Using User Models for Explanation: Introducing ThUMS

Lisa Dent, Amar Sanmugasunderam, Bruce Spencer
Department of Computer Science

Research Report CS-87-38
June 1987

Using User Models for Explanation:

Introducing ThUMS

Lisa Dent, Amar Sanmugasunderam, Bruce Spencer

Submitted as a final project

CS 685, Winter 1987

Instructor: R. Cohen

June 19, 1987

Contents

List of Tables

List of Figures

1 Motivation

User modelling is a popular topic in current Artificial Intelligence research. This technique of storing user characteristics enables a system to respond more intelligently to a particular user.

There are many different techniques of user modelling which have been proposed. We are interested in the approach proposed by [Cohen and Jones, 1986], and also in [Cohen, 1987], and in the related work of [Van Beek, 1986]. In the approach of Cohen and Jones, the user's background knowledge is modeled by the use of stereotypes and specific information which overrides the stereotype. Thus the user model keeps track of more than just the level of expertise of the user; it is possible to differentiate several aspects of the user's background simultaneously. The information about the user's background is used to avoid telling the user what is already known. In the approach of van Beek, the user's goals are tracked as well. Explanations generated by the system will address these goals, even if they are not explicitly stated.

The aim of this project is to explore the two parts of the proposed user model in an attempt to develop an overall control structure which combines user background and user goals. The interaction between the two components is of special interest. By implementing a small user modelling system (ThUMS, the Theorist User Modelling System), we test the ideas presented in Cohen and Jones, and some of our own ideas about the interaction between background and goals.

2 The User Model

The user model is used to store information about the current user of the system. There are two main components to the model: the *background* knowledge that the user has, and the *goals* of the user as they relate to the system.

Each user may have different background knowledge. In order to produce appropriate explanations, it is necessary to know how the user's knowledge compares to the knowledge of the system. To make this task manageable, the system knowledge is partitioned into categories, where knowledge of a particular category is likely to be all known or all unknown to the user.

Stereotypes are then used to provide a starting point to characterize the user. Each stereotype defines a different set of domain knowledge that a class of users is likely to have. As the session proceeds, *specific* information is added to the stereotype which pertains to the particular user. The system uses the information about the background of the user in generating the response by checking each part of the response against what the user knows. If the user already knows something, it will be deleted from the response in order to avoid providing repetitive information.

During a session with the expert system, the user may have goals at several different levels. The overall goal of the session is known as the *domain goal.* The domain goal is generally closely related to the purpose of the expert system, and typically does not vary according to the user. Thus, we do not consider different domain goals in our user modelling system.

Different users may have different *preferences*, which are session-wide subgoals that they would like to see addressed. We believe that these preferences are typically a result of the user's background, and therefore they are initially set by the user stereotypes. If the user's preferences are met, he should be made aware of it, and if they are not met, the explanation from the system

should include the reason why.

As well as session-wide goals, the user has goals which change during the session, according to the specific question being asked (see [Van Beek, 1986]). These goals must be determined by the user modelling system for each question; they are not part of the stored user model. We define the *stated goal* as the literal interpretation of the query asked by the user, in the context of the current session. The stated goal and the relevant user preferences should be addressed in an explanation. Generally information is added to the explanation due to the user's goals.

The *intended goal* is the underlying motive behind the user's stated goal. Based on the stated goal, and the user's background, the system attempts to derive the intended goal. The system then responds to this derived intended goal, in an effort to provide the user with an answer to the question he meant to ask. In the remainder of this paper we will refer to the derived intended goal simply as the intended goal.

The two parts of the user model both influence the response which the system gives the user. They are separate factors, but they are not independent. The interaction between the user's background and the user's goals is one of the focal points of our user modelling system.

3 Domain

In order to experiment with the type of user model described, a particular application domain was chosen: the domain of educational diagnosis. We have implemented a user modelling system which tailors explanations generated by an educational diagnosis Expert System to the particular user of the system. This educational diagnosis Expert System is the same system used by Cohen and Jones in their discussion of user modelling.

The purpose of educational diagnosis is to determine the exact nature of the learning problems a student appears to be experiencing. The diagnostic procedure is difficult and often requires specialized knowledge which may not be available. The expert system provides this specialized knowledge. Given data about a student, the system comments on possible learning disabilities, and recommends further testing to help isolate the problem. The purpose of the system is to discover the students disabilities (through continued testing), in order to develop a remedial program for the student. The most common user of the system would be a school psychologist. However the system could also be very useful to teachers who do diagnosis when a psychologist is not available, to school administrators who are interested in the effects of the diagnoses made by the system on the school as a whole, and to parents who may be interested in the system's explanations of their child's difficulties.

Due to the wide variety of system users, the educational diagnosis expert system is a good candidate for user modelling. The users fall into well-defined classes which can be stereotyped to provide an initial model of the user. These classes of users differ by background rather than just level of expertise. For example, a parent cannot be expected to be familiar with the diagnosis domain, but should be quite familiar with the history of the student. On the other hand the

psychologist should be very knowledgeable about the diagnosis domain but will probably not know the particular history of the student. Each type of user may also have different preferences to be addressed by the system. For example, the parent may be concerned about the level of stress which a particular test imposes on the child. Thus the user model described in the previous section appears to be appropriate for this domain.

In order to concentrate on implementing a system which uses user models to generate explanations, we have assumed that the educational diagnosis expert system is available and will be able to generate reasonable explanations to queries. We have defined a subset of diagnostic knowledge, and a set of stereotypical users and example situations to test the system. Four stereotypes have been defined: Psychologist, Parent, Principal and Resource Teacher. The knowledge in the system has also been broken down into categories. There is educational diagnosis knowledge, which consists of diagnostic principles and test interpretation rules. There is also knowledge about the results of the student's tests so far in the diagnosis. Finally, there is data about the history of the student at home and at school. Table 1 shows the background knowledge in the initial stereotype for each class of users. The main (domain) goal of all users is to get the most accurate diagnosis possible, but the subgoals or preferences also vary according to the stereotype. Table 2 shows the initial preferences of each stereotype. The user model will begin as one of these stereotypes, and will be updated as the session proceeds.

None of the authors of this paper are well versed in the domain of educational diagnosis. Therefore, the examples should not be considered sound diagnostic procedures.

