Pre-publication draft of paper which was published in the International Journal of Man-Machine Studies (1988) **29**, 257-285, Academic Press Ltd.

A Knowledge-Based Approach to Computer-Aided Learning

Geoffrey. I Webb
Division of Computing and Mathematics, Deakin University,
Victoria, 3217, Australia

Abstract

This paper describes a methodology for the creation of knowledge-based computer-aided learning lessons. Unlike previous approaches, the knowledge base is utilized only for restricted aspects of the lesson - both for the management of flow of control through a body of instructional materials and for the evaluation of the student's understanding of the subject matter. This has many advantages. While the approach has lower developmental and operational overheads than alternatives it is also able to perform far more flexible evaluations of the student's performance. As flow of control is managed by a knowledge-based component with reference to a detailed analysis of the student's understanding of the subject matter, lessons adapt to each student's individual understanding and aptitude within a domain.

1 INTRODUCTION

The test and branch style of computer-aided learning (CAL), as typified by such authoring languages as PILOT (Khieriaty & Gerhold, 1980) and CAL systems as PLATO (Alpert, 1975) has several deficiencies, including:

1. poor evaluation of student performance;
2. restrictively rigid forms of interaction with the student; and
3. failure to modularize or explicitly separate logically independent aspects of a lesson such as domain knowledge, student evaluation, and teaching strategy.

These deficiencies are discussed in greater detail in section 2.

One response to these deficiencies has been the development of intelligent tutoring systems. Sleeman & Brown (1982) is a good reference point for examples of the types of approaches that have been pursued in this paradigm. However, intelligent tutoring systems are themselves not without shortcomings. Webb (1986) identifies the following deficits in most general approaches that have been adopted:

1. high developmental overheads;
2. high operational overheads;
3. inability to handle complex domains:
4. lack of methods for knowledge representation that enable necessary forms of knowledge to be recorded; and
5. need for the inclusion of unnecessary domain knowledge.

This paper presents an alternative approach to computer-aided learning - one which bases lessons on a representation of the knowledge to be taught, but does not attempt to generate all aspects of the instructional interaction from such a representation. By this means it is possible to avoid most of the pitfalls of both test and branch CAL and intelligent tutoring systems.

2 THE DEFICIENCIES OF TEST AND BRANCH CAL

Although in principle test and branch CAL can provide highly individualized instruction that is sensitive to learners' aptitudes and learning rates in a domain, in practice this is extremely difficult to achieve.

The primary form of evaluation that test and branch systems are capable of providing is a *raw performance score* at different levels of grouping within a lesson. A raw performance score is a simple tally of the number of correct and incorrect answers that the student provides. At best, this may be useful in determining a crude form of individualization on the basis of initial competence and learning rate. This can be achieved to some extent within test and branch CAL by providing branching in the educational material dependent upon mastery of individual components as determined by a raw performance score. Learning rate can be accommodated by accelerating the student's progress by respectively adding or subtracting optional subcomponents to the student's path through the instructional materials. This will only provide suitable individualization if the component level at which conditional branching is determined happens to correspond with a decomposition of the student's difficulties. If there is an underlying misapprehension that the student holds that does not neatly correspond with the material covered by one of the components, then the manner in which the system adjusts its tuition is not likely to be appropriate for remedying that problem. For example, the methodology is unable to detect the fact that a student is consistently misapplying one of the underlying skills or principles required throughout a lesson unless that skill or principle is the specific subject of a component of that lesson

CAL systems based on the test and branch approach are inevitably quite rigid in terms of their interactions with the student. Only one mode of interaction is possible, one where the computer is very much in control of proceedings. Of necessity the interaction is extremely one-sided. The computer presents information to the student and then the student makes a. response, usually a very brief response. The computer responds to the student's response and then the cycle is repeated.

There is very little scope for the student's response to influence the overall lesson in any major way. Certainly the model allows for the possibility, through very complicated branching routines and the utilization of a history of the student's responses, of different responses leading to different paths through the instructional material. However, in any significant sense, this is just not practical. The amount of coding that has to be produced to create a highly reactive lesson using test and branch CAL is simply too great to be practical for most applications.

Another problem with test and branch CAL is that it inextricably mixes domain knowledge, teaching strategy and student evaluation. For instance, in. a test and branch CAL system a branch instruction can represent any one of:

1. a logical, connection between two aspects of the knowledge being examined by a lesson (for instance, that the material in Unit 2 builds upon the material in Unit 1);
2. part of the teaching strategy (for instance, a particular mistake should receive negative reinforcement);
3. an aspect of student evaluation (do procedure *x* to determine if the student knows fact *y*); or, more often,
4. a mixture of 1 - 3.

This makes it extremely difficult to update any of these three aspects of a lesson once it has been created. With the pace at which domain-specific knowledge advances in some domains it can be seen that it can be quite important to be able to update this aspect of a lesson if it is not quickly to become outdated. Just as domain knowledge alters, so does pedagogical practice. A tutorial strategy or approach to student evaluation is as likely to become dated as is domain-specific knowledge. The inability to readily upgrade these aspects of a lesson is likely to reduce drastically its useful lifespan.

3 KNOWLEDGE-MANAGED CAL

One solution to these problems that has been attempted is the creation of CAL systems that base a lesson on a description of the knowledge to be taught and then generate lessons directly from that description. Such an approach evidently has at least the potential to overcome the problems with test and branch CAL outlined above. First, as the lessons are based on an explicit representation of the knowledge to be taught there is the possibility of forming a detailed profile of the student's understanding of the subject domain and thus of being able to adjust tuition directly to student's instructional needs. As the exact interactions to take place need not be specified in advance, further flexibility is provided by different forms of computer-student interaction being generated for each different situation. Finally, there is the potential for domain knowledge, student evaluation and teaching strategy to be explicitly represented by individual modules of the system, enabling each to be modified with minimal impact on the other. However, as demonstrated in Webb (1986) and discussed briefly above (in Section 1), these potentials have not been realized, and indeed, are not likely to be realized, so long as previous approaches are pursued.

In the ECCLES system (Richards & Webb, 1985) we attempted to tackle these problems by the use of *topic-structuring* - basing lessons on explicit descriptions of the sequences of actions that must be performed to solve the problems or exercises which the lesson addresses. However, this approach is not without its limitations. Most notably, if it is not possible to associate each aspect of the knowledge to be taught directly with one of tube actions which has been described then it is not possible to evaluate accurately which aspects of the relevant knowledge the student has and has not acquired. In this case it is only possible to determine which generic actions the student is able to perform correctly. An underlying student misapprehension that results in the failure to perform several distinct generic actions cannot be adequately diagnosed and treated. This is discussed in more detail in Richards, Webb & Craske (1988).

The approach advocated herein is the use of a description of the knowledge to be taught for the general management of a lesson rather than a description of the sequences of actions that must be performed. Unlike the usual approach utilized in intelligent tutoring systems, this knowledge base is not used for the complete generation of all aspects of a lesson. Rather, the knowledge base is used for student evaluation and for managing flow of control within a lesson, but is not used as the basis for generating low-level interactions with the student. These are left to whatever means the lesson author deems appropriate, be they test and branch, artificial intelligence based, simulation based or whatever. This approach will be referred to as *knowledge-managed CAL* (KMCAL).

The reason for not using the knowledge base (in general) for the generation of low-level interactions with the student is that current artificial intelligence and knowledge representation techniques are not yet advanced enough to make this feasible.

In KMCAL the actual interactions that take place with the student are embodied in a set of modules called *tutorial specifications*. Each tutorial specification is associated with a clearly delineated aspect of the domain knowledge. It is this and only this aspect of the domain that the tutorial specification concerns itself with. Each tutorial specification will usually provide some form of tuition that can be related to the indicated aspect of the domain.

A central assumption made in this approach is that most teaching will take place in the context of the examination of specific concrete examples or problems from the domain. This is in accord with most modern educational theory which places emphasis on the advantages of learning in applied settings rather than in the abstract (Piaget, 1970). As a result of this assumption, KMCAL uses *courseware abstraction,* Courseware abstraction involves the development of general courseware that is applied to many concrete examples from a domain, thus providing specific tuition in a general abstract framework. The advantages of courseware abstraction are discussed in Webb (1986).

