierarchies or subsets of hierarchies, and then search for these. It is also possible to carry out a search combining formal terms with sources.



For demonstrating the functions available in the Boolean Search, we have used the knowledge base 'soil'.

A Boolean search is carried out by selecting **Boolean Search** under the main **KB**. A 'Boolean search' dialog box appears (Figure 7.61).

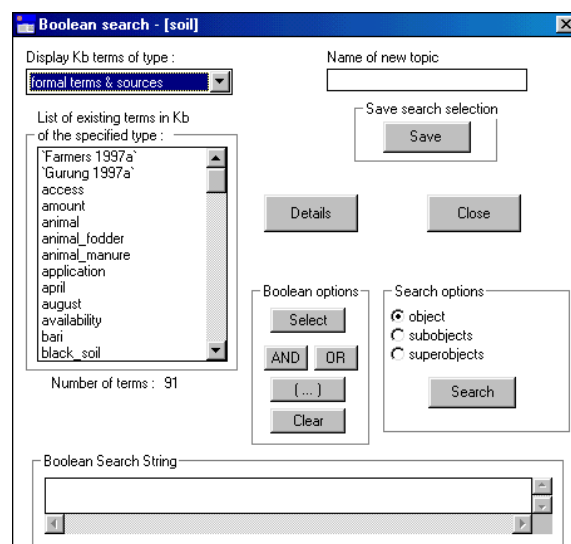


Figure 7.61 The 'Boolean Search' dialog box

The 'Display KB terms of type' box will give you the option of selecting one of the seven formal term types as well as sources, synonyms and topics. Highlight the type required. The 'List of existing terms in KB of the specified type' below will display all the terms under that type.

To conduct a Boolean search, highlight the term you wish to incorporate from the 'List of existing terms in KB of the specified type' box and press **Select** from the 'Boolean options' box. The chosen word will appear in the 'Boolean Search String' box at the bottom of the dialog box. The search is carried out by pressing **Search** in the 'Search options'.

7.10.1.a The use of 'and' and 'or'.

Further terms can be added to a Boolean Search String by clicking on **and** or **or** and then highlighting the next term and pressing **Select** again. The option **and** narrows a search by limiting the search to only those statements that contain all the terms selected. The option **or** broadens the search by including all statements that contain one or more of the formal terms selected.

7.10.1.b The use of brackets

If you use both **and** and **or** in your Boolean Search String, you must use brackets to construct the arguments logically, as one would in algebra. Using the knowledge base 'soil' imagine you wished to conduct a search which gave you all statements about degradation and soil and all the statements about erosion and soil. To do this you would use brackets in the Boolean Search String in the following way:

(degradation **and** soil) **or** (erosion **and** soil) (Figure 7.62)

7.10.1.c Studying the 'search results' using 'Diagram Selection Type'

Once you have formulated the Boolean Search String and pressed **Search**, the resulting pop-up menu, the 'Search Results', displays all the statements found by the search (Figure 7.63).

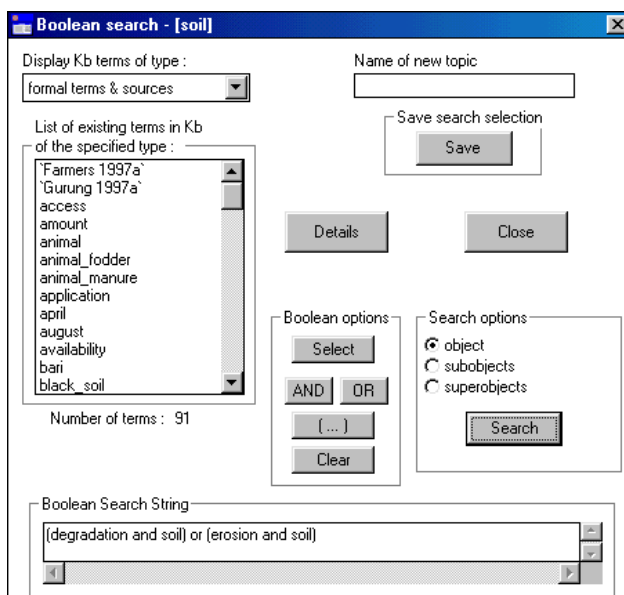


Figure 7.62 Carrying out a Boolean Search in 'soil'

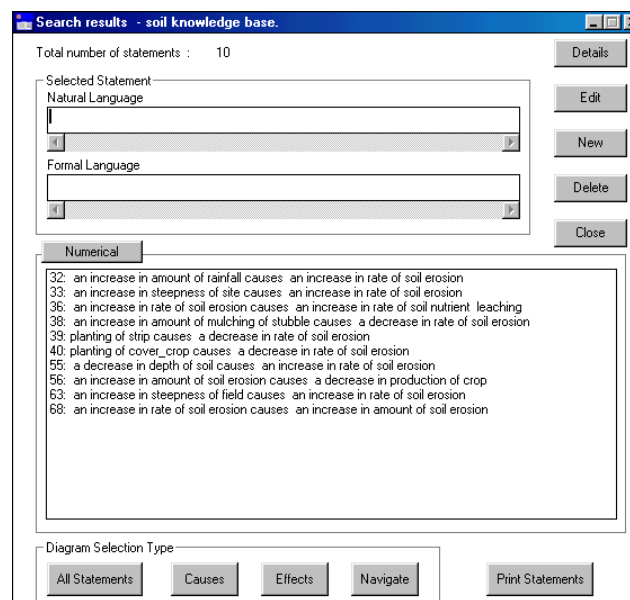


Figure 7.63 Search results from the Boolean Search in Figure 7.62

If you press **Diagram** in the 'Search Results' dialog box, a diagram of all the statements from that search, which can be diagrammatically represented, appears (Figure 7.64). (See Chapter 8 for further details about the diagramming interface).

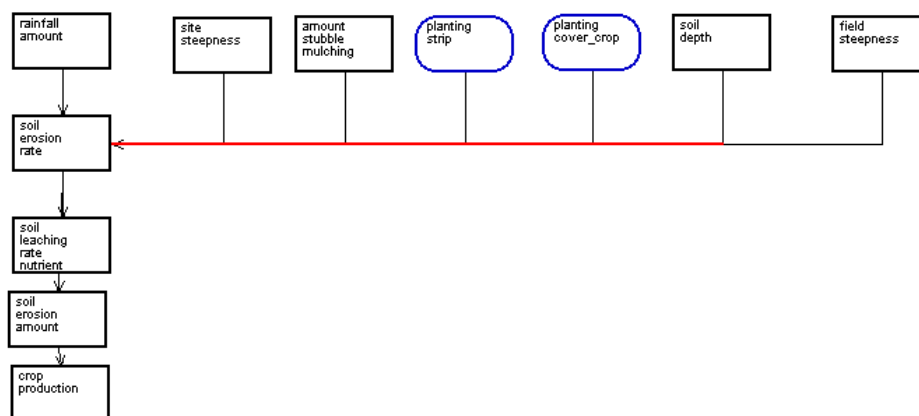


Figure 7.64 The diagrammatic representation of the Search Results in Figure 7.63

It is also possible to highlight any one of the statements in the 'Search Results' and then click on **Navigate**. This option will give you all the *immediate* causes and effects associated with that particular statement. Figure 7.65 gives a diagrammatic representation of all the nodes immediately associated with statement no.32 in the 'Search Results' (see Figure 7.63 above) - 'an increase in amount of rainfall causes an increase in rate of soil erosion'.

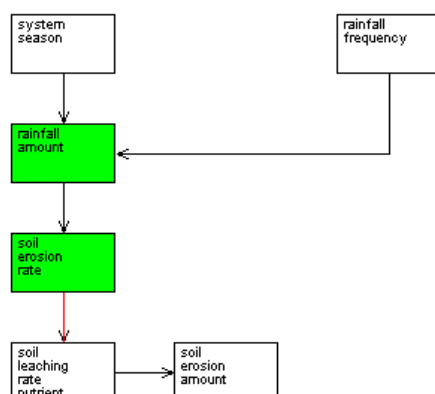


Figure 7.65a The diagrammatic representation of all the causal statements immediately connected to Statement No.32 'an increase in the amount of rainfall causes an increase in the amount of soil erosion.'

By selecting **Statements** on the left-hand side of the Diagram screen (see Chapter 8), you will get a full list of the statements incorporated into the diagram (Figure 7.65b)

5 of the 73 knowledge base statements are represented in this diagram.

- 32: an increase in amount of rainfall causes an increase in rate of soil erosion
- 36: an increase in rate of soil erosion causes an increase in rate of soil nutrient leaching
- 57: the system season is winter causes the amount of rainfall is low
- 65: a decrease in frequency of rainfall causes a decrease in amount of rainfall
- 68: an increase in rate of soil erosion causes an increase in amount of soil erosion

Figure 7.65b Statements incorporated in the diagram in Figure 7.65a

It is also possible to use the **Causes** and **Effects** buttons. In the following example we have highlighted the same statement as above, statement no. 32 'an increase in amount of rainfall causes an increase in rate of soil erosion' and then pressed **Causes**. The diagram in Figure 7.66a depicts the causes that flow *into* statement no.32 and Figure 7.66b presents them in list form.

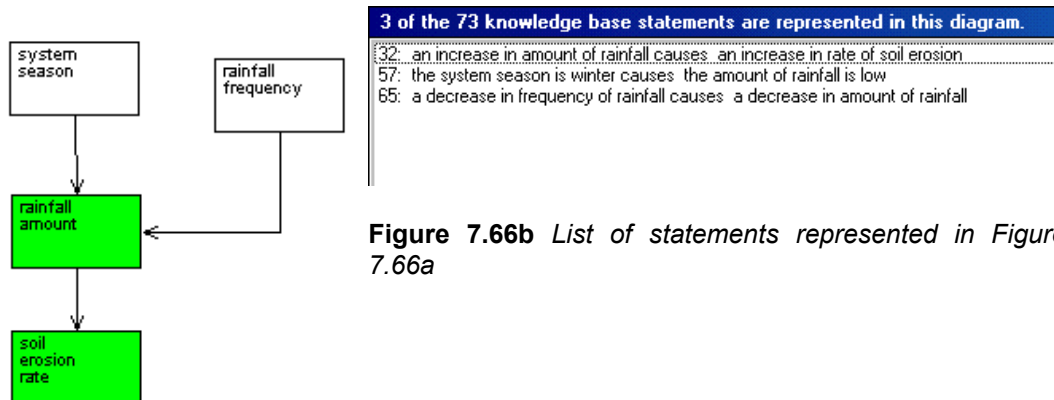


Figure 7.66b List of statements represented in Figure 7.66a

Figure 7.66a Causal flow into Statement No.32

Using the same statement no. 32 once more we pressed **Effects**. Figure 7.66c is a diagram of all the effects flowing *from* the statement no. 32 and Figure 7.66d presents them in list form.

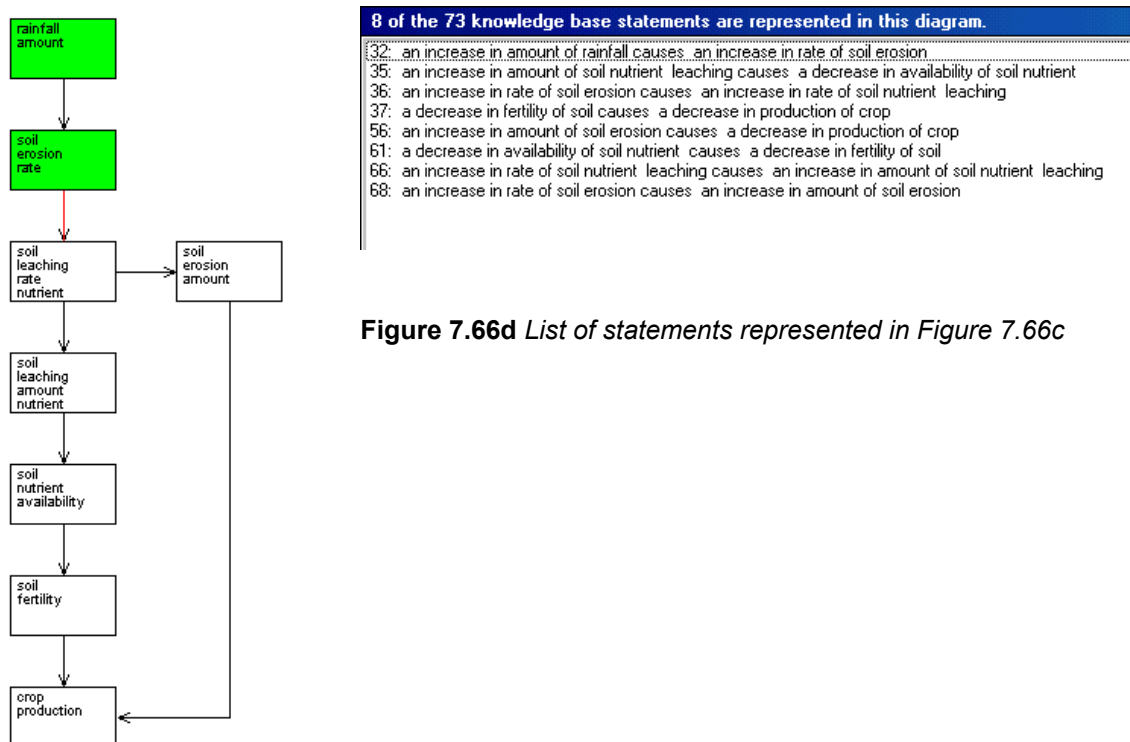


Figure 7.66d List of statements represented in Figure 7.66c

Figure 7.66c Causal flow out of statement no. 32



To summarize 'Diagram Selection Type'

All Statements– gives you a diagram of all the statements extracted by the Boolean Search which can be presented in diagrammatic form (i.e. causal and link statements)

Causes – gives you a diagram of *all* the causes of one statement selected from the statements extracted by the Boolean Search.

Effects – gives you a diagram of all the effects of one statement selected from the statements extracted by the Boolean Search.

Navigate – gives you a diagram of all the *immediate* causes and effects of one statement selected from the statements extracted by the Boolean Search.

7.10.2 USING THE BOOLEAN SEARCH ON OBJECT HIERARCHIES

For the objects amongst the formal terms selected in the 'Boolean Search String', a choice of search modes is available in the 'Search options' box; 'object', 'subobject' and 'superobject' (see above, Figure 7.62). By highlighting one or more of these options there are 7 possible combinations for the Boolean search. For example 'object' alone recognises only those statements in which the object selected is actually used; 'object' plus 'subobject' extracts unitary statements in which either the specified object or the children and descendants of that object in the object hierarchy are used; and 'object' plus 'subobject' plus 'superobject' extracts unitary statements in which either the specified object, its children and descendants, its parent, or ancestors all the way up the hierarchy tree, are used.

The collection of statements displayed as a result of a Boolean Search will assist in the interactive evaluation of the knowledge base contents, but it is worth noting that all of these search facilities can be invoked by one or more of the primitive tools (see chapter 9).

7.11 TOPICS

7.11.1. CREATING A TOPIC

It is possible to save any combination of search criteria to gather all the information about a particular topic within a knowledge base. To create a topic, first select your search criteria for the Boolean Search String, as described above (7.10.1). Then enter a topic name in the 'Name of new topic' box in the 'Boolean Search' dialog box (Figure 7.67) and press **Save**. The ensuing menu, 'Topic details' will permit you to add a description of the topic (Figure 7.68).

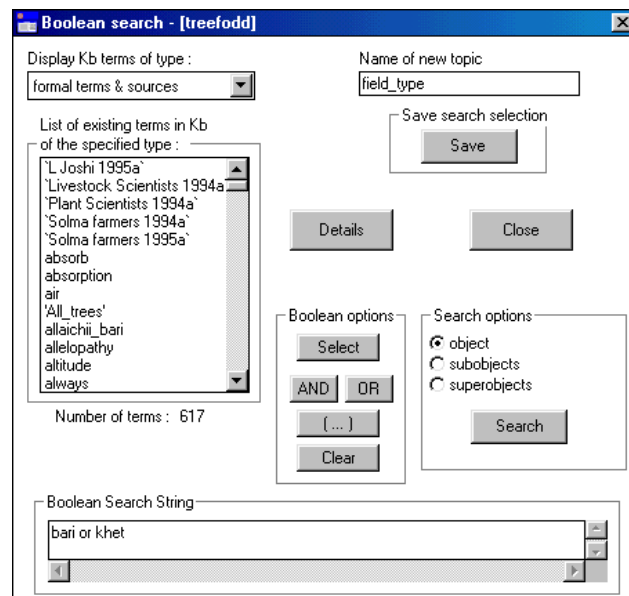


Figure 7.67 Creating a topic

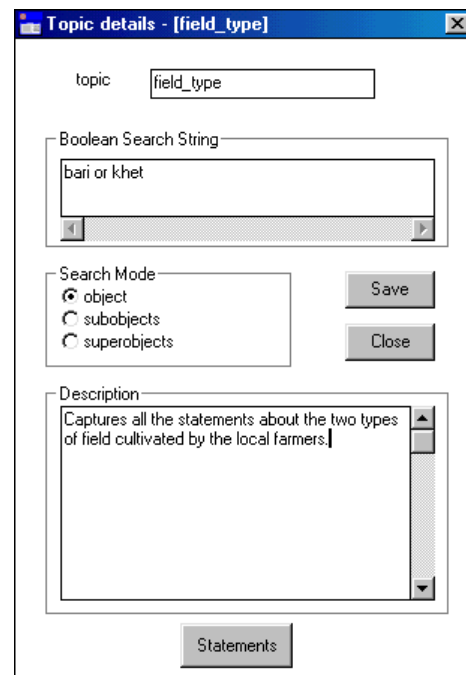


Figure 7.68 Filling in the 'Topic details' dialog box

If the search string incorporates hierarchic objects then the 'Search Mode' will allow you to specify further search criteria. Once you have filled in all the details, press **Save**. A message will appear (Figure 7.69) confirming that the topic has been saved.

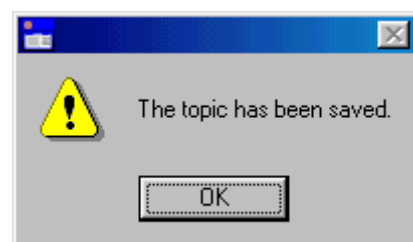


Figure 7.69 Message confirming that a new topic has been saved

Once you have pressed **OK** the screen reverts back to the 'Boolean Search' dialog box.

7.11.2 MANAGING TOPICS

To manage topics already created, it is necessary to select **Topics** from the main **KB** menu. The following dialog box then appears (Figure 7.70);

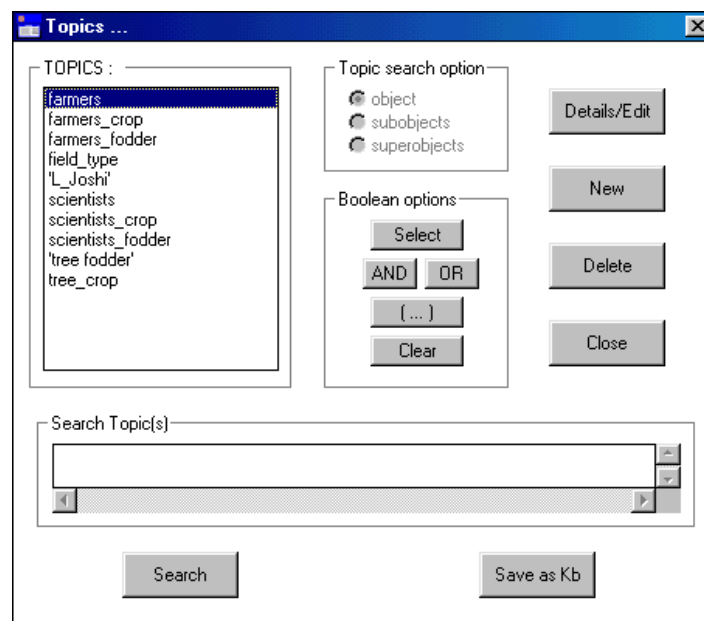


Figure 7.70 Topics dialog box

To see the details of a topic or to edit a topic, highlight the topic and press **Details/Edit**. This will bring up the 'topic details' box (see above 7.11.1 and Figure 7.68) which gives the Boolean Search String it uses, the search mode for objects and a description of the topic. If you wish to amend the details, it can be done from this dialog box, pressing **Save** when the amendments are complete. If you just wish to view the details, press **Close** when you have finished.

It is also possible to create a new topic by selecting **New**. If you select **New** the 'Boolean Search' dialog appears as above (see 7.10.1). To see the statements attached to a topic highlight the topic in the 'Topics' box, press **Select** so that the topic appears in the 'Search Topic(s)' box and then press **Search**. This brings up the 'Search Results' dialog box listing all the statements captured by that topic.

If you wish to delete a topic, you must first highlight the topic in the 'Topics' list and then press **Delete**. A message will then appear to confirm whether you want to delete the topic (Figure 7.71). Press **Yes** to confirm.

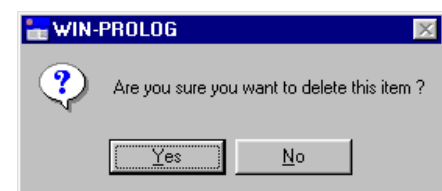


Figure 7.71 Message to confirm that the alias is to be deleted

7.11.3 USING TOPICS

It is possible to search for several topics at a time. If you wish to see statements of more than one topic, highlight the desired topics one at a time, press **Select** and then press **or** before adding another topic to the 'Boolean Search String'. If you wish to find only the statements which appear in two or more topics, use **and** to link the topics.

The 'Topic Search Option' box shows what combination of object, subobjects and superobjects was used to create the topic. This *cannot* be altered from within the 'Topics' dialog box. To alter the combination, you must press **Details/Edit** and revert to the 'Boolean Search' dialog box and alter it there.

Figure 7.72 is an example of combining two topics using OR. The two topics chosen were SoilColour and SoilTexture. The 'Boolean Search String' at the bottom of the dialog box gives the two topic names chosen. Once the selection is made, press **Search** and the 'Search Results' dialog box appears (Figure 7.73).

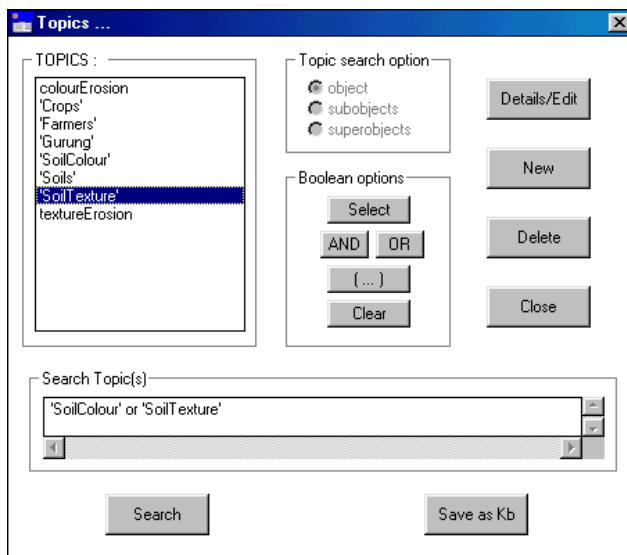


Figure 7.72 Boolean Search String for two topics combined, SoilColour and SoilTexture

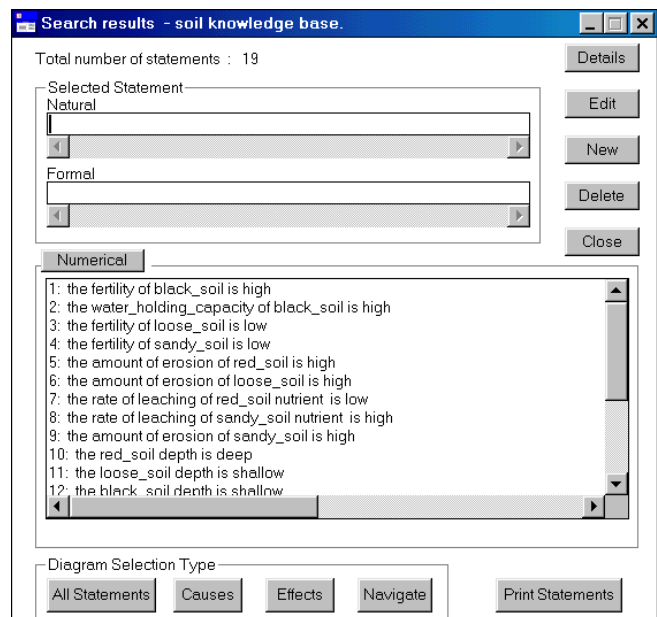


Figure 7.73 The 'Search Results' dialog box for the Boolean Search String in Figure 7.69

7.11.4 CREATING A NEW KNOWLEDGE BASE OUT OF A TOPIC

It is also possible to convert a topic, or a string of topics into a new knowledge base. This option is of use if the original knowledge base has become too large and unwieldy, or if so much information exists on a specific topic that it justifies creating a new, separate knowledge base. It may also be the case when a knowledge base is to be divided up between researchers, each with a separate remit, to study one particular aspect of the mother knowledge base.

In order to do this, highlight the topics you wish to incorporate into the new knowledge base - for this reason it is possible to highlight several topics at once. Then press **Save as KB**. This brings you to the 'Name of new KB' dialog box (Figure 7.74) where you can save the topics as a knowledge base under a new name and under a new folder/directory if necessary.

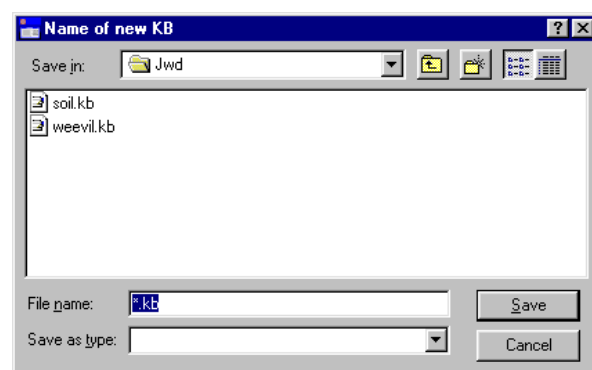


Figure 7.74 'Name of new KB' dialog box

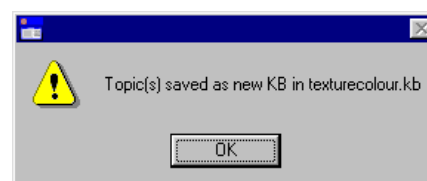
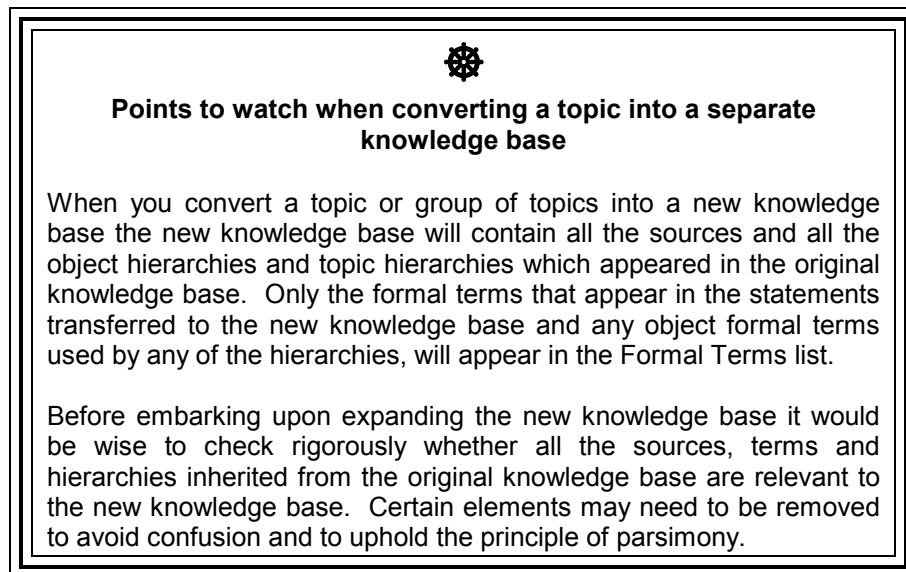


Figure 7.75 Message to confirm that the topics have been saved as a new knowledge base

Once you have entered a name for the new knowledge base and pressed **Save**, the screen reverts to the original knowledge base and a message appears that the topics have been saved as a new knowledge base (Figure 7.75).



7.12 TOPIC HIERARCHIES

As one can have object hierarchies, so one can create topic hierarchies. A single knowledge base may contain a number of discrete topic hierarchies and a single topic may appear in more than one hierarchy. When you first open your knowledge base the 'Topic Hierarchies' dialog box can be accessed directly via the Welcome Memo by pressing the **Topics** button. This is to give the user an idea of what topics are covered by the knowledge base. If you wish to return to it later, you enter **Topic Hierarchies** via the main **KB** menu.

7.12.1 VIEWING TOPIC HIERARCHIES

In order to view a topic hierarchy, enter **Topic Hierarchies** via the main **KB** menu. The 'Topic Hierarchies' dialog box appears listing the topic hierarchies that exist. If you highlight one of the topic hierarchies, the 'Topics in selected hierarchy' box produces a list of topics within the topic hierarchy (Figure 7.76).

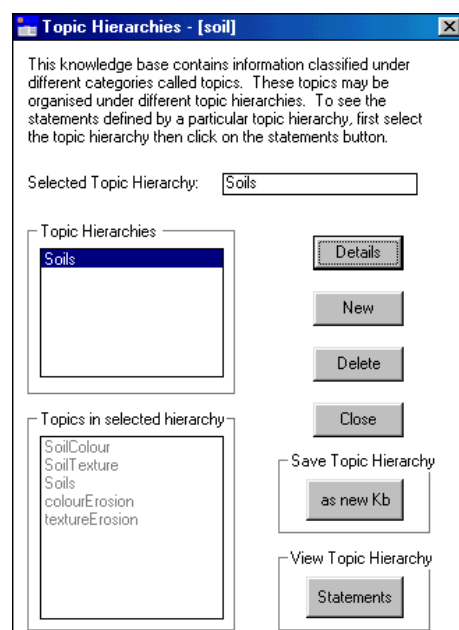


Figure 7.76 'Topic Hierarchies' dialog box

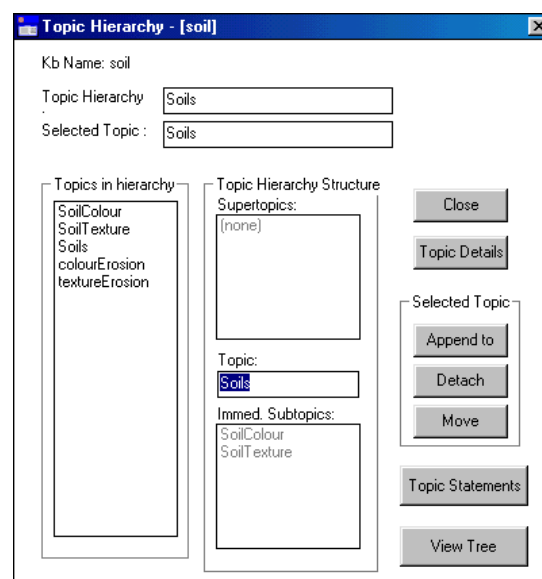


Figure 7.77 'Topic Hierarchy' dialog box

The relevant 'Topic Hierarchy' dialog box (Figure 7.77) appears automatically when you highlight a topic hierarchy in the 'Topic Hierarchies' dialog box and is very similar to the 'Object Hierarchy' dialog box explained above (see 7.5.2 and compare with Figure 7.29).

The **Topic Details** allows you to see the 'Topic Details' dialog box giving the Boolean Search String that made up the topic hierarchy, the search mode for objects and any description that might have been written on the topic hierarchy.

Topic Statements allows you to see and browse all the statements encompassed in the topic hierarchy. **View Tree** allows you to see the structure of the topic hierarchy.

7.12.2 CREATING TOPIC HIERARCHIES

In order to create a new topic hierarchy, select **New** in the 'Topic Hierarchies' dialog box (see above, Figure 7.76). A 'New Topic Hierarchy' dialog box appears (Figure 7.78).

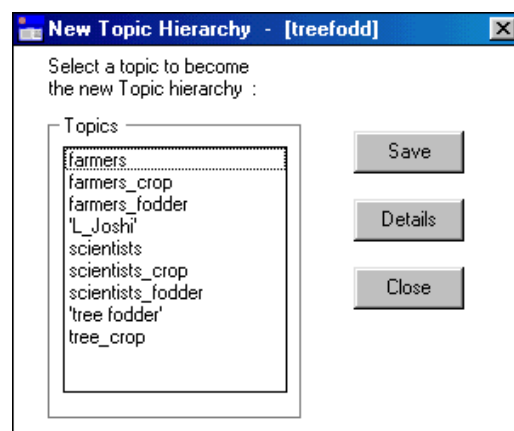


Figure 7.78 'New Topic Hierarchy' dialog box

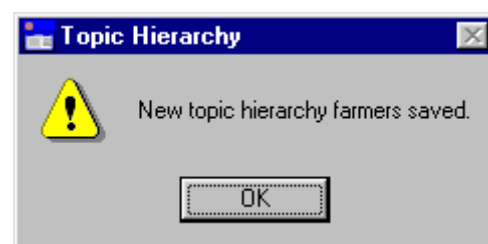


Figure 7.79 Message confirming that a new topic hierarchy has been saved

Unlike Object Hierarchies where it is possible to create a new name for the hierarchy, the name of the new Topic Hierarchy must come from one of the topics already created. Highlight the chosen topic in the 'Topics' box. If you want to make sure how the topic was originally defined, (i.e. the Boolean Search String used), select **Details**. After checking the details, press **Save**. A message will appear, confirming that a new topic hierarchy has been saved (figure 7.79) and the topic name will now appear in the 'Topic Hierarchies' list of the 'Topic Hierarchies' dialog box (Figure 7.80).