Knowledge Type	Psychologist	Teacher	Parent	Principal
Diagnostic Principles (simple)	knows	knows		
Diagnostic Principles (complex)	knows			
Test Rules (simple)	knows	knows		
Test Rules (complex)	knows			
Student Test Results	knows	knows		
Student Background			knows	

Table 1: Initial Stereotypes - Background

Psychologist	Teacher	Parent	Principal
standard tests		low stress tests	funded tests

Table 2: Initial Stereotypes - Preferences

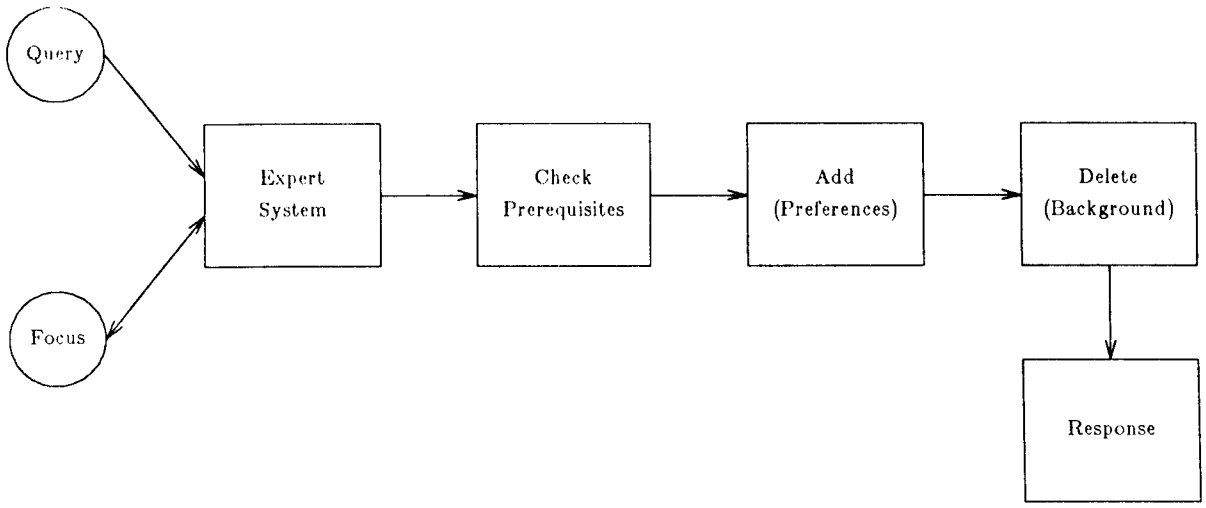

Figure 1: General Block Diagram of ThUMS

4 Architecture of ThUMS

Previous implementations have either modeled the user by tracking his goals and plans [Van Beek, 1986] or by using stereotypes of the user, which are updated with specific knowledge as it becomes available [Cohen and Jones, 1986]. In order to investigate the relative merits of each technique, and the interaction between them, they have been combined in ThUMS: Theorist User Modeling System.

4.1 Overview of ThUMS

In very general terms, the operation of ThUMS is illustrated by the block diagram in Figure 1. The user's question is converted to an internal form, and the present focus of attention is added to it, producing the user's stated goal. This query is forwarded to the expert system, which returns an appropriate reply. If the user has preferences which are applicable to the query, they are then addressed. Finally, information the user is expected to know is deleted from the explanation, to

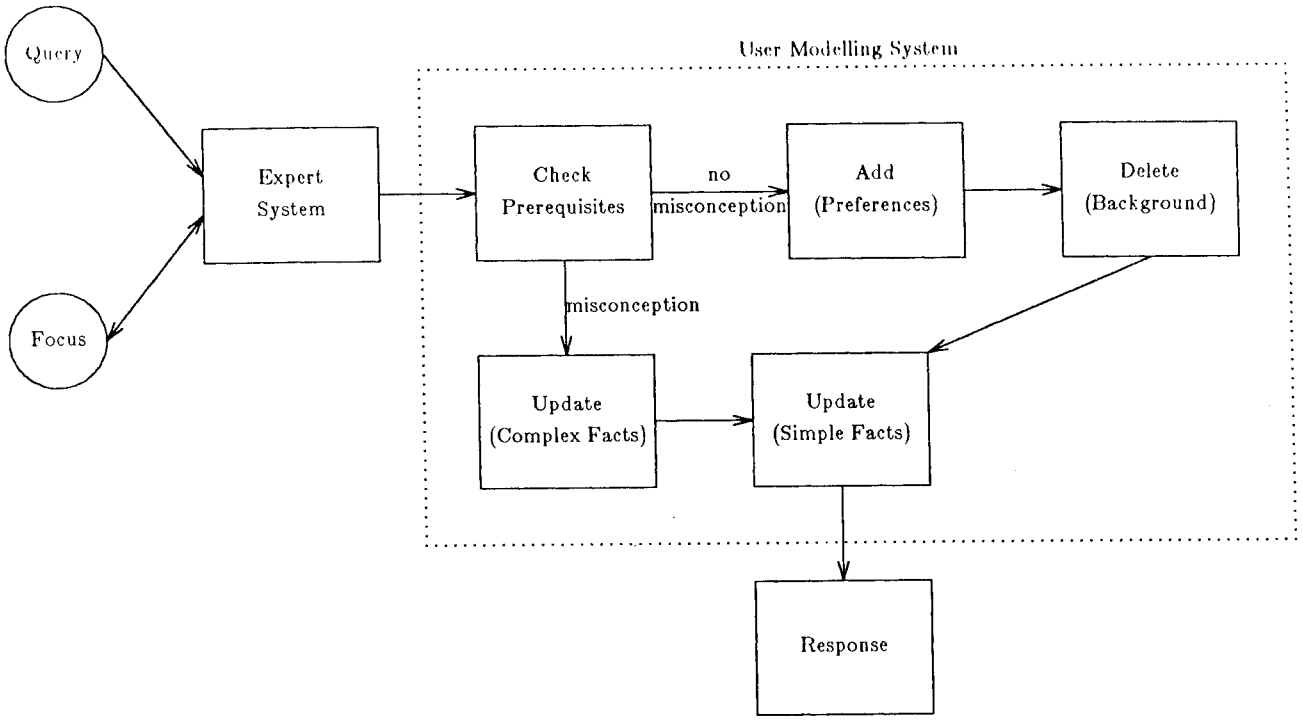

Figure 2: Detailed Block Diagram of ThUMS

avoid redundancy. The resulting explanation, which is still in internal form, is then converted to a suitable surface form for display, and output.