As a result of the use of courseware abstraction, most tutorial specifications in a KMCAL system will provide general treatments of some aspect of a domain which can be applied to any appropriate example or problem from that domain. (The qualification "appropriate" in the preceding sentence is important. Clearly a tutorial specification concerning some aspect of the carry operation in addition should not have a requirement that it be capable of being applied to multiplication problems!) Each tutorial specification also has the responsibility for evaluating the student's performance in its restricted context. Clearly it would be ridiculous for any other part of the system to have such responsibility. A subsystem that could observe a student's interaction with arbitrary CAL modules and produce a useful analysis of the student's performance would be little short of miraculous!

A knowledge-based lesson driver (which shall be referred to as *the driver)* has the responsibility for taking each tutorial specification's assessment of the student's performance and producing an integrated assessment of the student's understanding of the lesson as a whole. Based on this assessment, the driver manages the selection of examples or problems for the student to examine and the selection of tutorial specifications with which the examination is to take place.

In general, one manner in which it is possible to view the knowledge that a lesson covers is to regard it as having three parts:

1. a set of concepts that the student must master;
2. a set of discriminations (alternatively choices or categorizations) that must also be mastered; and
3. a set of operations that must be able to be performed.

To provide a simple illustration of these three aspects of knowledge consider a simple problem from elementary arithmetic: the addition of 16 + 16. In terms of concepts, in order to be able to solve this problem the student must obviously have some concept of "a number" and of "arithmetic operations" (among others.) One of the first discriminations that must be made is the choice of the arithmetic operation to apply - in this case addition. The application of this operation requires the choice of what low-level operation to apply, the appropriate selection being the single digit addition of 6 + 6. The result of this single digit addition is 12. The next discrimination is which part of this result to incorporate in the final solution, the answer being the digit 2.

Next the student must choose what to do with the remaining digit, 1. The correct discrimination is to apply the carry operation. And so the process continues.

The claim being made is that most domains can be analysed in terms of these three types of knowledge. The only evidence being offered is a lack of counter-examples - domains for which this principle does not hold.

The approach to knowledge-based CAL developed in this paper covers the tuition of discriminations and operations. A mastery of the relevant concepts is generally assumed. This is not to be taken to imply that the tuition of concepts is not important or theoretically interesting, merely that it is yet to be treated in this context.

The next section examines *feature networks,* a knowledge representation formalism that enables the description of systems of features. When those features represent discriminations that students must make and operations that they must apply, feature networks serve as an ideal knowledge representation formalism for KMCAL.

4 FEATURE NETWORKS

Feature networks are a formalism for representing systems of ordered and related features (or properties) that apply to the elements of a domain of instances. They specify which combinations of features can be exhibited by instances from the domain - that is, which combinations of properties can apply to instances from the domain and how the presence or absence of a feature or property affects the possibility of other features or properties applying to an instance

Feature networks are a variant of system networks, which were initially developed by M. A. K. Halliday as a means of describing systemic grammars. The earliest formal definition of system networks appears in Halliday (1973). A variant more closely related to feature networks is described in Winograd (1983).

A domain (as the word is used herein) is an area of knowledge. The example domain that will be used in this paper is the English Pronouns Domain. It covers the syntactic features that English pronouns may exhibit. It is the corpus of knowledge about the features that different English pronouns exhibit and the constraints on, and interactions between; the occurrences of different combinations of syntactic features for English pronouns. This domain was chosen as it provides a simple example of the various important aspects of a feature network.

A feature network for a domain encodes that area of knowledge. For instance, a feature network for the English Pronouns Domain describes the exact relationships between the different features that an English pronoun may exhibit: the constraints and interactions between the combinations in which they can occur.

Any given feature network is unlikely to be the only correct description of its domain. Rather, it will generally reflect one view of the relevant relationships between the features in that domain. It may be possible to describe any given domain by many different feature networks. In such a case the author of the network must decide which network best captures those aspects of the domain that s/he is seeking to explicate.

Feature networks are a. highly constrained form of knowledge representation. As a result, the types of knowledge that they can represent is restricted. However, this restriction is offset by the corresponding efficiency with which feature networks allow access to, and manipulation of, the knowledge that they represent.

5 A FORMAL DESCRIPTION OF FEATURE NETWORKS

5.1 FEATURE NETWORKS

S= $\langle \mathrm{e, F, O, C, D, P, enters} \rangle$ defines a feature network where:

C is an element called the *network entry point;*
F is a set of elements called *feature choices;*
O is a set of elements called *simultaneous branches;*
C is a set of elements called *conjunctive entry conditions;*
D is a set of elements called *disjunctive entry conditions;*
P is a set of elements called *features;*
{e}, F, O, C, D, and *P* are all disjoint;
enters is a binary function from
$(\{e\}\times(F\cup O))\cup((O\cup C\cup D\cup P)\times(F\cup O\cup C\cup D))\cup(F\times P) to \{true,\ false\}$

Provided:

1. there is exactly one element *x* such that *enters* (e, x);
2. for every feature choice $f \in F$ there is exactly one element *x* such that *enters(x,f)* and at least one $p\in P$ such that *enters(f,p)*;
3. for every simultaneous branch $o\in O$ there is exactly one element *x* such that *enters(x,o);* and there is more than one element *y* such that *enters* (o, y);
4. for every conjunctive entry condition $c\in C$ there is more than one element *x* such that *enters(x,c);* and there is exactly one element *y* such that *enters(c,y);*
5. for every disjunctive entry condition $d\in D$, there is more than one element *x* such that *enters(x,d*); and there is exactly one clement *y* such that *enters(d,y)*;
6. (for every feature p$\in P$ there is exactly one feature choice $f\in F$ such that *enters(f,*p);. and at most one element *x* such that *enters(p,x)*;
7. there is no element *x* such that *descends(x,x),* where *descends* is a relation over $(\{e\}\cup F\cup O\cup C\cup D\cup P)\times(F\cup O\cup C\cup D\cup P)$ It is defined as follows:
   (a) for any two elements *x* and *y* such that enters(*x,y*), *descends(x,y*);
   (b) for all elements *x, y,* and *z,* if *descends(x,*y) and *descends(y,z)* then descends(*x, z*)

Informally, this defines a network with one entry point (a node that descends from no other nodes), multiple terminal points (nodes from which no other nodes descend), and which has five types of internal nodes such that no node descends from multiple nodes and has multiple nodes descending from it. Only features may be terminal nodes. The network does not loop back on itself in that it is not possible to trace a path through the network that passes twice through any one node.

If *enters(x,y)* then *x* is referred to as *y*'s *parent* and *y* is referred to as *x*'s *child*. If *descends(x,y)* then x is referred to as *y*'s ancestor and y is referred to as *x*'s dependent.

The next section provides an interpretation of this formalism that enables it to be used for knowledge representation.

6 THE SEMANTICS OF FEATURE NETWORKS

D = $\langle$[illegible]$\rangle$ defines a *domain* where:

I is a set of instances;
P is a set of properties such that for all **p** $\in$ **P**, there is at least one **i** $\in$ **I,** such that **p** is a property of **i**.

The instances may be either concrete (such as physical objects) or abstract (such as concepts or events). The features in a feature network represent the properties in a domain. Feature networks represent the relationships between the properties in a domain.

A feature network S = $\langle \mathrm{e, F, O, C, D, P, enters} \rangle$ is a feature network for a domain **D** = $\langle$[illegible]$\rangle$ iff for all $p \in P$ there is exactly one **p** $\in$ **P** such that p represents **p** and for all **p**$\in$ **P** there is exactly one p $\in$ P such that p represents **p**.

Here we provide a formal description of a domain and start to describe how a feature network relates to its domain. Informally, a domain is a set of instances (either abstract or concrete *things)* and a set of properties that apply to those instances. A feature network is a feature network for a domain if and only if every feature in the feature network represents exactly one of the properties of a domain. In the CAL context, the instances in a domain will be the examples or problems that the student is to examine. Each property in the domain will, be a. discrimination that the student must be able to make with regard to those examples or problems and each feature will represent exactly one such discrimination.

It is important to note that not all properties of all instances in a domain need he included in a domain. For example, in a domain created for a lesson on motor mower repair procedures, the instances may be different faulty motor mowers. Clearly, each faulty motor mower will have many properties that are not likely to be relevant to the lesson, for example, its colour, its owner's great grandmother's second name and the number of times that it has been through 114^{o} turns while in use. These will clearly not be included in the domain for such a lesson. Examples of properties that are likely to be included are the amount of fuel in the tank, how the motor behaves when the starter cord is pulled and whether a grass catcher is fitted.

