# The Unification Tutor - An Intelligent Educational System in the Classroom

Geoffrey I. Webb
*Department of Computing and Mathematics, Deakin University, Geelong 3217*
Geoff Cumming
*Department of Psychology, La Trobe University, Bundoora 3083*
Thomas J. Richards
Kwok-Keung Yum
*Department of Computer Science, La Trobe University, Bundoora 3083*

## Abstract

The Unification Tutor is experimental Intelligent Tutoring System for the domain of the unification of Prolog terms. It demonstrates the interactive use of Feature-Based Modelling - an approach to cognitive modelling that has been presented at previous ASCILITE Conferences (Webb, 1988b.) The Unification Tutor has been used by Third Year Computer Science students at La Trobe University during September 1989. This paper describes the Unification Tutor and evaluates its performance at La Trobe.

**Keywords:**

Artificial Intelligence, Computer Assisted Teaching

## 1 Introduction

The Unification Tutor is an experimental Intelligent Educational System. The primary objective in designing the Unification Tutor has been to evaluate the FBM (Feature-Based Modelling) approach to student modelling (Webb, 1989.) This paper describes the Unification Tutor and discusses preliminary results of its use at La Trobe University during second semester 1989.

## 2 The Unification of Prolog Terms

Unification is the key procedure in the implementation of the Prolog language. If it is possible to unify two terms *X* and *Y* then it is possible to deduce *X* from *Y* (where *Y* is universally quantified and *X* is existentially quantified) using the Universal Modus Ponens law of logical deduction. Thus, unification is used to prove queries and sub goals from a Prolog program.

Prolog has three types of terms:

- *variables*, written as a word starting with an uppercase letter, such as Variable;
- *atoms*, written as a word starting with a lowercase letter, such as atom; and
- *compound terms*, written as a functor and a list of one or more *arguments*.
  A *functor* has a name which is written as a word starting with a lowercase letter.
  A functor also has an *arity* which is the number of arguments it requires. An argument to a compound term can be any term. The arguments to a term are written surrounded by brackets and separated by comments. The following is an example of a compound term: a(Compound, term, with(functor, Arity, three)).

A *substitution* is a list of *substitution pairs*. Each substitution pair has a variable as the left element and any term as the right element. The following is an example of a substitution: {A=B, X=a(Y, z)}. A substitution cannot contain the same variable on both the left and the right of substitution pairs.

A substitution can be applied to a term giving a result. The result of applying a substitution $\{V_1 = T_1, V_2 = T_2 \ldots V_n = T_n\}$ to a term *X* is the result of replacing every occurrence of $V_i$ (for $i = 1 \ldots n$) in *X* with the corresponding $T_i$. The application of a substitution *S* to a term *T* is denoted by *TS*. For example, a(X, E) {X=a(b, c), Y=d(E, F)} = a(a(b, c), E).

A term *I* is an *instance* of a term *T* if there is substitution *S* such that *I* = *TS*. For example, a(a, Y) is an instance of a(X, Y).

Any term *C* that is an instance, of both a term *X* and another term *Y* is a *common instance* of *X* and *Y*. For example, a(a, b) is a common instance of a(a, X) and a(Y, b).

A term X is *more general* than a term *Y* if *Y* is an instance of *X* but *X* is not an instance of *Y*. For example, a(X, Y) is more general than a(Z, Z).

A substitution *S* is a *unifier* for two terms *X* and *Y* if *XS* = *YS*. For example, {X=b, Y=a} is a unifier for a(a, X) and a(Y, b).

A substitution *M* is a *most general unifier* for two terms *X* and *Y* if *M* is a *unifier* for *X* and *Y* and there is no other unifier *U* for *X* and *Y* such that *XU* is more general than *XM*. For example, {X=Z, Y=Z} and {Y=X, Z=X} are both most general unifiers for a(a, X) and a(Y, b) but {X=a, Y=a, Z=a} is not.