By design, all domain knowledge resides in the expert system, and knowledge about the users resides in the user modeling subsystem. Neither system component has access to the other's knowledge base.

4.2 ThUMS In More Detail

Unfortunately, the straightforward approach presented in the previous section turned out to be too simplistic to handle more complex examples, such as those involving misconceptions on the part of the user. Handling examples of this kind required the structure of the system to be changed as shown in Figure 2.

Initially, the user's stated goal must be derived from his input. In ThUMS this would be implemented as a two step operation. First, the input would be parsed using a simple DCG or ATN. Second, the focus of attention implied by the question is identified. If the input does not explicitly indicate a change of focus, we assume that the focus is unchanged, and the current focus is used. However, because deriving the user's stated goal is a relatively straightforward operation which is not relevant to our thesis, this component was implemented in a very limited form. ThUMS does not have the ability to detect changes in focus from the user's input, nor does it parse the stated goal to obtain the corresponding internal form. We assume that the user's goal is presented in internal form, and derive the stated goal by adding the current focus of attention.

For example, if the system's focus is currently restricted to diagnosing reading disorders, and the TOWL is a suitable diagnostic, the question

```
Question:  Shall I administer the TOWL?
```

should be posed to ThUMS as

```
Internal Form:  is_suitable(towl, diagnosed_reading_disorder)
```

In our system, we assume that during a single consultation only one student will be discussed, and therefore there is no need to explicitly identify the student in the focus as well.

Having computed the internal form of the stated goal, we now turn to the expert system to determine a suitable response. The expert system assumes that all users have the same domain goal: to diagnose the student's disability. Thus, the expert system will always respond to the stated goal by explaining how it will take the user closer to the domain goal (as identified by the current focus of attention). If the user is under a misconception, then it is possible that the stated goal

will not aid the user in fulfilling his domain goal. Initially, however, let us assume that the user's stated goal is compatible with the domain goal.

In order for a particular explanation to make sense to the user, it may be necessary for the user to have certain background knowledge. This will be expressed by the expert system in the form of *prerequisites* that have to be met for the explanation to be applicable.

When the expert system returns a response, ThUMS will verify that the user's background includes the prerequisite knowledge for the explanation to be meaningful. If these prerequisites are not met, ThUMS will force backtracking into the expert system, which is then expected to return an explanation with different prerequisites. To avoid failure, the expert system must be capable of always generating an explanation which requires no background information as a last resort.

When the expert system formulates a response to the user's stated goal, it infers some of the user's intentions. If the explanation is subsequently rejected by the prerequisite checker, this can be interpreted to mean that these intentions were incorrectly deduced by the expert system. The expert system is then responsible for re-formulating the intentions and producing an alternate explanation.

After the basic explanation has been generated, ThUMS determines the user's preferences which are relevant to his query. The expert system is called again (using a different interface) to explain how these preferences fit into the explanation it previously generated. It might respond that the preference was not a factor in determining the previous response, or explain that the user's preference was met, or that it was not met. If it was not met, the factor that over-rode the preference is identified.

Lastly, the explanation being produced is compared for redundancy with the user's background.

Facts presented in the explanation which the user already knows are deleted from the explanation. The explanation to be presented to the user has now been computed in its entirety.

The facts in the user modeling system have been classified as either simple or complex. Simple facts are those which can be understood and used by the user if conveyed to him. Complex facts are those which cannot be completely explained by ThUMS (information about how to administer diagnostic tests, for instance). If an explanation relays one or more simple facts to the user, knowledge is added to the model of this user indicating that he knows this fact. This prevents ThUMS from repeating itself in later answers.

Let us now consider the case where the user's stated goal is not relevant to the current domain goal (for instance, he asks whether to use a test which is not relevant to the student's problems). In this case, the expert system deduces that the user is lacking certain domain knowledge, and returns two responses: one explaining why the user's stated goal has no relevance to the domain goal, and another suggesting alternative steps which will fulfill the domain goal (this is similar to [Van Beek, 1986]). Together they comprise the explanation that the user will be given. The user's preferences are not addressed because they would detract from the more important task of clearing up the user's misconception.

If ThUMS uses any complex facts when explaining why the user's stated goal is not relevant to the domain goal, ThUMS can only advise him of those facts. It is not capable of explaining complex facts to him. Therefore, these complex facts are flagged as not known by this particular user.

4.3 Theorist

Theorist is a system which formulates scientific theories to explain observations. It is given a set of defaults, and a set of facts, and it attempts to explain a given observation using the facts and a subset of the defaults which is consistent with the facts. For more details, see [Poole, Goebel, and Aleliunas, 1986].

The stereotypes of our users are stored as Theorist defaults. When ThUMS tells the user a simple fact, or determines that he doesn't know a complex fact, this information is added as a Theorist fact. ThUMS can determine if the user knows information I by asking Theorist to explain that the user knows I. If Theorist finds a suitable explanation, then the user knows I; if it fails to find a suitable explanation, the user doesn't know I.

5 Examples

In each of these examples, the full algorithm, described in section 4.2, is in effect. The discussion of each example will only focus on a particular aspect of the algorithm.

5.1 User Preferences

In the first example, we will concentrate on how the user's preferences contribute to the overall explanation. The system has diagnosed a reading disorder and has recommended WISCR. The user asks "Why do you recommend WISCR?"