So far we have only defined how one type of element of a feature network, the feature, relates to a domain. The broader relationship between a feature network and its domain is defined in terms of the *feature sets* of the domain. These can be used to specify the possible combinations of features that instances in the domain can exhibit. Feature sets are defined in terms of *traversals*.

6.1 TRAVERSALS

Let $S = \langle \mathrm{e, F, O, C, D, P, enters} \rangle$ be a feature network. Then, a subset T of $(\{e\} \cup F \cup O \cup C \cup D \cup P)$ is a *traversal* of S iff:

1. e is an element of T;
2. for all feature choices $f \in F$, $f \in T$ iff there is an element x such that $x \in T$ and *enters* (x, f);
3. for all feature choices $f \in F$ and $f \in T$, there is exactly one feature $p \in P$ such that *enters*(f, p) and $p \in T$;

4. for all conjunctive entry conditions c $\in$ *C*, *c* $\in$ *T* iff all elements *x* such that *enters(x, c)* are elements of *T;*
5. for all disjunctive entry conditions *d* $\in$ *D*, *d* $\in$ *T* iff there is an element x $\in$ *T* such that *enters(x,d).*
6. for all simultaneous branches *o* $\in$ *O*, *o* $\in$ *F* iff the element *x* such that *enters(x,o)* is an element of *T.*

A traversal can be considered to define a path through a feature network, where the nodes in the traversal are the nodes in the path, starting always from *e*, the network entry point. These paths represent the relationships between the nodes in a feature network. The interpretation of such a path relies on the concept of a node *applying* to an instance. Each path selects out a consistent set of nodes all of which may simultaneously apply to an instance. This path will always be such that:

1. The network entry point is in every path - it applies to all instances in a domain.
2. Every feature choice that has a parent in the path will also be in the path.
3. For any feature choice in the path, there will be exactly one of the features that are its children in the path. This means that a feature choice is a choice point from which exactly one of the descending branches must be chosen. It should be noted that feature choices are the only non-deterministic points in a traversal of a feature network and that they only enter features. Thus, all points of choice in the traversal of a feature network require the selection of one from a range of features. As features represent properties, a feature choice represents a set of properties exactly one of which applies to any instance to which the feature choice applies.
4. All and only those conjunctive entry conditions for which all the nodes from which they are entered are in the path, will be in the path. This means that conjunctive entry conditions are points at which feature networks merge and from which a traversal only continues if all of its parents are traversed. That is, the sections of a feature network that descend from a conjunctive entry condition only apply to those instances to which all sections of the network that enter the conjunctive entry condition apply.
5. All disjunctive entry conditions that are entered by nodes in the path are also in the path. As disjunctive entry conditions are entered by multiple nodes, this means that they are points at which multiple paths converge, and from which a traversal will continue if any of the entering paths is being traversed. Sections of a feature network that descend from a disjunctive entry condition will apply to any instance to which any of those sections of the network that enter it apply.
6. All simultaneous branches that are entered by nodes in the path will also be in the path. As simultaneous branches enter multiple nodes, this means that simultaneous branches are points at which traversals branch' to traverse multiple paths concurrently.

6.2 FEATURE SETS, PROPERTIES AND DOMAINS

Let $S = \langle \mathrm{e, F, O, C, D, P, enters} \rangle$ be a feature network. A subset A of P is a *feature* set of S iff there is a subset T *of* $(\{e\} \cup F \cup O \cup C \cup D \cup P)$ that is a traversal of S and $A{=}P \cap T$.

Informally, a feature set is any set of features that contains all and only those that are included in the one traversal.

Feature sets are used to relate feature networks to the domains that they describe. Each feature set for a domain defines an allowable combination of properties for instances from the domain. The set of feature sets for a domain defines all of the allowable combinations of properties that instances in the domain may exhibit.

A feature set A is a feature set for a set of properties $\mathbf{P}$ if and only if for all $\mathbf{p} \in \mathbf{P}$ there is an $a \in A$ such that a represents $\mathbf{p}$ and for all $a \in A$ there is exactly one $\mathbf{p} \in \mathbf{P}$ such that a represents $\mathbf{p}$. That is, a, feature set represents the set of properties that its features represent.

The set of feature sets for a feature network for a domain is complete if and only if there is no instance from the domain that exhibits a set of properties that is not a feature set for the feature network. If the set of feature sets for a feature network is complete then the feature network will be considered complete. If a feature network is not complete for a domain then it does not correctly represent the domain, For example, the English Pronouns Domain feature network (shown in Fig, 2) would not be complete if it did not include the feature set (Personal, Plural, Third Person, Subjective) that is, if it did not allow for the pronoun “they”. Unless otherwise stated, it will always be assumed, that any feature network is complete for the domain that it describes.

The set of feature sets for a domain is *exhaustive* if and only if it contains no feature set for which there is no instance from the domain that exhibits that set of properties. If the set of feature sets for a feature network is exhaustive then the feature network will be considered exhaustive. In general it is not assumed that feature networks are exhaustive. This is because a feature choice may apply to a class of instances even though not all of the features from the feature choice may apply to instances from that class. For example, in the motor mower repair domain referred to earlier, all instances which exhibit the property *fuel tank is empty* will also exhibit the property *motor does not fire* from the *response to pulling the starter* cord feature choice. Thus, this feature choice must apply to instances with the former property. However, no such instance will exhibit any of the other properties at this feature choice.

The semantics of a feature network vary depending on the type of domain being represented. However, in general, a feature network can be said to describe its domain if it defines which combinations of properties from the domain are possible for the instances in the domain and, the ways in which the properties from the domain interact.

6.3 DESCRIBING DOMAINS OF OBJECTS

Domains of objects are domains where the set of instances contains only objects (either abstract or concrete). These are to be contrasted with domains of processes or events. The feature network describes the domain in that it captures the significant relationships between the instances in the domain and the sets of properties that they may exhibit.

Figure 1 shows just such a domain, the extremely simple domain of the syntactic features of personal pronouns in English. Figure 2 shows a feature network for the English Pronouns Domain. Figure 3 shows the set of feature sets for the English Pronouns Feature Network.

If a feature network is complete and exhaustive, each possible feature set for a domain defines a combination of properties that could possibly be exhibited by some instance from the domain. The Pronouns Feature Network is complete and exhaustive as its set of feature sets includes all and only the possible combinations of NUMBER, PERSON and GENDER for English pronouns.

However, a feature network provides a far richer description of the domain that it describes than just this simple specification of the sets of allowable properties for instances from the domain. This deeper level of detail is provided by the structure that the network imposes on the combinations of features that are possible for the domain.

Relationships between features and feature choices in the network encode relationships in the domain being described. Feature choices encode more than simply the existence of a disjoint set of properties. They encode *dimensions of categorization* in the domain. A dimension of categorization is a range of related and contrasting categorizations that apply within a domain. For example, colour could be a dimension of categorization which included such categories as *yellow* and *blue.* By contrast, there is unlikely to be for any domain a dimension of categorization that included the categories *yellow* and *round,* even for a domain in which the properties *yellow* and *round* applied to disjoint sets of instances.

$\langle$ {'he', 'she', 'I', 'you', 'you'$_2$, 'us', 'them'}

{Singular, Plural, First, Second, Third, Masculine, Feminine} $\rangle$

Note, 'you' and 'you'$_2$ are respectively the singular and plural forms of 'you'

Fig. 1. The English Personal Pronouns domain

$\langle$ *nep*,
{NUMBER, PERSON, GENDER},
{SB1},
{CE1),
(},
(Singular, Plural, First, Second, Third, Masculine, Feminine),
enters $\rangle$

where *enters* maps the following pairs to *true*.

nep	*SB1*
SB1	NUMBER
SB1	PERSON
NUMBER	Singular
NUMBER	Plural
PERSON	First
PERSON	Second
PERSON	Third
Third	*CE1*
Singular	*CE1*
CE1	GENDER
GENDER	Masculine
GENDER	Feminine

Fig- 2. A feature network for the English Personal Pronouns domain.

It is beyond the scope of this exposition to discuss the epistemological status of dimensions of categorization, in particular to argue whether dimensions of categorization encode more than simply the necessity of the disjointness of the sets of all instantiations of the properties in them. Suffice it to say that they capture the network designer's intuitions as to the epistemological structure of the domain. If a feature network is complete and all of its feature choices encode dimensions of categorization from the domain then it is said to *describe* the domain. Thus, the Pronouns Feature Network describes the Pronouns Domain.