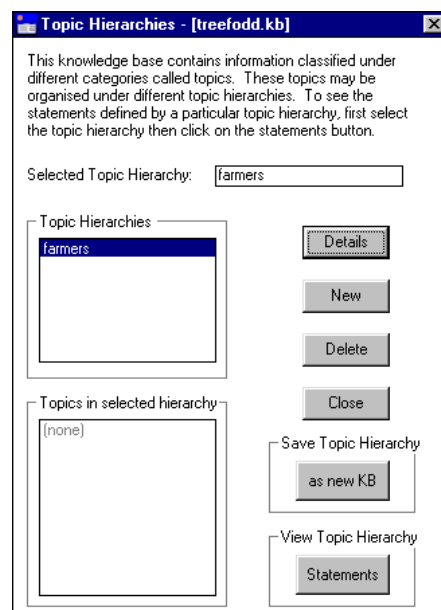


Figure 7.80 The new topic hierarchy appearing in the 'Topic Hierarchies' list box

To view all the statements, which form part of the topic hierarchy, select **Statements**. This will bring up a 'Search Results' dialog box with all the statements in the topic hierarchy listed in natural language.

To delete a topic hierarchy, highlight the relevant one and press **Delete**. A message will appear confirming that the topic hierarchy has been deleted.

7.12.3 APPENDING TOPICS TO A TOPIC HIERARCHY

In order to append topics to a topic hierarchy, highlight the chosen topic hierarchy in the 'Topic Hierarchies' dialog box. In the new dialog box 'Topic Hierarchy', select **Append**. An 'Append Topic' dialog box will appear (Figure 7.81)

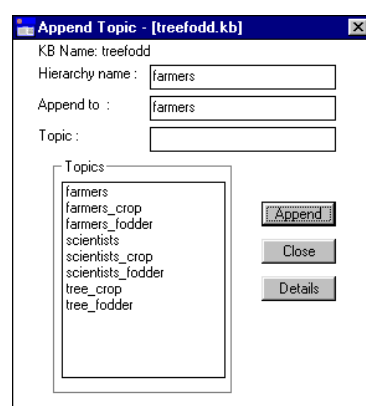


Figure 7.81 'Append Topic' dialog box

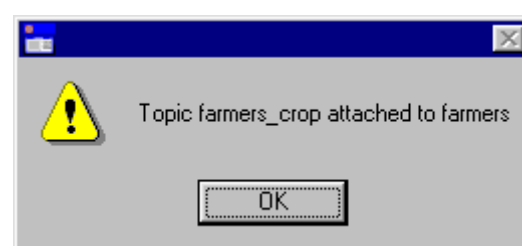


Figure 7.82 Message confirming that the topic has been appended to the topic hierarchy

Select the chosen topic to be appended and press **Append**. A message will appear confirming that the topic has been appended to the topic hierarchy (Figure 7.82). In the example used here, a topic hierarchy 'Farmers' was created in the 'treefodd' knowledge base, to which the topics 'farmers_crop' and 'farmers_fodder' were appended. The 'Topic Hierarchy' dialog box gives the structure of the hierarchy (Figure 7.83).

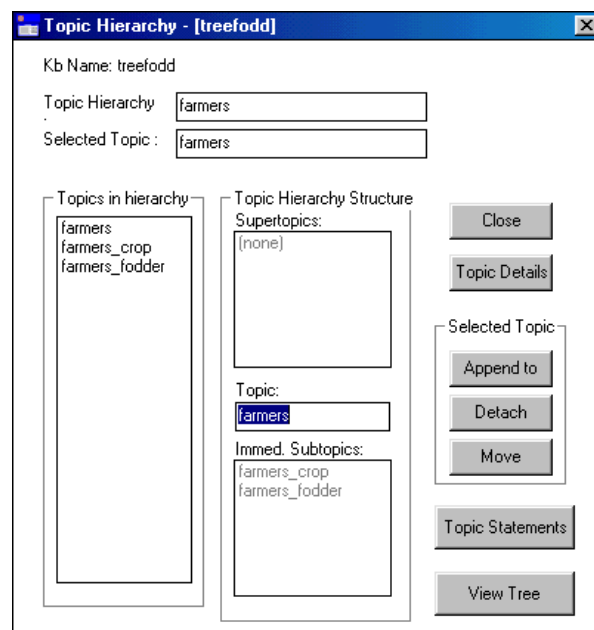


Figure 7.83 The subtopics 'farmers_crop' and 'farmers_fodder' appended to the topic hierarchy 'farmer'

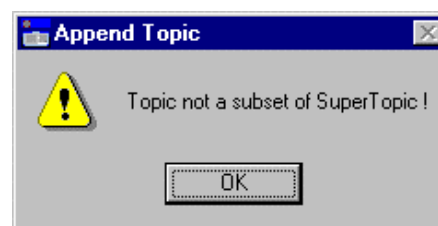


Figure 7.84 Message appearing when it is not possible to append a topic to a topic hierarchy.

Topics can *only* be appended to a topic hierarchy if they are a subset of that topic hierarchy. In other words their 'Boolean Search String' can only contain formal terms/sources also included in the topic hierarchy. In this case 'farmers_crop' and 'farmers_fodder' are both subsets of 'farmers'.

If you attempt to append a topic with a 'Boolean Search String' which is not a subset of that of the super-topic, for instance, if we try to append 'scientists' as a sub-topic of 'farmers', a message will appear warning you that the topic is not a subset of the super-topic (Figure 7.84).

7.12.4 CREATING A NEW KNOWLEDGE BASE OUT OF A TOPIC HIERARCHY

Just as one can save a topic as a new knowledge base, it is also possible to save a topic hierarchy as a new knowledge base. In order to do so, highlight the topic hierarchy in question in the 'Topic Hierarchies' dialog box, and select **as new KB** from the 'Save Topic Hierarchy' box. The procedure is then identical to that of saving a topic as a new knowledge base (see above 7.11.4).

7.13 PRINTING AND SAVING SEARCH RESULTS

In many cases it may be desirable to print out or to save search results made via the **Formal Terms** or **Boolean Search** dialog boxes. In order to do so, press **Print Statements** at the bottom of the 'Search Results' dialog box (Figure 7.85). A message appears (Figure 7.86) asking whether you wish to save the output to a text file.

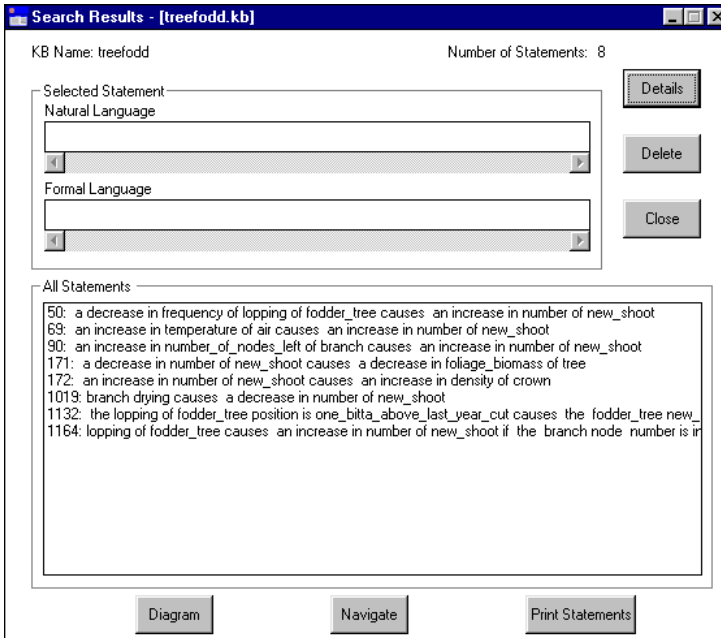


Figure 7.85 'Search Results' dialog box for the formal term 'new_shoot'

If you select **Yes** then the following 'Save output in file' screen will appear (Figure 7.87);

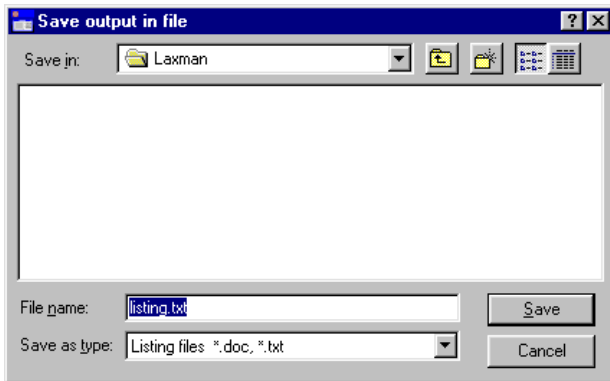


Figure 7.87 Saving the Search Results statements to a file

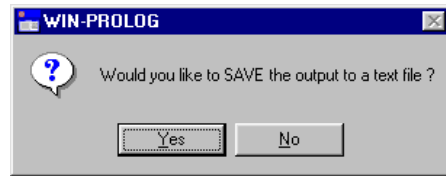


Figure 7.86 Message which appears when you select **Print Statements**

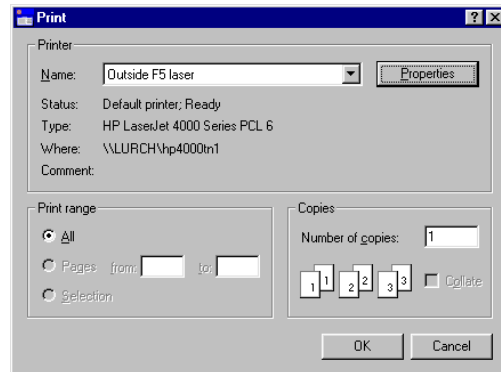


Figure 7.88 'Print' dialog box

This allows you to save the text as a text file in a directory and folder of your choice. Once you have named the file and saved it in the correct place, press **Save**.

If you decide not to save the file and press **No** in Figure 7.86 above, the printer details dialog box will appear, allowing you to change its properties before printing.

7.13.1 PRINTING IN LANDSCAPE

If the statements are particularly long, it may be better to print out the statements in landscape. In order to do so, select **Properties** and then **Basics** from the printer details dialog box (Figure 7.89) and select the **Landscape** radio button under 'Orientation'. Then press **OK**. This will return you to the 'Print' dialog box as above (Figure 7.88).

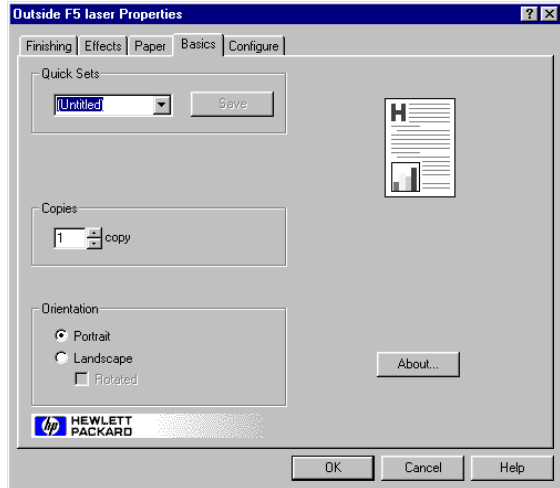


Figure 7.89 Selecting the Landscape option in print 'Properties'

7.14 SAVING A KNOWLEDGE BASE

7.14a Saving a knowledge base on the hard drive

If you wish to save any additions or alterations to a knowledge base you have made since creating it or opening it, you must always select **Save KB** before you close the knowledge base down by selecting **Close KB**.

7.14b Saving a knowledge base to a floppy disk or a cd

If you wish to save a knowledge base to a floppy disk or cd, *do not* save it directly to the disk because it will take a very long time. Save it to the hard drive first and then, using Windows Explorer, select the saved file, press **Ctrl c**, open the floppy disk drive / cd drive and press **Ctrl v**.

7.14c Saving the knowledge base under a different name

If you want to change the name of a knowledge base, or create two versions of a knowledge base, then before closing the knowledge base you should select **Save KB As**. This allows you to rename the knowledge base and retain the original version, by giving the most recent version a new name, or to save the knowledge base in another folder or directory.



WARNING

If you decide to change or edit the name of any knowledge base, it must only be done from *within* AKT via the **Save KB As** option. If it is done any other way then AKT will not be able to load the knowledge base correctly.

7.15 MOVING FROM ONE KNOWLEDGE BASE TO ANOTHER

It is possible to load more than one knowledge base onto the program at a time. The advantage of this is that it enables you to switch quickly from one knowledge base to another, without exiting the program, something which is of particular value when running the tools (see Chapter 9).

In order to load the knowledge bases onto the program, simply select **Open KB** from the main KB menu, and then choose the knowledge base you want. You load the knowledge bases one at a time, selecting **Open KB** each time. The most recent knowledge base to be loaded will be the one that appears on the screen. To move from one knowledge base to the other select **Select KB** from the main **KB** menu. The following dialog box appears (Figure 7.90).

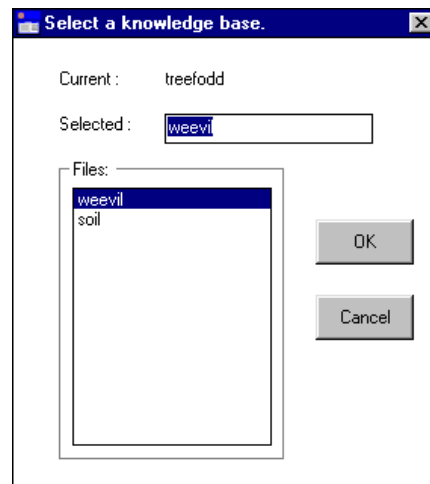


Figure 7.90 *Moving from one knowledge base to another.*

In this example the current knowledge base is 'treefodd'. The other two knowledge bases loaded, 'weevil' and 'soil', appear in the 'Files' list. To change the current knowledge base, highlight one of the knowledge bases in the 'Files' list and press **OK**. The selected knowledge base will now automatically become the current knowledge base.

CHAPTER EIGHT - THE DIAGRAM INTERFACE

8.1 INTRODUCTION

The diagram interface serves a number of purposes. It provides a means of easily entering knowledge into the knowledge base as diagrammatic, linked sets of unitary statements. It also gives a pictorial view of the unitary statements entered into the knowledge base via the statement interface and it provides a number of functions to manipulate and organise the knowledge represented in the diagram.

Whenever a new statement is added by joining two diagram nodes with a link, a corresponding formal language unitary statement and its natural language equivalent are generated in the knowledge base. Equally, whenever a causal or link statement is entered into the knowledge base via the statement dialog, any relevant diagrams are automatically updated to include the new statement.

Only causal or link statements (including conditional statements) can appear in diagrammatic form. The system will generate a complete diagram of the whole knowledge base, i.e. one that includes all the causal and link statements. However it is possible to build up a set of diagrams showing different sections of the knowledge base. This is particularly useful in large knowledge bases where a diagram can show a subset of the knowledge base such as a particular topic or a set of statements that are of interest (see section 8.3 below).

A diagram represents a particular view of the internal knowledge base. There can be many views co-existing, showing differing aspects of the knowledge. It is important to remember however, that if you add or delete nodes and links from any diagram, the knowledge base will reflect that change. The knowledge base and any views of that knowledge will always be consistent.

8.2 VIEWING A DIAGRAM

The diagram menu (Figure 8.1) shows the options available in the diagram interface.

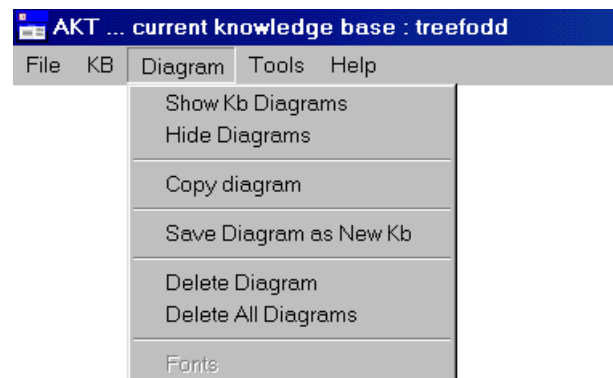


Figure 8.1 The main *Diagram* menu

If you are working on a new knowledge base, **Show Kb Diagrams** will give you a blank screen on which you can begin to create a knowledge base from scratch, using the diagram functions. (For creating a new diagram see 8.8).

If you are working on a previously created knowledge base where no sub diagrams have been developed previously, then the 'Select Diagram' dialog box will not appear, instead the program will automatically generate a diagram for the complete knowledge base (Figure 8.2). (See 8.3.4 for viewing subdiagrams).

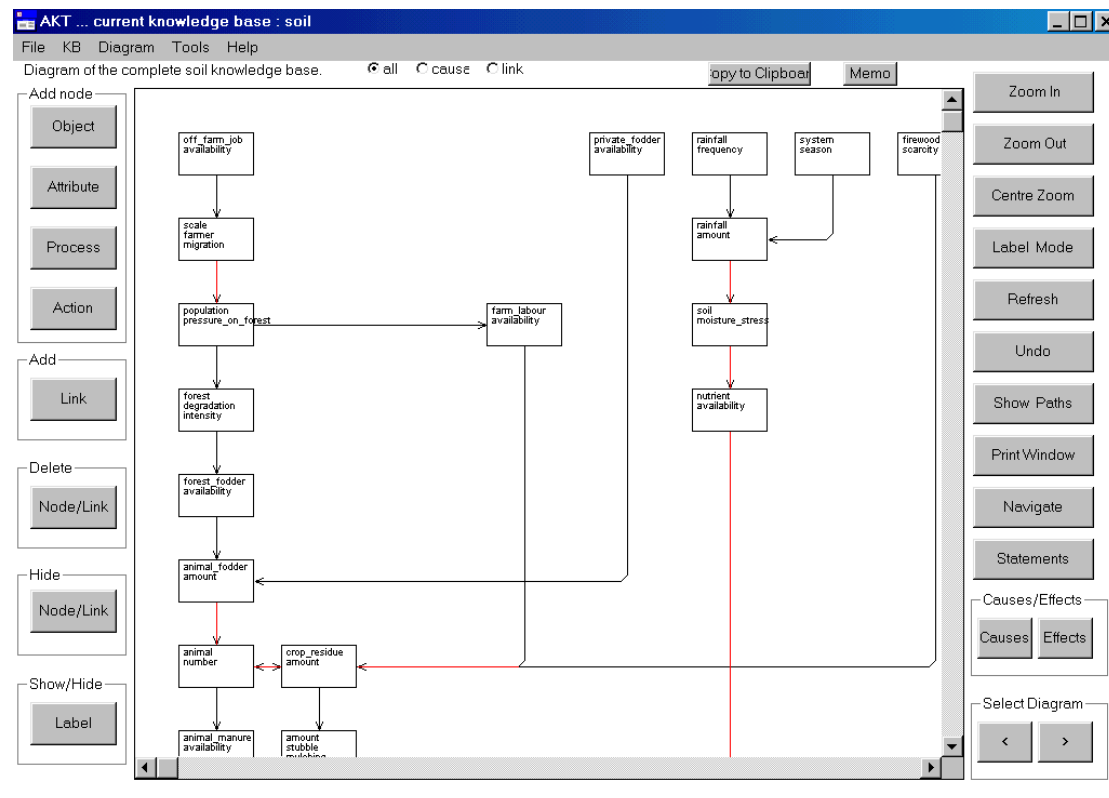


Figure 8.2 Part of the diagram for the complete 'soil' knowledge base

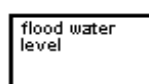
Figure 8.2 above shows a diagram screen, the diagram in the middle of the screen, and buttons either side. The buttons on the left-hand side are for node and link creation and deletion, the buttons on the right hand side are for viewing and manipulating the diagrams. In this chapter we will first explore how to view and manipulate a pre-existing diagram. In the second half of the chapter (8.8 onwards) we will concentrate on creating a knowledge base via the diagram interface.

8.2.1 DIFFERENT NODES

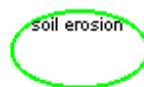
If you look at the diagrams generated by 'soil' and 'treefodd' you will see that there are four different node shapes, one each for Object, Attribute, Process and Action respectively, and, on screen, they are also different colours, so that the type of node can be identified immediately in a diagram (Figures 8.3a to 8.3d).



(yellow)
Figure 8.3a
Object Node



(black)
Figure 8.3b
Attribute Node



(green)
Figure 8.3c
Process Node



(blue)
Figure 8.3d
Action Node

The size of these nodes can be altered by placing the mouse over the node, pressing down the right-hand mouse button and dragging the mouse either outwards to enlarge the node or inwards to shrink the node. This facility is particularly useful if the node text is too large for the original node as in Figures 8.4a and 8.4b.

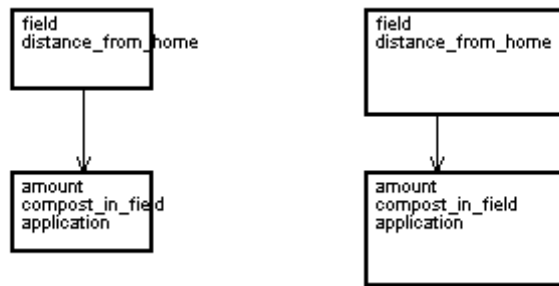


Figure 8.4a Nodes with overlapping text **Figure 8.4b** Enlarged nodes

8.2.2 DIFFERENT LINKS

On screen the links between nodes are either black or red. Black links are single links between two nodes, red links appear where two or more links overlap.

8.2.3 THE ZOOM

To view more of the diagram it is possible to move the scroll bar at the bottom of the diagram screen from left to right, and the scroll bar on the right hand side of the diagram up and down. However, to get a clearer picture of all the interconnections, zoom options have been provided.

8.2.3.a Zooming in and out

Figure 8.2 above is the visible portion of the diagram of the complete 'soil' knowledge base. In order to get a complete picture of the interactions between the different nodes, the **Zoom Out** button will reduce the diagram as in Figure 8.5.

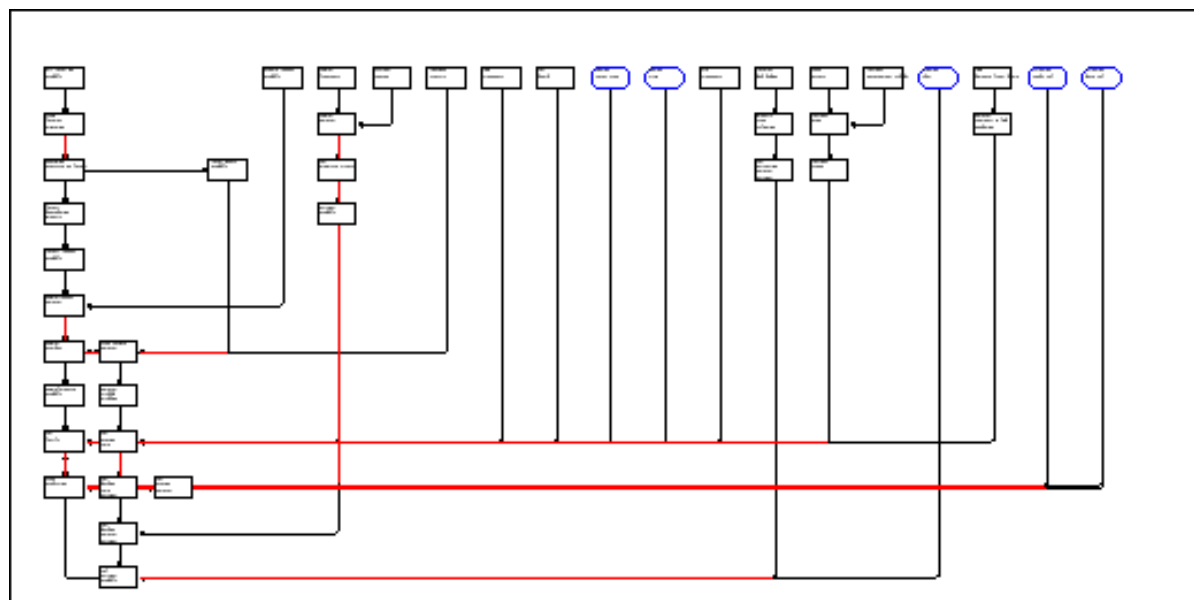


Figure 8.5 The reduced diagram of the 'soil.kb' knowledge base, using Zoom Out

In order to enlarge the image again, **Zoom In** is selected. There are four zoom levels with ratios of 1:1, 2:1, 4:1, and 7:1.

8.2.3.b Centre zoom

It is possible to recentre and magnify the diagram instantly by selecting **Centre Zoom** and then placing the mouse pointer over the node you wish to appear in the centre of the diagram and clicking once with the left mouse button.

8.2.4 LABELLING

8.2.4.a 'causes1way', 'causes2way' labelling

A diagram can be presented in three forms. The simplest form is as the diagram above (Figure 8.2) where only the link and nodes are drawn. By clicking on the **Label Mode** once, the diagram changes to Figure 8.6. The '1' or '2' beside the links indicates whether the link is a 'causes1way' or 'causes2way' causal statement. The ↑ arrow indicates an increase whilst the ↓ arrow indicates a decrease. Where something else other than an increase or decrease of the first node causes an increase or decrease in second node, this is indicated by a dot (•)¹.

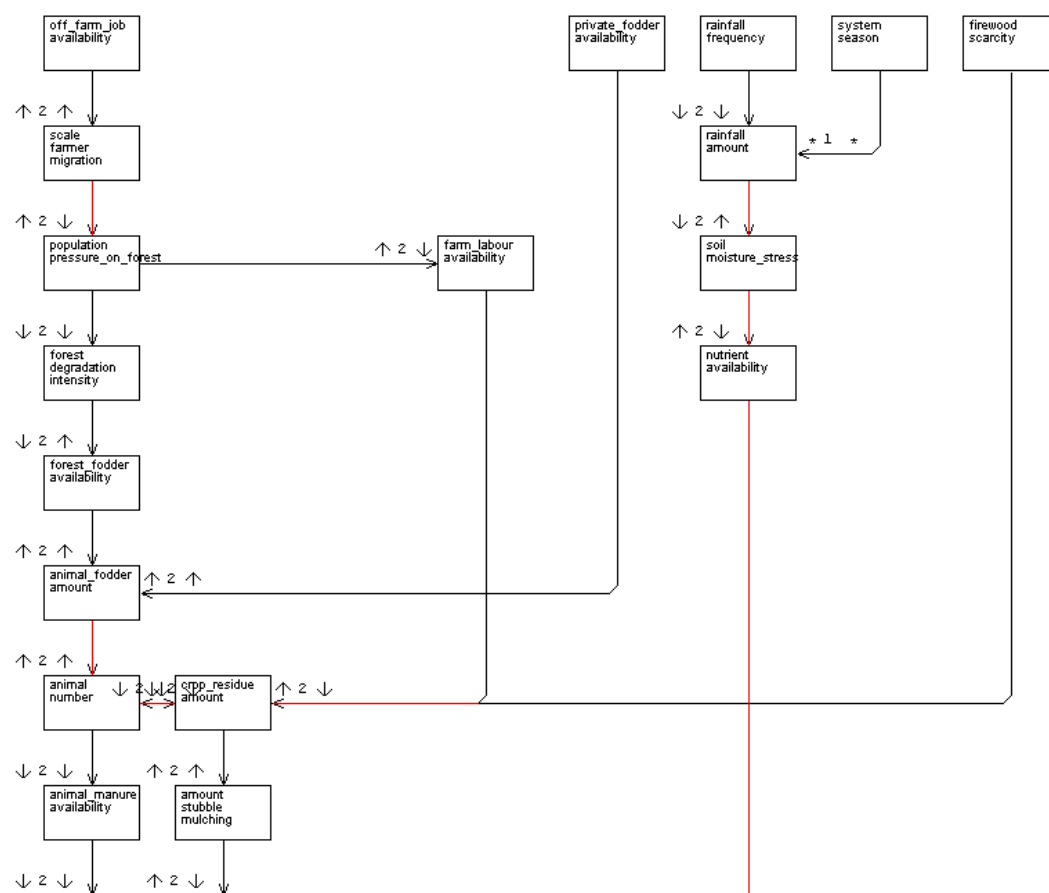


Figure 8.6 *Diagram with labelling*

Where there are several links entering one node from the same direction, i.e. where the link line appears red on the screen, the symbols (e.g. ↑2↑, or ↓1↑) may overlap as in Figure 8.7a. In order to see the symbols properly, select the symbol with the left-hand mouse button and drag the symbol a little way to reveal the symbol underneath, as in Figure 8.8b.

¹ Thus ↑ 2 ↑ would indicate 'an increase in X causes2way increase in Y', whilst • 1 ↓ would indicate 'something (other than an increase or decrease) in X causes1way decrease in Y'.

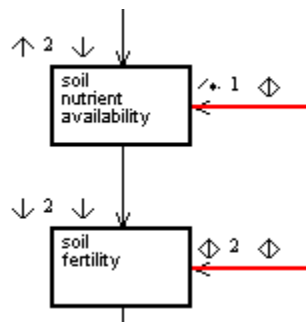


Figure 8.7a *Overlapping symbols*

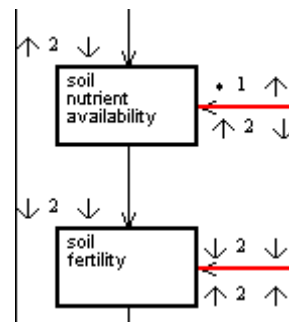


Figure 8.7 b *The symbols properly revealed*

8.2.4.b Link labelling

If the connection between two nodes is not a causal link, but a simple link (as in 'sheep eat clover') instead of symbols appearing, the names of all the links will be displayed (Figure 8.8).

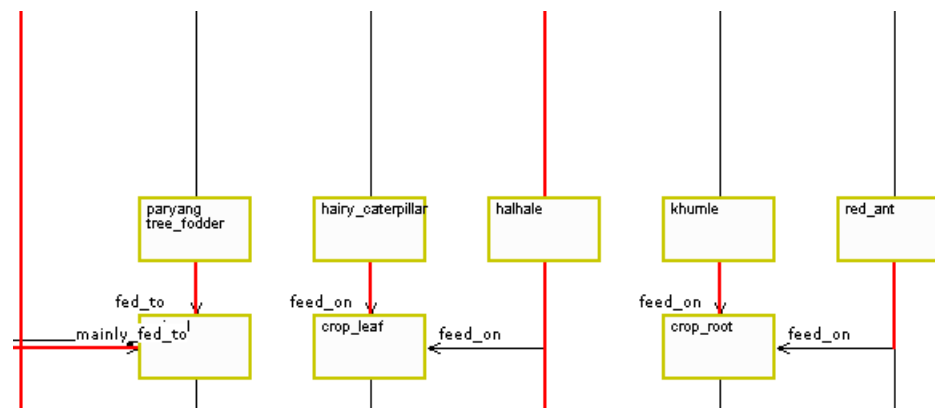


Figure 8.8 *Labelling of simple links (i.e. non-causal links) between nodes (in 'treefodd')*

8.2.4.c Showing only causal statements or only link statements

At the top of the diagram screen are three radio buttons, **all**, **causal**, **link**. If you select **all**, all the statements that can be represented in the diagram are represented. If you select **causal**, only the causal statements will appear in the diagram. If you select **link**, only the link statements will appear in the diagram. Try it out by selecting one radio button at a time, using the **Zoom Out** function to see the difference between each selection.

8.2.4.d Statement labels

Pressing the **Label Mode** button a second time will reveal the full natural language statement represented by the link and node diagram (as in Figure 8.9). If a condition has been attached to a statement via the 'New Statement' dialog box then this condition will appear in the natural language labels.