In the case of the psychologist, the intended goal is "Why do you recommend WISCR over a standard test, which I prefer?" ThUMS uses a special explanation facility of the expert system which extracts from the full chain of reasoning those links that refer to standard tests. The expert system returns that it considered standard tests, but rejected them in favor of WISCR which is more accurate. Thus the special explanation facility recognizes that the user's preferences are in conflict with the final recommendation, so it returns the fact that there was a conflict and that WISCR was chosen for its high accuracy. Both of the pieces of information are used in the final response, "not(standard_test) but (wiscr_is_high_accuracy_test)."

```
ex1(freud, Response);

Query why_recommend(wiscr,diagnosed_reading_disorders)
  Answer to Stated Goal: [wiscr_appropriate_for_reading]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: standard_test
  Response to Preferences: [wiscr_is_high_accuracy_test]
Final Response: [not(standard_test),but,[wiscr_is_high_accuracy_test]]
```

For the principal, the implied question is "Why do you recommend WISCR, when my preference is to use tests for which funding is available?" But in this case the recommendation was made without regard to funding because the student is not eligible to receive funding anyway. Thus the special explanation that has to do with funding indicates that funding is not applicable, and gives the reason. The response generated is "dont need to consider funding_available because [student_is_not_eligible_for_funding]."

```
ex1(bigwig, Response);

Query why_recommend(wiscr,diagnosed_reading_disorders)
  Answer to Stated Goal: [wiscr_appropriate_for_reading]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: funding_available
  Response to Preferences: [student_is_not_eligible_for_funding]
  Adding fact: knows(bigwig,wiscr_appropriate_for_reading)
Final Response: [wiscr_appropriate_for_reading,dont,need,to,consider,
funding_available,because,[student_is_not_eligible_for_funding]]
```

For the parent, tests are preferred if they impose little stress on the student. In this case, the student's stress was considered in recommending WISCR, so the special explanation facility indicates that this preference is addressed and is in accord with the recommendation. The response includes "wiscr_low_stress".

```
ex1(bruce, Response);

Query why_recommend(wiscr,diagnosed_reading_disorders)
  Answer to Stated Goal: [wiscr_appropriate_for_reading]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: low_stress
```

```
  Response to Preferences: [wiscr_low_stress]
  Adding fact: knows(bruce,wiscr_appropriate_for_reading)
Final Response: [wiscr_appropriate_for_reading,and,[wiscr_low_stress]]
```

In these three responses, we have seen three ways that the preferences may interact with the recommendation. For the parent, the preference was met by the recommendation. For the psychologist, it was not met, and for the principal, there was a reason given why it was not applicable.

In some future examples, preferences may not be addressed at all, if the special explanation facility says that the preference did not enter into the chain of reasoning that lead to the recommendation.

Apart from user preferences, in this example some background information is used to reduce the answer. Also updates are done to the parent's and principal's background, when each is told the simple fact that "WISCR_is_appropriate_for_reading".

5.2 User Background

In these responses, we will concentrate on pruning the response according to the user's background. The system has diagnosed that the student has pronunciation errors. The user asks "Could these pronunciation errors be due to dialect?"

When we suggest to the expert system that dialect may be a cause, it responds that this is possible, with two reasons: the student has a dialect and in general dialect may be a cause of pronunciation errors.

Since the user does not need or want to hear unsolicited confirmation of things he already knows, we prune out the facts that are part of his background. The parent is assumed to know that the student has a dialect. Therefore he only needs to be told that in general a dialect can lead

to mispronunciation, since this is not usually considered part of the background of the parent. The response includes "dialect_implies_errors".

```
ex2(bruce, Response);

Query is_cause(dialect,pronoun_errors,mistakes)
  Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: nil
  Response to Preferences: []
  Adding fact: knows(bruce,dialect_implies_errors)
Final Response: [yes,dialect_implies_errors]
```

The psychologist is not assumed to know specific details about the student, but is assumed to know about dialect and the possible results, such as mispronunciation. Responses to him would only include "student_has_dialect".

```
ex2(freud, Response);

Query is_cause(dialect,pronoun_errors,mistakes)
  Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: nil
  Response to Preferences: []
  Adding fact: knows(freud,student_has_dialect)
Final Response: [yes,student_has_dialect]
```

The principal is not assumed to know either details about the student or about the tests. The answer formulated for him includes "student_has_dialect" and "dialect_implies_errors".

```
ex2(bigwig, Response);
```

```
Query is_cause(dialect,pronoun_errors,mistakes)
  Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: nil
  Response to Preferences: []
  Adding fact: knows(bigwig,student_has_dialect)
  Adding fact: knows(bigwig,dialect_implies_errors)
Final Response: [yes,student_has_dialect,dialect_implies_errors]
```

After the principal is told that the student has a dialectic problem and that this may be the cause of errors, these facts are added to his background since he can be expected to understand them. There is no reason to tell him if he asks again.

```
ex2(bigwig, Response);

Query is_cause(dialect,pronoun_errors,mistakes)
  Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: nil
  Response to Preferences: []
Final Response: [yes]
```

5.3 Goals and Background

In this example, we emphasize how the user's subgoals and background may both play a role in determining the output. There is no direct interaction between them here; that discussion is postponed until the next section.

The system has diagnosed an intelligence disorder of a hearing impaired student. Thus user asks "Should I administer the Peabody Picture Vocabulary Test?"

The full response generated by the expert system is that PPVT is a good IQ test for a hearing impaired student, and that the student is hard of hearing. In responding to the principal, the system notices that his preference is to hear about funding. It also checks his background and sees that he does not know the student's background, the use of PPVT, or the eligibility of the PPVT for funding. So the final response generated is "yes, ppvt_good_for_hearing_impaired, student_is_hard_of_hearing and [ppvt_is_eligible_for_funding]".

```
ex3(bigwig, Response);

Query is_suitable(ppvt,diagnosed_iq_disorders)
  Answer to Stated Goal: [yes,ppvt_good_for_hearing_impaired,
  student_is_hard_of_hearing]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: funding_available
  Response to Preferences: [ppvt_is_eligible_for_funding]
  Adding fact: knows(bigwig,student_is_hard_of_hearing)
Final Response: [yes,ppvt_good_for_hearing_impaired,student_is_hard_of_hearing,
and,[ppvt_is_eligible_for_funding]]
```

The psychologist's response demonstrates another type of indirect interaction. The psychologist is assumed to know about PPVT and when it should be used. Based on his preference, the fact that PPVT is a standard test is added to the response, but is removed again because of his background.

```
ex3(freud, Response);

Query is_suitable(ppvt,diagnosed_iq_disorders)
  Answer to Stated Goal: [yes,ppvt_good_for_hearing_impaired,
student_is_hard_of_hearing]
  If Conflict, Additional Response: []
  PreRequisites: []
  Preference: standard_test
```

```
  Response to Preferences: [ppvt_is_standard_test]
  Adding fact: knows(freud,student_is_hard_of_hearing)
Final Response: [yes,student_is_hard_of_hearing]
```

5.4 Goals and Background with Interactions

This example demonstrates how the user's goals and background can interact to produce customized answers.