{{Plural, First}, {Plural, Second}, {Plural, Third}, {Singular, First},
{Singular, Second}, {Singular, Third, Masculine}, {Singular. Third, Feminine}}

Fig. 3. The set of feature sets for the English Personal Pronouns feature network.

6.4 DESCRIBING DOMAINS OF PROCESSES

Feature networks can also be used to define families of processes. In order to achieve this it is necessary that feature choices, simultaneous branches, conjunctive entry conditions, and disjunctive entry conditions in the network be used to define states, that features be used to define events and that descent in the network be used to define temporal precedence (in that a node that enters another node precedes that node in a sequence of events).

The network . entry point for such a network represents the initial state of the process. Feature choices represent states at which one of the several events that are represented by the nodes that they enter can take place. Simultaneous branches represent states after which multiple concurrent events take place - the events being those represented by the nodes that they enter. Disjunctive entry conditions represent states that are reached on the conclusion of any of the several events represented by the nodes that enter them. Conjunctive entry conditions represent states that are reached on the conclusion of all of the several events that are represented by the nodes that enter them.

As the enters function represents temporal relationships between events in a domain of processes it is necessary to be able to identify temporally distinct states when analysing any given process within a domain of processes- These are represented by *markings* and *firings.*

6.5 MARKINGS, FIRINGS AND THE FOLLOWS FUNCTION

Let *S* = $\langle \mathrm{e, F, O, C, D, enters} \rangle$ be a feature network.

M is a *marking* for *S* iff $M \subset (\{e\} \cup F \cup O \cup C \cup D \cup P)$ and *M = e* or *follows* (*e, M*)

follows is a function from subsets of $(\{e\} \cup F \cup O \cup C \cup D \cup P)$ to subsets of $(\{e\} \cup F \cup O \cup C \cup D \cup P)$. *follows (A, B)* iff either:

1. there is an element $x \in (\{e\} \cup F \cup O \cup C \cup D \cup P)$ such that $x \notin A$ and $B = A \cup \{x\}$ and either
   a. $x \in P$ and there is an element $y \in A$ such that *enters(y,* x) and there is no element z C A such that *enters(y, z);*
   b. $x \in (F \cup O \cup C \cup D \cup P)$ and there is an element $y \in A$ such that *enters*(*y, x*); or
   c. $x \in C$ and for all elements y such that *enters(y, x)*, $y \in A$

   (If these conditions hold then *x* is a *firing* of *A); or*
2. there is a marking *C* of S such that *follows (A, C)* and *follows(C, B).*

Markings allow the definition of sequential traversals through a feature network. A marking can be considered to be a partially completed traversal of a feature network. Those nodes in a marking represent the nodes that have been visited to date during the partial traversal that the marking represents. All traversals start from the network entry point, so all markings include the network entry point. The firings of a. marking are the nodes that can be visited next during a traversal that has already visited those nodes in the marking. The markings that follow a marking *M* are those markings that occur subsequently to the state represented by *M* in a traversal of *S.*

For a domain of processes, each marking represents the state of a process at a particular instant. Each firing for a marking represents a possible event (change in state) for the process.

The initial state of a process is represented by the network entry point, so a scan through the network always starts with a marking including only the network entry point.' Any of its firings can then be added to the marking, representing the enactment of an event. This process is continued until a marking is obtained with no more firings. The process represented is then complete.

$\langle$ *nep*,
{FIRST-CHECK, FUEL-TANK-STATUS, NEXT-CHECK, EMPTY-ACTION },
{ },
{ }
{ },
{Check-fuel-tank, Full, Empty. Fill, Check-fuel-pump, Check-carburetor},
enters $\rangle$

where *enters* is true for the following pairs

nep	FIRST-CHECK
FIRST-CHECK	Check-fuel-tank
Check-fuel-tank	FUEL-TANK-STATUS
FUEL-TANK-STATUS	Full
FUEL-TANK-STATUS	Empty
Empty	EMPTY-ACPION
EMPTY-ACTION	Fill
Full	NEXT-CHECK
NEXT-CHECK	Check-fuel-pump
NEXT-CHECK	Check-carburetor

Fig 4. A feature network for the Fuel domain.

Figure 4 shows a feature network for the Fuel Domain. This is a subdomain of the Mowers Domain, a domain of motor mower repair procedures for which a KMCAL lesson has been created. In the Mowers Domain, each instance is a fault with a motor mower and its properties are the symptoms that the mower exhibits for the fault and the set of actions that should be taken to test and, repair it. The Fuel Domain is a subset of the Mowers Domain intended solely for illustrative purposes. It only includes some simple problems with the fuel supply for the mower. The FIRST-CHECK feature choice represents the first test that the repairman must make. In the Mowers Domain this includes many different tests, each appropriate for a different symptom that the mower is exhibiting. The only appropriate first test for the Fuel Domain is to test if the fuel tank is empty. The FUEL-TANK-STATUS feature choice represents the result of this test. Either the tank is empty or it is full (a state that represents any fuel level other than empty). If it is empty then there is only one appropriate action to pursue - fill the tank, and the mower is then fixed!

This interpretation of feature networks bears some similarities to Petri nets (Petri, 1980). However, the similarity is only superficial. Petri nets have no counterpart for the formal distinction in feature networks between disjunctive and conjunctive entry conditions and between feature choices and simultaneous branches. Unlike Petri nets, feature networks are highly structured in such a way as to emphasize choice points (the points at which only one of several options may obtain) and the consequences of each option at such points.

6.6 LIMITATIONS OF FEATURE NETWORKS

6.6.1 Discrete Instances

For a domain to be represented by a feature network it is necessary for it to be possible to identify discrete instances that belong to that domain. Instances from a domain must be discrete in that they must be distinguishable from one another and must each have an identifiable set of features from those incorporated in the domain. This condition is necessary because the meaning of a feature network is given in terms of its relationship to instances from the domain that it describes, If it is not possible to identify instances from a domain, then it is not possible to define the meaning of a network for that domain.

One consequence of this restriction is that it is not possible to define domains that cover any type of continua. For instance, it would not be possible to have a feature network that defined the features of air at varying temperature, as varying temperature cannot be regarded as a set of discrete instances. (This excludes the possibility of regarding it as an infinite set of instances, one each for every temperature. It is not clear how such a set of instances would capture the essential nature of a change in air temperature.) However, such domains can be described indirectly by resorting to a domain that only includes a finite set of the instances that the continuum covers. For instance, the domain of changes in air temperature could be modelled by reference to a set of discrete changes in air temperature.

6.6.2 Discrete properties

For a domain to be represented by a feature network it is necessary for there to be discrete properties identifiable within the domain. This is because a feature network for a domain must explicitly contain a feature for every property from that domain. If it is not possible to identify discrete properties in the domain then it is not possible to relate features to properties for the domain.

It is, of course, possible to cover continua of properties. These can be simply represented by infinite sets of features. For instance, if temperature was a relevant property of the instances in some domain, it could be represented by a feature choice which entered an infinite set of features, one for each possible temperature. It should be noted that the discrete properties requirement does not prevent the definition of feature networks for domains where the properties have vague separating boundaries. In these cases, it is always possible to use only paradigmatic instances.

An example of a set of properties that it might be desirable to include in some domain, but the elements of which do not have clear separating boundaries, is a set of colours. In some cases it may not be possible to identify some instance as being clearly blue rather than green, for example. In the case of such a set of colours being used as properties for a domain, those hues that are clearly identifiable as a particular colour could be selected as the paradigmatic properties that were to be used in the domain. The domain would then only include instances that clearly exhibited those properties that were in the domain.

6.6.3 Single feature assignment per feature choice

Feature networks do not allow instances to be assigned more than one feature at a feature choice. This prevents the definition of feature choices from which instances can exhibit any of several properties, For instance, in the Mower Domain there are several instances in which several tests would be equally appropriate given the information available to the repairman. In these cases it has been necessary to choose one as being correct and to force its choice. It should be noted, however, that the feature network formalism could be extended to allow such feature choices to be defined. See Webb (1986).