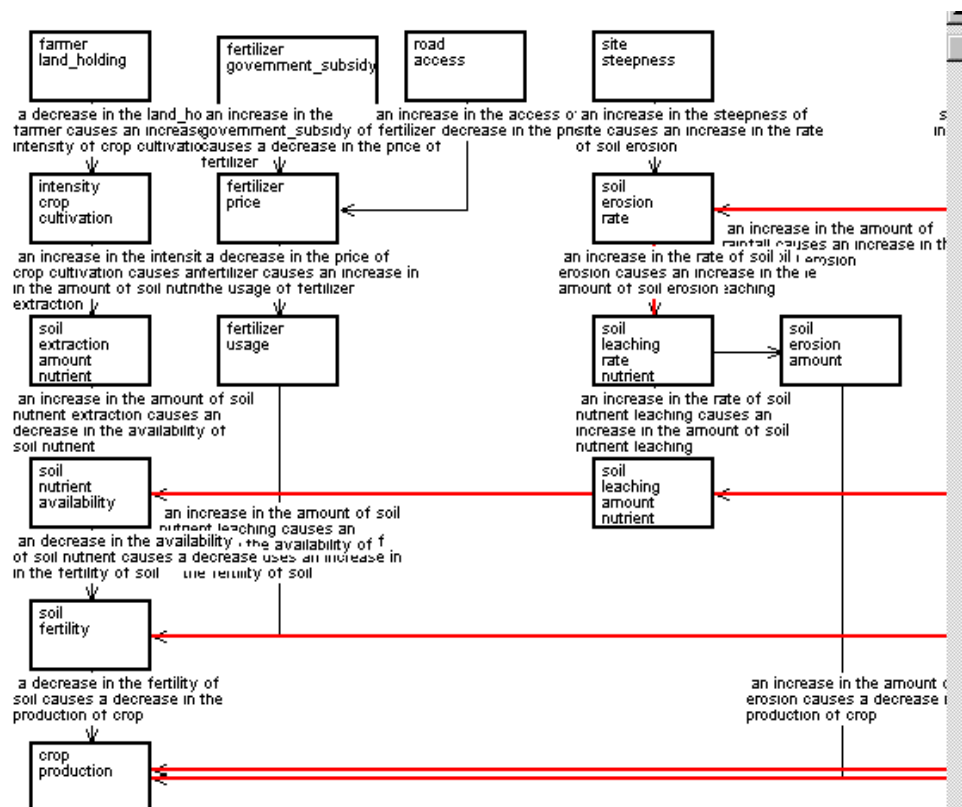


Figure 8.9 Diagram giving the full natural language statement

As in Figure 8.7a where symbols overlapped, here, in Figure 8.9 statements overlap. To reveal the statements hidden behind other statements, select the superimposed statement with the left-hand mouse button, and keeping the button down, drag the statement to the side until the statement below is fully visible.

8.2.4.e Hiding labels

Whilst it is possible to move the labels around, it is obvious from Figure 8.9 that a diagram with all the natural language statements visible can rapidly become impenetrable. Not all the labels may be necessary, the nodes being self-explanatory, or the processes already well known and thus not requiring notation. In order to improve the clarity of the diagram it is possible to turn off / on the labelling for a selected link. Select **Label** from the 'Show/Hide' box on left hand side of the diagram screen then press the left-hand mouse button down on the first node and holding the button down, drag the mouse down to the end node and release. Figure 8.10 demonstrates how judicious use of the **Label** 'switch off' function can render a diagram much clearer.

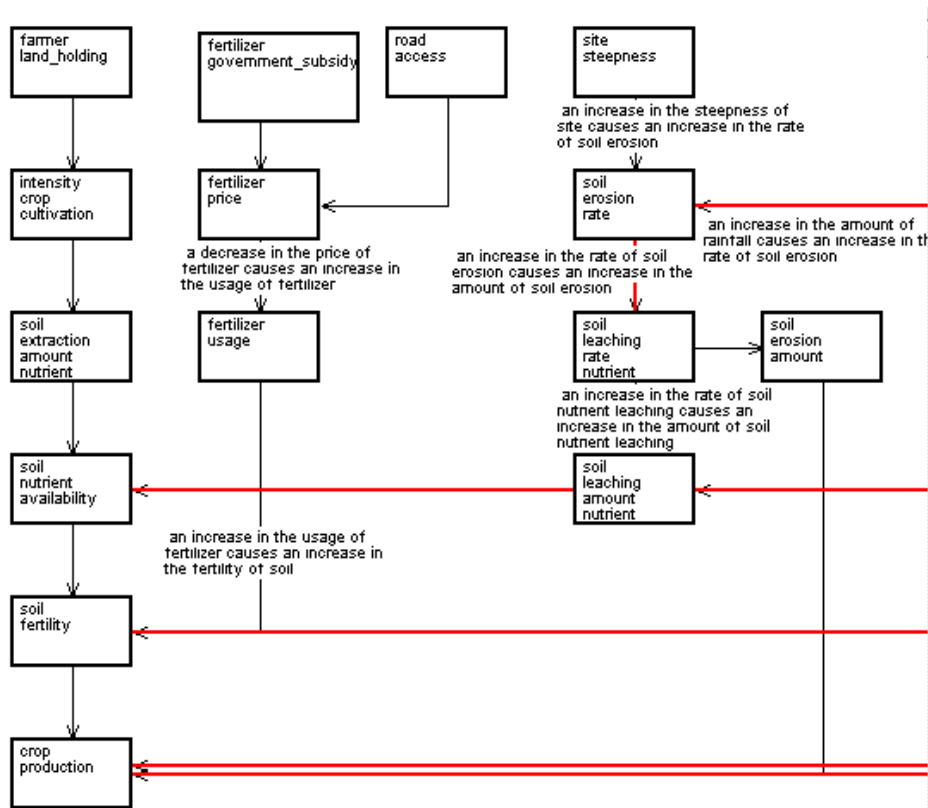
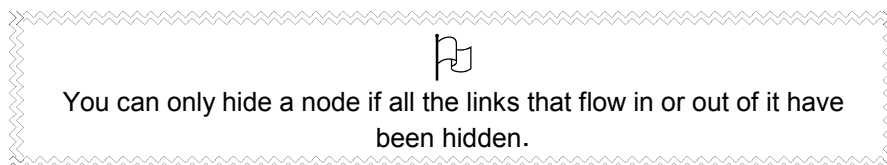


Figure 8.10 A diagram with some of the link labels switched off

8.2.5 HIDING NODES AND LINKS

Sometimes it may be desirable to hide various nodes and links on a diagram, to enhance its clarity, or to remove nodes of only secondary interest. The Hide Node/Link button on the left hand side of the screen will hide links and nodes in a particular diagram without removing them from the knowledge base.

In order to hide a node and link you must hide the link first. Therefore, press on the **Node/Link** button in the 'Hide' box and then, in the diagram window, press the cursor down at one end of the link and release it at the other end of the link, taking care to drag the cursor in the direction of the causal flow. Once the link has been hidden you can then hide the node by pressing the Hide Node/Link button once more and selecting the node.



8.2.6 REFRESHING A DIAGRAM

Occasionally a diagram may become corrupted through other Window dialogs overlapping the diagram. To restore the current diagram, select the **Refresh** button.

8.3 CREATING SUB DIAGRAMS

8.3.1 SHOW PATHS

Even in such a relatively small knowledge base as 'soil', the diagram soon becomes crowded and it is sometimes difficult to see if or how nodes are connected. In the example below it was required to ascertain whether there was any relation between the nodes 'field steepness' and 'crop production'. The two nodes are highlighted (Figure 8.11) by double clicking of the right mouse button (which turns them green) and then select the **Show Paths** button.

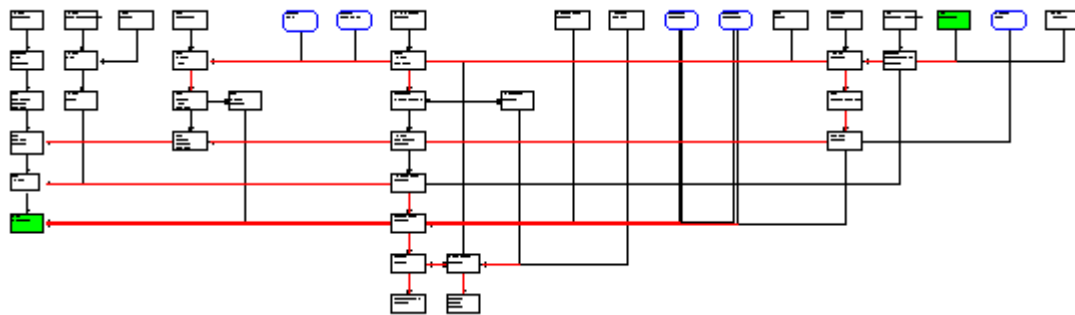


Figure 8.11 The complete diagram of 'soil' knowledge base with nodes 'field steepness' and 'crop production' highlighted.

A new diagram appears showing the pathway between the two nodes (Figure 8.12).

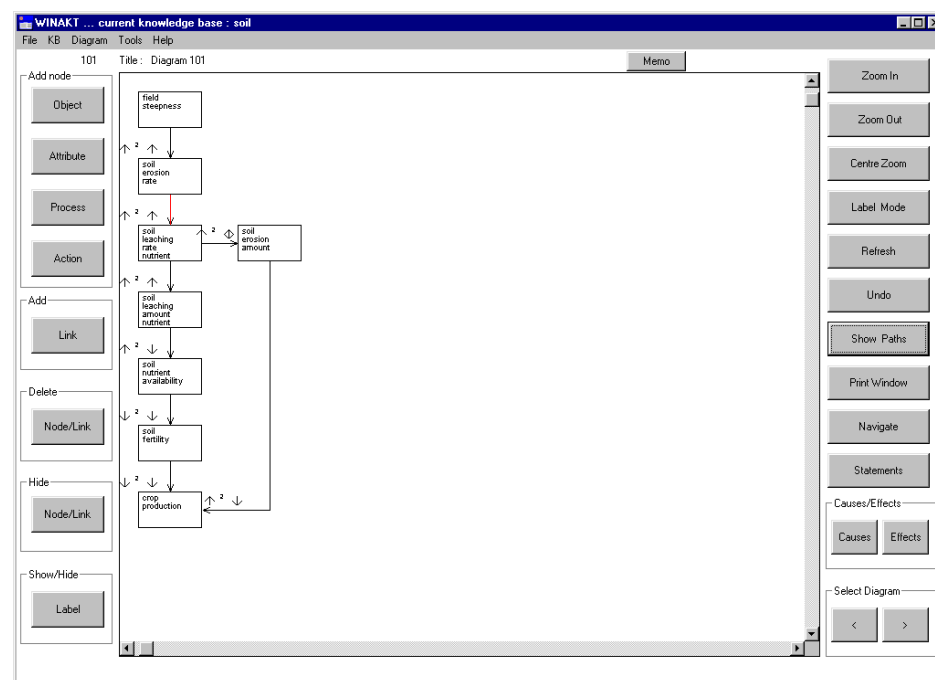


Figure 8.12 New diagram screen showing the pathway between 'field steepness' and 'crop production'

In some cases the new diagram might be too big to fit into the screen without using **Zoom Out** to reduce it. In such circumstances it is possible to rearrange the diagram by selecting a node with the left-hand mouse button, then holding it down, dragging the node across the page to reposition it. Figure 8.13 below is the same diagram as Figure 8.12 above but demonstrates how a diagram might be rearranged.

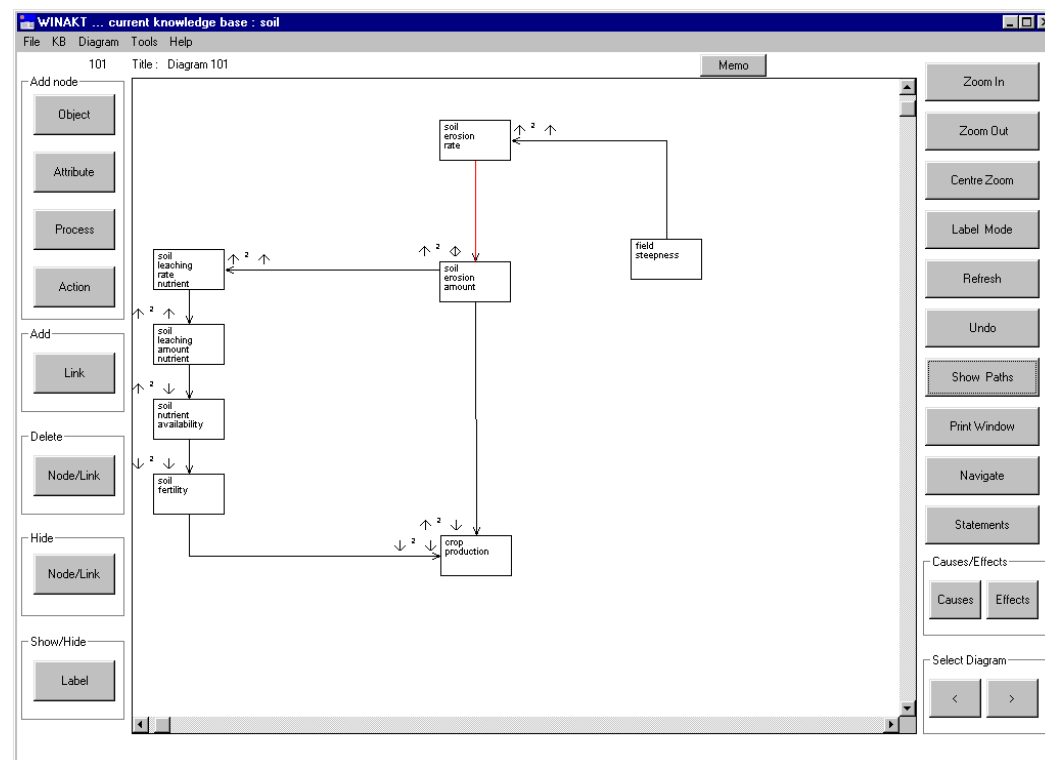


Figure 8.13 *Rearranged diagram of the pathway between 'field steepness' and 'crop production' rearranged to fit onto the screen at full scale*

It is possible to highlight more than two nodes before selecting **Show Paths**. However, the more nodes highlighted the more complicated the subsequent pathways in the sub-diagram become.

8.3.2 NAVIGATING

If a subsection of the main diagram is required for further investigation, it is possible to build one up, level by level, to incorporate those nodes of particular interest, by using the **Navigate** button. When you select **Navigate** and click it over a chosen node, all the *immediate* causes and effects of a particular node appear.

As an example, take the livestock element in the 'soil' knowledge base:

1. In the main diagram highlight two adjacent nodes related to livestock, in this case 'animal fodder amount' and 'animal number' (Figure 8.14)

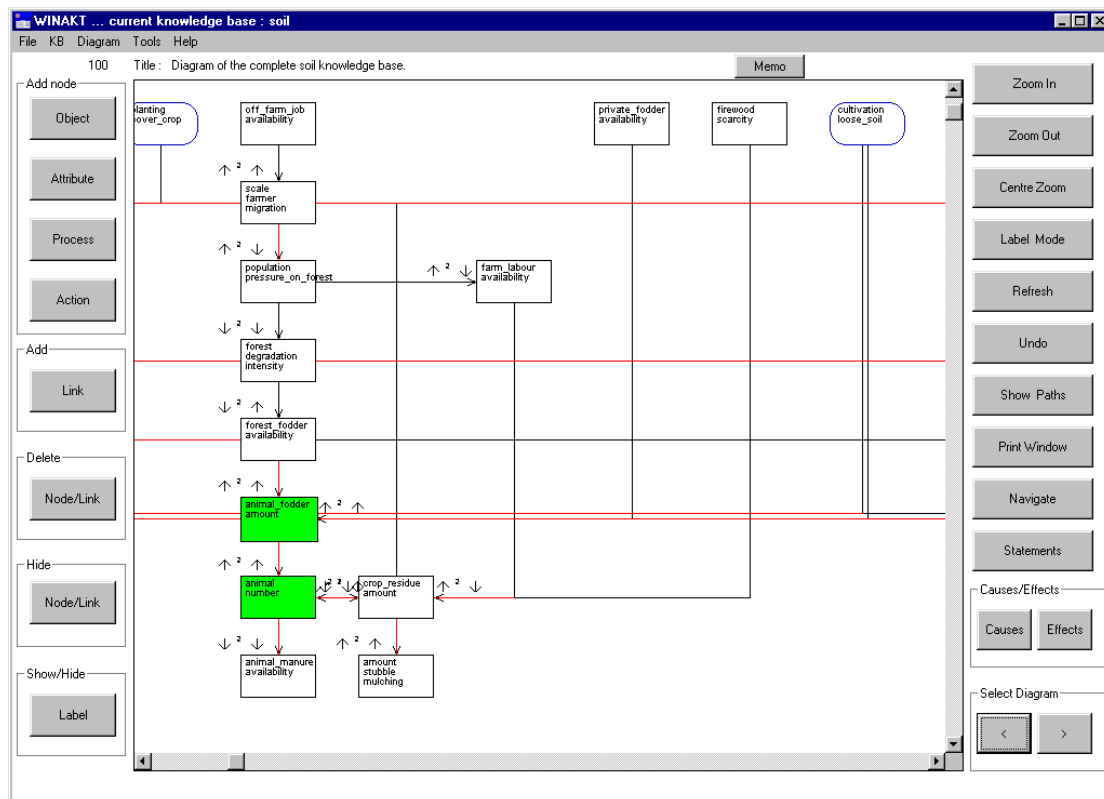


Figure 8.14 Adjacent nodes, 'animal fodder amount' and 'animal number' highlighted in the main 'soil.kb' diagram

2. Press the **Show Paths** button to open a new diagram with just the one statement of interest displayed (Figure 8.15)

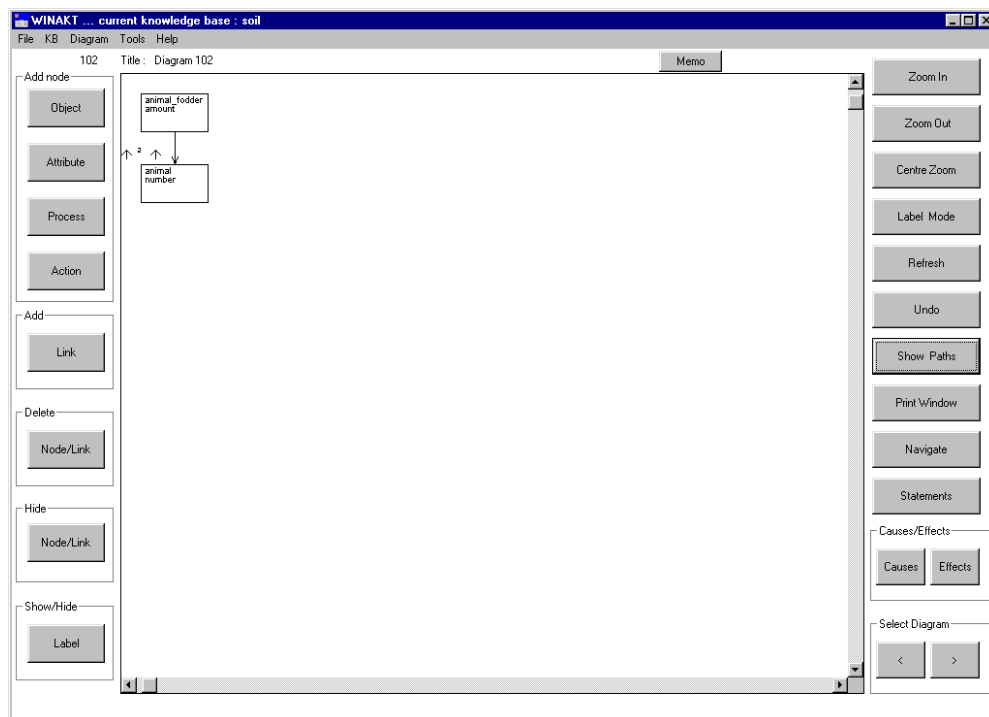


Figure 8.15 Diagram of the statement 'An increase in the amount of animal fodder causes2way increase in the number of animals'

- Press **Navigate**, and a doubled ended arrow appears (↔). Place this over the node of interest (in this case 'animal number') and click with the left-hand mouse button. All the immediate causes and effects of the selected node will then appear (Figure 8.16a). This diagram can then be built up step by step by selecting **Navigate** and then pressing it over each node in turn. Figures 8.16b, 8.16c and 8.16d demonstrate how the diagram was further enlarged, first by clicking the Navigate arrow over 'animal fodder amount' then 'animal manure availability' and finally 'soil fertility'.

Thus, by the repeated use of **Navigate** on different nodes, a diagram can be built up to show a particular aspect of the knowledge base.

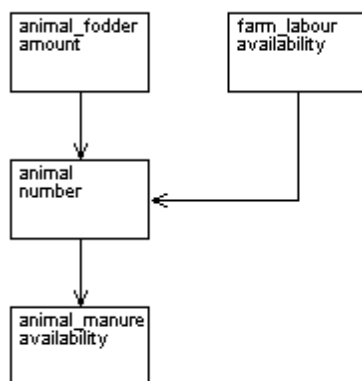


Figure 8.16a Pressing the **Navigate** arrow on 'animal number'

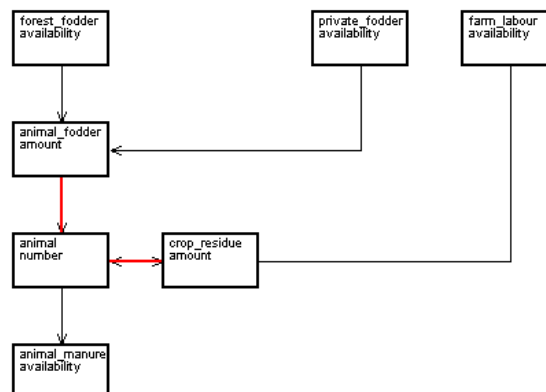


Figure 8.16b Pressing the **Navigate** arrow on 'animal fodder amount'

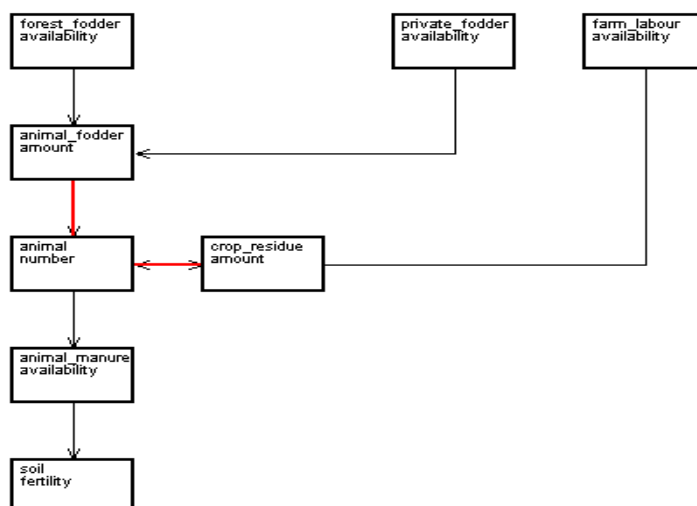


Figure 8.16c Pressing the **Navigate** arrow on 'animal manure availability'

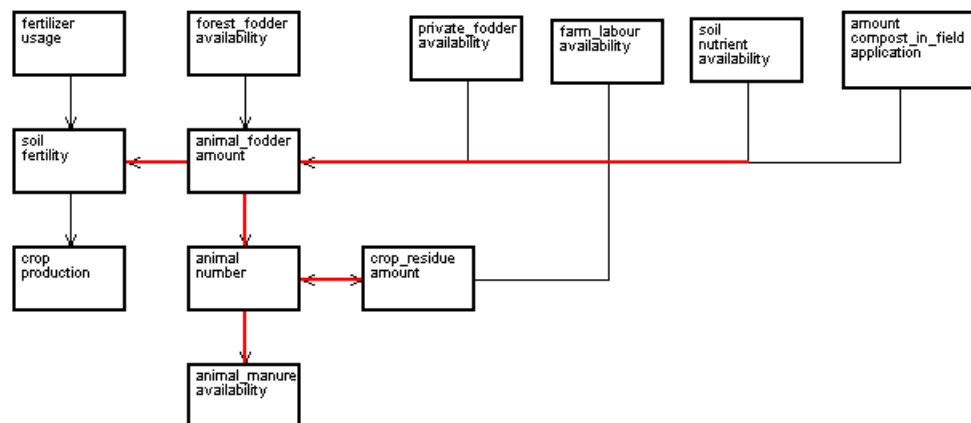



Figure 8.16d Pressing the **Navigate** arrow on 'soil fertility'

8.3.2.a Navigating via the 'Statements' or 'Search results' dialog boxes

It is also possible to build up a diagram using the **Navigate** facility via the 'Statements' and 'Search Results' dialog boxes. Enter **Statements** via the main **KB** menu and then highlight the statement around which you wish to build up the diagram and press the **Navigate** button at the bottom of the screen. This will generate a diagram giving the immediate causes and effects of that statement, which you can then build upon as described above. The principle is exactly the same for using the **Navigate** option in the 'Search Results' dialog box of the Boolean Search (see Chapter 7, section 7.10.1 for an example).

8.3.2.b Making a mistake when navigating

When you are building up a sub-diagram, it may be that you inadvertently click the wrong node, and get links you did not want in the sub-diagram. The **Undo** button on the right side of the diagram window allows you to cancel the last operation carried out in the current diagram.



WARNING

It is important to remember that you cannot simply delete a node from a sub-diagram by pressing **Delete**, because that would remove the node, not only from the sub-diagram, but from the earlier diagram and also from the underlying knowledge base.

8.3.3 CAUSES / EFFECTS DIAGRAMS

The **Causes** and **Effects** buttons on the right hand side of the diagram window allows you to create a sub-diagram showing all the causes of a particular node and all the effects of a particular node in two strokes. Select two adjacent, linked nodes in the main diagram, highlight them and press **Show Paths**. (Figure 8.17 uses the same statement as Figure 8.15, 'An increase in the amount of animal fodder causes2way increase in the number of animals'). Press **Causes** and then place the cursor (which takes the form of an upward arrow) over the first node (animal fodder amount) and click. This gives *all* the elements that *affect* animal fodder amount (Figure 8.17a). Then press **Effects** and place the cursor (which takes the form of a diagonal arrow) over the second node (animal number) and click. This will add to the diagram *all* the elements that are *affected* by animal numbers (Figure 8.17b).

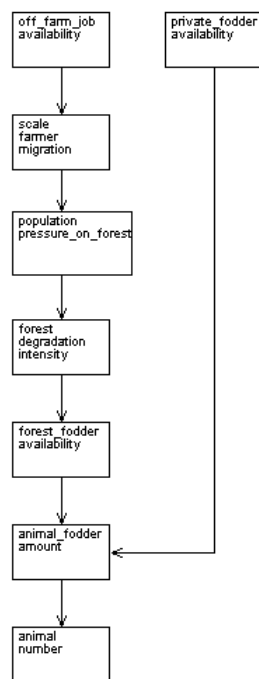


Figure 8.17a Using the Causes Button – all the nodes that affect ‘animal_fodder amount’

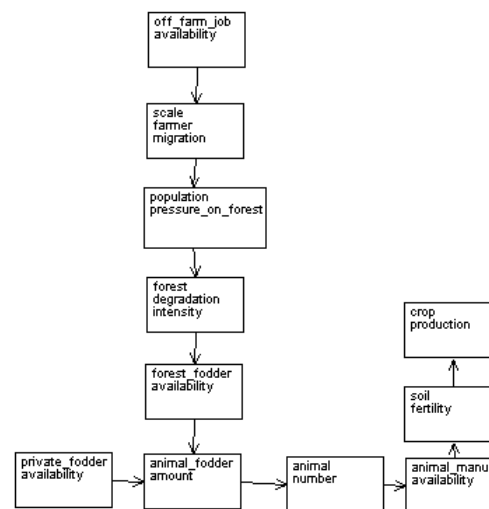
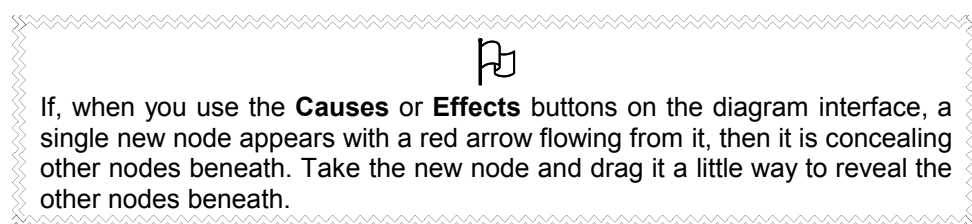


Figure 8.17b Using the Effects Button – Showing Figure 8.17a plus all the nodes affected by ‘animal number’

8.3.3.a Causes / Effects diagrams via the ‘Statements’ or ‘Search results’ dialog boxes

As with **Navigate**, there is also a **Causes/Effects** button at the bottom of the ‘Statements’ dialog box and the ‘Search Results’ dialog box generated by the Boolean Search. See above, 7.10.1.c.



8.4 VIEWING DIAGRAM STATEMENTS

8.4.1 STATEMENTS FOR THE COMPLETE DIAGRAM

To view all the statements represented by any diagram, the **Statements** button on the bottom right-hand side of the diagram screen is selected.

In Figure 8.18 the statements represented by diagram in Figure 8.2 are all listed and it is possible to access the details of each statement by highlighting the statement and then selecting **Details**. At the very top of the pop-up menu is stated how many of the formal statements have been drawn in the diagram. In this case of a complete diagram, 48 out of 73 statements have been used. This means that the remaining 25 statements could not be drawn because they were neither causal nor link statements.

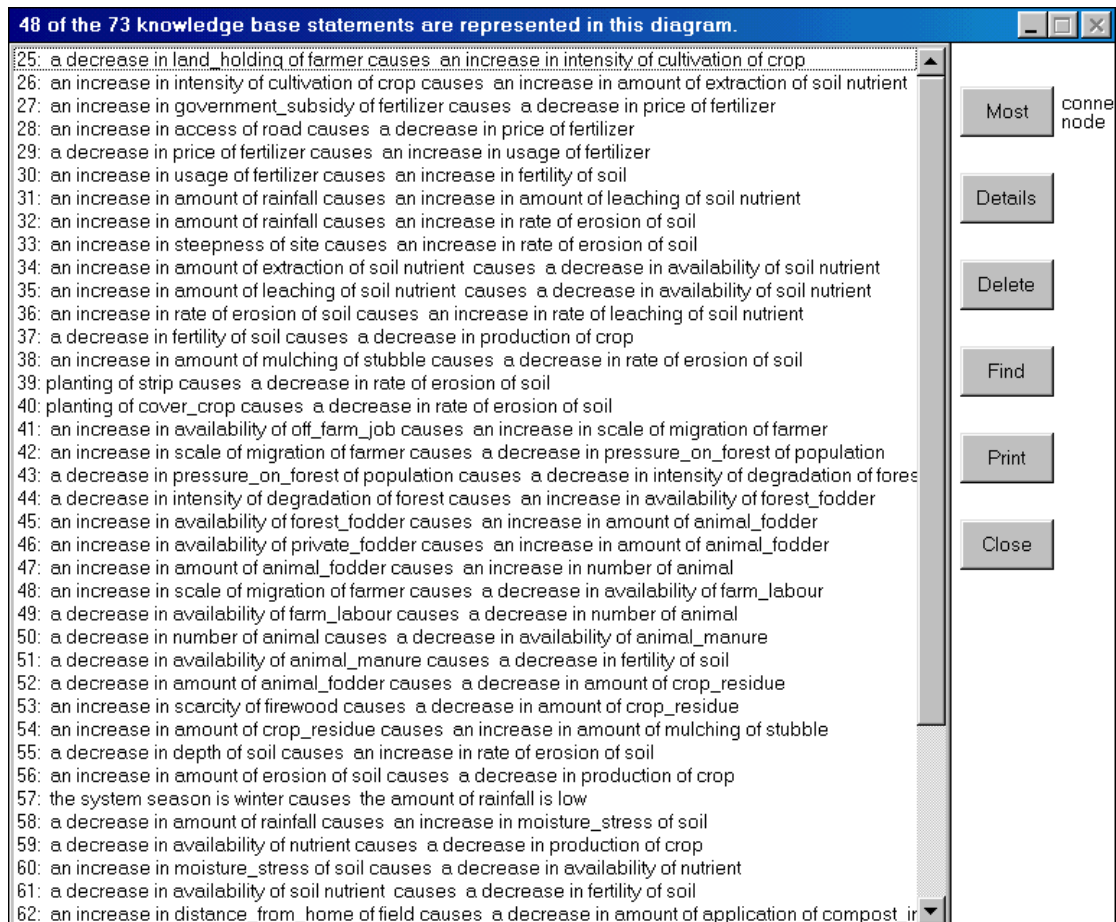


Figure 8.18 *The diagram statements*

This statement card offers the following options:

Most connected node: If you press this button, a diagram will appear showing the most connected node with all its connections. In soil.kb the most connected node is 'soil erosion rate'.

Details: If you highlight a statement and press Details, it will bring you to the Statement Details dialog box, from where it is possible to access the statement formal terms, the statements sources and statement memo.

Delete: It is also possible to delete a statement, by highlighting the statement and then pressing **Delete**. However, **BE WARNED!** If you delete a statement from this dialog box you not only remove the statement from the diagram interface, you also remove it from the knowledge base.


Find: If you want to find the position of a statement within a diagram, highlight the statement and press Find. The diagram will then return to the screen with the two nodes of the selected statement highlighted.

Print: If you want to print out a list of statements, press Print. You can then either save the output to a text file or print it directly from the screen.