A reading disorder has been diagnosed. Prior to recommending treatment, however, further tests are required to determine the exact nature of the problem. A suitable test to administer is the TORC (Test of Reading Comprehension), which can be administered by either a resource room teacher or a psychologist. A psychologist may also choose to administer the TOWL (Test of Written Language). Used in the standard way this test will not obtain any useful information about the student's reading disorder, but psychologists can be expected to know about a special way to administer the test that allows it to gather useful information. For the purposes of this example, let us assume there is no reason for the expert system to prefer one test over the other.

A resource room teacher (Mr_Resource) and a psychologist (Freud) ask ThUMS the same question: "Should I administer the TOWL?"

```
ex4(freud, Response);

Query is_suitable(towl,diagnosed_reading_disorders)
  Answer to Stated Goal: [yes,
      towl_must_be_used_in_special_way_for_reading]
  If Conflict, Additional Response: []
  PreRequisites: [towl_in_special_way_for_reading]
  Preference: standard_test
  Response to Preferences: [towl_is_standard_test]
Final Response: [yes,towl_must_be_used_in_special_way_for_reading]
```

```
ex4(mr_resource, Response);

Query is_suitable(towl,diagnosed_reading_disorders)
  Answer to Stated Goal: [yes,
      towl_must_be_used_in_special_way_for_reading]
  If Conflict, Additional Response: []
  PreRequisites: [towl_in_special_way_for_reading]
  Answer to Stated Goal: [no,towl_appropriate_for_writing]
  If Conflict, Additional Response: [torc_appropriate_for_reading]
  PreRequisites: []
  Adding fact: not(knows(mr_resource,towl_appropriate_for_writing))
Final Response: [no,towl_appropriate_for_writing,however,
    torc_appropriate_for_reading]
```

From the stated goal, the expert system decides that the user's intent is to use TOWL in a non-standard way to diagnose reading disorders. Thus, the first explanation produced by the expert system addresses this interpretation of the stated goal. However, it can only be used in this way if a prerequisite is met: the user must be capable of administering it in this special way. Freud is capable of administering it in this way, and as a result ThUMS' response to him is simply the explanation returned by the expert system.

Mr. Resource does not have the background to enable him to administer TOWL in a special way, and therefore he could not have intended the same thing Freud did. This explanation is rejected by the prerequisite checking component, and the expert system is asked to generate a new explanation based on an alternate intention. It attempts to do so, but fails, and concludes that the user has a mistaken view that TOWL can be used to diagnose reading disorders. The response from the expert system pinpoints the user's error: it explains that TOWL is used for writing disorders, not reading disorders. The expert system also volunteers the extra information

that TORC is appropriate for reading disorders. Since this explanation has no prerequisites, it is accepted.

The last thing ThUMS does is update the stereotype of Mr. Resource with the specific knowledge that he does not know how to use TOWL.

6 Limitations

ThUMS has only been tried on the examples presented here, but we feel that the algorithm will work for all of the examples in Cohen and Jones, except for Dialog V, page 13. In that example, the expert system is expected to assimilate the results of a different test than it recommended. We did not consider dialogues where the user presents test results.

We do not show how to change the focus as the dialog progresses.

In the remainder of this section, we describe design decisions which should be reviewed before further work is done. For example, the domain facts are split into simple and complex facts. This allows ThUMS to represent the meta-knowledge that the user will not know something even after ThUMS's explanation. But it has been implemented so that a user cannot convince ThUMS he knows the fact. A fact should, perhaps, be classified with regard to the user's background. A psychologist may be able to use a fact he is told without further explanation, whereas a parent may not.

There is no allowance made for situations such as a parent with a psychology degree who should be credited with knowing most testing procedures. This is a dual stereotype where the user's background is the union of the parent's and the psychologist's. Although it would not be difficult to use two backgrounds at once, there is no easy way to know when the stereotype should be extended, or shifted from one to another.

Occasionally, the entire response to a query is already contained in the user's background. In this case, there is nothing to print. The user, presumably, asked the question for a reason, to find out some piece of information. This is weak evidence that some parts of the background are not actually known to the user. One strategy is to output the entire response, and remove knowledge

about these facts from the user's background. Another is to assume the user just wanted to confirm something; no updates to his background are required, and an indication of accordance is the only output required. We have not implemented a strategy for this case.

Although removing facts from the explanation will prevent ThUMS from telling the user what he already knows, it may remove an important fact in a chain of reasoning meant to convince the user of a subsequent fact. Thus it could destroy the coherency of an argument. We do not have a method for retaining coherence in an explanation, so this situation is not treated specially. It may be that the surface form generator can retain coherence, but to our minds, this is an open issue.

7 Conclusions

The experience gained in this implementation has convinced us that the technology we present here is both feasible and flexible. Although we have not implemented a full user modelling system, we feel our methods can be extended to do so. Such a system would not need to be related to educational diagnosis; it could be applied to any explanation-based task because we have localized the domain knowledge in the expert system.

We found that the ideas concerning background and goals presented by [Cohen and Jones, 1986] were suitable for user modelling, and easy to implement. In particular they allow different aspects of the user to be modeled. Misconceptions are detected, the user's goals are addressed and redundant output is avoided. We were able to implement these ideas in one control structure.

The experience of implementing these ideas led us to some conclusions which are not already part of the literature. One of the aspects we wanted to explore was how the goals and the background interact. In general, the goals add to the response and the background deletes from it. One observed interaction was that the background can remove parts of explanations that were added by the goals. A more direct interaction occurs when the user's intended goal depends on his background, as in example 4.

We also had to define an interface to the domain expert system which returns desired explanations. Two interfaces were required, one to obtain a general (top-level) explanation, and one to obtain an explanation specific to a particular topic. In addition, the top-level explanation generator had to be capable of formulating an alternate explanation if the user's background did not support the first one. Although most expert systems in use today cannot produce suitable explanations, we feel that an expert system embodying deep knowledge will be capable of doing so.

We felt that Theorist was well suited to user modelling. Defaults provide a useful way of expressing general stereotypes, which can then be overridden by specific facts about a particular user. In addition, Theorist provided a convenient way to organize the knowledge in the educational diagnostic domain as an inheritance hierarchy with exceptions.

There is much work yet to be done in the areas of formulating the user's plans and goals, tracking the current focus of attention, and maintaining an accurate description of the user's background.