6.6.4 No Looping

Feature networks do not allow feature choices to depend upon features which they enter. This prevents the networks from looping back upon themselves. This is desirable for domains of objects where the ability to loop back on themselves would make the semantics of feature networks unworkable. It does, however, impose a considerable restriction on domains of processes for which it is frequently desirable to allow the one state to obtain at different temporal removes. For instance, the current Mowers Domain does not allow for situations in which the mower to be repaired suffers from more than one problem. Thus, if the fuel tank is initially empty and is then filled, then the mower is considered fixed. It would be possible for this initial problem to be but one of the problems with the mower, and once the tank was full, for the repairman to have to return to the FIRST-CHECK feature choice in order to start diagnosing the further problems. This limitation is discussed further in Webb (1986).

7 INSTANCE DESCRIPTIONS

So far we have described how to formalize feature networks for a domain. It is also necessary to be able to specify formally how instances from a domain relate to the feature network description of the domain. The formalism for doing this is called an instance description. An instance description identifies an instance from a domain and specifies the exact set of properties from the domain exhibited by that instance. By doing the latter it indirectly specifies a single unique path through the feature network for the domain.

7.1 HOW INSTANCE DESCRIPTIONS ARE SPECIFIED

An instance description has two parts. The first part uniquely identifies the instance being described. This is called the instance identifier. The second part exhaustively lists the properties from the domain that the instance exhibits. This is called the instance features.

The exact form that the instance identifier takes is not important, except that it must uniquely define the instance being described.. The reason that a general format for specifying the instance identifier is not defined is practical rather than theoretical. Because there are no limitations on the types of instances that problem domains can apply to, there is no form that can be defined that can be guaranteed to identify uniquely an instance from any domain.

In practice, instance identifiers are normally given as textural descriptions of the instance in question. However, graphics or any other medium could be used with equal validity. The exact form that an instance identifier takes is left up to the person specifying the domain.

The instance features are specified by a set of properties from the domain. This set must be a feature set for the network that describes the domain. If this condition is not met, either the network is not complete for the domain, or the set of features for the instance is not a possible combination of features for the domain**.** Figure 5 shows some instance descriptions for the Pronouns Domain.

Identifier	Features
‘he’	{Singular, Third, Masculine}
‘she’	{Singular, Third, Feminine}
‘I’	{Singular, First}
‘you’ $_1$	{Singular, Second}
‘us’	{Plural, First}
‘you’ $_2$	{Plural, Second}
‘them’	{Plural, Third}

Fig. 5. Some instance descriptions for the Pronouns domain

7.2 HOW FLOW INSTANCE DESCRIPTIONS RELATE TO FEATURE NETWORKS

The instance features for an instance define a unique feature choice set for that instance for the domain. This is the set of feature choices and the features chosen for them that must be included in the traversal of the network for that instance. In other words, in terms of traversing a feature network, it specifies the exact set of choices made for the instance from those choices possible in the domain. Thus, the instance features for an instance uniquely specify the set of choices that must be made for that instance.

In terms of the semantics of feature networks the instance features identify the exact set of properties that the instance exhibits from those in. the domain. Thus, the instance features for the pronoun “she” identify that it is a Feminine Third Person Singular pronoun.

8 HOW FEATURE NETWORKS DIFFER FROM SYSTEM NETWORKS

Other than restriction 7 from their definition, feature networks are structurally isomorphic with system networks as defined by Winograd (1983) which are in turn a subset of those defined by Halliday (1973). However, system networks have only one interpretation, as a means of describing systems of choices. By comparison, feature networks are given two interpretations. One as a means of describing possible combinations of properties for instances from a domain, the other as a means of describing the flow of operations in simple processes.

Further, system networks are a graphic representation of choice systems. By comparison, feature networks are a set theoretic construct for which a graphic representation is defined.

System network’s simple systems map directly onto feature network’s feature choices, features, and the enters relationships that hold between feature choices and features.

System network’s entry conditions map onto feature network’s disjunctive entry conditions and conjunctive entry conditions and the nodes that they enter.

System network’s simultaneous systems map directly onto feature network’s simultaneous branches and the nodes that they enter. Halliday’s unmarked features do not map directly into any aspect of the feature network formalism However, this does not entail any difference in representational power between the two formalisms as unmarked features can be represented explicitly in terms of the other mechanisms of feature networks.

System networks contain no restriction on networks looping back on themselves. Restriction 7, from their definition, explicitly excludes this possibility in feature networks.

9 LESSONS BASED ON FEATURE NETWORKS

As was argued in Section 3, most bodies of knowledge that it may be desirable to teach can be analysed as consisting of sets of concepts, operations and categorizations. Disregarding concepts of which the tuition is not the concern of this paper, both operations and categorizations can be represented by feature choices from feature networks. Categorizations are straightforward to represent each feature in the feature choice represents one of the possible categories for a given dimension categorization. Operations are also straightforward to represent, even though the manner in which it is done may be less intuitive a feature choice may represent an operation if all of its feature represent possible outcomes of that operation. In terms of properties of instances (which all features must represent) such a feature will represent the outcome of the operation when it is applied to an instance.

Feature networks do more than just explicitly encode the dimensions of categorization and operations represented by the feature choices and features. The other nodes of the network explicitly depict how those dimensions of categorization and operations relate to one another.

By encoding the categorizations and operations that a student is required to be able to perform, feature networks aid the identification of the relevant aspects of an instance. If it is known which categorizations and operations the student experiences difficulty with under which conditions then instances that require those categorizations and operations to be performed can be identified and selected for presentation to the student. Because they provide an explicit representation of the relationships between the different categorizations and operations, feature networks provide a convenient means of determining the order and conditions under which each categorization and operation should be presented to the student when examining an instance.

The methodology that has been developed for the use of feature networks in KMCAL is called *feature network based courseware design.* By this methodology the lesson author creates a description both of the underlying structure of the domain to be taught and of the specific instances from the domain. A *lesson presentation system* then uses these descriptions to drive detailed lessons that can interactively assess the student's underlying assumptions about the domain.

As outlined above, unlike most knowledge-based CAL systems, the knowledge base is not used to generate the entire instructional sequence. Rather, it is used to analyse the student's understanding of the domain and to determine the flow of control within the instructional process. Given the use of the courseware abstraction methodology, flow of control has two aspects:

1. *extra-instance flow of control* - the selection and sequencing of instances to examine; and
2. intro-instance *flow of control* - the selection and sequencing of instructional activities while examining an instance.

Feature networks efficiently encode exactly the information that is required for these two purposes.

The first stage of lesson authoring under this methodology is to create a feature network for the domain to be taught. This, feature network describes the epistemological structure of the pedagogically relevant aspects of the domain. It specifies all of the relevant features that an instance from the domain may exhibit and how those features are related to one another.

Associated with each node of the network are one or more tutorial specifications. Each tutorial specification for a node encodes a method of teaching how the knowledge encapsulated by that node and, possibly by its dependents applies to any given concrete instance from the domain. Different tutorial specifications for the one node encode different approaches to teaching the same knowledge.

Next, a set of individual instances from the domain is specified. This is called the *instance set.* Each instance is related to the feature network for the domain by specifying which features it exhibits, as is described in Section 7.

By referring to the feature network, the tutorial specifications, and the instance set, the lesson presentation system can then present lessons on the domain to the student.

Automatic analysis of student performance can be used to determine whether the student can correctly identify which features from the domain the instance exhibits. If the student makes an error then the exact features from the domain that the student has incorrectly ascribed to the given instance are known. Over a series of exercises, each based on a different instance from the domain, the lesson presentation system is able to detect regularities in the occurrence of these incorrect ascriptions. In particular, it can detect if the student regularly substitutes particular combinations of features for other set combinations of features, is over- or under-generalizing the applicability of features, or simply does not understand certain aspects of the domain. Thus, the lesson presentation system is able to construct a profile of the exact underlying misapprehensions which the student has with regard to the domain. This is a simple cognitive model. It describes the regularities to be found in the outputs of the student's cognitive system. Webb (1988) describes the form of analysis in more detail.

This form of student model contrasts strongly with other current established courseware methodologies. These methodologies can only, at best offer overlay models of a student's understanding (Goldstein & Carr, 1977). In an overlay model, the student's understanding of the domain can only be described as a subset of the system's. As is demonstrated in Webb (1988), feature-network-based courseware can produce student models that provide a detailed analysis of how the student's analysis of a domain diverges from the system's.

The use of feature-network-based CAL is necessarily limited by the power of feature networks as a means of knowledge representation. However, it is a very efficient means of providing highly versatile, robust, and. responsive courseware for analysing and debugging student's knowledge of those domains feature networks can readily model.