Close: If you press Close you will return to the diagram screen.

8.4.2 STATEMENTS FOR A SUB-DIAGRAM

When a sub-diagram has been created, selecting **Statements** will give you all the statements used in creating that sub-diagram. Figure 8.19 is a list of the statements making up the sub-diagram in Figure 8.16d.

 If you use the 'Hide' function for several nodes and links within the sub-diagram, the nodes and links which you have hidden will not appear in the statement list of the sub-diagram.

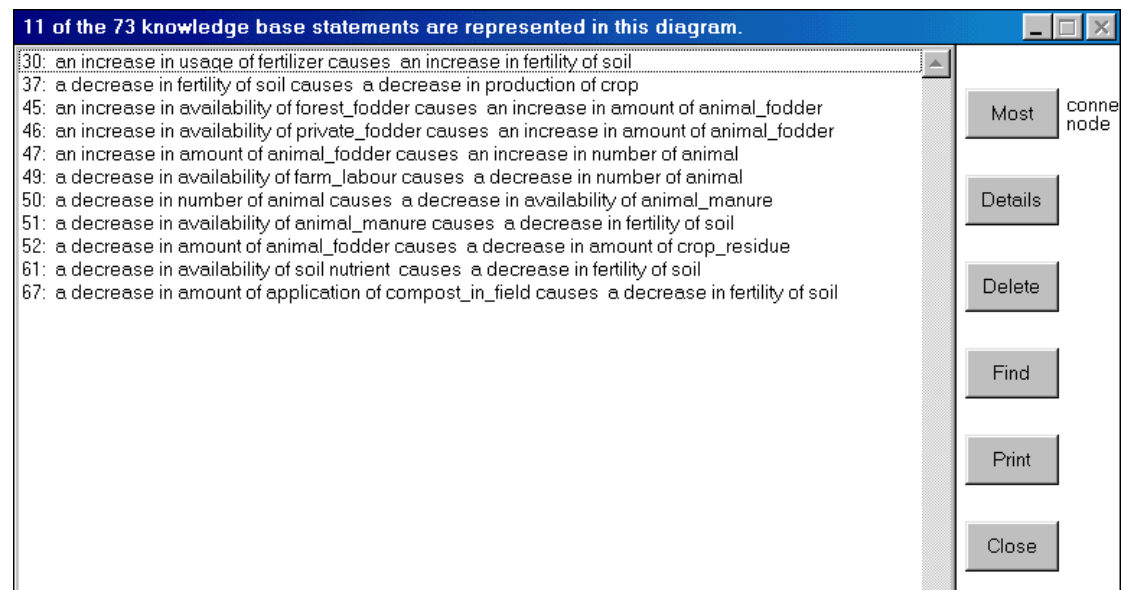




Figure 8.19 The statements that are incorporated into the diagram in Figure 8.16d

8.5 MOVING BETWEEN DIAGRAMS

In order to move between diagrams there are two arrows '<' and '>' in the 'Select Diag.' box in the bottom right-hand corner of the screen which allow you to move backwards and forwards between diagrams. The diagrams are numbered chronologically. If a sub-diagram is deleted, that diagram number is deleted as well.

 **DIAGRAM SIZE** 
It is worth bearing in mind: diagrams with more than 40 statements have limited use as a means of trying to illustrate the interconnectivity of the statements.

8.6 SAVING DIAGRAMS

Any diagrams that you have created will be automatically saved with the knowledge base whenever you select **Save KB** or **Save KB As**. However, if you do not select **Save KB**, but merely press **Close KB**, all the diagrams you have generated will be lost.

8.6.1. LABELLING DIAGRAMS

All diagrams are numbered. However, as the number of diagrams generated increases, it is more useful to give each diagram a title. In order to do so, select **Show KB Diagrams**. A list of the numbered diagrams will appear (Figure 8.20).

Highlight the number of the diagram you wish to label and then enter the title into the 'Title' box. Press **Save Title**. To save the title permanently, press **Save KB**.

It is easier to give a diagram a title when you have the diagram in front of you. When you have completed a diagram and wish to give it a title, keep the diagram open and select **Show KB Diagrams**. This way you can make sure that the number in the top right hand corner of your diagram corresponds to the number that you have highlighted on the list of diagrams.

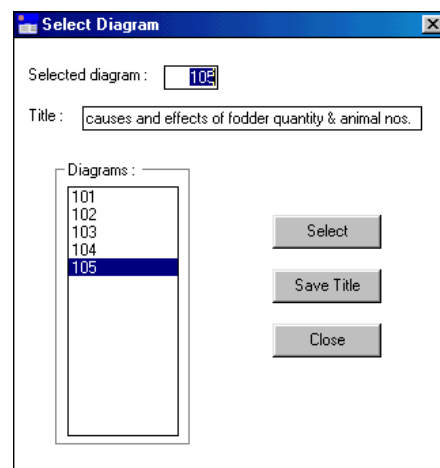


Figure 8.20 Saving titles for the diagrams

If the diagram titles has been saved permanently, when you open the knowledge base again and select **Show KB Diagrams**, each title will appear in turn in the 'Title' box as you run down the diagram list, highlighting the diagram numbers one after the other.

8.6.2. MEMO FIELD FOR EACH DIAGRAM

For those diagrams which you wish to save it may be worth recording what the diagram shows, how it was constructed (e.g. by Navigate or by Causes and/or Effects), and what is of interest or importance in the diagram. For this reason each diagram has a separate **Memo** button attached in the top right hand side of the diagram screen. This memo field functions exactly as the memo fields attached to the sources and the individual statements. When you press **Memo**, a dialog box appears with a blank memo field in which you can type all necessary details about the diagram. Once you have filled in the memo field, press **Save**, to save the information. **Delete** will delete the memo field and **Close** will close the memo field.

Remember: You only select **Close** if you were simply looking at the memo field. If you wrote or edited a memo, you must always select **Save**.

8.6.3. COPYING DIAGRAMS

If you wish to copy a diagram, open the diagram and then select **Copy diagram** from the main diagram menu. The copy will immediately appear on the screen. If you then select **Show KB diagrams** from the main **Diagram** menu, you will see that the copy appears as the last diagram in the list and has the title 'Copy of diagram...' and the number of the diagram that was copied.

If you wish to copy a diagram to a different package, for example to be incorporated into a word processing document, or to be used in a presentation, press the **Copy to Clipboard** button at the top right-hand side of the diagram. A message will appear 'Copy of diagram sent to clipboard'. You must then open the other package and paste the diagram into to document/presentation immediately, by selecting **Edit** and then **Paste** (or **Ctrl v**). (If you do not paste it immediately, the diagram will be overwritten by the next item that you copy).

8.6.4. SAVING DIAGRAMS AS A SEPARATE KNOWLEDGE BASE

If you wish to save a particular view and subset of a knowledge base as shown in your diagram as a separate knowledge base, then select **Save Diagram as New KB**. This will only save the statements and formal terms shown in the diagram, but will retain information about all the sources and object hierarchies in the original knowledge base.

When creating a new knowledge base in this manner, it is wise to go through it carefully and weed out any unnecessary object formal terms, hierarchies and sources before starting work on it as a new knowledge base.

8.7 DELETING DIAGRAMS

If you wish to delete a sub-diagram that you have created, press **Delete Diagram** in the main **Diagram** menu. If you wish to delete all the diagrams you have created press **Delete all Diagrams** in the main **Diagram** menu. If you do this, when you next select the main **Diagram** menu and then **Show KB Diagrams**, the only diagram to appear will be the diagram for the whole knowledge base generated afresh.



By selecting **Delete Diagram** or **Delete All Diagrams** you are **NOT** deleting the statements from the knowledge base. Only if you delete an individual node/link from the diagram do you delete the equivalent statement from the knowledge base.

8.8 CREATING NEW DIAGRAMS

Before creating a new diagram it is necessary to have loaded a knowledge base. Therefore select **KB** from the opening menu and create a new knowledge base by selecting **New KB** and giving the new knowledge base a name.

Selecting **Show KB Diagrams** from the main **Diagram** menu (Figure 8.1), will give you a blank diagram screen (Figure 8.21) because there is not yet any information in the knowledge base.

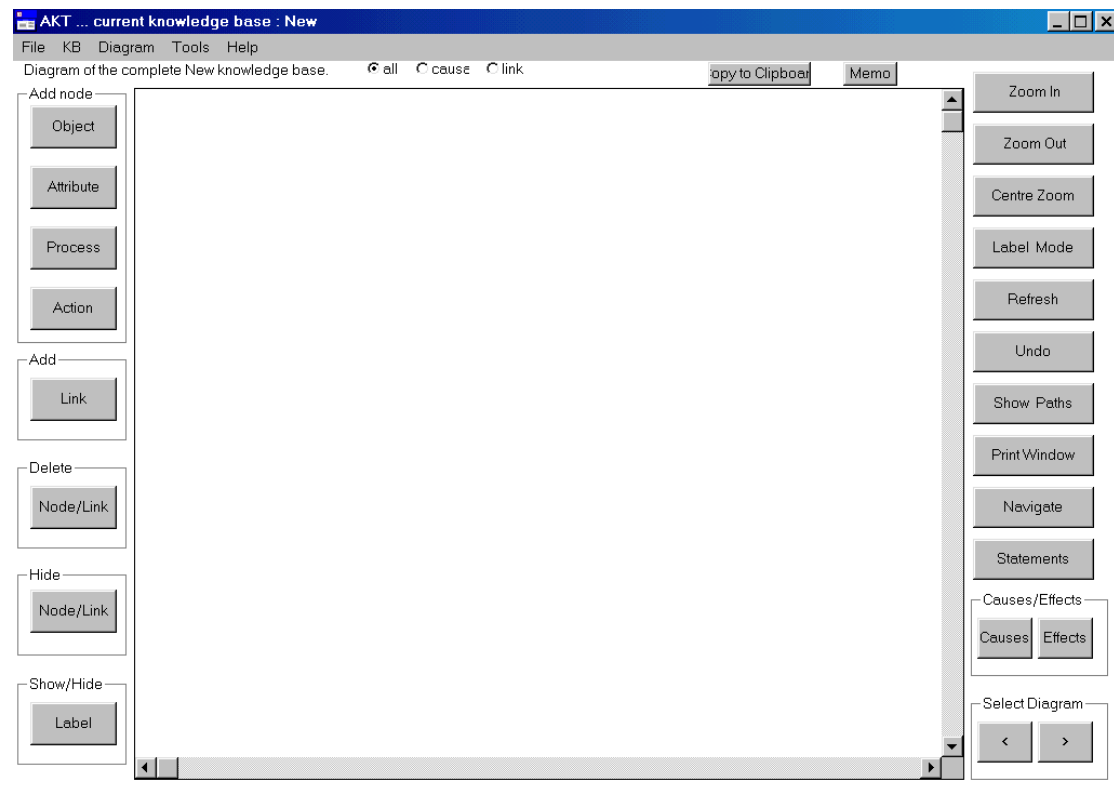


Figure 8.21 The empty 'Diagram Screen' visible when setting up a new knowledge base

8.8.1 CREATING A DIAGRAM

The method of entering a unitary statement through the diagram interface is best illustrated by example. Consider the statement: '....deforestation results in a decrease in the rate of evapotranspiration....'. 'Deforestation' is a process, 'results in' is the link, and 'decrease' is a value of the attribute 'rate' which describes the process 'evapotranspiration'. This statement can be represented on the diagram by two nodes joined by a link. As 'deforestation' is a process, the **Process** button from the 'add node' box is selected. Click the cursor on the drawing area and a 'process_dialog' box appears (Figure 8.21a).

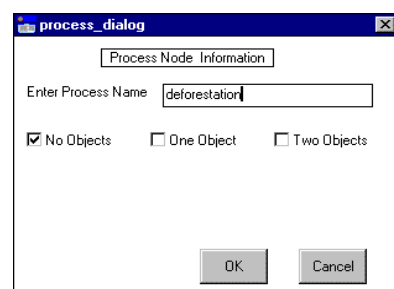


Figure 8.21a The 'process_dialog' box

The 'process_dialog' box requires the process name and whether or not it is attached to one or two objects². In this case there are no objects. When the box has been completed, **OK** is

² In this example 'deforestation' is without any accompanying objects. 'Trampling by sheep' is an example of a process attached to an object, 'stabilization of bunds by salt grass' is an example of a process attached to two objects.

selected. On the diagram area a node appears with a green outline bearing the legend 'deforestation'.

The second half of the statement, 'a decrease in the rate of evapotranspiration' is an attribute value. Therefore the **Attribute** button is selected from the 'add node' box, the cursor is placed in the appropriate position in the diagram area and the mouse clicked. The 'attribute_relation_dialog' box appears (Figure 8.21b). In this example, the attribute relates to the process 'evapotranspiration', so the 'Process' box is ticked.

As soon as **OK** is selected in the 'attribute_relation_dialog', the 'attribute_process_dialog' box (Figure 8.21c) will appear.

In this example the process name is 'evapotranspiration' and the attribute name 'rate'. There are no objects in the node.

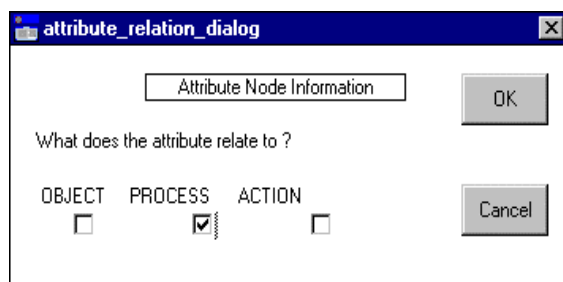


Figure 8.21b 'attribute_relation_dialog' box

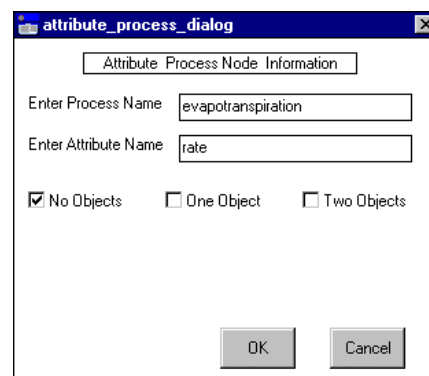


Figure 8.21c 'attribute_process_dialog' box

Once the **OK** button has been selected the main diagram screen will show the following diagram, the process node appearing as a oval outlined in green bearing the legend 'deforestation' and the attribute node appearing as a rectangular box with a black outline and with the legend 'evapotranspiration rate' (Figure 8.22).

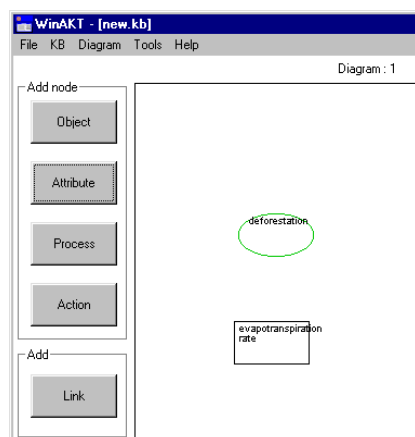


Figure 8.22 Partial view of the main diagram screen with the two nodes.

In order to link the two nodes, the **Link** button, on the left-hand side of the diagram screen, is selected (Figure 8.22), then the left-hand mouse button is pressed down on the 'deforestation' node and released on the 'evapotranspiration rate' node.

When the two nodes have been connected in this way, the 'Choose Link' dialog appears (Figure 8.23). In this case the 'causes1way' is the correct selection. In the 'Effect Value' box, 'decrease' should be typed.

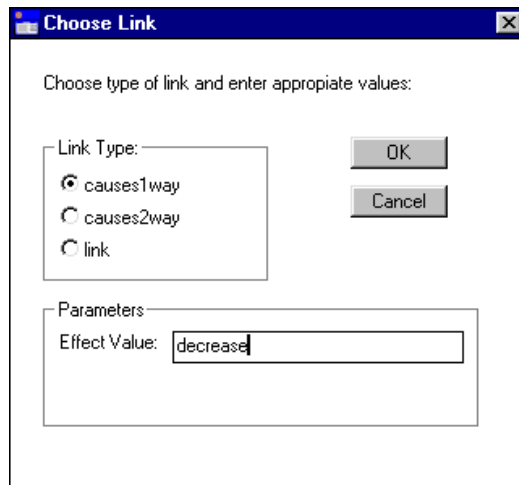


Figure 8.23a 'Choose Link' dialog box

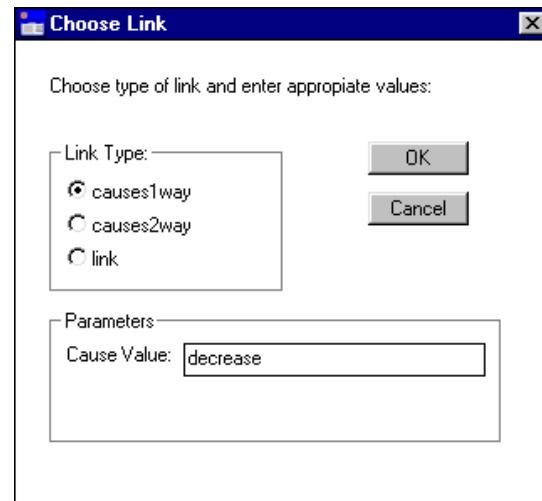


Figure 8.23b 'Choose Link' dialog box, if the cursor is dragged in the wrong direction

It is important to drag the cursor in the right direction between the two nodes, i.e. *in the direction of the causal flow*. If you drag it in the wrong direction, i.e. from 'evapotranspiration rate' to 'deforestation' a 'Choose Link' dialog box as in Figure 8.23b will appear with 'Cause Value' instead of 'Effect Value' in the 'Parameters' box. If it is completed in exactly the same manner as in Figure 8.23a, it will give the statement 'a decrease in rate of evapotranspiration will cause deforestation' which is clearly nonsense.

As soon as **OK** has been selected from the 'Choose Link' dialogue box (Figure 8.23a), the 'Information Source' dialog box appears (Figure 8.24). This is identical to the 'Information Source' dialog box that appears before entering a unitary statement via the main **KB** menu (See Chapter 7.4.2) and must be completed in exactly the same way if the statement is to be accepted.

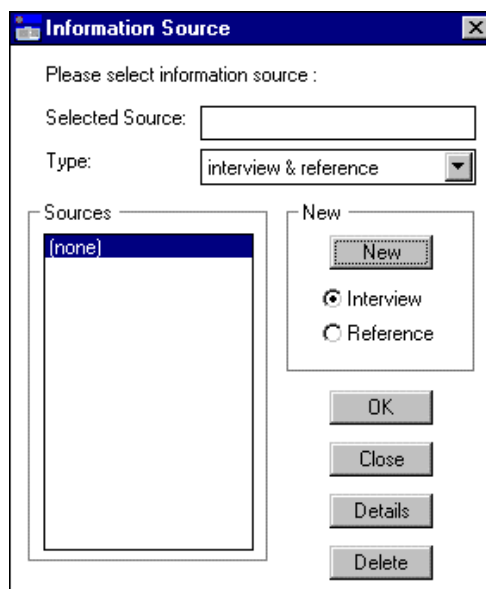


Figure 8.24 'Information Source' dialog box

Once the 'Information Source' dialog box has been completed as in the manner described in Chapter 7.4.2, the 'New Statement' dialog box immediately appears (Figure 8.25) giving both the formal language representation and the natural language representation of the nodes and link created in the diagram interface.

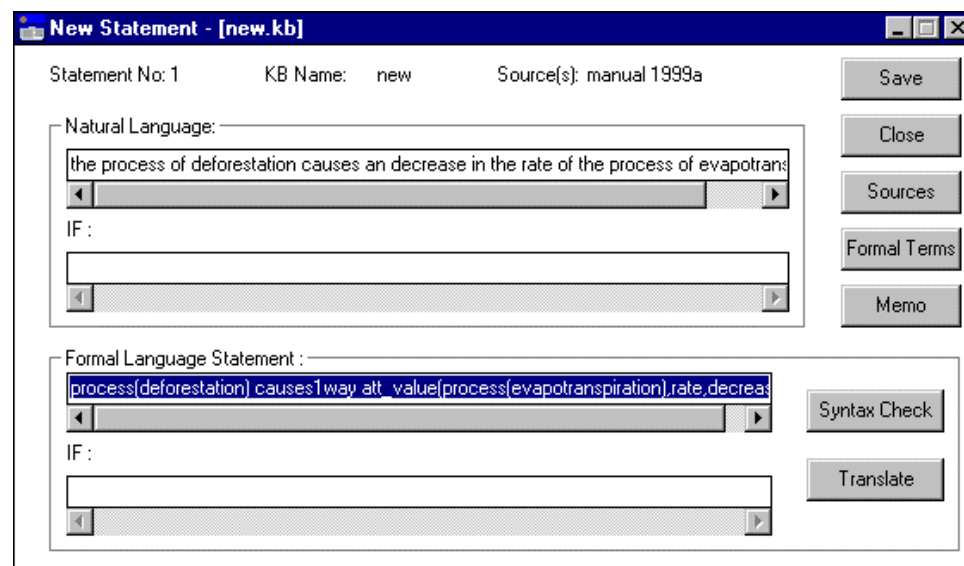
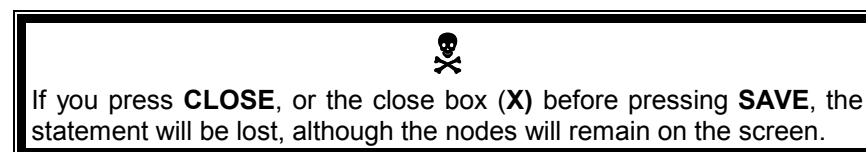


Figure 8.25 'New Statement' dialog box

From the 'New Statement' dialog box it is possible to edit the Formal Language Statement, for example, changing 'causes1way' to 'causes2way', or even changing the various formal terms. The source can also be modified by selecting **Sources** and a memo can be attached to the statement by selecting **Memo**³. It is also in this dialog box that any conditions attached to the statement may be added

In order to save the statement press **SAVE**.



If no conditions were added in the 'New Statement' dialog box, once **SAVE** has been pressed, the following message then appears (Figure 8.26), asking whether the statement should be saved without a condition. (In principle, unitary statements are generally more useful when qualified by conditions.) It is not possible to enter conditions diagrammatically, therefore any conditions have to be entered in the formal language statement 'IF' box.

In this simple example there are no conditions, so **No** is selected.

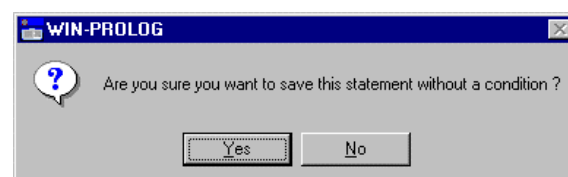


Figure 8.26 Message that appears when saving a statement without any conditions

Then a message will appear for each new formal term you have created, e.g. – 'Do you wish to create a new formal terms 'deforestation''. Press **Yes**.

The screen then reverts to the Diagram Interface and displays the two nodes connected by a link (Figure 8.27).

³ Although it is possible to select the **Formal Terms** button it is not possible to modify anything as the 'Statement Formal Terms' box will be empty, because the statement has as yet not been saved and the formal terms thus not yet entered into the knowledge base.

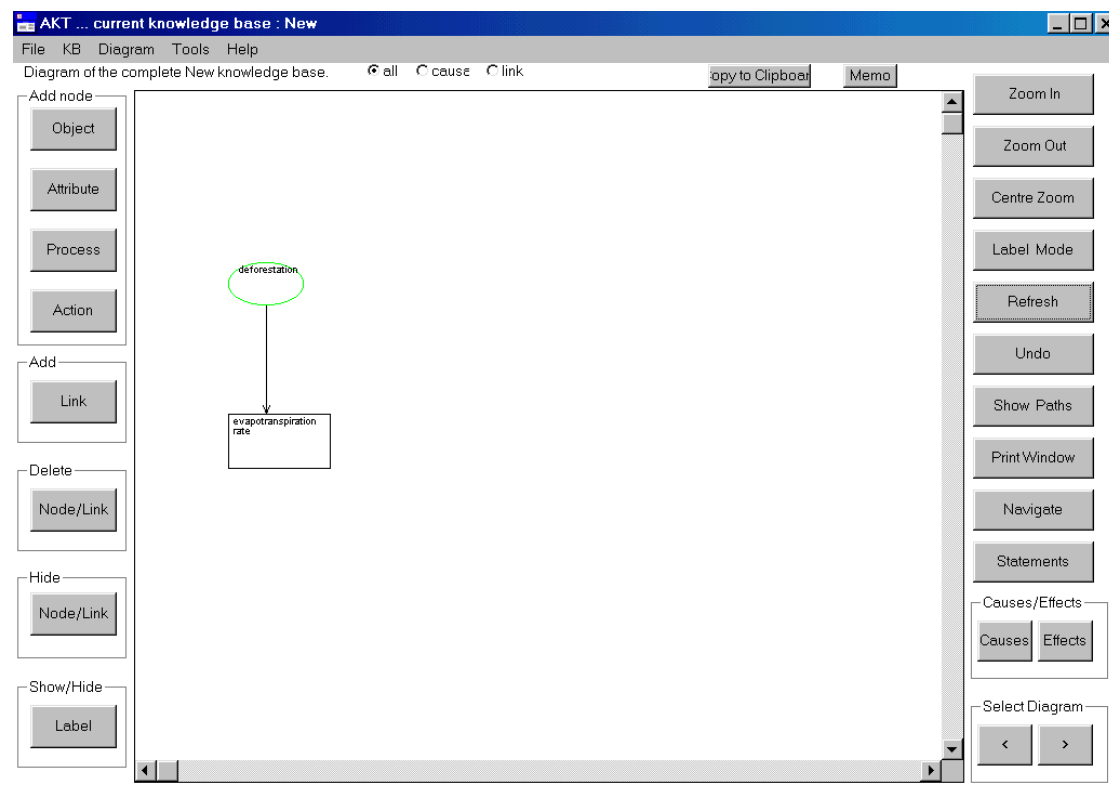
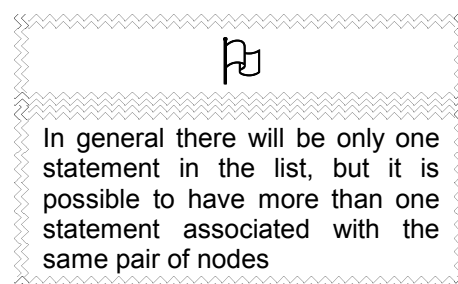
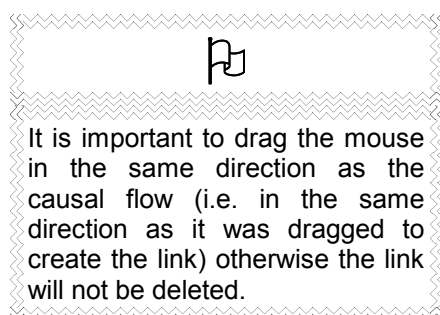


Figure 8.27 A basic link and node diagram on the screen

8.9 EDITING/DELETING A DIAGRAM STATEMENT

In order to delete a statement represented by two nodes and a link, you must delete the link first. Select the **Delete** button from the diagram screen and then press the left hand button of the mouse down on the first node and release it on the second node.



A dialog box will then appear in which the sentence to be deleted must be selected. Once the sentence is highlighted, press **Delete**.

Once **Delete** has been selected, the following message appears (Figure 8.28), giving the natural language version of the statement and checking whether it is to be deleted.

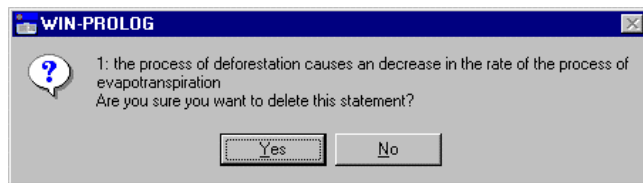


Figure 8.28 Message requesting you to confirm that the statement should be deleted

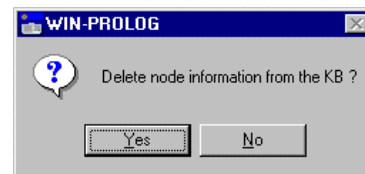



Figure 8.29 Message asking whether the node information should be deleted from the knowledge base as well

If **No** is selected the whole process is abandoned. If **Yes** is selected, a message as in Figure 8.29 appears, checking whether the nodes are to be deleted as well.

If **Yes** is selected, both the nodes will disappear, as well as the link, if **No** is selected, the nodes will remain, unattached, as in Figure 8.30. It would then be possible to create a different link, either a 'causes2way' or a simple 'link' link, or to add new nodes and rearrange the links between them.



Figure 8.30 Unattached nodes


 Any changes made via the diagram interface will result in the same changes being made to the statement interface and vice versa.

8.10 ALTERNATIVE ROUTE TO EDITING DIAGRAMS

An alternative way of altering a diagram is to select **Statements** from the main **KB** menu. From the 'Statements' dialog box it is possible to edit the statement in question (See Chapter 7.4.3). Once the statement has been edited and saved, closing the 'Statements' dialog box will return you to the Diagram screen, with the diagram updated where appropriate.

PRINTING DIAGRAMS

If you wish to print a diagram directly from the screen, press the **Print Window** button on the right hand side of the screen.

If you wish to export the diagram to another package before printing, do **Copy to Clipboard** as explained above in **8.6.3**.

CHAPTER NINE - REASONING WITH AKT TOOLS

9.1 INTRODUCTION

An important part of the AKT application is the task language. This consists of a number of primitive or system tools together with a choice of control structures in which the tools can be combined to perform tasks defined by the user. The resulting definition can then be saved as a user-defined tool for future use. The tools defined in this way can also be incorporated into more complicated tools so allowing complex tasks to be completed automatically.



These task definitions are similar to the idea of macros used in other computer applications

9.1.1 WHAT IS A TOOL?

A tool is created by writing a 'definition'. A definition is a set of 'functions'¹ linked with appropriate punctuation and control structures and entered into the definition field.

A function calls the underlying program code to perform a specified task. Some functions are part of inference mechanisms that undertake automated reasoning tasks on the knowledge base, others are concerned with the management of inputs and outputs (interfacing) to and from inference mechanisms. Functions are defined by a name one or more arguments (variables) associated with it.

9.1.2 WHAT DO THE TOOLS DO?

The task language allows the user to interrogate the knowledge base and carry out assessments of the knowledge it contains. These can be simple queries such as;

How many conditional statements are there?
How many statements are not represented in the diagram?
What comparison statements are there?

or a more sophisticated analysis of the knowledge base, for example;

What are the immediate causes of soil erosion?
What are all the effects of shading?
Using hierarchical information, which livestock are suitable for small landholdings?

9.1.2.a What is a primitive?

Primitives are small program segments, pieces of generic code, from which a tool can be constructed. They are classified into a number of categories, Cause and Effect, Display, Formal Term, Hierarchies, List, Miscellaneous, Sources, Statement and Test. These primitives can carry out searches on the knowledge base, handle lists and manipulate the formal terms and statements.

¹ A function is a generic term that could represent either a primitive, system tool or a user defined tool

9.1.2.b What is a control structure?

Control structures define special relationships between a set of functions, thereby allowing different actions to be taken depending on the circumstances and allowing repetitive tasks to be specified efficiently. Control structures allow primitives to be ‘threaded’ together in ‘if_then_else’; ‘repeat_until’ or ‘foreach_in_do’ type of structures to create tools, which can carry out more complex tasks that a single primitive might achieve.

9.1.2.c What is a tool?

A tool is a collection of primitives together with control information defining the sequence in which they should be carried out. They are classified as;

- a) **Systems tools** - tools supplied with the AKT application that have been constructed using primitives and control structures to perform some generally useful function which is far too complex for a simple primitive.
- b) **User tools** – tools created by you, the user, using the primitives, systems tools and control structures provided, to carry out tasks not previously envisaged by the programmers. Once defined, user tools themselves can be used as primitives to create larger tools.

This chapter looks at tools and the mechanics behind the primitives and systems tools. Chapter 10 looks at user tools and teaches you how to create your own.

9.2 WORKING WITH TOOLS

To display the tools provided, select **Tools** from the main **Tools** menu. Then, from the drop down menu, select the type of tool you want:

- Control structures
- Primitives
- System tools
- User tools

9.2.1 PRIMITIVES

9.2.1.a Opening primitives

In order to work with primitives, select **primitives** from the drop-down menu. When **primitives** has been selected another drop-down menu appears, listing the types of primitives available (Figure 9.1).