References

[Cohen and Jones, 1986] Cohen, Robin; and Jones, Marlene. Incorporating User Models Into Expert Systems for Educational Diagnosis. *Research Report CS-86-37*, Department of Computer Science, University of Waterloo, September 1986.

[Cohen, 1987] Cohen, Robin. Varying the content of responses based on user's perspective of domain. *Unpublished draft*, Department of Computer Science, University of Waterloo, January 1987.

[Poole, Goebel, and Aleliunas, 1986] Poole, David; Goebel, Randy; Aleliunas, Romas. Theorist; a logical reasoning system for defaults and diagnosis. *Research Report CS-86-06*, Department of Computer Science, University of Waterloo, February, 1986.

[Van Beek, 1986] Van Beek, Peter. A Model for User-Specific Explanations from Expert Systems. *Research Report CS-86-42*, Department of Computer Science, University of Waterloo, September 1986.

8 Appendix A: The Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%       The psychiatrist stereotype                                            %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

default psych_knows_diagnostic_principles :
        knows( User X ) <-
                psych( User )
                diagnostic_principle( X );

default psych_knows_test_rules :
        knows( User X ) <-
                psych( User )
                test_rule( X );

default psych_knows_student_test_results :
        knows( User X ) <-
                psych( User )
                student_tests( X );

default prefers_standard_tests :
        prefers( User test standard_test ) <-
                psych( User )
                ;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%       The Resource Teacher stereotype                                        %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


default resource_teacher_knows_diagnostic_principles :
        knows( User X ) <-
                resource_teacher( User )
                simple_diagnostic_principle( X );

default resource_teacher_knows_test_rules :
        knows( User X ) <-
                resource_teacher( User )
                simple_test_rule( X );

default resource_teacher_knows_student_test_results :
        knows( User X ) <-
                resource_teacher( User )
                student_tests( X );



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%       The parent stereotype                                                  %
%                                                                              %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

default parent_knows_student_background :
        knows( User X ) <-
                parent( User )
                student_background( X );

default prefers_low_stress_tests :
        prefers( User test low_stress ) <-
                parent( User )
                ;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%       The principal stereotype                                             %
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

default prefers_funded_tests :
        prefers( User test funding_available ) <-
                principal( User )
                ;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%       Domain Knowledge                                                     %
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fact simple_diagnostic_principle( dialect_implies_errors );
fact simple_test_rule( wiscr_appropriate_for_reading );
fact test_rule( wiscr_low_stress );i
fact simple_test_rule( ppvt_is_standard_test );
fact simple_test_rule( wiscr_is_high_accuracy_test );
fact test_rule( ppvt_good_for_hearing_impaired );
fact simple_test_rule( towl_appropriate_for_writing );
fact test_rule( towl_in_special_way_for_reading );
fact test_rule( torc_appropriate_for_reading );

fact diagnostic_principle(X) <- simple_diagnostic_principle(X);
fact test_rule(X) <- simple_test_rule(X);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%       Domain Meta-Knowledge                                                %
%                                                                            %
%  We divide knowledge into two sorts:  that which can be                    %
%  conveyed in its entirety to the user, and that which cannot.              %
%  For instance, when the psychologist is told that the student              %
%  has dialectal differences, this knowledge can be used.                    %
%  On the other hand, telling a resource teacher that TOWL can               %
%  be used to diagnose reading disorders may be interesting to               %
%  him, but not useful.                                                      %
```

```
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

default n(easy_to_comprehend(X));
fact easy_to_comprehend(X) <- simple_diagnostic_principle(X);
fact easy_to_comprehend(X) <- simple_test_rule(X);
fact easy_to_comprehend(X) <- student_background(X);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%       Knowledge about the particular people involved                         %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fact student_background( student_has_dialect);
fact student_background( student_is_hard_of_hearing );

fact psych( freud );
fact parent( bruce );
fact principal( bigwig );
fact resource_teacher( mr_resource );

end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                          Explanation:                                        %
%       Generate the answer to the user's intended query                       %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

explanation( User Question Focus UserOutput ) <-
        % translate question to stated goal
        determine_stated_goal( Question Focus Query )
        printf( "Query %\n" [Query] )
        % get response to stated goal from expert system
        es( Query Conflict_Indicator Answer PosExplanation NegExplanation
            PreRequisites )
        append( Answer PosExplanation Output )
        printf( "       Answer to Stated Goal: %\n" [Output] )
        printf( "       If Conflict, Additional Response: %\n" [NegExplanation])
        printf( "       PreRequisites: %\n" [PreRequisites] )
        % make sure prerequisites for explanation are met
        check_prereq( User, PreRequisites )
        !   % avoid backtracking into Theorist
        % check for misconceptions on the part of user
        explanation_continued( Conflict_Indicator User Query Answer
                               PosExplanation NegExplanation, UserOutput )
       ;


%  This clause is for the case where no user misconception has been noted
%  (ie: query is consistent with present focus).

explanation_continued( yes, User, Query, Answer, PosExplanation, NegExplanation,
                       UserOutput ) <-
        % add information according to user preference
        relates_to_preference(User Query Pref PrefAnswer PrefExplanation)
        printf("       Preference: %\n", [Pref] )
        printf( "       Response to Preferences: % \n", [PrefExplanation] )
        % delete information already known to user (check his background)
        relevant( User PosExplanation Response )
        relevant_pref( User PrefAnswer PrefExplanation PrefResponse )
        % update user model with knowledge the user will be told
        update_user_model_with_simple_facts( User, Response )
        % produce "surface form".
        formulate_answer( [[yes, Answer, Response, NegExplanation]
                           [PrefAnswer Pref PrefResponse]],
                          UserOutput )
        printf("Final Response: %\n\n\n" [UserOutput])
        ;

%  This clause is for the case when the user's question indicates a
%  misconception
%  (ie, query is inconsistent with the focus).
%  Prefernces and background are not considered, since first the
%  misconception must be addressed.

explanation_continued( no, User, Query, Answer, PosExplanation, NegExplanation,
                       UserOutput ) <-
```