10 COMPONENTS OF FEATURE NETWORK BASED COURSEWARE DESIGN SYSTEMS

A system for authoring and presenting lessons based on the feature-network-based. CAL methodology has five major components.

1. *A feature network editor.* This allows the specification and modification of feature networks.

2. *A tutorial specification editor.* This allows the creation and modification of tutorial specifications which are associated with the nodes of the feature network.
3. *An instance editor.* This allows instances to be specified. It also allows these instances to be related to the feature network for a domain and enhanced as required by the lesson.
4. *A feature network lesson presentation system.* This takes the feature network, the tutorial specifications, and the instance set and uses them to present lessons to the student. It maintains records of the student's performance.
5. *An analysis system.* This takes the student records, the feature network, and the instance set, and generates analyses of the student's understanding of the domain being taught.

The domain-analysis-based instruction system (DABIS), an implementation of a feature-network-based CAL system, is described in Webb (1986)

11 MODES OF PRESENTATION

The one feature network description of a domain can be utilized for many different forms of lesson. The form of lesson produced depends upon the form of the tutorial specifications utilized by the system in presenting the lesson. The following sections describe some of the forms that such lessons can take. It should be noted, however, that feature-network-based lessons are in no way limited to these forms of presentation.

11.1 INTERROGATIVE MODE

A powerful pedagogical method involves the use of strategically selected questions to establish a. profile of the student's understanding of a domain and then the provision of tuition in those areas in which the student demonstrates weaknesses. This pedagogy is realised in feature-network-based CAL by *interrogative mode.* This mode of operation involves the presentation of judiciously selected instances to the student so as to both determine the student's weaknesses with regard to the domain and provide tuition that concentrates directly on problems that exercise those weaknesses.

To present an instance in this mode the lesson, presentation system steps through the feature network and executes tutorial specifications that test the student's understanding of the instance. Feedback is given as appropriate.

To step through a. feature network *S* for an instance *I*, the system starts with a marking of *S* which initially contains only the network entry point for *S*. This marking is called the *current marking.* An initially empty set of elements of *S* is also maintained. It is called the *block*. Each firing of the, current marking which is not also an element of the block is added one at a time to the current marking. When a firing is added to the current marking, if it is a feature choice then a tutorial specification is invoked that describes the feature choice to the student and requests her/him to select the correct feature from the feature choice for the current instance. If the student selects the correct feature then it is also added to the current marking. If the student selects the wrong feature, then all of the features that the feature choice enters are added to the block. This process continues until there are no more firings of the current marking which are not also elements of the block.

The tutorial specification may either be *explicit* or *implicit.* Explicit tutorial specifications directly refer to the choice being made. Figure 6 shows a transcript of a possible student interaction with an explicit tutorial specification for the Pronoun feature choice from the English Pronouns domain. In this example, the student is asked to identify explicitly the pronoun type of the instance (which is in this case the word “he”).

By contrast, implicit tutorial specifications require the student to identify which feature s/he is assigning to an instance without explicit reference to the feature choice or features in questions. Figure 7 shows a transcript of a possible (simple) student interaction with an implicit tutorial specification for the Pronoun Type choice from the English Pronouns domain. In this case the student is given a word and a set of sentence templates and asked to select which template the word best occupies. Each template best takes a word with one of the features at the feature choice. Thus, the student is specifying by her/his choice, exactly which features s/he is implicitly identifying for the instance. This tests the student’s competence in judgments about the feature choice rather than her/his formal knowledge as is tested in the explicit case.

The tutorial specification may utilize any form of interaction with the student so long as it serves both to test the student’s understanding of how the feature choice applies to the instance and to provide suitable feedback. (It should be noted that the multiple choice format featured in Figs 6 - 8 is only one of the possible approaches that can be utilized. These simplistic forms of interaction should not be taken to indicate the limits on the sophistication in student interactions that are possible under the methodology. Rather, they should be taken as simple illustrations of the different presentation modes. The major limitations on the form of interaction possible are imposed by the physical capacities of the computer and the imagination and skill of the course author.)

‘he’

Is ‘he’:

a. Demonstrative;
b. Personal;
c. Question; or
d. Quantified?

? c

No, a question pronoun can appear as the question element in a clause.
The pronoun ‘he’ cannot. An example of a question pronoun is ‘which’.

Have another try

Fig. 6. A simple explicit tutorial specification for the Pronoun Type feature choice

‘he’

Select which sentence slot this Word best fills.

A. ‘...is the right direction.’
B. ’…is going now.’
C. ’…of these is the answer?’
D. ‘Here are … of the answers.’

? C

No, a. pronoun like ‘which’ that is used for introducing questions fits much better into the slot in sentence C than does the pronoun ‘he’

Have another try

Fig. 7. A simple implicit tutorial specification for the Pronoun Type feature choice

The student learns in interrogative mode in two ways. First, feedback is provided that corrects and analyses mistakes and confirms and reinforces correct responses. Second, the student receives practice in the analysis of the domain being examined.

Appropriate feedback is given to the student based on the system’s knowledge of the correct response for the instance at the feature choice and the student’s response. This feedback can either be specified by the course author or automatically generated by the lesson presentation system. When generating this feedback the lesson presentation system has available to it not only the details of the student’s choice and the correct choice, but also details of the student’s performance to date while analysing the current instance and a model of their general understanding of the domain. As a result it is relatively simple for the lesson to provide very detailed remedial feedback.

The interrogative presentation mode ensures that the student examines each feature choice that applies to an instance. At each feature choice the associated tutorial specification is presented for the instance. The use of the block ensures that no feature choice is presented until all of the relevant features which are its ancestors have been correctly identified by the student. This is because it is likely that a feature choice is going to appear inappropriate to a student if she/he has not yet considered or, even worse, has failed to identify the features from which it descends. For example, in the case of the English Pronouns domain, the student is never asked to identify the person of a personal pronoun until they have correctly identified it as being a personal pronoun. After all, if the student believes that a personal pronoun is a demonstrative pronoun it is necessary to remedy this misapprehension before the task of identifying what type of personal pronoun it is becomes appropriate.

11.2 TESTS

An important aspect of tuition is the ability of the teacher to evaluate the student’s understanding of a domain. Interrogative mode allows for the evaluation of students’ deep-seated understandings of a domain. However, this evaluation is not generally suitable for evaluation purposes as the student is given feedback that is intended to improve her/his understanding of the domain. This means that the results of the student’s interaction with the system after a period of tuition will not accurately reflect the student’s understanding of the domain before the tuition.

For many purposes this is not important. Sometimes, however, it is desirable to be able to test students in such a manner as to not allow their results to be influenced by the testing system. An example of such an occasion is an examination for accreditation purposes.

Due to the system’s knowledge of a domain, it is possible for a feature-network-based courseware system to provide detailed analyses of a student’s understanding of a domain. As a result, it can provide a very powerful testing and student evaluation system. Test mode enables such evaluation and testing to be automatically conducted by the lesson presentation system.

Test mode is identical to interrogative mode except that the tutorial specifications do ‘not provide feedback to the student. Thus, the results of the test are not confounded by the student receiving corrective feedback during the testing process.

For some purposes it is desirable to have large batteries of tests on a particular subject (see, for example, Cooper & Lockwood, 1981; Derevensky & Cartwright, 1981; Ariew, 1982).

It is desirable to be able to both deliver and evaluate these batteries by computer. Systems for the delivery and evaluation of test batteries, such as those discussed in the citations above, typically require huge development efforts. However, given that an interrogative mode lesson has been specified for a domain and sufficient numbers of instances have been specified, test mode can be obtained simply by turning off the responses given in interrogative mode! Given these conditions, there is no extra overhead in authoring time for the delivery and evaluation of such tests.

11.3 DECLARATIVE MODE

Interrogative and test modes involve the evaluation of the student through close questioning with optional feedback. Another important pedagogical method is direct instruction, where the teacher actively provides information to the passive student. Feature networks can be utilized to this end by the creation of declarative lessons.

As in interrogative lessons, in declarative lessons the student is stepped through the feature network for the domain in which instruction is taking place, and a tutorial specification is invoked for every feature choice encountered- The difference from the interrogative mode is that a declarative tutorial specification describes the dimension of choice represented by ‘the ‘feature choice and how the instance being presented relates to it. These tutorial specifications do not question the student in any way. Thus the student is given a detailed description of the overall domain in the context of concrete examples of how individual instances relate to the domain as a whole.