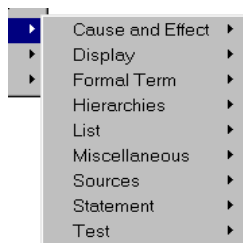


Figure 9.1 Drop-down menu for primitives

9.2.1.b Different categories of primitives

The following is an explanation of the different categories of primitive available:

Cause and Effect	Functions for the extraction of information about all types of nodes and the pathways between them.
Display	Functions that allow the user to display the results of running a tool in a text window in various ways. Also allows tools to display messages and request input from the user.
Formal Term	Functions for the manipulation of formal terms.
Hierarchies	Functions for extracting information about objects within object hierarchies.
List	Provide a variety of functions for operating on lists such as sorting, appending and combining lists.
Miscellaneous	These are assorted primitives that do not fall into any of the above categories.
Sources	Function for examining the details of any sources in the knowledge base.
Statement	Functions that operate on statements. Allows searching and manipulation of any statements which exist within the AKT environment.
Test	Functions that allow true/false checks to be made and a way of aborting the running of a tool

Generally primitives are not run in isolation but are used within tools. Some primitives however can be run independently and give useful results.

All primitives and system tools have individual descriptions of their functions and parameters together with an example of their use available via the **Details** button on the category dialog box. A full list of the tools, including the same details as above, are also available via the Help/ToolList menu. This list incorporates all primitives system and any user tools.

9.2.1.c Understanding the Details box

In order to demonstrate the 'Tool Details' dialog box, we have selected a primitive, ***formal_term(KB, Term, Type, Result)*** from the **Formal Term** category of primitives. Click on this tool, and the tool details will automatically appear (*Figure 9.2*)

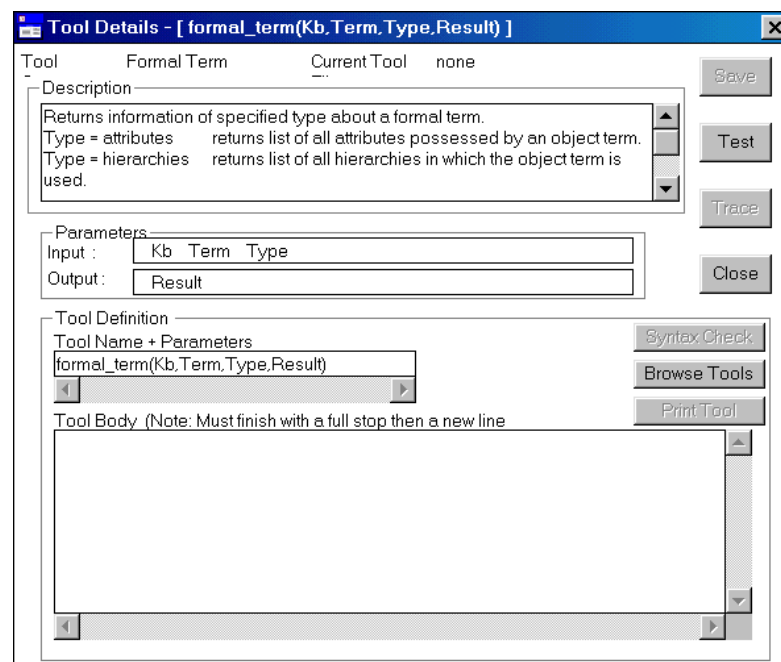


Figure 9.2 'Tool Details' dialog box of the primitive **formal_term(Kb, Term, Type, Result)**

The dialog box title shows the tool category, in this case 'Formal Term', and the current tool file if a user tool file is loaded (in this case there isn't).

Taking the output dialog box apart bit by bit, you have:

Description

Here the function of the tool is described together with a relevant example. Below is the information provided in the Description box for the tool **formal_term(Kb,Term,Type, Result)**:

Returns information of specified type about a formal term.
 Type = attributes returns list of all attributes possessed by an object term.
 Type = hierarchies returns list of all hierarchies in which the object term is used.
 Type = synonyms returns list of all the synonyms used for a formal term.
 Type = type returns the type of a formal term.
 Type = use returns 'used' or 'unused' depending whether term is used by a statement.
 Type = values returns list of all the values assigned to an attribute formal term.
 eg: formal_term(treefodd,erosion,type,Type) will return **Type=process**
 formal_term(treefodd,area,values,Values) will return **Values=[decrease,increase]**.

Parameters

This shows the parameters that are required as inputs to the tool and the parameters that the tool uses to return information. In the case of the tool **formal_term(KB, Term, Type, Result)** the parameters required as inputs are:-

- the name of the knowledge base
- the term (the object) to be investigated
- the type of output sought (i.e. attributes or hierarchies)

The output parameter will be:

- the result (i.e. the attributes or hierarchies sought)

In the 'Tool Definition' box the name of the tool and its parameters (in brackets) are shown. The name of the tool must always begin with a **lower case letter** or be enclosed in single quotes eg: 'JohnsUserTool' if it is necessary to use a capital letter for a real name. The name of each parameter listed after the tool name must always begin with a **capital letter** to denote a *variable*. If the parameter is an *atom*, the first letter will either be in lower case (e.g. *soil*) or it will be enclosed in single quotes for proper names (e.g. 'Nepal').

The 'Tool Body' box in this instance remains blank, because the tool `formal_term` is a primitive and cannot be broken down into smaller parts. The systems tools and user tools will have details of the body of the tools showing the chain of primitives and control structures (and possibly simpler systems tools and user tools) from which the current tool is constructed.

On the right hand side of the 'Tool Details' box are three active buttons, **Test**, **Close** and **Browse Tools** for primitives and system tools. The remaining buttons will only be active for User tools.

Browse Tools allows the user to browse any tool and will be dealt with in Chapter 10, below.

The **Close** button removes the 'Tool Details' dialog box

The **Test** button allows you to run an existing tool or try out a new user tool.

If you press **Test**, a dialog box appears listing all the input parameters which need to be filled in. In this case, we are searching for the attributes of the object 'badahar' (a Nepalese tree), in the knowledge base 'treefodd'. The input parameters are entered into the template (Figure 9.3). Then press **Continue**. The 'Tool output' dialog box then appears (Figure 9.4). Under 'Result' are the two attributes describing the tree badahur, height and growth_rate.

Figure 9.3 Dialog box with input parameters filled in for the tool `formal_terms(Kb, Term, Type, Result)`

Figure 9.4 Tool Output box for the *primitive formal_terms(Kb, Term, Type, Result)* used on the knowledge base 'treefodd'.

(In the 'Tool output' dialog box, it gives the tool name as `formal_term/4`. This is merely a shortened form, the '/4' referring to the number of parameters (input and output) associated with the tool).

This time try out the tool for real. Press **Close** on the 'Tool output' dialog box and **Close** on 'Tool Details' dialog box. The screen will revert to the 'Tools' dialog box. The tool should still be highlighted in the list of tools. Press **Run**. A dialog box appears requiring you to fill in the input parameters. Fill it out, as in Figure 9.3 but this time instead of entering *attributes* in the 'Type' box, enter *hierarchies*. Press **Continue**. The 'Tool output' dialog box appears, giving a list of all the hierarchies to which the tree badahar belongs:

```
All_trees;
Fodder_animal_type;
Fodder_overripening_month;
```

```

Fodder_quality;
Fodder_ripening_month;
Leaf_flushing_month;
Leaf_shedding_month;
Posilo_kam_posilo_fodder_trees;
crop_land_trees

```



Although the user has the option to run any tool including the primitives as shown above, normally only system or user tools will be run like this, primitives tend to be used within the bodies of other tools.

9.2.2 CONTROL STRUCTURES

There are six control structures available in the AKT task language for use in the creation of reasoning tools. To view the available control structures, select the control structures from the drop down menu under Tools in the main Tools menu.

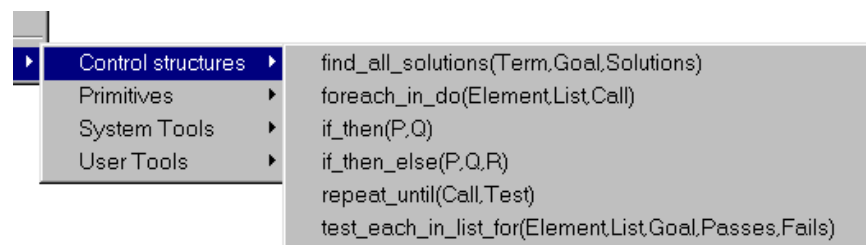


Figure 9.5 Drop-down menu for control structures

Control structures are pieces of code, which are used to link one primitive to another, to construct a tool – they are, if you like, the ‘glue’ that binds the primitives together.

Name of control structure	Function of control structure
Find..all..solutions(Term, Goal, Solutions)	This structure allows you to find all solutions to a specified goal. The goal can be a primitive, a system or a user tool or any combination of them.
Foreach..in..do(Element, List, Call)	The operation detailed in Call (or Goal) is executed in turn for every item in the specified list
If..then(P, Q)	In this structure ‘P’ is a primitive or tool or any combination of primitives and tools, which returns a response ‘true’ or ‘false’. If the response is <i>true</i> then ‘Q’ (a primitive, tool or a combination of both) is invoked before continuing with the next primitive in the tool definition. If the response is <i>false</i> then ‘Q’ is ignored and the tool continues with the next primitive in the definition.
If..then..else(P,Q,R)	It is possible to elaborate on the above tool by using the if_then_else control structure. This is similar to the if_then structure except that, if the test fails, the ‘else’ call is implemented rather than simply exiting the control structure tool.
Repeat..until(Call, Test)	This control structure enables you to repeat a Call (i.e. an instruction or chain of instructions) until a Test is successful.

Test..each..in..list..for(Element, List, Goal, Passes, Fails)	This control structure tests each item in a list for a specified condition and returns two lists, one containing the items that meet the condition and one containing the items that do not meet the condition.
---	---

* a 'call' is goal consisting of an instruction or a sequence of instructions where an instruction can be a primitive or system or user tool

9.3 SYSTEM TOOLS

System tools are the tools already created and supplied within the AKT program. If you select **System Tools** from the 'Tools' dialog box the following drop-down menu appears (Figure 9.6):

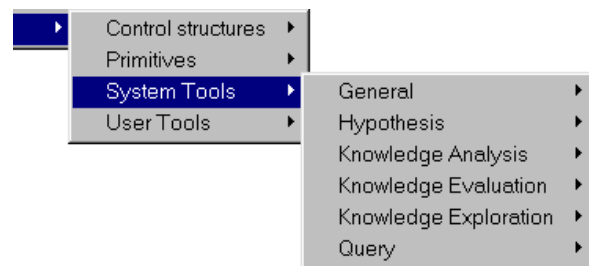


Figure 9.6 The drop-down menu for System Tools

The following is an explanation of the different categories of systems tool available:

General	These are general systems tools not incorporated into the other categories
Knowledge Analysis	Analyses the effect(s) of a statement or formal term(s) on the knowledge base
Knowledge Evaluation	Evaluates the contents of the knowledge bases and checks for example inconsistencies or redundant information within the knowledge bases
Knowledge Exploration	Explores the knowledge base for the implications of processes and management actions
Query	Allows the user to determine if anything in the knowledge base matches any type of formal statement specified by the user

9.3.1 SOME EXAMPLES OF SYSTEMS TOOLS

a) System Tool Category: Knowledge Evaluation

System Tool: kb_report/1

The System Tool kb_report/1 is in the category Knowledge Evaluation (Figure 9.7);

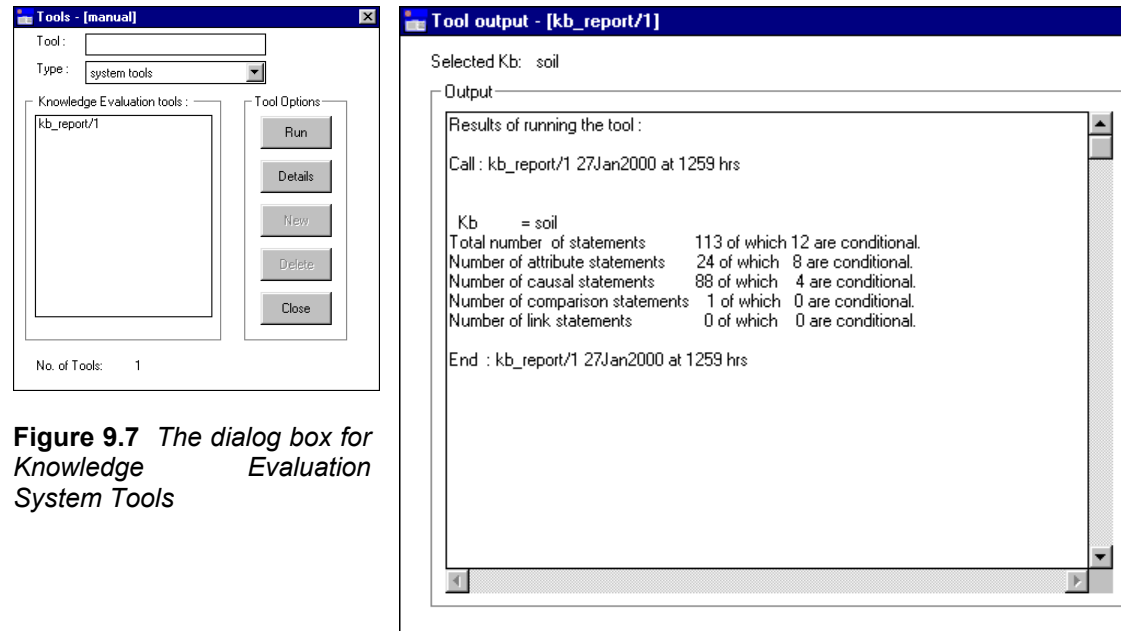


Figure 9.7 The dialog box for Knowledge Evaluation System Tools

Figure 9.8 Output screen for kb_report/1 on the 'soil' knowledge base

This is a simple system tool requiring you only to press **Run** and enter the name of the knowledge base to be interrogated. Alternatively it can be incorporated within a user tool as just part of a more extensive user report.

Running the tool gives a report (Figure 9.8) on how many statements within the specified knowledge base fall into the following categories:

- All statements of which n number are conditional
- Simple attribute value statements of which n number are conditional
- Causal statements of which n number are conditional
- Comparison statements of which n number are conditional
- (User defined) Link statements of which n number are conditional

b) System tool category: Query

statement_query(Kb): This tool allows the user to analyse the knowledge base by submitting a query of arbitrary complexity. The query is specified in the form of a formal statement that conforms to the AKT grammar, for example, typing

att_value(Ob1,Att1,Val1) causes2way att_value(Ob2,Att2,Val2) in the query box will allow the user to find all the causal statements in the knowledge base that match the query.

In the example below (Figure 9.9 and 9.10) all the 'soil' knowledge base statements have been examined and any attribute phrases matching the AKT expression att_value(O,A,V) have been found.

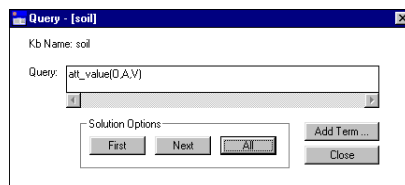


Figure 9.9 Using *statement_query.0* to find all attribute value phrases of a certain format

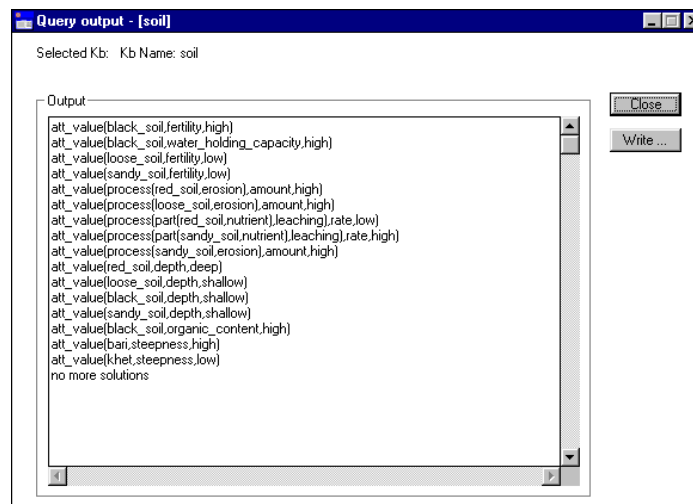


Figure 9.10 The output window for the statement query: *att_value(O,A,V)*

c) System tool category: General

merge_External_Statements: This tool enables you to merge statements from an external knowledge base into the current knowledge base. If you select this tool and press **Run**, a directory appears, with the legend 'Kb containing the statements to be merged?' from which you select the knowledge base to be integrated into current knowledge base.



Although this tool makes the merging of two knowledge bases extremely easy, it should be used with caution. Only if the definitions of formal terms used in both knowledge bases are identical is it possible to merge the two knowledge bases without fear of misunderstandings or internal contradictions arising.

CHAPTER TEN – CREATING YOUR OWN TOOLS

10.1 INTRODUCTION

The facility for creating new tools in the AKT software is provided for the user whose particular needs are not met by existing tools. This chapter introduces the basic principles of creating a new tool, and gives some examples.

10.2 TOOL FILES

10.2.1 CREATING A NEW TOOL FILE

If you wish to create a new tool you must first create a tool file in which to save the new tool(s). To do so, select **Tools** from the main menu and then **New Tool File**. The following screen will appear (*Figure 10.1*), requiring that the new tool file be given a name. The suffix is always automatically '.mcr' (for 'macro'). Once the file has been named, press **Save** and the following message will appear (*Figure 10.2*):

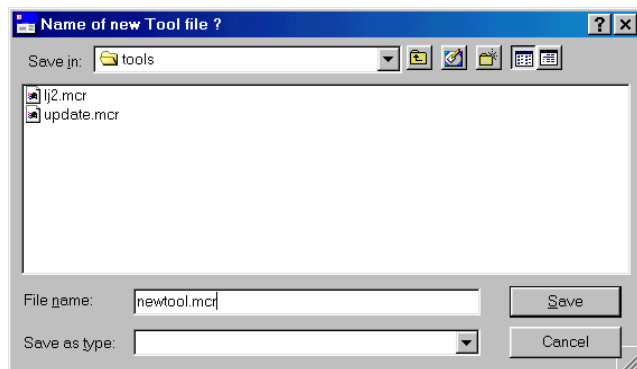


Figure 10.1 Naming a new tool file

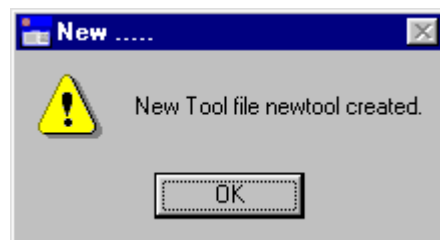
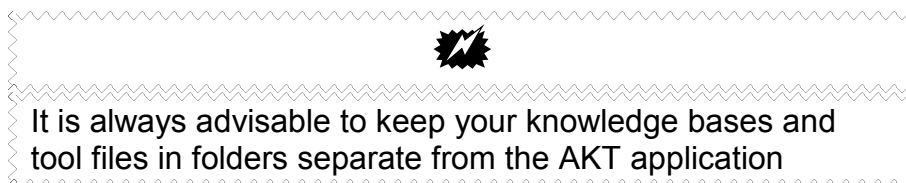


Figure 10.2 Confirming that a new tool file has been created

10.2.2 OPENING A PREVIOUSLY CREATED FILE

To open a previously created tool file, select **Tools** from the main menu, then **Open Tool File**. Then browse through your folders and files to select the correct file. Highlight the file required and press **Open**. The tool file will then automatically open.



10.2.3 KEEPING TRACK OF THE TOOL FILES

It is possible to have more than one tool file loaded at a time, simply by creating or opening more tool files. To change the tool file in use, select **Tools** from the main **Tools** menu, then **User Tools** and then the tool file required (*Figure 10.3*).

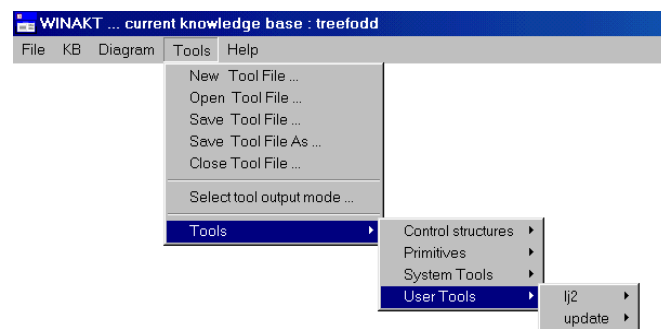


Figure 10.3 Changing the tool file in use

10.2.4 SAVING TOOL FILES

It is only necessary to save a tool file if you have created or edited existing user tools. When you save the new or edited tool, a message appears asking if you wish to save an updated tool file as well as keeping a copy in memory. If you wish to preserve the newly created tool, you *must* save the tool file as well, therefore select **Yes**.

If you wish to save the new tool but not in the current tool file, then save the tool file under a new name.

It is possible to save a tool file at any time by opening the main **Tools** menu and selecting **Save Tool File**. To save it under a different name, or in a different folder or different directory, select **Save Tool File As**.

10.2.5 CLOSING TOOL FILES

In order to close a tool file select **Close Tool File** from the main **Tools** menu. If you have more than one tool file open, the dialog box that appears (*Figure 10.4*) allows you to select the tool file to be closed from the 'Files' list. Once selected, press **OK**.

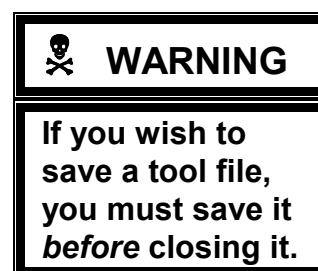
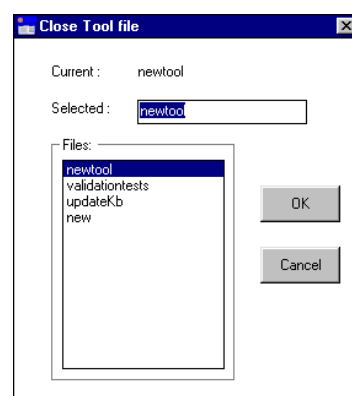


Figure 10.4 Closing a tool file

10.3 AN EXAMPLE OF A USER DEFINED TOOL

Below we give an example of a user defined tool, from a user defined tool file, Ghana Tools¹ which we will run on the treefodd knowledge base. Make sure you have the Ghana Tools tool file loaded and then select the tool **search_Topic_Statements**. This tool allows the user to search for a term, or several terms in a given topic. First we will run the tool, and then we will analyse how it has been built up.

To run the tool, press **Run** (Figure 10.5). A dialog box appears, requesting you to select a topic (Figure 10.6). Select your topic, in this case 'scientists' and press **OK**.

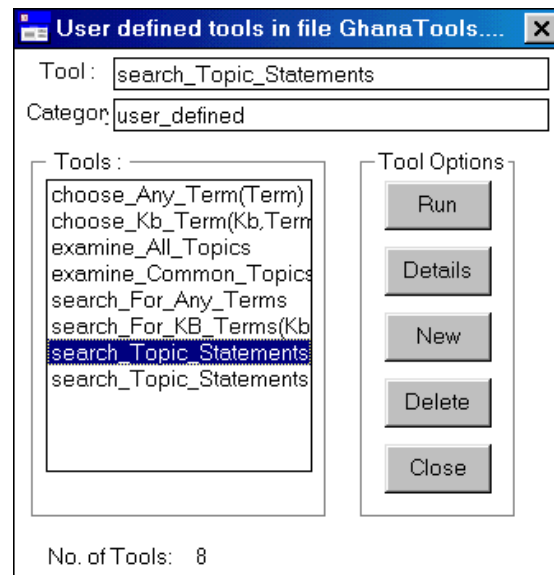


Figure 10.5 User-defined tools

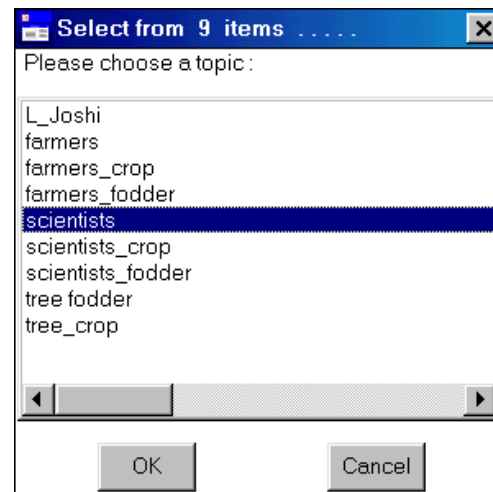


Figure 10.6 The tool **search_Topic_Statements** in action

A new dialog appears, requesting you to type in the formal term/terms that you want to search for, in this case 'soil' (Figure 10.7). Press **OK**. The tool output (Figure 10.8) will then give you all statements containing the term 'soil', in natural language, from the topic 'scientists'.

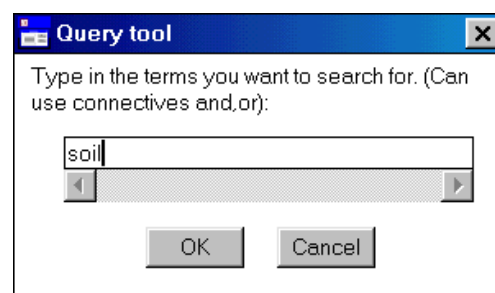


Figure 10.7 The tool **search_Topic_Statements** in action continued

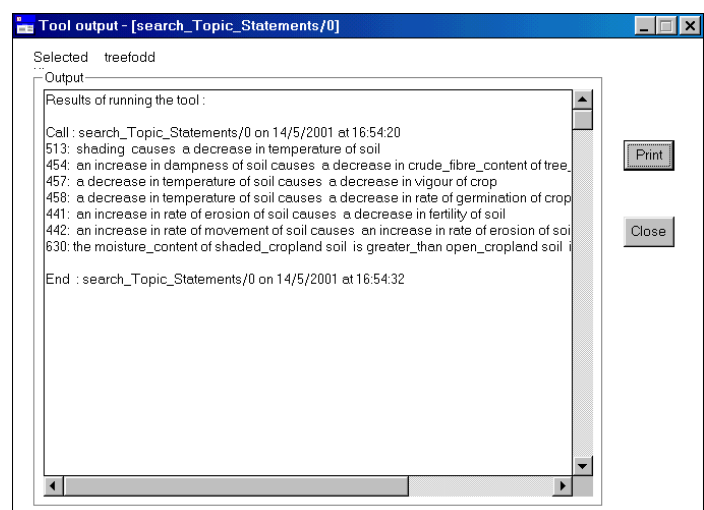


Figure 10.8 The Tool output

¹ These tools were created for the Atwima knowledge base. Both the knowledge base and tool file can be downloaded from our website: www.safs.bangor.ac.uk/afforum

Now we will look at the body of the tool. If you return to the tool **search_Topic_Statements** and press **Details**, the following dialog box appears (Figure 10.9):

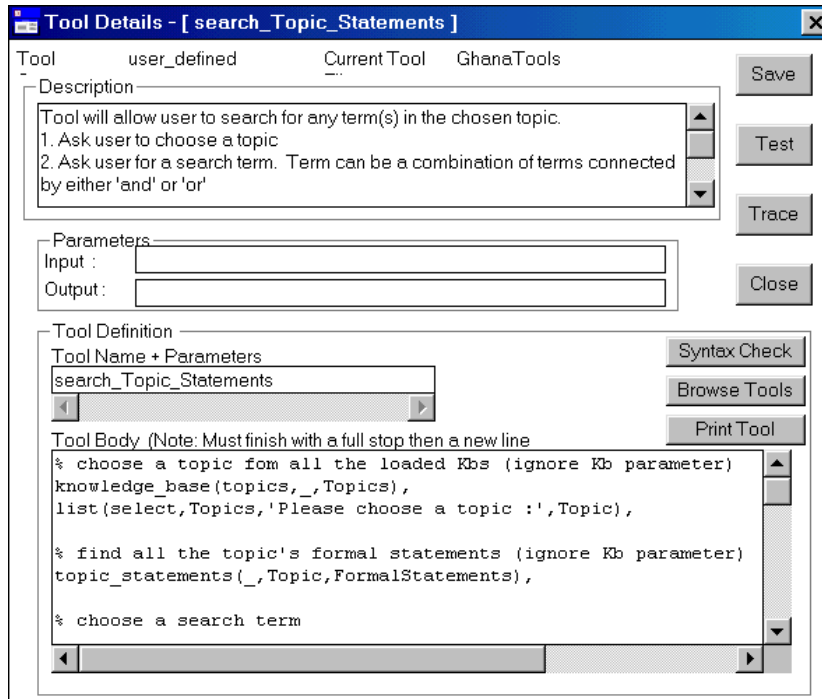


Figure 10.9 Tool details for the tool **search_Topic_Statements**

The full tool body looks like this:

```
% choose a topic fom all the loaded Kbs (ignore Kb parameter)
knowledge_base(topics,_,Topics),
list(select,Topics,'Please choose a topic :',Topic),

% find all the topic's formal statements (ignore Kb parameter)
topic_statements(_,Topic,FormalStatements),

% choose a search term
ask('Type in the terms you want to search for. (Can use connectives and,or)',SearchTerm),

% search the formal statements for selected term (ignore Kb parameter)
statements_search(_,SearchTerm,_,object,FormalStatements,TermStatements),

% list all the numbered translations of the Formal statements
foreach Statement in TermStatements
do ( statements_convert(numbered,_,Statement,Translation),
    show(Translation),show(nl)
    ).
```

We will now go through the tool, step by step. As we go through the primitives that make up the tool, look at each primitive in turn, by using the **Browse Tools** button on the 'Tool Details' dialog box.

Several things to note before we start:

- a) A line beginning with '%' is a comment and not an executable part of the tool. Thus '% choose a topic from all the loaded Kbs (ignore Kb parameter)' tells you that the next step will instruct the program to permit you to choose any one topic from all the

available topics in the loaded knowledge bases. (In the following step by step analysis of the tool, these descriptions have been removed).

- b) Each primitive or tool may have parameters which can be any combination of input and output parameters. The output parameter can contain a result which is returned as the output from the tool or it can in turn be used as an input parameter to another primitive or tool later in the definition.



NOTE: An **input parameter** is one where the primitive or tool expects the parameter to be specified before starting the tool. An **output parameter** is one where the primitive or tool determines the value and returns this value as the output at completion of the tool.

Here goes:



Take one

```
knowledge_base(topics,_,Topics),
list(select,Topics,'Please choose a topic : ',Topic),
```

The first step enables the user to specify the topic that he wishes to investigate. The topic may come from any of the loaded knowledge bases, as the knowledge base parameter is left unspecified.

This primitive is from the **Miscellaneous** category and its generic form is **knowledge_base(Type, Kb, Result)**. In this tool you will see that there is an anonymous variable '_' where the Kb parameter would normally be. This means that the knowledge base parameter will match any loaded Kb name, i.e. the tool will search all the loaded knowledge bases, rather than a specific knowledge base.

```
list(select,Topics,'Please choose a topic : ',Topic),
```

This primitive is from the **List** category and its generic form is **list(Operation, List, Item, Result)**. The variable 'Operation' is replaced by the term 'select'. (In the Description box of the primitive is a list of all the possible terms that can be used for the variable 'Operation' – 'select' is one of the possibilities). The variable 'List' is replaced by the variable 'Topics' which will give a list of all the topics found by the previous primitive. The variable 'Item' is specified as the message 'Please choose a topic' which has to be enclosed in inverted commas. Finally the variable 'Topic' will contain the name of the topic selected by the user



Remember: parameters which start with a capital letter are variables (unless they are enclosed in inverted commas), whereas parameters which start with a small letter are atomic terms and do not change.



Take two

```
topic_statements(_,Topic,FormalStatements),
```

This step will find all the formal statements about the topic selected in step one.

The primitive is from the **Statements** category and its generic form is **topic_statements(Kb, Topic, Statements)**. The first term of the primitive is an anonymous variable ('_') and will match with any knowledge base. The output variable 'FormalStatements' contains a list of the statements about the topic.



Take three

```
ask('Type in the terms you want to search for. (Can use connectives and,or)',SearchTerm),
```

This step enables the user to specify the search term/terms

This primitive is from the **Display** category and the generic form is **ask(Question, Answer)**. The parameter 'Question' is used to contain a message. To distinguish the message from a variable, it is written within inverted commas. The output variable 'SearchTerm', contains the term(s) typed in by the user.



Take four

```
statements_search(,SearchTerm,_,object,FormalStatements,TermStatements),
```

This step will search the formal statements about the selected Topic for the user specified search term.

This primitive is from the **Statement** category and its generic form is **statements_search(Kb,Term,Hierarchy,SearchOption,StatementsIn,ListFormalStatements)**

The first and third parameters are anonymous variables because we are not restricting the search to a particular knowledge base or object hierarchy. The variable SearchTerm contains the output from step 3 and the search option is set to object because we are looking for statements that contain the specified search term(s) and not statements that might contain the superobjects or subobjects of the search term. FormalStatements, contains the output of Step 2 and TermStatements will contain any of these statements that contain the search term(s).



Take five

```
foreach Statement in TermStatements
do ( statements_convert(numbered,_,Statement,Translation),
    show(Translation),show(nl)
    ).
```

This last step lists all the statements containing the specified formal term. It translates them from the formal syntax into natural language and then displays them on the screen prefixed by the statement number.