```
        % update user model with knowledge the user will be told
        update_user_model_with_simple_facts( User, PosExplanation )
        update_user_model_with_simple_facts( User, NegExplanation )
        % update user model with facts the user does not know
        % and cannot be told
        update_user_model_with_complex_facts( User, PosExplanation )
        % produce "surface form".
        formulate_answer( [[no, Answer, PosExplanation, NegExplanation]
                           [not_applicable nil []]],
                          UserOutput )
        printf("Final Response: %\n" [UserOutput])
        ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                          Determine stated goal:                              %
%                 Translate user question to stated goal.                      %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

determine_stated_goal( Question Focus Query ) <-
        % translate user's question to internal form
        translate( Question Internal )
        % append current focus
        add_focus( Internal Focus Query )
        ;

add_focus( Internal Focus Query ) <-
        functor( Internal List )
        append( List [Focus] Newlist )
        functor( Query Newlist )
        ;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                          Check_prereq:                                       %
%                 Make sure that the user has the background                   %
%                 required for an explanation to be applicable.                %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

check_prereq( User, [] );

check_prereq( User, [P|PreReqs] ) <-
        explain( [knows( User, P )], _, _ )
        check_prereq( User, PreReqs );


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                 Update_user_model_with_simple_facts:                         %
%        If the user is told a "simple" fact, update his user                  %
%        model to indicate that he knows it.                                   %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
update_user_model_with_simple_facts( User, [] );

update_user_model_with_simple_facts( User, [E|Es] ) <-
        % check whether the fact is simple
        explain( [easy_to_comprehend(E)], _, _ )
        ! % avoid backtracking into Theorist
        % add fact to user model
        fact( knows(User E), [] )
        printf( "       Adding fact: %\n", [knows(User E)] )
        update_user_model_with_simple_facts( User, Es );

update_user_model_with_simple_facts( User, [E|Es] ) <-
        % fact is complex - do not add to user model
        update_user_model_with_simple_facts( User, Es );


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%               Update_user_model_with_complex_facts:                           %
%               If the expert system determines that the user does              %
%       not know a complex fact which he should (according to his               %
%       background), then his background is updated to reflect this.            %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


update_user_model_with_complex_facts( User, [] );

update_user_model_with_complex_facts( User, [E|Es] ) <-
        % check whether the fact is complex
        explain( [n(easy_to_comprehend(E))], _, _ )
        % check whether the user is supposed to know the fact
        explain( [knows( User, E )], _, _ )
        ! % avoid backtracking into Theorist
        % add fact to user model
        fact( n(knows( User, E )), [] )
        printf( "       Adding fact: %\n", [not(knows(User E))] )
        update_user_model_with_complex_facts( User, Es );

update_user_model_with_complex_facts( User, [E|Es] ) <-
        % fact is simple, or user is not supposed to know it -
        % do not add to user model
        update_user_model_with_complex_facts( User, Es );


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%               Formulate_answer:                                               %
%               Generate pseudo-surface form                                    %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

formulate_answer( [[yes, Answer, PositiveAns|_], Prefs], Output ) <-
        formulate_preference( Prefs, OutputPrefs )
        append( Answer, PositiveAns, NewPositiveAns )
```

```
        append( NewPositiveAns, OutputPrefs, Output );

formulate_answer( [[no, Answer, PosExpl, NegExpl], Prefs], Output ) <-
        append( Answer, PosExpl, NewPosExpl )
        append( NewPosExpl, [however| NegExpl], Output );

formulate_preference( [_, _, []], [] );

formulate_preference( [yes, Pref, [Expl|Expls]], [and [Expl|Expls]] );

formulate_preference( [no, Pref, [Expl|Expls]],
                      [not(Pref) but [Expl|Expls]] );

formulate_preference( [not_applicable, Pref, [Expl|Expls]],
                      [dont need to consider Pref because [Expl|Expls]] );


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                          Relevant:                                           %
%       Delete the portions of a positive explanation which                    %
%       are already known to the user.                                         %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

relevant( _ [] [] );

relevant( User [Reason | List] Explanation ) <-
        % check whether user knows this part of the explanation
        explain( [knows( User Reason )] X Y )
        ! % avoid backtracking into theorist
        % user already knows this fact - remove from explanation
        relevant( User List Explanation )
        ;

relevant( User [Reason | List] [Reason | Explanation] ) <-
        % user does not know this part of the explanation - retain it
        relevant( User List Explanation )
        ;

% An explanation about a preference is relevant in the response if it explains
% why it wasn't met. Otherwise its relevant only if the user doesn't already
% know the reason why his preference was met.
relevant_pref(User yes Expl_In Expl_Out) <-
        relevant(User Expl_In Expl_Out);

relevant_pref(User no Expl Expl);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                Relates_to_preference:                                        %
%       Returns information related to the user's preference                   %
% Simplifying assumption:  that the user will only have one preference.        %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
relates_to_preference(User Query Pref Answer Response) <-
        % get the object of the query
        treatment(Query Treatment Type)
        % find the user's preferences concerning the object queried
        preference(User Type Pref)
        % get an explanation fomr the expert system of how the
        % user's preference relates to the query
        es_sub(Query Answer Pref Response)
        ! % avoid backtracking into theorist
        ;

% If no preferences could be found, this clause will avoid failure
% of the whole query.
% No preference response is returned.
relates_to_preference(User, _, nil, not_applicable, []) <-
        ;

% Find the user's relevant preference
preference(User Type Pref) :-
        explain([prefers(User Type Pref)] X Y)
        ! % avoid backtracking into theorist
        ;

% A treatment is something recommended
treatment(why_recommend(Treatment Focus) Treatment Type) <-
        treatment_type(Treatment Type)
        ;

% A treatment is something suitable
treatment(is_suitable(Treatment Focus) Treatment Type) <-
        treatment_type(Treatment Type)
        ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%                Domain Specific Knowledge                                      %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

treatment_type( wiscr, test );
treatment_type( ppvt, test );
treatment_type( torc, test );
treatment_type( towl, test );


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%                Expert System                                                  %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

es( is_cause( dialect pronoun_errors mistakes)
    yes
    [ yes ]
    [ student_has_dialect, dialect_implies_errors ]
    []
```