This mode of operation is not envisaged for use by the student for extended periods of time. It is viewed more as an aid to the tuition process that can be invoked as needed. An example of when this mode may be appropriate is if a student is found to be unable to analyse certain classes of instances. In this case, declarative mode can be used to demonstrate to the student how to analyse correctly instances of that class.

11.4 IMMEDIATE MODE

A totally different mode of lesson presentation from those discussed above can also be based on the feature network and instance set for a domain- It is called the *immediate presentation mode.*

The interrogative, test, declarative, and mixed modes of presentation are all *domain decomposition* modes of presentation. They analyse the instance being presented separately in terms of each individual dimension of categorization in the domain that is relevant to it. By contrast, the *immediate presentation mode* takes a holistic approach to the analysis of instances. Rather than working through the network selecting each of the features for an instance, this mode requires the student to make a sing1e choice which specifies all of the features that apply to an instance.

By way of a simple example, suppose that a lesson is being presented on English verb tenses and the domain consists only of two feature choices that descend from a simultaneous branch the first between Past, Present and Future and the second between Simple and Continuous. The student can then be presented with a series of exercises such as that in Fig. 8. The form selected specifies the full set of features that the student deems appropriate for the given situation.

A series of such exercises allows the presentation system to construct a profile of the student's deep-seated apprehensions and misapprehensions with regard to the domain in exactly the same manner as is done by the modes discussed above. The difference is that the profile is based on a holistic view of the domain, rather than the detailed but itemized view that the other modes operate on.

As is the case with interrogative mode, lessons in immediate presentation mode can be either implicit or explicit. However, it is difficult to envisage why it would be desirable to present explicit lessons in this mode. It would seem more natural to examine each feature choice explicitly, rather than explicitly choosing the whole feature set for an instance at the one time.

I…the ball then threw it to John.

a. bounce
b. was bouncing
c. will bounce
d. will be bouncing
e. bounced
f. bouncing

Select the correct form to fill the gap in the sentence.

Fig. 8. A simple implicit tutorial specification for the Pronoun Type feature choice

It should be emphasized that the simple multiple choice format presented in Fig. 8 is used solely for illustrative purposes and should not be taken as representative of the scope of the approach.

11.5 CONSTRUCTIVE MODE

A radically different use of feature networks is provided by the *constructive presentation mode*. This mode utilizes tutorial specifications which allow students to construct feature networks to describe the domain of knowledge being explored. The system then critiques the constructed model on grounds of simplicity of construction and completeness. The former can be tested in terms of number and complexity of nodes and the latter in terms of the model's ability to account for a judiciously selected set of example instances and counter-example instances (those with combinations of features that should not be allowed). Both feature set and feature choice set equivalence could, be used in evaluating the ability of a student's feature-network to account for a given example instance.

The sets of example and counter-example instances must be specified by the course author. Such specification is all that the course author must do to enable the system to present lessons in, this mode after a feature network has been specified for a domain.

11.6 REFERENCE MODE

Another presentation mode is the *reference presentation mode*. This mode enables the student to query the system's knowledge base. The types of information retrieval that the system can readily support include:

the features that an instance exhibits;
the dimensions of categorization (feature choices) from a domain;
the allowable combinations of features from a domain;
the features that must be exhibited by an instance if it exhibits a given feature or combination, of features;
the features that can be exhibited, by an instance if it exhibits a given feature or combination of features; and
example instances with given features.

All that a course author must provide to enable a lesson to be presented in this mode is a feature network description of the domain and a set of instances to use as examples and to refer to when requested for the features of an instance. Thus, once a lesson has been prepared for a domain in one of the other modes, no additional work is required of a course author to enable the system to present lessons in this mode.

11.7 MIXED MODE

In some cases, of course, it is desirable to mix different modes of presentation in the one exercise. The order in which the different modes are inter-mixed depends upon the aims of the course author.

If it is desired simply to test the student's understanding of a domain of knowledge then only interrogative or test mode exercises are given.

If it is desired to use the lesson for remedial work then interrogative exercises can initially be presented. Then once the lesson presentation system has identified problem areas for the student, declarative exercises can be given by way of follow-up for instances having the combinations of features with which the student had difficulties. This process can continue through several cycles, with the system presenting series of interrogative exercises after each series of declarative exercises in order to determine how the student’s understanding of the domain has been affected. These series of interrogative exercises can then be used to determine problem areas to be concentrated on in further declarative exercises - and so the cycle continues.

If it is desired to use the lesson for instruction then the lesson author has several choices. Declarative exercises can be presented, with interrogative exercises optionally being used as a follow-up to assess the student’s understanding of the domain. Alternatively, interrogative lessons can be written, with much emphasis being placed on the feedback texts associated with each feature with the aim of having the student learn from her/his mistakes.

It is, of course, possible to mix presentation modes within an exercise by associating with different nodes tutorial specifications that utilize different modes.

12 CHOOSING INSTANCES FOR PRESENTATION

Section 11 discusses various strategies for utilizing feature networks to determine intra-instance flow of control. This section outlines how they may be used, to manage extra-instance flow of control. Many different strategies may be appropriate depending upon the aims of the lesson.

An extremely simple strategy that may be used is simply to select instances from the instance set so as to provide a broad sample of the combinations of features present. This may be achieved by determining the available feature sets and then randomly selecting instances that exhibit each feature set. This provides the student with a wide coverage of the domain being taught. However, this simple strategy fails to take full advantage of the knowledge available to the system.

Alternatively, the lesson author may favour some order in which instances from the domain should be exposed to the students. This ordering may even be responsive to student’s performance as the lesson progresses. Conditions could be imposed on which different alternative streams of instances are selected

A more sophisticated strategy involves tile identification of the student’s cognitive misapprehensions about the domain and the selection of instances that provide the student with the opportunity to confront and resolve those misapprehensions. The first step in such a strategy is the identification of the student’s misapprehensions. To this end, a sequence of instances must be chosen that will enable the system to detect the difficulties that the student is experiencing. This may either be selected by determining the set of feature sets for the domain and randomly selecting examples of each, or by an initial set of instances specified by the lesson author for this purpose. Detailed analyses of the type outlined in Webb (1988) can then be performed to determine the cognitive misapprehensions that the student holds.

Having identified the feature choices, features and feature combinations for which misapprehensions exist, the system is able to select instances that will require the student to perform the type of analysis with which they are experiencing difficulty. This provides them with the opportunity to confront the misapprehensions that they hold and to rectify them.

However, virtually the same effect can be achieved without the computational overhead of having to construct a detailed student model while presenting the lesson. Rather, it is possible to use the feature sets of the instances for which the student makes errors to select further instances for remedial examination. When the student makes an error it is reasonable to assume that it reflects a cognitive misapprehension on their behalf. Indeed, this assumption is central to any attempt to construct a profile of the student’s comprehension of a domain from a record of their actions while examining that domain. Given that a misapprehension exists, it is the system’s responsibility to assist the student to remedy that misapprehension. Given Piaget’s principle that learning is best facilitated in the context of the practical application of the principles to be learnt, the system should provide the student with the opportunity to apply the correct principles, and assist her/him to apply them correctly. If the examination of an instance with one particular set of features has caused the misapprehension to be exercised then it is probable that further instances with the same combination of features will have the same effect- Thus, the selection of instances with the same feature set as the instance for which the erroneous analysis was performed is likely to provide the student with opportunities to correct the misapprehension that caused the error. This can be achieved without any significant computational overhead.

The only forms of error for which this strategy will not select appropriate remedial instances are over-generalizations and errors arising from extraneous causes. (For a fuller discussion of these forms of error see Webb, 1988.) By definition, the latter of these two forms of error is due to factors which the system is unable to determine and thus cannot be adequately remedied. The former, where the error arises from the student’s over-application of an incorrect principle rather than from the failure to apply the correct principle, can be dealt with by selecting remedial instances that exhibit the same feature set as the student has erroneously identified for an instance as well as the correct feature set for that instance. This provides the student with practice with, both cases where the principle should be applied and where it should not.

13 THE VALUE OF FEATURE NETWORKS AS A BASIS FOR COMPUTER AIDED LEARNING

Feature networks provide a very simple model of the knowledge that they represent. This has the disadvantage that some forms of knowledge cannot be readily represented by the formalism. On the other hand, however, it is this very simplicity that lends much to the power of feature networks as a basis for computer-aided learning.