For this final step, the control structure **foreach...in...do..** has been combined with two primitives, one from the **Statements** category, **statements_convert(Mode,Kb,Original,Converted)** and one from the **Display** category **show(Term)**

Note: if the 'do' part of the control structure contains more than one primitive or tool then they need to be enclosed within parenthesis

This sequence of instructions is

- a) **statements_convert(numbered,_,Statement,Translation)**. The first input parameter 'Mode' is set to 'numbered', one of the mode options². The second input parameter is the anonymous variable again and the third input parameter 'Statement' Contains the formal statement to be translated and numbered.

- b) **show(Translation)** This primitive simply shows the numbered translation on the screen.

Show(nl). This last part simply instructs the tool to output a new line (nl) and carriage return after the translated statement. The control structure **foreach...in...do..** insures that the instructions a, b and c are carried out for each statement in the selected topic, containing the specified term until all such statements have appeared on the screen.

Finally, note that the last line contains the closing bracket for the control structure and a full stop. You must always add a carriage return (↵) after the full stop to complete the tool.

² See the 'Description' box for the primitive **statements_convert(Mode,Kb,Original,Converted)** to see the full list of mode options.

The Trace button.

To observe the various functions of the different steps, select the tool **search_Topic_Statements** once more and in the 'Tool Details' dialog box, press the **Trace** button. This will take you through the tool step by step. This function is particularly useful if a tool does not function properly. You can see at which step the error lies.

10.4 CREATING YOUR OWN TOOL

In order to create a new tool in AKT it is necessary for a user file to be opened to contain the tools. Open the main **Tools** menu and select either **New Tool File** or **Open Tool File** (see above, section 10.2) Once a new tool file has been created, or an existing tool file opened, a dialog box will appear showing a list of 'User defined tools' (*Figure 10.10*). If no tools have been previously created by the user, this dialog box will be empty.

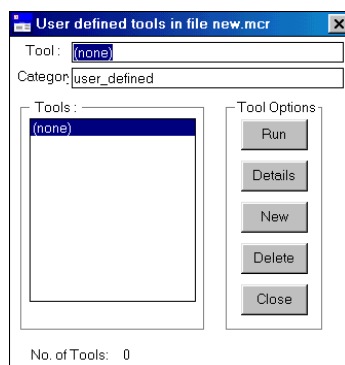


Figure 10.10 'User defined tools' dialog box

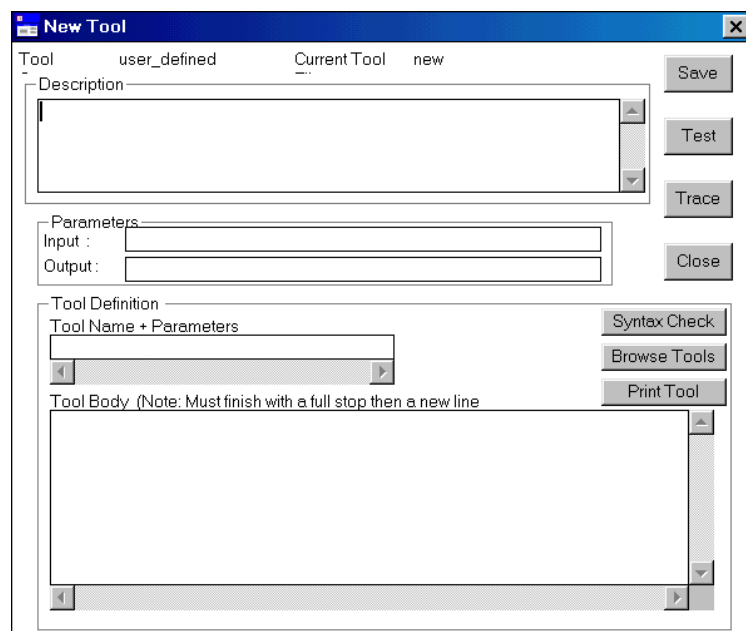


Figure 10.11 Dialog box for the creation of a new tool

In order to create a user defined tool, select **New**. A 'New Tool' dialog box appears (*Figure 10.11*)

The information that must be specified about a tool includes:

- Description of the tool's function
The 'Description', written in plain text should describe the tool's function and give instructions for its use. It is worth remembering that if the knowledge base is to be used by others then the description should be as informative as possible.
- Input and output parameters
The input and output parameters (arguments) required by the tool. Parameters are the means of passing information into or out of the tool. Each tool can have any number/combinations of input and output parameters. A tool does not necessarily need input parameters nor output parameters – it depends on its function. You only need any parameters when you are passing information into or extracting information from a tool.
- Tool Definition –a) Tool name and parameters
b) tool body

- a) The 'toolname' must start with a lower case letter e.g. **find_trees** or if it is required to start with a capital letter then the whole toolname must be enclosed in single quotes e.g. **'Find_trees'**. The arguments or parameters contained in parenthesis after the toolname generally begin with a capital letter indicating a variable but they can be a fixed value if required. For example *Kb* and *Date* are variables but *attributes* and *'John Smith'* are fixed values. The toolname should express succinctly the function carried out by the tool.
- b) The 'tool body' contains all the functions, control structures and appropriate punctuation that allows the tool to perform the required process. If the tool body does not have the correct form or syntax then AKT will report a 'Syntax Error' whenever an attempt is made to save the tool or the syntax check button selected.

10.4.1 INCORPORATING EXISTING TOOLS / PRIMITIVES WITHIN A NEW TOOL DEFINITION

There are various ways of adding a primitive, systems tool, control structure or previously defined user tool to the definition of the new tool; two possible ways using cut and paste functions are described below.

Method 1: Select the **Browse Tools** box from the 'New Tools' dialog box (see above, *Figure 10.11*). This will display a list of all the tools. Highlight the primitive you think you need and then select **Details**. The 'Description' within the 'Tool Details' will enable you to decide whether or not it is indeed the primitive you want to use. Once you have the desired primitive highlight the 'Tool Name + Parameters' box and press **Ctrl c** to copy it. Return to the 'New Tool' dialog box and taking the cursor to the 'tool body' box, press **Ctrl v** to paste it into the tool.

Method 2: Another method of copying a primitive, once you are familiar with the program, is to select **Help** from the main menu, and from the dropdown menu that appears, select **Tool List**. The following screen will then appear (*Figure 10.12*):

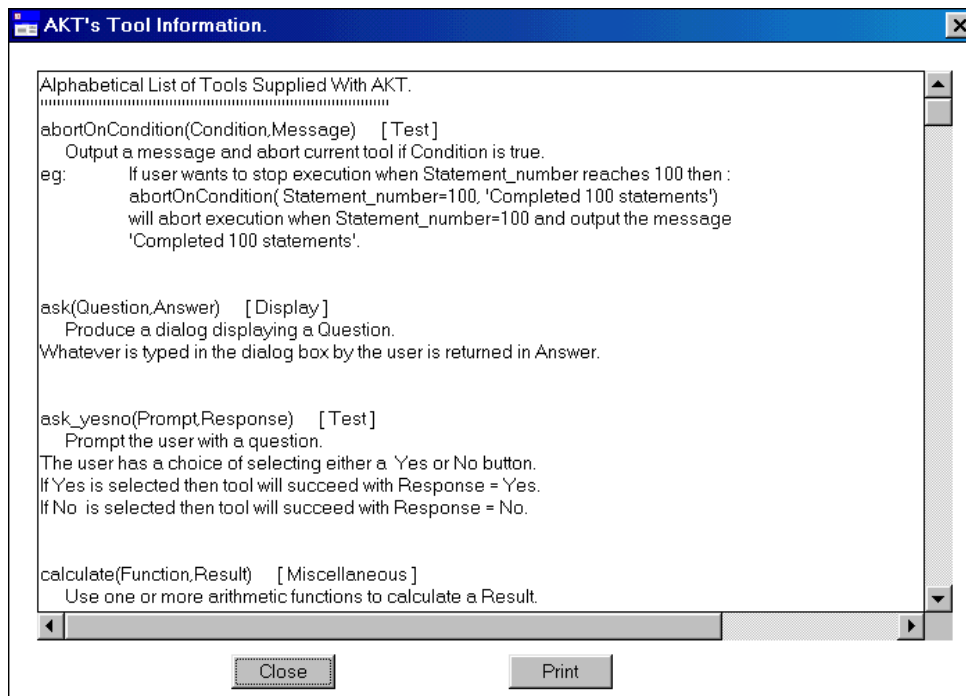


Figure 10.12 List of Tools available in **Help** on the main menu

The list is alphabetical and gives the tool name and the category to which it belongs and the description of what it does. Using the scroll bar enables you to select the tool you require.

Highlight the tool name and parameters (but not the category, nor the definition) and press **Ctrl c**. Return to the 'New Tool' dialog box and setting the cursor in the 'tool body' box, press **Ctrl v** to paste it into the tool.

10.4.2 TESTING THE SYNTAX OF A NEW TOOL

In order to check whether or not the syntax of the tool is correct, select **Syntax Check**. If there is an error in the syntax anywhere within the tool one of the following messages will appear (*Figures 10.13 and 10.14*):

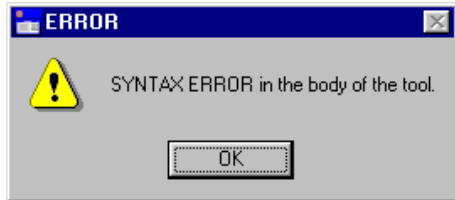


Figure 10.13 Error message which appears if the format of the control structure is wrong, or the punctuation incorrect

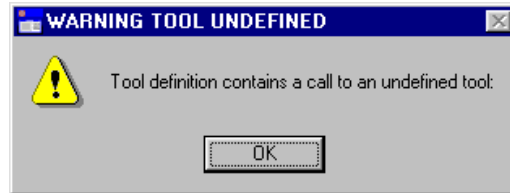


Figure 10.14 Message received if a non-existent tool has been used

A **syntax error** occurs when the format of the tool head or body does not correspond to the Prolog syntax which is used by the code underlying the AKT application. A **call to an undefined tool**, on the other hand, means that the name of a tool used in the function does not exist. It may have been incorrectly spelt or the number of parameters may be wrong.

As the message does not tell you specifically where the error lies within the tool, it is useful to carry out a syntax check after each new line is added. In order to do this you can temporarily put a full stop at the end of the new line and follow it with a carriage return (↵); then press **Syntax Check**. If a message 'Tool syntax is ok' appears then you can continue building the tool, first removing the full stop and replacing it with a comma before starting a new line. If a message appears as in *Figure 10.13* or *10.14* above, then you should not proceed to a new line until the current line of instructions has been corrected.

10.4.3 AN EXAMPLE OF CREATING A TOOL

The best method of explaining how to create a tool is by demonstration.

In this example we are creating a tool which will list all the statements related to a specified term (or terms) and then saving these statements in the form of a new knowledge base.

Select **New** from the 'user defined tools' Tool dialog box (as above, *Figure 10.10*) and the 'New Tool' dialog box appears (as above, *Figure 10.11*).

First give a description of what the tool does in the 'Description' box. In this case a sentence such as 'This tool collects all statements relating to a user specified term/terms and then saves them as a new knowledge base' will express its function adequately.

The toolname 'saveStatementsToKb' acts as a sufficient 'aide memoire' for its function. Note that the first letter in the tool name is in lower case. In this instance there are two externally supplied parameters needed to make the tool run; the knowledge base you are interrogating, the term or terms you are looking for. Thus the full name of the tool will be saveStatementsToKb(Kb,Term).

Then move the cursor down to the tool body. The tool we wish to create should do the following things;

- a) search for all the statements about a given term (or terms) in a specified knowledge base
- b) save the formal statements as a separate knowledge base
- c) convert the formal statements into natural language
- d) display the statements in natural language.



BEFORE YOU START - SYNTAX CHECK

As you build a new tool you should carry out a syntax check after *each new line*, so that any source of error can easily be traced. See above, 10.4.2

For a) we need the primitive:

statements_search(Kb,Term,Hierarchy,SearchOption,StatementsIn,ListFormalStatements).

We access this by pressing **Browse Tools** in the 'New Tool' dialog box (as above, Figure 10.11), and selecting **Primitives** then **Statement**. We then highlight the primitive required and press **Details**. From the 'Details' dialog box we highlight the toolname plus the arguments and press **Ctrl c**. Returning to the 'New Tool' dialog box we move the cursor to 'tool body' and press **Ctrl v**³.

Looking at the primitive, we need to tailor it to our needs.

The description of the primitive is:

This tool enables the user to search the complete knowledge base or a subset of it, and identify statements that contain any specified formal term, synonym, source, topic or a combination of these items. The tool has the same capability as the interactive Kb / Boolean Search menu option. It collects together in a list any formal statements from the StatementsIn list that contain Term or its superobject or subobject depending on SearchOption.

The user can restrict the effect of the SearchOption to a particular hierarchy by specifying the Hierarchy parameter. However this parameter can be ignored for more general searches (set Hierarchy = _). If user is not interested in the effect of object hierarchies then SearchOption should be set to object.

If StatementsIn = all then search will be carried out on all statements in Kb knowledge base.

This description box allows us to see how we can implement the primitive. Thus, as we are not interested in hierarchies, we will replace the Hierarchy variable with the anonymous variable ('_') and the SearchOption is set to 'object'. Finally, as we want the search to be carried out on all statements in the knowledge base, the StatementsIn variable is replaced by 'all'.

The tailored primitive will therefore look like this:

statements_search(Kb,Term,_,object,all,ListFormalStatements),

Carry out a syntax check (as recommended in the box above).

For b) we need the **Statement** primitive, **statements_save(Kb,Statements).**

Enter the primitive details via the **Browse Tools**. Highlight the tool name plus parameters and, using **Ctrl c** and **Ctrl v**, place the primitive as the second line in the new tool, add a comma and press carriage return.

³An alternative, quicker route is Method 2 as described above, 10.2.1.

The growing tool now looks like this:

```
statements_search(Kb,Term,_,object,all,ListFormalStatements),  
statements_save(Kb,Statements),
```

When you are combining primitives together to create a new tool there are two points to remember

- The parameter names in the tool body must be identical to the parameter names in the tool head when referencing the input or output parameters.
- When a primitive in the new definition is using the output parameter of a primitive used earlier on in the tool body as an input parameter, the two parameter names must be identical.

Thus, the input parameter 'Statements' in this second primitive is replaced by 'ListFormalStatements', the output parameter from the primitive above. When amended, the second line of the new tool looks like this:

```
statements_save(Kb,ListFormalStatements),
```

Do a syntax check and then go on to c)

For c) we need the **Statement** primitive **statements_convert(Mode,Kb,Original,Converted)**. Enter the primitive details via the **Browse Tools** and copy and paste the tool name and parameters as above in a). Then add a comma and press carriage return. The growing primitive should now look like this:

```
statements_search(Kb,Term,_,object,all,ListFormalStatements),  
statements_save(Kb,ListFormalStatements),  
statements_convert(Mode,Kb,Original,Converted),
```

The description box below suggests ways in which the primitive **statements_convert(Mode,Kb,Original,Converted)** might be modified.

Allows user to change the format of either a single or a list of statements. The choices are :

(a) Mode = formal Convert list of statement identifiers to a list of formal statements.
(b) Mode = identifier Convert list of formal statements to a list of statement identifiers.
(c) Mode = translate Translate a list of formal statements to their natural language equivalents.
(d) Mode = numbered Convert list of statement identifiers or formal statements to numbered list of natural language equivalents.

eg: statements_convert(formal, soil,[1,25,73],FormalStatements)
 statements_convert(identifier,soil,att_value(black_soil,fertility,high),Number)
 statements_convert(translate, soil,[att_value(black_soil,fertility,high)],Translation)
 statements_convert(numbered, soil,att_value(black_soil,fertility,high), Number)

NOTE: when converting formal statements to their equivalent statement number it is possible to have more than one solution if the knowledge base contains duplicate statements from different sources or more than one knowledge base is loaded.

As the statements should retain their identifying number, the Mode variable is replaced by the option 'numbered'. The second variable Kb remains the same. The variable 'Original' is replaced by the input parameter from the previous primitive 'ListFormalStatements'. The variable 'Converted' we can give whatever name we choose – in this instance 'TranslatedStatements' would seem appropriate.

Thus, the third line, when amended, should look like this:

```
statements_convert(numbered,Kb, ListFormalStatements, TranslatedStatements),
```

Do a syntax check and then go on to d)

Finally to d). The tool output will not automatically be displayed in the tool output menu. To display the tool output, you must incorporate instructions into the tool itself. To do this we add the Display primitive **show(Item)**. Whatever appears in place of the Item variable will appear on the screen. We need to show two things, (i) a message saying what the output is and (ii) the output itself. Therefore we will need the primitive **show(Item)** twice.

The tool would now look like this:

```
statements_search(Kb,Term,_ ,object ,all,ListFormalStatements),  
statements_save(Kb,ListFormalStatements),  
statements_convert(numbered,Kb, ListFormalStatements, TranslatedStatements),  
show(Item),  
show(Item)
```

The variable *Item*, in the first **show(Item)** primitive would be a message. This message must be in inverted commas. 'This is a numbered natural language version of the statements' would be an appropriate message.

The variable *Item*, in the second **show(Item)** primitive should be the statements themselves. Thus Item should be replaced by the output parameter of the third primitive TranslatedStatements.

Thus the final tool will look like this:

```
statements_search(Kb,Term,_ ,object ,all,ListFormalStatements),  
statements_save(Kb,ListFormalStatements),  
statements_convert(numbered,Kb, ListFormalStatements, TranslatedStatements),  
show('This is a numbered natural language version of the statements'), show(nl),  
show(TranslatedStatements).
```

You will note that show(nl) has been added to the fourth line, in order for the statements to appear on a new line below the message, rather than directly appended to it. The new line is not necessary for the tool to work, it simply makes the output easier to read. Finally, the tool is finished with a full stop and carriage return (↵).



A tool can *only* be completed by putting a full stop at the end of the last line and entering a carriage return (↵).

Carry out **Syntax Check** on last time. Your screen should look like *Figure 10.15* below. If the message 'Tool syntax is ok' appears, you can proceed to the next step.

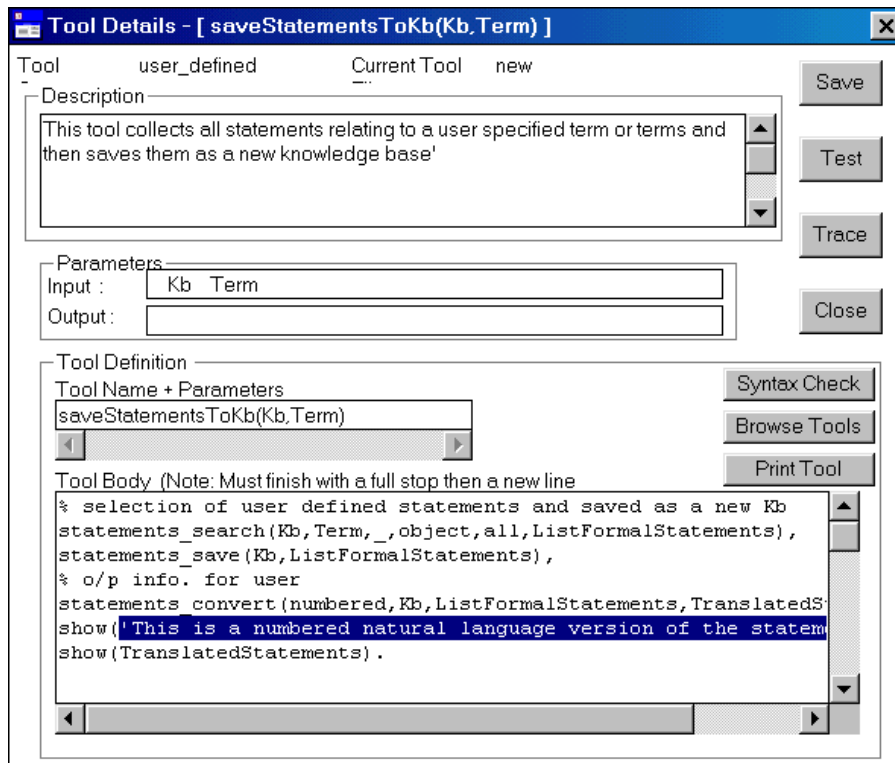


Figure 10.15 The new tool, *saveStatementsToKB(Kb,Term)*, completed

Then press **Save**. A dialog box will appear (Figure 10.16) with the two parameters in the toolname, requesting you to specify which are the input parameters. In this case both parameters are input parameters, so tick the two and then press **OK**.

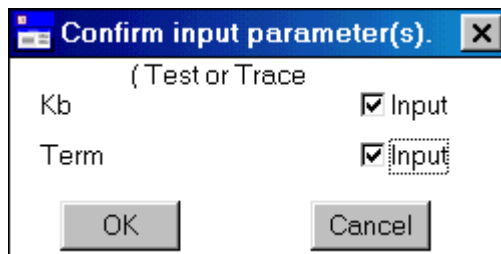


Figure 10.16 Dialog box requesting the input parameters.

10.4.4. TRACING AND TESTING A NEW TOOL

Once a new tool has been created it is important to test whether it works properly. This can be done by using either **Trace** to step through the tool one line at a time or **Test** to run the whole tool at once. In this example we will look for all the statements containing 'soil' in the 'treefodd' knowledge base and save them as a separate knowledge base called 'treesoil'.

10.4.4.a The trace key

If, after following the example of the new tool above (10.4.3) you select **Trace**, the same dialog box as Figure 10.16 above will appear, in which the input parameters are specified. Tick both and press **OK**. Then another dialog box appears requesting you to enter the input parameters (Figure 10.17). For this example the parameters are entered as follows:

- Kb: soil
- Term: soil and erosion

Then press **Continue**. The following screen appears (Figure 10.18).

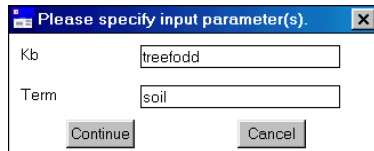


Figure 10.17 Specifying the input parameters for use of the `saveStatementsToKB(Kb,Term)` tool

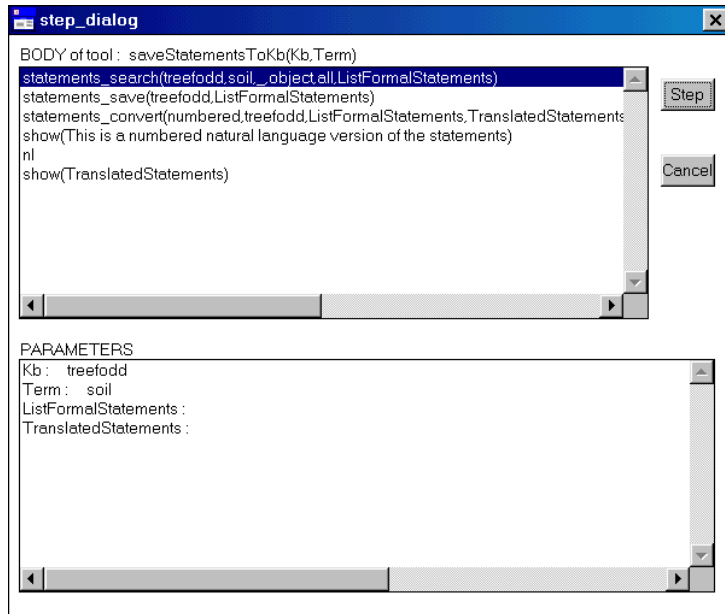


Figure 10.18 Using the Trace option

Each time you press **Step**, the highlighted bar will travel down one step in the tool and carry out the instructions on that line, the results appearing in the box beneath marked 'Parameters'. By the time you have reached the last line, the screen will look like this (Figure 10.19);

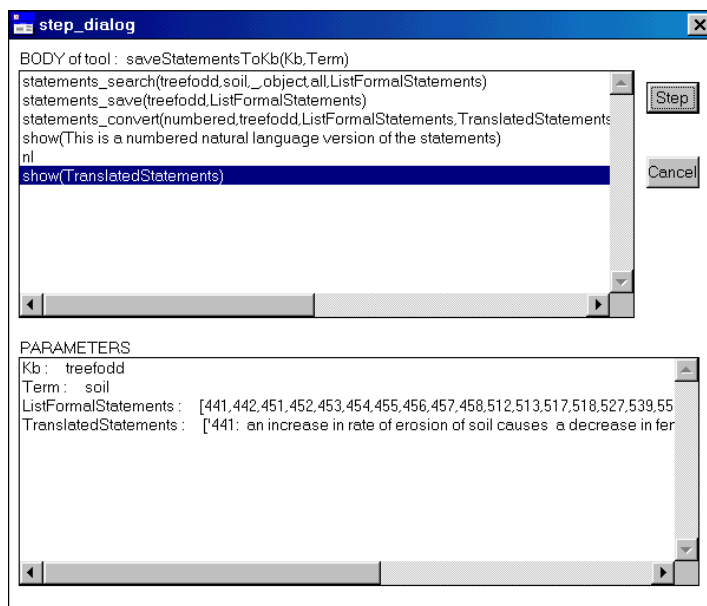


Figure 10.19 Using the Step button, going though the tool line by line

The **Trace** function allows you to follow the tool step by step and to monitor the actions of each line of the tool. The **Trace** function is enabled only if the tool is syntactically correct. Therefore, if the tool is not giving the required output it means that one of the individual functions is probably being used incorrectly. The step by step facility allows you to determine where it goes wrong by observing the parameter values in the lower box.

10.4.4.b The Test button

If you do not wish to test your new tool line by line you can instead select **Test** for a trial run of the tool. As in **Trace** the same dialog boxes appear, the first one specifying the input parameters (see above *Figure 10.16*) the second one requesting the input parameters to be entered (see above, *Figure 10.17*). After pressing the **Continue** key the 'Save Kb As' dialog box appears, requiring you to enter a name for the new knowledge base.

Once you have chosen and entered a name for the new knowledge base, press **Save**. The next dialog box to appear is then the 'Tool output' screen, giving all the statements containing 'soil'. Table 10.1 lists all the statements that appear in the Tool output screen:

Table 10.1 *Statements containing the formal terms 'soil' in the knowledge base 'treefodd' retrieved by the tool saveStatementsToKb(Kb,Term)*

Results of running the tool :

Call : saveStatementsToKb/2 on 22/5/2001 at 10:40:38

Kb = treefodd

Term = soil

This is a numbered natural language version of the statements

441: an increase in rate of erosion of soil causes a decrease in fertility of soil
442: an increase in rate of movement of soil causes an increase in rate of erosion of soil
451: the soil condition is moisture_stressed causes the crop condition is moisture_stressed
452: an increase in dampness of soil causes a decrease in vigour of crop
453: an increase in dampness of soil causes an increase in rate of infestation of crop_pest
454: an increase in dampness of soil causes a decrease in crude_fibre_content of tree_leaf
455: the dampness of soil is low causes the soil condition is moisture_stressed
456: a decrease in fertility of soil causes a decrease in vigour of crop
457: a decrease in temperature of soil causes a decrease in vigour of crop
458: a decrease in temperature of soil causes a decrease in rate of germination of crop seed
512: shading causes an increase in dampness of soil
513: shading causes a decrease in temperature of soil
517: the land site_quality is malilo causes an increase in fertility of soil
518: a decrease in quantity of manure causes a decrease in fertility of soil
527: an increase in effect of shading causes an increase in dampness of soil
539: a decrease in rukhopan of tree_leaf causes an increase in fertility of soil
556: an increase in competitiveness of fodder_tree causes a decrease in fertility of soil
568: an increase in soil_binding_ability of fodder_tree causes a decrease in rate of movement of soil
573: a decrease in competitiveness of tree causes an increase in fertility of soil
574: a decrease in difficulty of ploughing of crop_land causes an increase in fertility of soil
630: the moisture_content of shaded_cropland soil is greater_than open_cropland soil if the system season is hiudae_crop

End : saveStatementsToKb/2 on 22/5/2001 at 10:41:28

10.4.5 COMPLETING THE TOOL

Either before or after selecting Trace or Test you can save the new tool both locally to memory or to a disk file by pressing **SAVE**. If you have already saved once before the following message will appear (*Figure 10.20*):

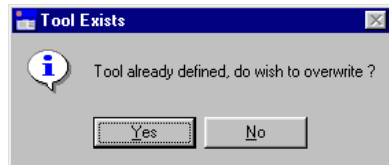


Figure 10.20 *Message that appears when saving a tool for the second time.*

Select **Yes**. Once the tool has been saved, select **Close**. If you now go to **Tools** under the main **Tools** menu, you will find that under 'User defined tools' your new tool is listed as **saveStatementsToKb(Kb,Term)**.

10.5 EDITING TOOLS

Only user defined tools can be edited. Primitives, control structures and systems tools cannot be edited. To edit a user defined tool, simply enter 'User defined tools' in the 'Tools' dialog box and select the tool in question. Select **Details**. You can then freely alter both the tool description and the tool definition. However, remember to press **Save** in the 'Tool Details' box to preserve the changes, otherwise any alterations will be lost.

10.6 DIRECTING TOOL OUTPUT TO A FILE

It is possible to direct any tool output to a file, rather than to a screen window. This is particularly useful for tool outputs larger than 64 Kb which is the limit for screen output. In order to do this go to the main **Tools** menu and select **Select tool output mode**. The program will then allow you to save the tool output in a file of your choice.

CHAPTER ELEVEN - INCORPORATING PICTURES AND DIAGRAMS INTO THE KNOWLEDGE BASE

In AKT 5 there is the option to incorporate pictures and diagrams into the knowledge base. The Atwima knowledge base below shows what can be done. *Figure 11.1* shows the Welcome Page with a thumbnail picture. This picture can be enlarged by double clicking on the image with the left mouse button (*Figure 11.2*).

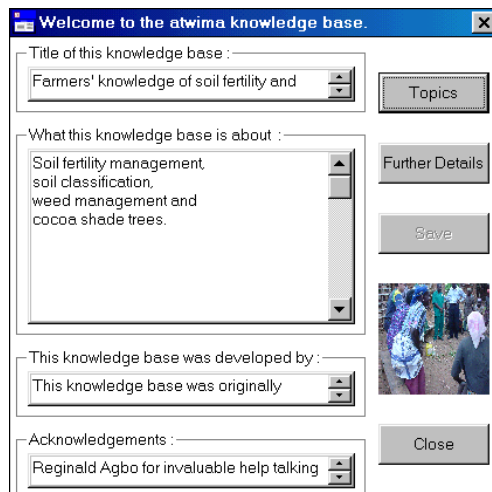


Figure 11.1 Welcome Page of the Atwima knowledge base with the thumbnail picture

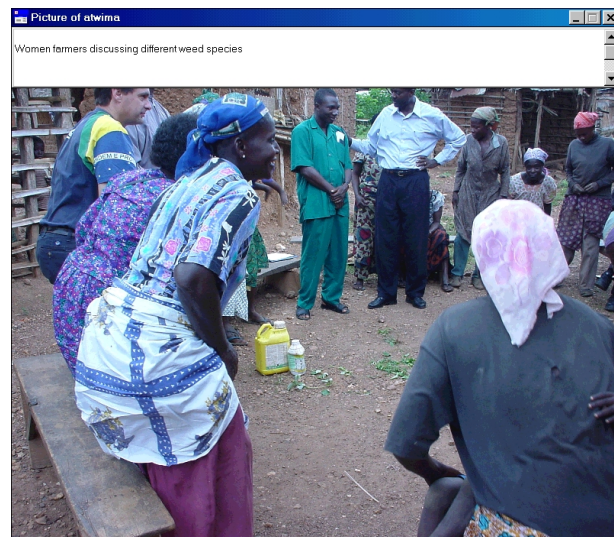


Figure 11.2 The thumbnail picture enlarged with a description of the image

By pressing **Further Details**, the second part of the Welcome Page appears (*Figure 11.3*), at the bottom of which is a button **Pictures/Diagrams**. Selecting that button, a new dialog box appears (*Figure 11.4*) from which you can select one of the pictures or diagrams on the menu, by highlighting one and pressing **Select**.

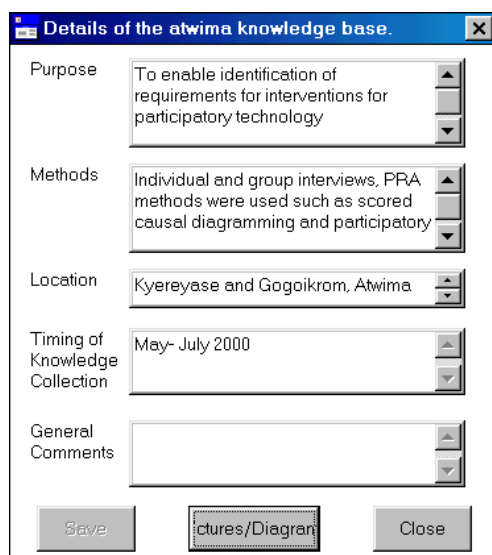


Figure 11.3 The second part of the Welcome Page, with the Pictures/Diagrams button

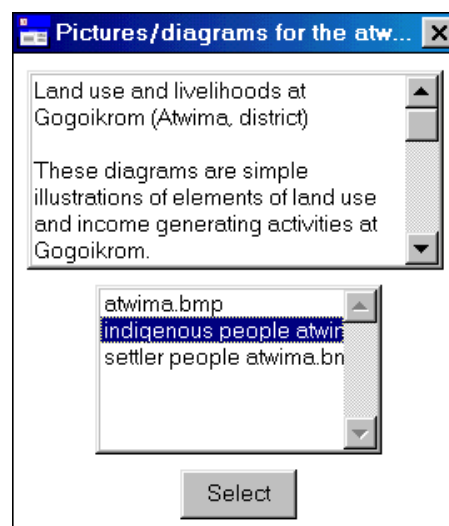


Figure 11.4 Selecting a diagram from the dialog box.