```
    []
    );

es( why_recommend( wiscr diagnosed_reading_disorders)
        yes
        []
        [ wiscr_appropriate_for_reading ]
        []
        []
        );

es( is_suitable( ppvt diagnosed_iq_disorders )
        yes
        [ yes ]
        [ ppvt_good_for_hearing_impaired, student_is_hard_of_hearing ]
        []
        []
        );

es( is_suitable( towl diagnosed_reading_disorders )
        yes
        [ yes ]
        [ towl_must_be_used_in_special_way_for_reading ]
        []
        [ towl_in_special_way_for_reading ]
        );

es( is_suitable( towl diagnosed_reading_disorders )
        no
        [ no ]
        [ towl_appropriate_for_writing ],
        [ torc_appropriate_for_reading ]
        []
        );

es_sub( why_recommend( wiscr diagnosed_reading_disorders)
        yes
        low_stress % Select low_stress information
        [ wiscr_low_stress ]
        );

es_sub( why_recommend( wiscr diagnosed_reading_disorders)
        no
        standard_test % select "standardness" of test
        [ wiscr_is_high_accuracy_test ]
        );

es_sub( why_recommend( wiscr diagnosed_reading_disorders)
        not_applicable
        funding_available % Select funding info
        [ student_is_not_eligible_for_funding ]
        );

es_sub( is_suitable( ppvt diagnosed_iq_disorders)
        yes
        standard_test % Select standard test info
```

```
        [ ppvt_is_standard_test ]
        );

es_sub( is_suitable( ppvt diagnosed_iq_disorders)
        yes
        funding_available % Select funding info
        [ ppvt_is_eligible_for_funding ]
        );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%               Translation of user's question to internal form                %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


translate( dialect_query is_cause( dialect pronoun_errors ) );
translate( recommend_wiscr_query why_recommend(wiscr) );
translate( ppvt_suitable_query  is_suitable(ppvt) );
translate( towl_suitable_query  is_suitable(towl) );


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%               Run the examples                                               %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ex1(User X) <-
        explanation(User recommend_wiscr_query diagnosed_reading_disorders X)
        ;

ex2(User X) <-
        explanation(User dialect_query mistakes X)
        ;

ex3(User X) <-
        explanation(User ppvt_suitable_query diagnosed_iq_disorders X);

ex4(User X) <-
        explanation(User towl_suitable_query diagnosed_reading_disorders X);
```

9 Appendix B: The Script

```
ex1(freud, Response);

Query why_recommend(wiscr,diagnosed_reading_disorders)
        Answer to Stated Goal: [wiscr_appropriate_for_reading]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: standard_test
        Response to Preferences: [wiscr_is_high_accuracy_test]
Final Response: [not(standard_test),but,[wiscr_is_high_accuracy_test]]


ex1(bigwig, Response);

Query why_recommend(wiscr,diagnosed_reading_disorders)
        Answer to Stated Goal: [wiscr_appropriate_for_reading]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: funding_available
        Response to Preferences: [student_is_not_eligible_for_funding]
        Adding fact: knows(bigwig,wiscr_appropriate_for_reading)
Final Response: [wiscr_appropriate_for_reading,dont,need,to,consider,funding_ava
ilable,because,[student_is_not_eligible_for_funding]]


ex1(bruce, Response);

Query why_recommend(wiscr,diagnosed_reading_disorders)
        Answer to Stated Goal: [wiscr_appropriate_for_reading]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: low_stress
        Response to Preferences: [wiscr_low_stress]
        Adding fact: knows(bruce,wiscr_appropriate_for_reading)
Final Response: [wiscr_appropriate_for_reading,and,[wiscr_low_stress]]


ex2(freud, Response);

Query is_cause(dialect,pronoun_errors,mistakes)
        Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: nil
        Response to Preferences: []
        Adding fact: knows(freud,student_has_dialect)
Final Response: [yes,student_has_dialect]


ex2(bruce, Response);

Query is_cause(dialect,pronoun_errors,mistakes)
        Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: nil
        Response to Preferences: []
        Adding fact: knows(bruce,dialect_implies_errors)
```

```
Final Response: [yes,dialect_implies_errors]


ex2(bigwig, Response);

Query is_cause(dialect,pronoun_errors,mistakes)
        Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: nil
        Response to Preferences: []
        Adding fact: knows(bigwig,student_has_dialect)
        Adding fact: knows(bigwig,dialect_implies_errors)
Final Response: [yes,student_has_dialect,dialect_implies_errors]


ex2(bigwig, Response);

Query is_cause(dialect,pronoun_errors,mistakes)
        Answer to Stated Goal: [yes,student_has_dialect,dialect_implies_errors]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: nil
        Response to Preferences: []
Final Response: [yes]


ex3(bigwig, Response);

Query is_suitable(ppvt,diagnosed_iq_disorders)
        Answer to Stated Goal: [yes,ppvt_good_for_hearing_impaired,student_is_ha
rd_of_hearing]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: funding_available
        Response to Preferences: [ppvt_is_eligible_for_funding]
        Adding fact: knows(bigwig,student_is_hard_of_hearing)
Final Response: [yes,ppvt_good_for_hearing_impaired,student_is_hard_of_hearing,a
nd,[ppvt_is_eligible_for_funding]]


ex3(freud, Response);

Query is_suitable(ppvt,diagnosed_iq_disorders)
        Answer to Stated Goal: [yes,ppvt_good_for_hearing_impaired,student_is_ha
rd_of_hearing]
        If Conflict, Additional Response: []
        PreRequisites: []
        Preference: nil
        Response to Preferences: [ppvt_is_standard_test]
        Adding fact: knows(freud,student_is_hard_of_hearing)
Final Response: [yes,student_is_hard_of_hearing]


ex4(mr_resource, Response);

Query is_suitable(towl,diagnosed_reading_disorders)
        Answer to Stated Goal: [yes,towl_must_be_used_in_special_way_for_reading
]
        If Conflict, Additional Response: []
```

```
        PreRequisites: [towl_in_special_way_for_reading]
        Answer to Stated Goal: [no,towl_appropriate_for_writing]
        If Conflict, Additional Response: [torc_appropriate_for_reading]
        PreRequisites: []
        Adding fact: knows(mr_resource,towl_appropriate_for_writing)
Final Response: [no,towl_appropriate_for_writing,however,torc_appropriate_for_re
ading]


ex4(freud, Response);

Query is_suitable(towl,diagnosed_reading_disorders)
        Answer to Stated Goal: [yes,towl_must_be_used_in_special_way_for_reading
]
        If Conflict, Additional Response: []
        PreRequisites: [towl_in_special_way_for_reading]
        Preference: nil
        Response to Preferences: []
Final Response: [yes,towl_must_be_used_in_special_way_for_reading]
```