A feature network clearly does not contain all the information necessary to teach a domain to a student. A feature network for a domain specifies how the different aspects of a domain relate to one another, but in no way does it specify what each aspect of the domain is. For example, the Pronoun Type feature choice from the English Pronouns Domain specifies that all instances in the English Pronouns Domain have one of four features, Demonstrative, Personal, Question or Quantified. However, it does not specify what those features are or, indeed, anything about them other than that they belong to the one dimension of categorization within the domain and relate in certain ways to other features in the domain.

Nevertheless, although it does not contain all the information necessary to conduct the entire process of tuition, a feature network does contain exactly the information needed to be able to produce very powerful models of a student’s understanding of a domain and to select which aspects of the domain the tuition process should concentrate on. With a feature network as the underlying basis of a lesson, all that is required to be able to conduct intelligent tutorial interactions with the student is a means of conveying the knowledge described by each aspect of the feature network to the student and of evaluating her/his understanding of that aspect of the network This is achieved through the creation of tutorial specifications for each aspect of the network.

A major issue in AICAL is the identification of the underlying misconception(s) that cause a student's error. When the model of knowledge that a system uses is highly complex, it is not possible to determine readily how the student's model of the knowledge in question differs from the system's. By the intelligent utilization of such a simple knowledge representation medium as feature networks, it is possible for a computer-based system to quite readily build up a complex model of a student's understanding of a domain (Webb, 1988). This model, unlike alternative approaches to student modelling, provides a detailed description of how the student's understanding differs from the system's without the need for prior specification by the lesson author of the possible divergences that may occur.

Another major problem for the designer of any form of computer-aided leaning courseware, and in particular for designers of AICAL systems, is the extreme difficulty of specifying and validating the courseware. A rule-of-thumb figure that is touted around the computer-aided learning community is that traditional forms of courseware take 200 hours of specification for one hour of finished courseware. No figures are available for comparison with most of the much more complex Al based systems that have been designed, but given their sheer size and complexity, they must be in orders of magnitude higher.* By contrast, feature-network-based CAL has a very low authoring-to-student time ratio. One lesson on English word classes has demonstrated a ratio of 12.5 to 1 (Webb, 1986). Although this lesson supports only very limited forms of interaction with the students, it is highly responsive to their understanding of the domain and can be used, to construct detailed student models.

The low, authoring-to-student time ratio can be attributed to three factors:

First, when specifying courseware based on feature networks, the course author is concerned primarily with issues of fully specifying an appropriate description of the knowledge to be taught and not with extraneous issues such as how to specify to the computer the low-level details of how to present the information to the student and how to respond to student errors Such extraneous issues tend to dominate authoring time under traditional computer-aided learning methodologies. The author's time is not expended in such ways when using feature-network-based CAL.

Second the one specification of the knowledge to be taught can be utilized for many different forms of lesson, with at most only a minimal need for the author to make explicit allowance for the possibility. The manner in which this can be achieved is discussed in Section 11. A result of this ability to use the one knowledge base for many forms of lesson is that the one investment of authoring time pays many dividends in the quantity and type of lesson that is generated. Traditional authoring methodologies require explicit authoring of all material with which the student is to he presented.

* Anderson & Skwareki (1.986) state that remedial material is being produced for their LISP tutor at the rate of one hour of lesson for 40 hours of coding. However, as this is remedial material, presumably each hour's worth will be viewed by only a very small number of students, making the average student-hour's worth of material that is being produced per authoring hour very much lower than the ratio of one for every 40 that this figure initially suggests.

Finally, with feature-network-based CAL, the bulk of authoring time is taken with the specification of a generalized lesson that is then applied to many individual tasks. Thus, it is abstracted courseware and has all the advantages associated with courseware abstraction that are outlined in Webb (1986).

Another highly problematic issue for the designer of AICAL is how to respond to the detection of a student's misconceptions with regard to a domain of knowledge. Feature networks provide a convenient framework for the automatic selection of remedial tasks for the student. They also provide for the explicit description of student misconceptions, either directly to the student, or to the course coordinator, for later use in tailoring the student's curriculum to suit her/his particular instructional needs. As is described in Webb (1988) a student's misconceptions can be well characterized by a description of the features that instances exhibit when the student selects particular features.

Another advantage of feature-network-based courseware is that, due to its simplicity, the highly responsive lessons that it provides can be presented by quite small computer programs. These can thus be realistically run on both the heavily utilized time-shared computers and the small, highly limited micro-computers typically found in educational institutions. This compares favorably with the gigantic systems that are usually associated with AICAL. These monolithic systems can perform acceptably only on lightly utilized large computers. Many require dedicated, large computers.

At a global level, feature-network-based CAL explicitly separates domain knowledge, teaching strategy and student evaluation. The domain knowledge is represented in the feature network. The teaching strategy is represented in the strategy used to determine flow of control. Student evaluation is represented by the analysis subsystem. The only level at which these three may not be explicitly separated is in tutorial specifications. The methodology does not specify how these should be managed so any approach may be taken. If the test and branch style of CAL is used in the tutorial specifications then domain knowledge, teaching strategy and student evaluation are likely to be intermixed. However, each tutorial specification addresses one small aspect of the knowledge that is to be examined, so the domain knowledge that it represents is very localized. Similarly, the student evaluation that it embodies has a very specific purpose, to evaluate the student's comprehension of one localized issue with respect to a specific instance from the domain. Changes to these two aspects of a lesson will be relatively simple to make. The only type of change that will require major alterations to the lesson will be a change in the teaching strategy that is to be employed at feature choices.

Feature-network-based CAL provides genuine individuation between students on the grounds of initial competence and learning rate. The methodology very quickly identifies the student's particular difficulties with regards to the domain and focuses its attention upon them. As the student masters an aspect of a domain, tuition rapidly adjusts accordingly. As is demonstrated in Webb (1986) the methodology also enables a certain degree of individuation on the basis of learning style and educational goals. However it is the problem of providing individuation on the basis of initial competence and learning rate that the methodology primarily addresses.

I would like to thank Tom Richards out of whose initial research on the ECCLES tutoring system grew the research reported on herein. His assistance and encouragement have both contributed significantly to this research.

References

[1] Alpert, D. (1975) The PLATO IV system use: a progress report. In Lecarme, O. and Lewis, R. Eds., *Computers in Education.* Amsterdam: North-Holland.

[2] Anderson, J>R> and Skwarecki, E. (1986). The automated tutoring of introductory computer programming. *Communications of the ACM*, **29**, 842.

[3] Ariew, R. (1982). A management system for foreign language tests. *Computers and Educations*, **6**, 117.

[4] Cooper, A. and Lockwood, F. (1981). The need for provision and use of a computer-assisted interactive tutorial system. In N. Rushby, Ed., *Selected Readings in Computer Based Learning*, pp 127-131. New York: Kogan Page.

[5] Derevensky, J. and Cartwright, G. (1981). The use of a computer-assisted testing in an introductory course of educational psychology. In N. Rushby, Ed., *Selected Readings in Computer Based Learning,* pp 218-221. New York: Kogan Page

[6] Goldstein, I.P. and Carr, B (1977). The computer as a coach: an athletic paradigm for intellectual education. *Proceedings of the 1977 Annual ACM Conference*, Association for Computing Machinery, pp 227 – 233

[7] Halliday, M. (1972). *Explorations in the Functions of Language*. London: Edward Arnold.

[8] Khieriaty, L. and Gerhold, G. (1980) *COMMON PILOT Language Reference Manual.* Bellington WA: Western Washington University.

[9] Petri, C. (1980) Introduction to the general net theory. In W. Bauer, Ed. *Net Theory and Applications*, pp 1 – 20. Berlin, Springer-Verlag.

[10] Richards, T.J. and Webb, GI.I. (1985) ECCLES: an “expert system” for CAL. *Proceedings of the 1985 Western Educational Computing Conference*, pp 151 – 157. Oakland, CA

[11] Sleeman, D. and Brown, J.S. eds (1982) *Intelligent Tutoring Systems*. London: Academic Press

[12] Webb, G.I. (1986) Knowledge Representation in Computer-Aided Learning: The Theory and Practice of Knowledge-Based Student Evaluation and Flow of Control. *PhD. Thesis*, La Trobe University, School of Mathematical and Information Sciences.

[13] Webb, G.I. (1988) Attribute-based cognitive diagnosis. *To be published*.

[14] Winograd, T. (1983) *Language as a Cognitive Process*. Vol. 1: Syntax. Reading, MA: Addison-Wesley.