Figure 11.5 gives you the diagram¹ selected. There is also a memo field in which to explain the meaning of the diagram or picture, and its relevance to the knowledge base.

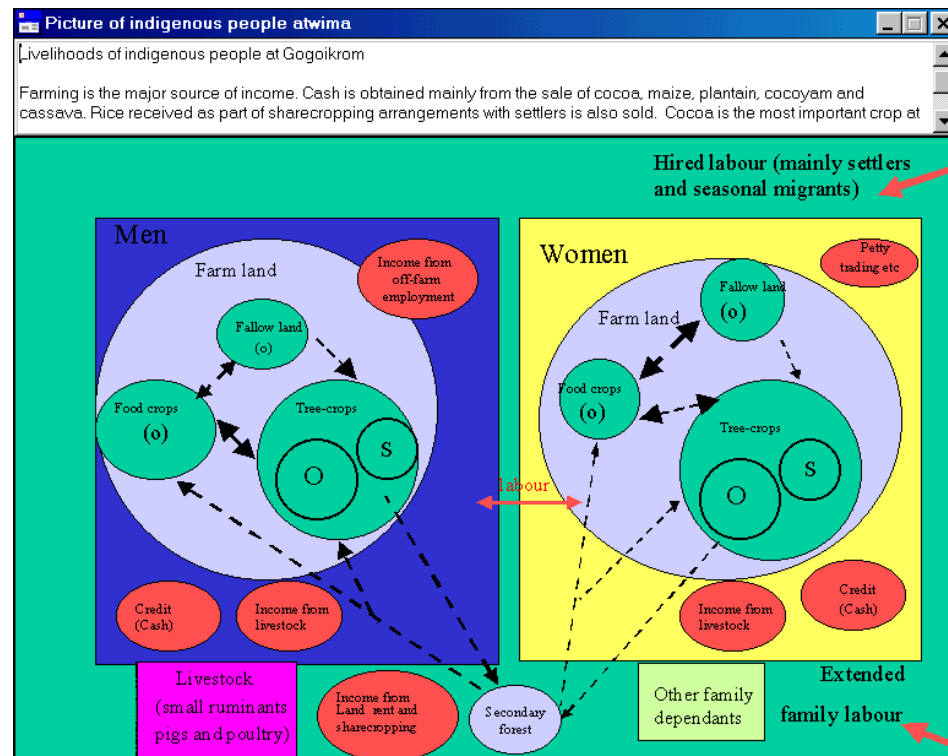


Figure 11.5 An example of a diagram stored in the knowledge base, with an explanation in the memo field

In order to incorporate diagrams and pictures into a knowledge base, you must adhere to the following procedures:

- i) All the diagrams and pictures that are to be incorporated into the knowledge base must be in the form of bitmap files.
- ii) All these bitmap files must be in the same folder as the knowledge base.
- iii) The thumbnail picture or diagram to go on the Welcome Page must have exactly the same name as the knowledge base - in the above example atwima.bmp for the knowledge base atwima.kb
- iv) All the other pictures and diagrams must contain the name of the knowledge base somewhere within the file name e.g. indigenous people atwima.bmp; settler people atwima.bmp
- v) These file names will then automatically appear when you select **Pictures/Diagrams** in the second part of the Welcome Page.

¹ The diagrams in this chapter are taken from: Obiri-Darko, B., Ayisi-Jatango, J., Anglaaere, L., Cobbina, J., Moss, C., McDonald, M., Sinclair, F., and Young, Einir., 2000. Livelihood systems and farmers ecological knowledge in Ghana: a report on three districts. Shortened Bush-fallow Rotations for Sustainable Livelihoods in Ghana (DFID Project R7446). School of Agricultural and Forest Sciences University of Wales, Bangor, U.K.

CHAPTER TWELVE – THE HELP FACILITY

The Help menu (Figure 12.1) offers you the following facilities:

- A summary of the AKT formal grammar.
- A list of all the tools available to AKT including any tools in the current User Tool file.
- A list of the publications on which this program is based.
- Information about the AKT version being used. (Quote for any problems or queries)

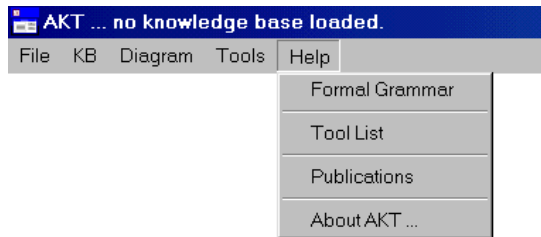


Figure 12.1 *The Help menu*

12.1 THE FORMAL GRAMMAR

The formal grammar in Figure 12.2 is exactly as in Table 4.1 in Chapter 4. It is available via the **Help** menu so that, when formalising unitary statements, you can quickly refer to it to see if the format of the unitary statement you are proposing is valid.

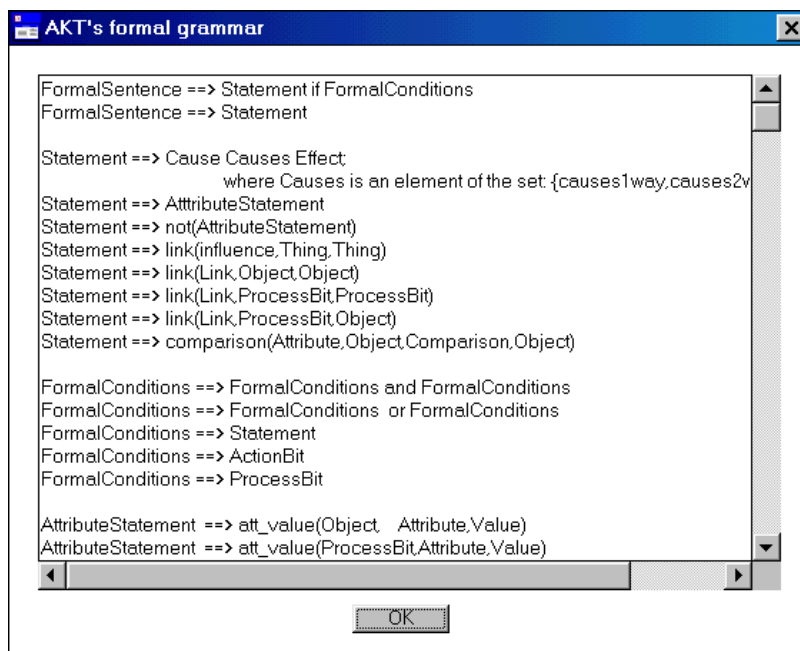
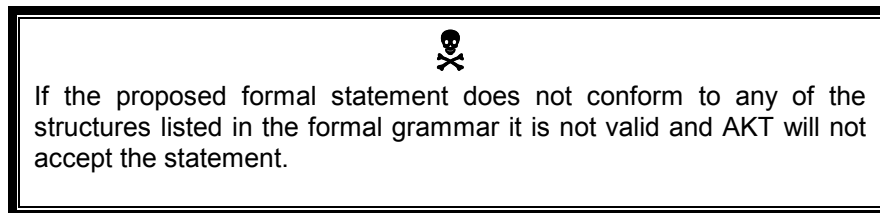


Figure 12.2 *The formal grammar as it appears in the Help menu*

12.2 THE TOOL LIST

The 'Tool Information' dialog box (Figure 12.3) is simply an alphabetically ordered list of the tools supplied with AKT plus any user tools if loaded. The list gives the name of the tool, the names of the parameters, the category to which it belongs and a description of its function.

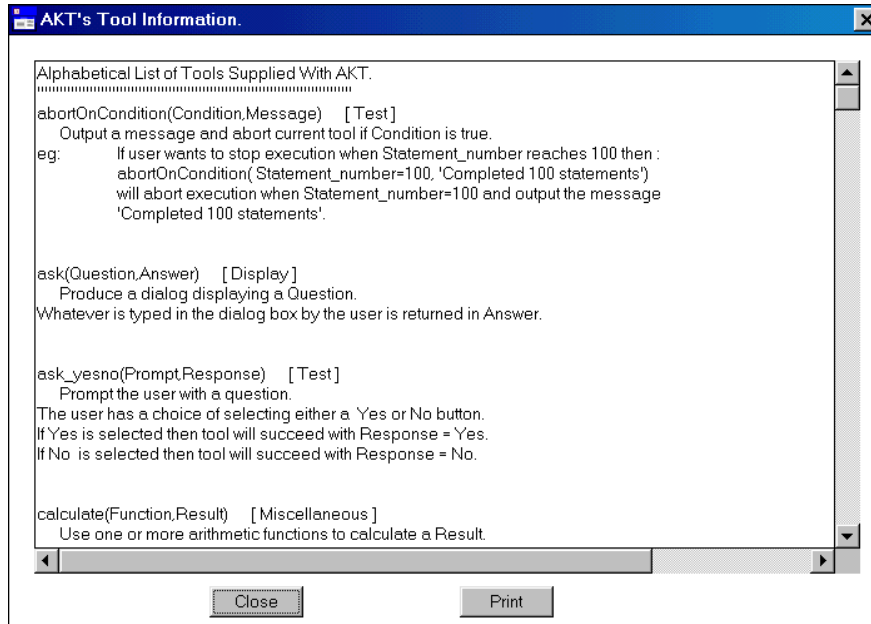


Figure 12.3 'Tool Information' accessed via the Help menu

Once you are familiar with writing tools it is of use as a short cut for copying primitives, control structures and systems tools and pasting them into your own tools. To do this, highlight the tool name, (but *not* the tool category or description) as demonstrated in Figure 11.3 with **attribute_values(Kb,Attribute,Values)** and press **Ctrl c**. You then return to the 'tool body' of the 'New Tool' dialog box and press **Ctrl v** to paste it in as described above in 'Method 2', Chapter 10.2.1.

12.2.1 SAVING 'TOOL INFORMATION' AND PRINTING

To save the whole 'Tool Information' list to a text file press **Print**. The following message appears (Figure 12.4). If you select **Yes**, you will be prompted for the name of a text file and the directory and folder in which to place it. If you select **No**, the 'Print' menu will appear in which you can select the orientation, number of copies, etc.

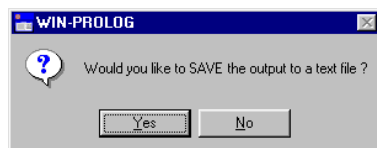


Figure 12.4 Message asking you whether you want to save the 'Tool Information' to a file

12.3 SOME POPULAR MISTAKES AND DIFFICULTIES

☹ *My knowledge base won't open properly.*

😊 *This is usually a problem with large knowledge bases. Check that you have increased the memory available to AKT by adding the correct parameters to the executable shortcut, as described on page iii, 'How to Install AKT' at the front of this manual.*

☹ *I get the error message '1 error(s) during compilation' when trying to load a knowledge base*

😊 *Is often happens when a knowledge base has been emailed to you. The reason for it is obscure, but it is easy to fix:*

Open Wordpad or any other text program. Open your knowledge base within that program. Scroll down to the very last line of the knowledge base. Make sure that the last line has a full stop at the end. Then press carriage return (↵). Save the file and then close it. Open AKT and try to open the knowledge base. It should now work.

☹ *My knowledge base opens but all my statements are missing.*

😊 *There are three possible reasons for this:*

- 1) You saved all your statements but did not save your knowledge base (using Save KB from the main KB menu). If you do not save your knowledge base at the end of a session, you will lose all the work you have done since you last saved the knowledge base properly.*
- 2) You have changed the name of your knowledge base outside the AKT program (using Windows Explorer for example). You should only change the name of your knowledge base from within the program by selecting **Save KB As** from the main **KB** menu. To get the statements back, revert to the old knowledge base name.*
- 3) You are trying to open a knowledge base in an earlier version of the program than it was created in. You can open a knowledge base created in an earlier version of the program in a later version of the program, but you can't do it the other way round, i.e. open an knowledge base created in a later version of the program in an earlier version.*

☹ *My knowledge base has jammed.*

😊 *This sometimes happens if you have too many windows open at once. To get out of a jam, select **Ctrl + Break**.*

☹ *In diagrams I cannot move some of the Actions nodes.*

😊 *This is a quirk within Prolog. To circumvent it, make sure that both the vertical and horizontal scroll bars are on zero. Select **Zoom Out**. When the action node you wish to move appears on the screen, click on it with the left mouse button and drag it. It will now move.*

☹ *When I load a knowledge base, I get lots of messages before the Welcome Page appears.*

😊 *When a knowledge base is opened, the program checks for duplicate statements and inferred duplicated statements (by checking the causes2way statements). Usually the messages are for your information only.*

☹ *When I load a knowledge base, I get the message 'Cannot generate an inferred statement from statement 42' (for example)*

😊 *This means that you have entered 'causes2way' where the statement cannot be a causes2way statement. You have to check that both parts of the statement are attribute value statements and both parts contain the values decrease or increase. If this is not the case, then you must change the statement to a causes1way statement.*

12.4 ABOUT AKT

Finally, the last option in the Help menu is a small dialog (Figure 12.5) containing the version number of AKT and the Web address where AKT files and information are available. There is also an acknowledgement of the people involved in the development of AKT.

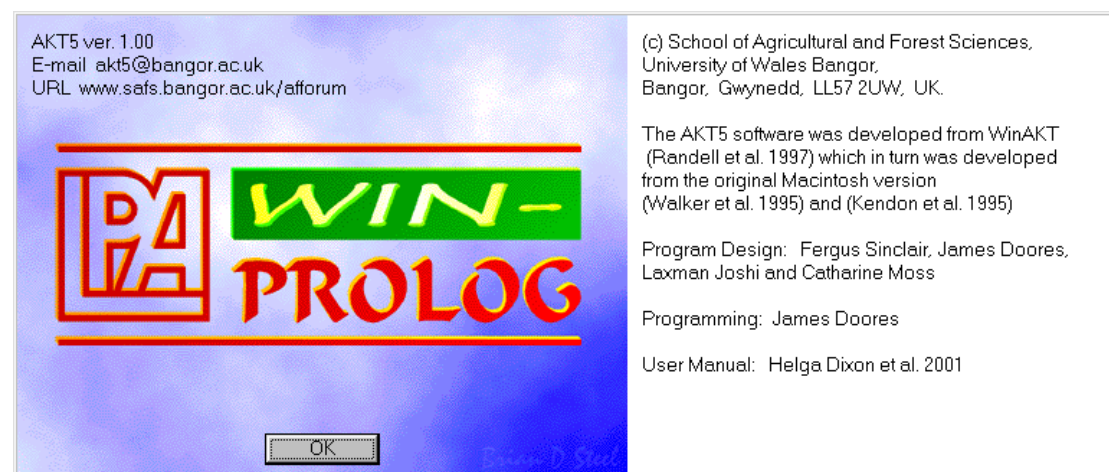


Figure 12.5 Details about the authorship of AKT

As a team, may we take this opportunity to wish you all possible success with AKT, and may it act as a useful and user-friendly support tool in your research and development work and your quest for local ecological knowledge. If you have any difficulties or queries, you can contact us directly at akt5@bangor.ac.uk.

CHAPTER THIRTEEN

A QUICK SIGHTSEEING TOUR AROUND AKT5

The purpose of this exercise is to let you have a go with a good knowledge base, and see what can be done with it. For this exercise we are using the Atwima knowledge base¹, a knowledge base created to explore farmers' knowledge on soil fertility management, soil classification, weed management and cocoa shade trees in the Atwima district of Ghana.

Getting started:

1. Load the AKT program (5.0) onto your computer (follow the instructions 'How to Install AKT 5' at the front of the manual).
2. Load the file atwima.kb onto your computer.
3. Open the AKT program.
4. Press **OK** on the first screen and the main menu will appear with the options at the top left hand corner.
5. Select **KB** → **Open Kb** → atwima.kb
6. Press **OK** at the various messages that appear until the Welcome dialog box appears.

Welcome Dialog Box

Read the Welcome dialog box, to get an idea what the knowledge base is about. Press **Further Details** for further details about the knowledge base. Press **Close** on each dialog box when you have finished reading the dialog boxes.

Press **Topics**. Then highlight in turn each topic hierarchy listed in the Topic Hierarchies dialog box. You will see in each new Topic Hierarchy dialog box that appears, all the topics which that particular topic hierarchy covers. For example, the topic hierarchy 'Fallows' contains the topics 'fallow length', 'fallows', 'fallows and soil types', and 'fallows on fertility'.

Question: what topics does the topic hierarchy 'Weeds' cover?

Press **Close** on both dialog boxes to return to the Welcome Memo and **Close** again to arrive at the main menu.

Sources

Go to the main menu and select **KB** → **Sources**. These are a list of all the sources interviewed for the knowledge base (No statement can be entered into the knowledge base, without the author of the statement being entered into the knowledge base first). Let us look at one of them. Highlight the name **Asmoah et al.** and press **Details**. A dialog box appears giving you the name of the interviewer, interviewee, sex of interviewee and date of interview. If you press **Memo**, it will give you any further details that the knowledge base creator felt to be important. Press **Close** on all three dialog boxes.

Topics

Return to the main menu and select **KB** → **Topics**. This gives you a list of all the topics in the knowledge base. Highlight '**Management actions**' and press **Details/Edit**. In the dialog box that appears you will see in the 'Boolean Search String' how the topic was created – it is a Boolean search string of all management actions – 'burning', 'clearing', 'cutting', 'harvesting', 'planting', 'uprooting', 'use', 'work'. If you press **Statements** at the bottom of the dialog box, a list of all the statements on management actions appear. There are 107 statements in all on that topic.

(You can try the same thing out with the topic '**Weed control**').

¹ Frost, W. 2000. Farmers' knowledge of soil fertility and weed management in Atwima district, Ghana. WinAKT Knowledge base. University of Wales, Bangor.

Question: How many statements are there on weed control?

As you can see 107 statements is an unwieldy number. To make better use of the knowledge we should look at it in smaller sections, which we will now do. Press **Close**, **Close** and **Close** again to get rid of all the dialog boxes.

Object Hierarchies

From the main menu select **KB** → **Object Hierarchies**. You now see before you a list of the object hierarchies in the knowledge base. (Object hierarchies are a way of sorting the knowledge, creating an indexing system of related objects.) Highlight **'tree'**. The Object Hierarchy dialog box for 'tree' appears. In the box on the left hand side, all the objects in the hierarchy are listed, on the right hand side you see the name of the object hierarchy and the immediately subobjects below.

Press **timber_tree** in the 'Objects in Hierarchy' list. You will see that it now appears in the 'Object' box with the super object (tree) above and the subobjects (odum, okoro, opam, wawa) below.

Press **View Tree**. This gives you the full hierarchy with all its objects and subobjects. Now press **Close** in both dialog boxes.

Formal Terms

Go to the main menu and select **KB** → **Formal Terms**. Press the downward arrow on 'Type' and see the different types of formal terms. Select **object**. The 'formal terms' list now gives a list of all the objects in the knowledge base. Scroll down and get an idea of the objects in the knowledge base. Highlight **nfofoa_kwae** and press **Details**. This gives you information about nfofoa_kwae which is the local name for secondary forest.

Press **Show Use in Hierarchies**. You will see that it appears in the object hierarchy land_types. Press **OK**.

Press **Show Use in Statements**. The 16 statements that appear are all the statements in the knowledge base that mention nfofoa_kwae. Under 'Diagram Selection Type' at the bottom of the dialog box press **All Statements**.

Introduction to diagrams

The diagram that you see before you will show you all the statements that can be represented diagrammatically, i.e. all causal statements.

Press the **Label Mode** button twice. This gives you the statements in full. You can make the statements more legible by dragging the nodes across the screen to separate them out.

Question: What are the immediate effects of burning nfofoa_kwae?

(If the script is too small, just press the **Statements** button to get a list of the statements then return to the diagram by pressing **Close** in the Statements dialog box).

Go to the main menu (top left hand corner) and select Diagram → Hide Diagrams. Maximize the Search Results dialog box again. Still exploring nfofoa_kwae, highlight statement 156 ('burning of nfofoa-kwae causes the trees log presence is positive') and press **Navigate** at the bottom of the dialog box. (The Navigate button gives you the immediate causes and effects of each node). A diagram will appear with the statements nodes highlighted. The immediate effect of this statement is unhighlighted (rain runoff rate). There are no immediate causes of this statement. Select the Navigate button on the right hand side of the diagram screen and click the double arrow that appears over 'rain runoff rate'. Carefully drag sideways all new

nodes to reveal any further nodes underneath (you do this by pressing the left hand mouse button over the node and dragging the node away). Select **Navigate** once more and click on 'vegetation debris presence'. Continue to build up the diagram by selecting **Navigate** each time and clicking on one of the new nodes.

Question: What, according to this diagram, does 'vegetation debris burning' affect?

Go to the main menu and select **Diagram** → **Hide Diagrams**

Boolean Search

Go to the main menu. Select **KB** → **Boolean Search**. Press on the downward arrow on 'Display KB terms of type' and see the different types available. Select **object**. Select 'asase_kokoo'. First press Details to see the term's synonym. Then press Close on the Formal Term Details dialog box.

Now press **Select** and 'asase_kokoo' will appear in the Boolean Search String at the bottom of the dialog box. Then press the **AND** button. Then select 'asase_tuntum' and press **Select** once more. (If you want to check the synonym for 'asase_tuntum', press **Details**.) Press **Search**. One statement will appear. It is the only statement in the knowledge base which includes both 'asase_kokoo' and 'asase_tuntum'.

In the Search Results dialog box press **Close**. In the Boolean Search dialog box press **Clear**. Now do the same thing again, selecting 'asase_kokoo' and 'asase_tuntum', only this time using **OR** instead of **AND**. Press **Search**.

Now you have 30 statements. This is because you have selected all the statements that include *either* 'asase_kokoo' or 'asase_tuntum'.

In the Search Results dialog box press **Close**. In the Boolean Search keep 'asase_kokoo' or 'asase_tuntum' in the Boolean Search String but this time press the radio button **superobject** in the 'Search Options' box so that it is highlighted in the same manner as **object**. Press **Search** once more. You will now have 157 statements because, besides statements using 'asase_kokoo' or 'asase_tuntum' you have also selected the statements related to the superobject of 'asase_kokoo' and 'asase_tuntum' ('soil').

Creating a topic

Staying in the Boolean Search menu, it is also possible to create topics through the Boolean Search String. Let us create a topic containing all the statements about red soils (asase_kokoo), all the statements about black soils (asase_tuntum) and all the statements about cocoa. Create a Boolean Search String with these three objects.

Question: should you put AND or OR in the search string?

Then in the box 'Name of new topic' enter 'soils_and_cocoa' and press **Save**. The topic details dialog box appears, showing the Boolean Search String that makes the topic and allowing you to write a description of the topic in the 'Description' box. Press **Save** once more (you should get a message saying that the topic has been saved).

Now go to the main menu and select **KB** → **Topics**. Select your new topic 'staple crops' and press **Select** and then **Search**. All the statements linked with that topic will appear. Press **All Statements** in the 'Diagram Selection Type' to get a visual idea of all the causal statements involved.

Closing a knowledge base and finishing off

Before you finish using the knowledge base you must save your knowledge base if you wish to save the changes you have made. Save is via the main menu, **KB** → **Save Kb** or, if you wish to keep both versions, **KB** → **Save KB as...**

When you have saved your knowledge base, close the knowledge base (**KB** → **Close KB**) and close AKT by going to the main menu **File** → **Exit AKT**.

ASSIGNMENT

Create two new topics exploring what women know about trees and what men know about trees. Compare the two knowledges and suggest reasons for the differences/similarities between them.

Big Hint:

Start off in the Boolean Search, selecting Sources. If you press Details for each source, you will see if the source is male or female. Link the sources together with OR, but link the 'tree' to the Boolean Search String with AND. (*Question: why is this?*). Remember to explore the subobjects of 'tree' as well.

CHAPTER FOURTEEN – TUTORIAL IN CREATING A SIMPLE KNOWLEDGE BASE

This tutorial guides you through the creation of a small knowledge base and then invites you to try out different ways of exploring it. The knowledge represented is abstracted from Bandy, D.E., Garrity, D.P. and Sanchez, P.A. (1993) The worldwide problem of slash-and-burn agriculture. *Agroforestry Today* Vol. 5 (3), 2-6. The tutorial is only an introduction to AKT; please use chapters 7 – 12 to guide you when you require further information on AKT facilities.

14.1 OPENING AKT

Open the AKT application. An introductory title page will appear. Press **OK**. A new screen will appear with a series of menus in the top left-hand section. Select **KB**. In the ensuing drop-down menu, select **New KB**, give your new knowledge base an appropriate name (e.g. *slash_and_burn*) and save it.

Now that you have an empty knowledge base prepared to receive information, we will address the text on which your knowledge base is to be based.

14.2 KNOWLEDGE FROM THE SOURCE

For the purposes of this tutorial, we will make use of the following text as our source knowledge. It may be more usual however, for source knowledge to be a tape-recorded interview. The following are extracts from the source text:

“...burning helps to control pests and diseases....the higher soil temperatures that follow clearing and burning also accelerate the decomposition of organic matter in the top layers of the soil.

About half of the nitrogen and phosphorus in the burnt material and nearly all the remaining nutrients are released to the soil from the ash after burning. These nutrients are flushed from the ash by the rain and have the effect of raising the pH of the upper layers of soil as well as incorporating the nutrients. Nutrients in concentrated form are thus available for one or two years after clearing. The quantity and quality of these nutrients depends on the native fertility of the soil...

Crops such as corn, rice, beans, cassava, yams and plantains are then planted....

As nutrients are removed by crop harvests or lost through leaching, soil fertility declines. At the same time, the relatively easily removed broad-leaved weeds are replaced by harder to manage grasses and increasing weed density quickly impedes further cropping. The fields are then abandoned for a period of fallow.

The secondary forest grows rapidly during the fallow, using nutrients remaining in the soil...Essential minerals (phosphorus, potassium, calcium, etc.) are extracted from lower soil layers during regrowth and brought to the surface by trees....

Soil erosion... is seldom a problem in shifting cultivation because the cleared areas are small and are always covered by some sort of vegetation. When unsustainable slash-and-burn is practised by newcomers, however, the soil is sometimes left uncovered. This can lead to major erosion problems, particularly in hilly areas.....

When the soil is bare and erosion a problem, the siltation rate of the waterways is increased and this often has a negative effect upon aquatic life forms and fish production.

Neither are weeds a difficulty in the traditional method because the land is left to fallow as soon as they become a problem. In the subsequent fallow period, weeds soon die out as the crown of the secondary forest closes. Migrant farmers, however, will remove all vegetation and the roots and stumps of the felled trees from the areas they clear for intensive farming.

These practices mean that when the land is finally abandoned there are no rootstocks left from which the trees may grow and grasses are the most common invaders of the open areas. This means that clearing of land by migrant farmers generally results in the permanent destruction of the rain forest."

The intention behind creating a knowledge base in AKT is to represent knowledge that provides a description of the ecology of a particular agroforestry practice. This is achieved by abstracting a set of unitary statements from the source knowledge that, as a set, represent a coherent description of the ecology of the practice. This process of breaking down the source knowledge into a set of unitary statements may demand considerable interpretation. Relying too literally on the exact statements in the source knowledge will tend to result in many statements in the knowledge base that become very difficult to interpret.

Consider, for example:

"In the subsequent fallow period, weeds soon die out as the crown of the secondary forest closes."

A literal abstraction of information might result in the statements that:

"Weeds soon die out in the fallow period"
"Weeds die out when the crown of the secondary forest closes"

Both statements are fairly literal abstractions and superficially reasonable and useful. However, as statements in a knowledge base they are problematic as both contain implicit information. The term 'soon' is relative to the rate of weed mortality on land, which is not fallowed. This is clear enough in the source knowledge but not as an explicit statement of ecological knowledge. Furthermore, in the second statement there is an implicit causality – however the statement only really identifies the coincidence of weed die back and canopy closure.

Interpretation might suggest that we should state instead that:

Forest canopy closure causes weeds to die back

and

Forest canopy closure occurs during the fallow period

Clearly this interpretation may require validation through further reference to the source knowledge. However, the knowledge expressed here is more fundamental and more useful than the above statements.

As a result of the interpretation involved, different sets of statements will be abstracted from the same source knowledge by different people with different objectives. It is important to realise that any abstraction of knowledge for representation in the knowledge base is open to debate. There can be no single correct abstraction, although there can certainly be incorrect

ones. The statements in Table 14.1 are an example of one set of statements abstracted from the text. These do not immediately and literally relate to the source knowledge but are designed to capture its ecological content. Some statements are speculative and demand validation while others can be expressed with more confidence.

Table 14.1 Example set of statements abstracted from the text

Burning causes a decrease in numbers of pests
Burning causes a decrease in crop disease levels
Clearing causes an increase in soil temperature
An increase in soil temperature causes an increase in the rate of decomposition of organic matter
Nutrient availability is high for two years after burning
Harvesting causes a reduction in soil nutrient levels
Leaching causes a reduction in soil nutrient levels
A reduction in soil nutrient levels causes a reduction in soil fertility
An increase in weed density causes a decrease in crop yield
Soil erosion is not a problem IF the soil is always covered with some type of vegetation
Soil erosion is severe IF soil is not covered with vegetation
Soil erosion is very severe on hills
An increase in soil erosion causes an increase in the siltation of waterways
An increase in the siltation of waterways reduces fish production
Weeds are not a problem IF land is regularly fallowed
Canopy closure causes the death of weeds

14.3 ENTERING KNOWLEDGE INTO THE KNOWLEDGE BASE

These statements can be entered into the knowledge base in two ways, through the **Show Kb Diagrams** interface in the main **Diagram** menu or through the **Statements** dialog box in the main **KB** menu.

14.3.1 ENTERING KNOWLEDGE THROUGH THE DIAGRAM INTERFACE

It is possible to enter all the statements in Table 14.1 into the knowledge base using the statement card. Alternatively, all the causal and link statements can be entered through the diagramming interface. We will start with the diagramming interface as this helps to ensure unambiguous connections between the individual statements (which is very important if a useful knowledge base is to result) and can be helpful to the beginner because statements entered through the diagramming interface are formalised automatically.

As an example of the use of the diagram interface we will enter the statements abstracted from the first paragraph of the text:

“...burning helps to control pests and diseases ... the higher soil temperatures that follow clearing and burning also accelerate the decomposition of organic matter in the top layers of the soil.”

Under the **Diagram** menu select **Show Kb Diagrams**. This will give you a blank diagram window with all the function buttons displayed on either side.

Entering a statement through the diagram interface demands some thought about the structure of that statement. For full details of the diagramming syntax see Chapter 4, section 4.4. The section on formal representation (Chapter 4, section 4.2) will also clarify the structures of statements supported.

The following five statements might be abstracted from this paragraph:

Table 14.2 Statements abstracted from the first paragraph:

- 1) Burning causes a decrease in pest numbers
- 2) Burning causes a decrease in disease levels
- 3) Clearing causes an increase in soil temperature
- 4) Burning causes an increase in soil temperature
- 5) An increase in soil temperature causes an increase in the rate of decomposition of organic matter

Each of these statements can be represented in the diagramming interface as a pair of nodes and a link between them. In the case of the first statement the two nodes are 'burning' –an action¹– and 'number of pests' – an attribute (number) of an object (pests). To represent this statement;

1. Select the **Action** button from the 'Add node' box on the left-hand side of the diagram screen.
2. Move your cursor onto the drawing area and click to create an action node. A dialog box will appear, asking for details of the action.
3. Enter the action name 'burning' and then look at the object boxes. (There are two options, 'One Object' or 'Two Objects'. In this case it is an action without a specified object but as all actions are accompanied by objects it is necessary to refer to the object being burnt. In this case 'site' or 'field' could be used).
4. Tick the 'One Object' box and enter 'site'.
5. Press **OK**. On the diagram screen a node appears bearing the legend 'site burning'.
6. Select the **Attribute** button from the 'Add node' box to create the second node. An 'attribute_relation_dialog' box appears. The attribute can relate to three things, objects, processes or actions. In this case it is related to an object (pests).
7. Tick the 'Object' box.
8. Press **OK**. An 'attribute_object_dialog' box then appears.
9. Specify 'number' as the attribute and 'pests' as the object.
10. Press **OK**.
11. Now select the **Link** button from the 'Add link' box.
12. Move the cursor to the 'site burning' node, left click and hold the mouse button down, drag the cursor to the 'pests number' node and release the mouse button.
13. A 'Choose Link' dialog box now appears. In this case for 'Link type' choose 'causes1way' and for 'Effect Value' in the 'Parameters' box choose 'decrease'.
14. Press **OK**. The 'Information Source' dialog box now appears.....

14.3.1.a Entering the Information Source

No statement can be accepted by AKT without a source attached, so here goes...

1. In the 'New' box, select the **Reference** radio button and then press **New**. The dialog box 'Create a new reference source' appears.
2. Fill in the details of the authors, the name of the journal title and the name of the article as given at the beginning of this chapter².
3. When you have completed filling in the details of the dialog box, press **Save**.
4. The statements sources dialog box reappears with the source highlighted. Press **OK**.

¹ 'Burning' can be either a process or an action. If it is used as a management tool, it is an action, otherwise it is a process. In any knowledge base it can only be used as one or the other, it cannot be entered as both action and process. If some statements also refer to burning brought about by natural causes then another term must be found, either 'fire' as an object, or 'conflagration' as a process.

² In the case of an article, the name of the journal should go in the 'Name' box.

5. The screen now automatically brings up the 'New Statement' dialog box, giving both the formal language representation and the natural language representation of the nodes and link created in the diagram interface. Press **Save**.
6. The following message will then appear: "Are you sure you want to save this statement without a condition?" In this case the answer is 'yes' so press **Yes**.
7. Dialog boxes will then appear, asking if you wish to create new formal terms for the various terms just entered. Press **Yes** on each occasion.

The screen then reverts to the diagram interface and displays the two nodes connected by a link.

Try entering the other four statements via the **Diagram** interface. Note that 'clearing', like 'burning', is an action and therefore needs an object. In this case as well, 'site' would be a valid object.

After entering the first 5 statements, and pressing **Label Mode** on the right hand side of the screen twice to get the full statements, you should end up with a diagram something like Figure 14.1.

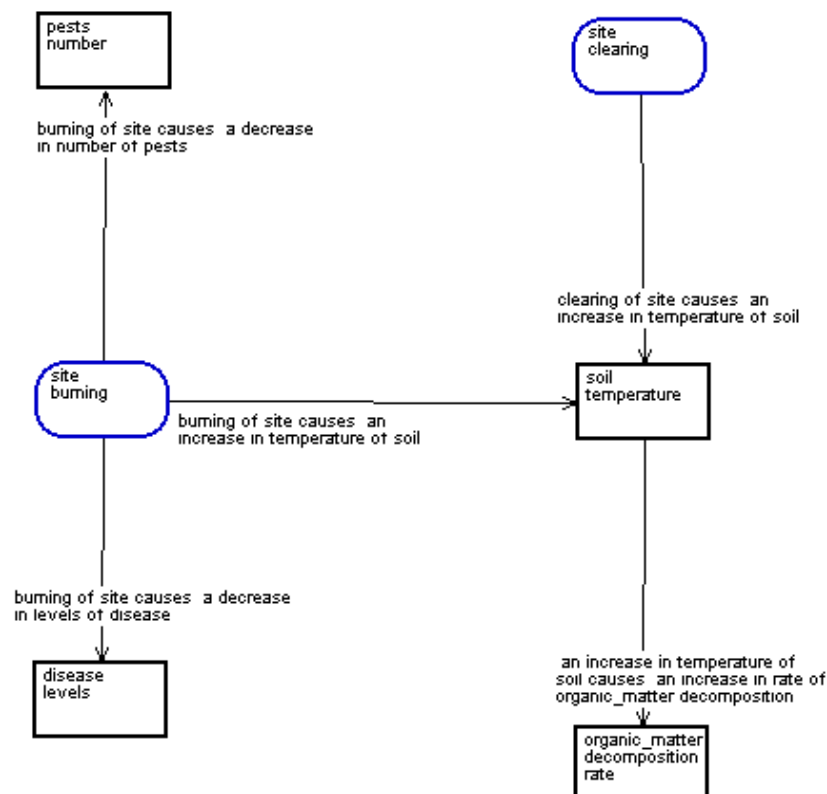


Figure 14.1 Diagram constructed from the statements in Table 12.2

This example illustrates one of the most important features of the diagramming approach: the way that it helps you to identify ambiguity in the knowledge being represented. While the five statements individually appear reasonably intuitive, their combination into a diagram suggests that important information is missing. Burning will certainly raise soil temperatures but it is likely that the warming of the soil referred to in the text is not actually a direct consequence of burning but of the removal of vegetation as a result of burning and the consequent increase in insolation. Modification of the above diagram may better capture this interpreted meaning (see Figure 14.2 below).

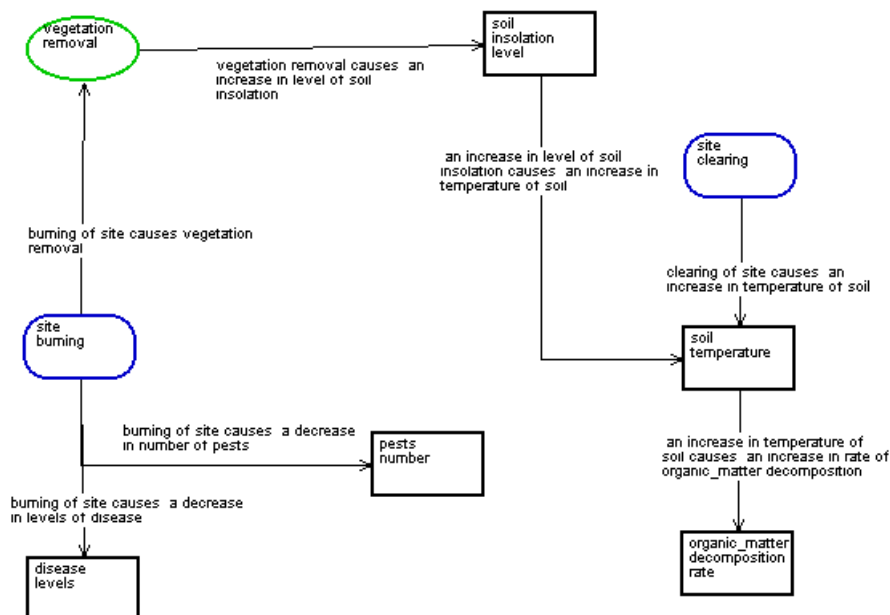


Figure 14.2 Figure 14.1 modified to better capture the interpreted meaning

14.3.2 ENTERING KNOWLEDGE THROUGH THE STATEMENT CARD

Although statements entered through the diagram interface are formalised automatically, any statements entered through the statement card must be formalised manually. The process of formal representation requires some practice. Read through Chapter 4 (part 4.2), which takes you through formal representation step by step, then try to formalise the extracted statements given in Table 14.1. Table 14.3 gives a formal version for each of the statements in Table 14.1. There are no definitive, correct answers in formal representation, although there are certainly incorrect ones.



NOTE: Remember to use the underscore (_), not a hyphen (-) when writing formalised statements, e.g. att_value

Table 14.3 Formal versions for each of the extracted statements in Table 14.1

	Extracted Statements	Formalised Statements
1	Burning causes a decrease in numbers of pests	action(burning,site) causes1way att_value(pests,numbers,decrease)
2	Burning causes a decrease in crop disease level	action(burning,site) causes1way att_value(crops_disease, level, decrease)
3	Clearing causes an increase in soil temperature	action(clearing, site) causes1way att_value(soil,temperature,increase)
4	An increase in soil temperature causes an increase in the rate of decomposition of organic matter	att_value(soil,temperature,increase) causes2way att_value(process(organic_matter, decomposition),rate, increase)
5	Nutrient availability is high for two years after burning	att_value(site, nutrient_availability, high) IF att_value(site, time_since_burning,range('0 years', '2 years'))

6	Harvesting causes a reduction in soil nutrient levels	action(harvesting, crops) causes1way att_value(part(soil, nutrients),level,decrease)
7	Leaching causes a reduction in soil nutrient levels	process(leaching) causes1way att_value(part(soil, nutrients),level, decrease)
8	A reduction in soil nutrient levels causes a reduction in soil fertility	att_value(part(soil, nutrients),level, decrease) causes2way att_value(soil, fertility, decrease)
9	An increase in weed density causes a decrease in crop yield	att_value(weeds, density, increase) causes2way att_value(crops, yield, decrease)
10	Soil erosion is minimal IF the soil is always covered with some type of vegetation	att_value(process(soil, erosion), rate,minimal) IF link(covers, vegetation, soil)
11	Soil erosion is severe IF soil is not covered with vegetation	att_value(process(soil, erosion), rate, high) IF link (not_covers, vegetation, soil)
12	Soil erosion is very severe on hills	att_value(process(soil, erosion),rate, severe) IF att_value(process(soil, erosion), location, hillside) and att_value(site, vegetation_cover, bare)
13	An increase in soil erosion causes an increase in the siltation of waterways	att_value(process(soil, erosion), rate, increase) causes2way att_value(process(waterways, siltation), rate, increase)
14	An increase in the siltation of waterways reduces fish production	att_value(process(waterways, siltation),rate, increase) causes2way att_value(action(production,fish), rate, decrease)
15	The weed population does not become a problem when land is regularly left to fallow.	att_value(action(fallowing,land),frequency,regular) causes1way att_value(weeds, population,decrease)
16	Canopy closure causes the death of weeds	process(canopy,closure) causes1way process(weeds, death)

Select **Statements** from the main **KB** menu. Then press **New**. Enter the formalised version into the 'Formal Language Statement' box. Where there are conditions in the statement (as in statements 5, 10, 11, 12, and 15) the condition is entered in the second half of the box, headed 'IF'. Press **Syntax Check** to check whether or not the syntax has been properly entered. Then press **Translate** to get a natural language version. When you are satisfied, press **Save**.



Warning!

If you press **Close**, instead of **Save** you will lose the statement and it will have to be entered again.

14.4 PLAYING WITH THE KNOWLEDGE BASE

Now that you have created a mini knowledge base, the following section gives you suggestions on ways in which to enhance its usefulness and to explore it. In each section are numbers, underlined and in brackets. These are the chapters and paragraphs to which you should refer for instructions on how to carry out the tasks.

✂ Defining formal terms (7.4.3.a)

You should always take care to define your formal terms clearly. Even simple terms should be defined as two people may understand something quite different under the same term. For example the action 'clearing' could mean clearing of undergrowth, clearing of weeds or complete clearing of a site. Let us assume that in this case 'clearing' means 'the complete removal of vegetation from the site':

1. Go to the main **KB** menu.
2. Select **Formal Terms**.
3. In the 'Type' box select 'action'.
4. Highlight *clearing* and press **Details**. You can now enter the definition into the definition box.
5. Press **Save**.

✂ Creating an object hierarchy (7.5.1)

You should wherever possible, structure the objects to identify any hierarchical relationships between them. The creation of object hierarchies allows you to develop a compact knowledge base because they allow knowledge to be recorded at its most general level of application, yet be used to consider more specific instances.

Two of the objects in the knowledge base you are creating here **vegetation** and **crops** clearly can be hierarchically associated. Furthermore, it is obvious from the source text that beans, cassava, corn, plantains, rice and yams are types of crop.

1. Select **Object Hierarchies** in the main **KB** menu.
2. Select **New**.
3. In the 'New Hierarchy Name' box, enter a name for the object hierarchy. You can either enter a name of your own choosing, (in this case we have called the hierarchy 'plants'), or select an object from the 'KB objects' list to act as the hierarchy name.
4. Press **Save**. The hierarchy name 'plants' now appears in the 'Object Hierarchies' list in the 'Object Hierarchies' dialog box.

✂ Appending objects to the object hierarchy

1. Highlight the name of your new Object Hierarchy.
2. In the 'Selected Object' box press **Append to**. A list of all the objects in the knowledge base then appears.
3. Select 'vegetation'.
4. Press **Append**. The formal term 'vegetation' then appears in the object hierarchy 'plants' as an immediate sub-object of 'plants'.

✂ Building up object hierarchies (7.5.3.)

Add the following statements to your knowledge base

Beans fix nitrogen
Corn does not fix nitrogen
Rice does not fix nitrogen

Now practise creating another level in the 'plants' object hierarchy:

1. Highlight **vegetation**, so that it appears in the 'Object' box of the 'Hierarchy Structure' section.
2. Press **Append To**.
3. In the 'Append Object' dialog box, select 'crops' from the 'Objects' list and press **Append**.
4. You now have an object hierarchy in which the formal term 'vegetation' has the parent 'plants' and the child 'crops'.

5. Try incorporating 'beans', 'corn' and 'rice' in the object hierarchy 'plants' as subobjects of 'crops' by highlighting 'crops' so that it appears in the 'Object' box of the 'Hierarchy Structure' section and then pressing **Append To**.
6. Continue as above.

The principal advantage of the hierarchy you have just created is that it can now be recognised during reasoning that because:

Soil erosion is not a problem if the soil is always covered with some type of vegetation

and

Rice is a type of vegetation

(information from the hierarchy) then:

soil erosion on a site covered in rice should not be a problem.

Conducting an Boolean Search (7.10.1)

A knowledge base may often contain many hundreds of statements (see for example, the knowledge base 'treefodd'). Consequently, exploring the contents of a large knowledge base through **Statements** in the main **KB** menu, or even via the **Diagram** interface becomes very demanding and confusing. However, AKT contains a **Boolean Search** facility that allows you to retrieve sets of statements that correspond to particular criteria.

Practise searching on an object using the Boolean Search:

1. Select **Boolean Search** from the main **KB** menu.
2. From the 'Display Kb terms of type' drop-down menu select 'object'.
3. From the list of objects displayed, highlight *crops* and press **Select**.
4. The term *crops* will appear in the 'Boolean Search String' at the bottom of the dialog box. Press **Search**.
5. A 'Search Results' dialog box then appears, giving all the statements retrieved containing the formal term 'crops'.

Boolean searches using and and/or or (7.10.1.a)

Boolean searches can include combinations of formal terms linked with **and** and/or **or**. Try it out on the formal terms *pests* and *disease*

6. Select **Boolean Search** from the main **KB** menu.
7. From the 'Display Kb terms of type' drop-down menu select 'object'.
8. From the list of objects displayed, highlight *pests* and press **Select**.
9. The term *pests* will appear in the 'Boolean Search String' at the bottom of the dialog box.
10. Press the **OR** button from the 'Boolean Options' box.
11. Now find the term 'disease', highlight it and press **Select**.
12. In the 'Boolean Search String' at the bottom of the dialog box should be 'pests or disease'. Now press **Search**.
13. You will be presented with all the statements in your knowledge base containing either the term 'disease' or the term 'pests'.
14. If you repeat this process but select the term **AND** instead of **OR**, no statements will be found as there are no statements in the knowledge base which contain both the term 'disease' and the term 'pests'.

When undertaking searches combining 'and' and 'or' it is necessary to use brackets as one does in algebra to give the order of precedence in the search. Experiment with 'burning' and 'pests' or 'burning' and 'disease'. To get all the statements mentioning either 'burning' and 'pests' or 'burning' and 'disease' the Boolean Search String should include brackets, thus:

(burning and pests) or (burning and disease)

The searches which we have practised so far have retrieved only statements that contain the specified formal term. However, because we have additional information about object terms in the form of the hierarchies, we can use alternative search strategies.

In the 'Search Options' box, it is possible to select:

- 1) 'object',
- 2) 'superobject'
- 3) 'subobjects'

in any combination.

Now try out all the different search options, one at a time. The first option will give you only the statements containing the word 'crops'. The second option will give you all the statements containing 'crops' or any of the subobjects, 'rice', 'beans' or 'corn'. The third option will give you all the statements containing any of the objects in the object hierarchy 'plants'.

Creating Topics (7.11.1)

You may wish to save a particular Boolean Search String to create a permanent subset of statements in your knowledge base. These subsets of knowledge are called **Topics** and are created in the same way as any Boolean Search String, the difference being that they are then saved as a topic.

Try creating a topic entitled 'management actions', from a Boolean Search String containing the terms 'burning' **or** 'clearing' **or** 'harvesting'.

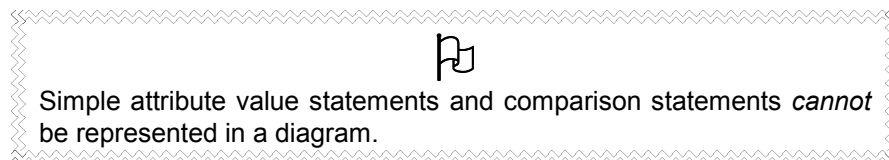
1. Create the Boolean Search String.
2. In the box marked 'Name of new topic' type in *management actions*.
3. Now press **Save** in the 'Save search selection' box.
4. In the Topic Details box press **Save** once more.

Go to **Topics** via the main **KB** menu and select *management actions*. When the Search Results appear, try out the different buttons on the 'Diagram Selection Type' at the bottom of the dialog box **(7.10.1.c)**

DIAGRAM GENERATION (8)

Diagramming was used earlier in the tutorial as a means of creating a knowledge base. You can also use diagramming as a means of exploring the content of a knowledge base.

1. Open the diagram interface (by selecting the **Diagram** from the main menu).
2. Try to build up sub diagrams by using the **Show Paths (8.3.1)**, **Navigate (8.3.2.)** and the **Cause and Effect (8.3.3.)** buttons.
3. Move between the different diagrams **(8.5)**
4. Give each diagram a different label **(8.6.1.)**.



Printing diagrams (8.11)

You may want to print a diagram to illustrate a point – just press **Print Window** on the right hand side of the diagram screen. Or if you want to transfer it to another package before printing it out, press **Copy to Clipboard (8.6.3)**

✂ Printing Statements (7.13)

Often it may be useful to print out some of the content of the knowledge base in order to help you in its exploration. Or you may wish to save your statements to another package and print them out from there. Try printing out all the statements in your Topic 'management actions';

1. Select the main **KB** menu.
2. Select **Topics**.
3. Select 'management actions'.
4. Press **Search**.
5. Print the **Print Statements** button at the bottom of the 'Search Results' dialog box.
6. Try printing the statements directly and try saving them as a text file as well.

✂ USING TOOLS (9 and 10)

As this is a tutorial for beginners and the knowledge base 'slash and burn' is only very small, you would not normally need the tools at this stage. However, just to give you a little taster of what tools can do, here are two for you to try out.

The following is a little exercise to introduce you to some of the tools and what they do. To get started select the main **Tools** menu and then press **Tools, Systems Tools** and finally **Knowledge Evaluation**.

1. The tool **kb_report(Kb)** gives you a summary of the categories of statements in your knowledge base. Press **Run**. A dialog box appears, requesting the name of the knowledge base. Enter the name of your knowledge base and press **Continue**. The tool output will give you the following output;

Total number of statements	22 of which 5 are conditional.
Number of attribute statements	5 of which 5 are conditional.
Number of causal statements	14 of which 0 are conditional.
Number of comparison statements	0 of which 0 are conditional.
Number of link statements	3 of which 0 are conditional.

Remaining within the **Systems Tools** category, go to **Knowledge Exploration**.

2. The tool **species_report/2** extracts information from the knowledge base about a particular species³. Press **Run**. An input parameter dialog box appears requesting the name of the knowledge base and of the species to be extracted. Enter the name of your knowledge base and enter 'crops' in the 'species' box. Then press **Continue**. You should get the following output;

Kb = slash_and_burn

Species= crops

INFORMATION SHEET FOR : crops

HIERARCHY : plants
vegetation
crops

ATTRIBUTE STATEMENTS :
none

CAUSAL STATEMENTS :
harvesting of crops causes a decrease in level of soil nutrients

COMPARISON STATEMENTS :
none

LINK STATEMENTS :

³ Species in this context can mean any object in the knowledge base.

none

CONDITIONAL STATEMENTS :

none

DERIVED STATEMENTS :

the erosion of soil rate is minimal if crops covers soil

the rate of erosion of soil is high if crops not_covers soil

burning of site causes removal of crops

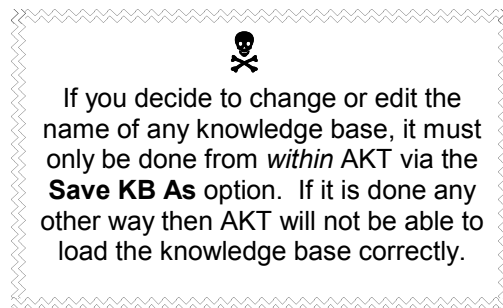
removal of crops causes an increase in level of insolation of soil

This tool not only tells you all about the selected 'species' (in this case 'crops') but it also gives you all the statements derived via the object hierarchy as well.

CLOSING YOUR KNOWLEDGE BASE (7.14)

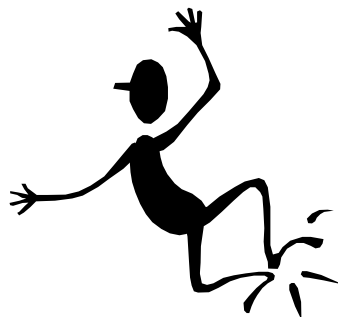
Now has come the time to close your knowledge base. Before closing it, you must decide if you want to save the changes you have made to it or not.

If you **do** wish to save the changes, then select **Save KB** or **Save KB As** from the main **KB** menu. Once you have saved the knowledge base, select **Close KB** from the main **KB** menu.



If you **do not** wish to save the changes you have made, then go straight to **Close KB**. Once you have closed the knowledge base go to the main **File** menu and click on **Exit AKT**.

YOUR TUTORIAL IS AT AN END



**THE CHALLENGE IS NOW TO MAKE A KNOWLEDGE BASE
OF YOUR OWN!**

APPENDIX 1

GLOSSARY OF TERMS

AKT – Agroecological Knowledge Toolkit – the name of the methodology and software.

Artificial intelligence: A field of computer science in which machines are developed and programmed to act as a human brain.

Automated reasoning: Where a machine (computer) systematically processes statements using rules of inference.

Boolean search: A search mechanism for retrieving statements containing particular keywords. Any combination of keywords may be defined using 'and' and 'or' operators.

Consultant: Those from the local community co-operating with the researcher building up the knowledge base.

Control structure: Program segments within AKT which control when and upon what knowledge the primitives are used.

Data: A set of observations, may be quantitative or qualitative.

Domain: Subject, or topic area under consideration.

Ecological knowledge: Knowledge about organisms, interactions amongst organisms and between them and their environment.

Explanatory knowledge: Knowledge providing an explanation of phenomena.

Emic: Internally generated explanation of phenomena (i.e. from within the affected community)

Etic: Externally generated explanation of phenomena (i.e. from outside the affected community)

Expert: The local farmer recognised as an authority in a particular aspect of the local farming system.

Formal Term: The key components that make up the unitary statements in the formal language. All words entered into the unitary statements that do not belong to the formal grammar are formal terms.

Generic: Describes something applicable or referring to a whole class or group.

Hierarchy: Classification of objects whereby all objects under a higher level object share all its attributes.

Indigenous Knowledge: Local knowledge which has been modified by the incorporation of cultural/religious beliefs and values.

Informant: The interviewee from the local community who has supplied a particular piece of knowledge.

Inheritance: Objects in a classificatory hierarchy sharing properties of higher level objects.

Input parameter: A parameter where the primitive or tool expects the parameter to be specified before starting the tool.

Instantiate: To assign a value to a variable.

Iterative: Repetitive

Knowledge: The outcome, independently of interpreter, of the interpretation of data/information.

Knowledge acquisition: Process of knowledge elicitation and representation

Knowledge base: An articulated and defined set of knowledge stored on a computer which can be accessed and processed systematically.

Knowledge-based system: The combination of a knowledge base/set of knowledge bases and knowledge-based system software.

Knowledge elicitation: Articulating knowledge for the purpose of abstraction and representation in a knowledge base.

Link: In a diagram interface the link links two nodes together. In causal statements the link is either **causes1way** or **causes2way**. In link statements the link is defined by the user.

Local Knowledge: Knowledge based on locally derived understanding, formed by experience and observation.

Node: These appear in the diagram interface. A node represents either an object, a process, an action or an attribute of an object, process or action. There are two nodes in a causal statement, linked by the reserved terms **causes1way** or **causes2way**. In a link statement the two nodes are linked by a user defined link.

Output parameter: A parameter where the primitive or tool determines the value and returns this value as the output at completion of the tool.

Parsimonious: Restricting formal terms used in the knowledge base to the minimum possible, without losing the meaning of the knowledge captured.

Primitive: Small program segments within AKT employed for running a tool.

Prolog (WinProlog): An artificial intelligence programming language used for developing AKT software.

Scientific knowledge: Knowledge generated in a formal manner by universities, research and other institutions.

Task language: A dedicated procedural programming environment in AKT software which allows users to build tools or programmes using primitives and control structures to interrogate a knowledge base.

Tool: A small computer program, either supplied with AKT software or developed by the user using primitives, control structures and pre-existing tools for interrogation and reasoning with a knowledge base.

Tractable: Easily managed, malleable

Unitary Statement: The smallest useful unit of knowledge which can stand alone without reference to other statements to be comprehensible.

WinAKT: Windows version of the Agroforestry Knowledge Toolkit (the former name of AKT)

BIBLIOGRAPHY

- Bandy, D.E., Garrity, D.P. & Sanchez, P.A. (1993) The worldwide problem of slash-and-burn agriculture. *Agroforestry Today* **5**(3): 2-6
- Bell, J., Hardiman, R.J. (1989) The Third Role – the naturalistic knowledge engineer. *Knowledge elicitation: Principles, techniques and applications* (ed Diaper, D.) Ellis Horwood Ltd, Chichester, UK.
- Benfer, R.A. & Furbee, L. (1990) Knowledge acquisition: lessons from anthropology. *AI Applications in Resource Management*, **4**(3): 19-26
- Brandt, C.J. (1989) The size distribution of throughfall drops under vegetation canopies. *Catena* **16**: 507.
- Breuker, J.A. & Wielinga, B.J. (1987) Use of models in the interpretation of verbal data. *Knowledge acquisition for expert systems: a practical handbook* (ed Kidd, A.) Plenum Press, New York
- Brokensha, D., Warren, D.M., & Werner, O. (eds) (1980) *African rural development: Indigenous knowledge systems and development*. University Press of America
- Chambers, R., Pacey, A. & Thrupp, L.A. (eds) (1989) *Farmer First: Farmer Innovation and Agricultural Research*. Intermediate Technology Publications, London.
- Conway, G.R. (1989) Diagrams for farmers. *Farmer First: Farmer Innovation and Agricultural Research* (eds Chambers, R., Pacey, A. & Thrupp, L.A.) Intermediate Technology Publications, London.
- Cooke, N.J. (1994) Varieties of knowledge elicitation techniques. *International Journal of Human Computer Studies* **41**: 810-849
- Cordingley, E. & Betsy, S. (1989) Knowledge elicitation techniques for knowledge based systems. *Knowledge elicitation: Principles, techniques and applications* (ed Diaper, D.). Ellis Horwood Ltd, Chichester, UK.
- Diaper, D. (ed) (1989) *Knowledge elicitation: Principles, techniques and applications*. Ellis Horwood Ltd, Chichester, UK.
- Frost, W. (2000). Farmers' knowledge of soil fertility and weed management in Atwima district, Ghana: The implications for participatory technology development. Unpublished MSc. Thesis. University of Wales, Bangor. 91 pp.
- Gupta, A.K. & IDS workshop (1989) Maps drawn by farmers and extensionists. *Farmer First: Farmer Innovation and Agricultural Research* (eds Chambers, R., Pacey, A. & Thrupp, L.A.) Intermediate Technology Publications, London.
- Haggith, M., Robertson, D., Walker, D.H., Sinclair, F.L. & Muetzelfeldt, R.I. (1992). TEAK: 'Tools for eliciting agroforestry knowledge'. *Proceedings of the British Computer Society Symposium on IT-enabled Change in Developing Countries*. London, July, 1992.
- Hall, R.L. & Calder, I.R., (1993) Drop size modification by forest canopies: measurements using a disdrometer. *Journal of Geophysical Research*, **98**(18): 465
- Hart, A. (1986) *Knowledge acquisition for expert systems*. Kogan Page, London.

- Johnson, N.E. (1989) Mediating representations in knowledge elicitation. *Knowledge elicitation: Principles, techniques and applications* (ed Diaper, D.). Ellis Horwood Ltd, Chichester, UK.
- Joshi, L. (1998) Incorporating farmers' knowledge in the planning of interdisciplinary research and extension, PhD thesis, University of Wales, Bangor, UK.
- Joshi, L. & Devkota, N. (1996) Effect of *Ficus auriculata* on maize and millet crops at mid-altitude *bari*. *PAC Working Paper 153*. Pakhribas Agricultural Centre, Dhankuta, Nepal.
- Knight, G.G. (1980) Ethnoscience and the African farmer: rational and strategy. *African rural development: Indigenous Knowledge Systems and Development* (eds Brokensha, D., Warren, D.M. & Werner, O.). University Press of America.
- Lightfoot, C., De Guia, O., Aliman, A. & Ocado, F. (1989) Systems diagrams to help farmers decide in on-farm research. *Farmer First: Farmer Innovation and Agricultural Research* (eds Chambers, R., Pacey, A. & Thrupp, L.A.) Intermediate Technology Publications, London.
- Lundgren, B.O. (1987) Institutional aspects of agroforestry research and development. *Agroforestry: a decade of development* (eds Stepler, H.A. & Nair, P.K.R.). ICRAF, Nairobi.
- Paudel, K.C., Pandit, R., Amatya, L.K., Gurung, D.B., Bhandari, H.S., Harding, A.H. & Bhattarai, S.P. (1997). *Agroforestry Research for Lumle Agricultural Research Centre, 1997 – 2001*, 37.
- Richards, P. (1980) Community environmental knowledge.) *African rural development: Indigenous knowledge systems and development* (eds Brokensha, D., Warren, D.M. & Werner, O.). University Press of America.
- Robertson, D.H., Bundy, A., Muetzelfeldt, R., Haggith, M. & Uschold, M. (1991). *Eco-logic. Logic-based approaches to ecological modelling*. MIT Press, Cambridge, Massachusetts.
- Sinclair, F.L., Walker, D.H., Joshi, L., Ambrose, B., & Thapa, B. (1993) Use of a knowledge based systems approach in the improvement of tree fodder resources on farmland in the eastern hills of Nepal. School of Agricultural and Forest Sciences, University of Wales, Bangor.
- Sinclair, F.L. & Walker, D.H. (1999) A utilitarian approach to the incorporation of local knowledge in agroforestry research and extension. *Agroforestry in Sustainable Agricultural Systems* (eds Buck, L.E., Lassoie, J.P. and Fernandes, E.C.M.) pp 245-275. Lewis Publisher, New York.
- Southern, A.J. (1994) Acquisition of indigenous ecological knowledge about forest gardens in Kandy district of Sri Lanka. MSc Thesis, University of Wales, Bangor, UK.
- Thapa, B. (1994) Farmers' ecological knowledge about the management and use of farmland tree fodder resources in the mid-hills of eastern Nepal, PhD thesis, University of Wales, Bangor, UK.
- Thapa, B., Sinclair, F.L. & Walker, D.H. (1995) Incorporation of indigenous knowledge and perspectives in agroforestry development. Part 2: Case-study on the impact of explicit representation of farmers' knowledge. *Agroforestry Systems* **30**:249
- Thapa, B., Walker, D.H. & Sinclair, F.L. (1997) Indigenous knowledge of the feeding value of tree fodder. *Animal Feed Science Technology* **67**: 97-114
- Thorne, P.J., Sinclair, F.L. & Walker, D.H. (1997) Using local knowledge of the feeding value of tree fodder to predict outcomes of different supplementation strategies. *Agroforestry Forum* **8** (2): 45

- Thornes, J. (1989) Solutions to soil erosion. *New Scientist* (3 June), 45.
- Walker, D.H., Sinclair, F.L., Joshi, L. & Ambrose, B. (1997) Prospects for the use of corporate knowledge bases in the generation, management and communication of knowledge at a frontline agricultural research centre. *Agricultural Systems* **54**:291-312
- Walker, D.H., Sinclair, F.L. & Muetzelfeldt, R.I. (1991) Formal representation and use of indigenous knowledge about agroforestry : pilot phase report, June 1991. School of Agricultural and Forest Sciences, University of Wales, Bangor.
- Warren, D.M. (ed) (1991) Indigenous agricultural knowledge systems and development. *Agriculture and human values* **VIII** (1&2)
- Warren, D.M., Slikkerveer, L.J., Titilola, S.O. (eds) (1989) Indigenous knowledge systems: implications for agriculture and international development. *Studies in Technology and Social Change, Technology and Social Change Program*, No. 11. Iowa State University.
- Warren, D.M., Slikkerveer, L.J. & Brokensha, D. (eds) (1995) *The cultural dimension of development*. Intermediate Technology Publications, London.
- Werner, O. & Schoepfle, G.M. (1987a) *Systematic Fieldwork Volume 1: Foundations of Ethnography and Interviewing*. Sage Publications.
- Werner, O. & Schoepfle, G.M. (1987b) *Systematic Fieldwork Volume 2: Ethnographic Analysis and Data Management*. Sage Publications.
- Whyte, A. (1977) Guidelines for field studies in environmental perception. *Man and the Biosphere: technical notes* 5. Paris, UNESCO.