## 3 Feature Based Modelling

Feature Based Modelling is an approach to student modelling that provides detailed analysis of an individual's understanding of a subject domain without recourse to a library of possible misunderstandings for the domain. This is achieved by analysing in terms of their significant features both the tasks being performed and the student's actions while performing those tasks. Machine learning techniques are then applied to these analyses to produce a model of the student's cognitive system at an input/output level. This model allows the tutoring system to make predictions about the student's probable actions when confronted with a given task.

FBM makes use of two types of features:

> *Task features* describe the significant features of a task and the context in which it is being performed.
>
> *Action features* describe the significant features of a student's actions while engaged in a task.

The *appropriate action features* for a task is the set of action features that describe appropriate actions for the student to perform when confronted with the task.

The *attributions* to a task is the set of action features that describe the student's actions while engaged in the task.

A *feature set* is a set of task features that fully specifies a task. Every task has a single feature set.

Every feature belongs to a *feature choice*. The features in a feature choice are mutually exclusive. An alternative nomenclature frequently used in the discipline of machine learning for a feature choice and its features are an attribute and its attribute values (Quinlan, 1986.)

The core of an FBM student model is a set of *erroneous associations*. An *association* relates a set of task features *I* to an action feature *a*. Its inclusion in the student model indicates that the system has sufficient evidence to believe that whenever a task exhibits all task features in *I*, *a* applies to the student's actions. Erroneous associations are associations that are not appropriate. Only erroneous associations are of interest for most purposes. Most use of the student model is further restricted to only considering *most general associations*. An association of *I* to *a* is most general if there is no association of a subset of *I* to *a*.

One of the major features of FBM is that it is *viewpoint independent*. A viewpoint is the operators, strategies and background knowledge that an individual applies to a domain (Wenger, 1987.) Most cognitive modelling techniques are tied to a single viewpoint by allowing only one correct model of problem solving for a domain. By modelling at a more abstract level than that of cognitive operators and strategies, FBM is not tied to a single viewpoint. FBM can accurately diagnose inappropriate problem solving in students irrespective of the viewpoint adopted.

Another major feature of FBM is that the instructional designer need not anticipate the forms of error that may occur. FBM does not require libraries of bugs (Brown & Burton, 1978) or mal-rules (Sleeman, 1982). Accurate student models can be formed even for cases in which the instructional designer has failed to anticipate the particular bug that a student has adopted.

FBM has been described in full by Webb (1989).

Although FBM has been successfully employed in a number of separate educational systems, including a lesson on English word classes (Webb 1989) and a piano scale tutor (Amato & Tsang, 1988), it has not previously been used for teaching in a problem solving domain. Nor has its effectiveness received formal evaluation. The Unification Tutor has been developed to demonstrate the application of FBM to a problem solving domain and to provide a test bed for its formal evaluation.

## 4 The Unification Tutor

The Unification Tutor examines the unification of terms from the Prolog programming language. This is a simple yet non-trivial problem solving task.

The cycle of interaction with the student is as follows.

- A problem is generated and presented to the student. Each problem consists of two Prolog terms to be unified.
- The student responds by entering a unifier for the two terms or typing ‘none’ to indicate that the two terms do not unify.
- The system provides comments in response to the student’s actions.

The Unification Tutor uses FMCD (Feature Managed Courseware Design, described in Webb, 1988.) A simple knowledge representation formalism, the feature network, is used to describe the domain. This is used for lesson management and student evaluation but not for generating instructional activities. Direct interactions with the student are managed by individually crafted procedures.

Appendix A details the features and feature choices used by the Unification Tutor.

The Unification Tutor takes full advantage of the ability provided by FMCD and FBM to be *viewpoint* independent. The Unification Tutor allows the student to adopt any viewpoint and provides comments and support that are viewpoint independent. The only exception to this rule is that one of the help facilities available to the student is a description of an algorithm for unification. This is necessarily grounded in a single viewpoint. However, the student is not required to examine this algorithm or to make use of it in any manner or form.

Table 1 shows some tasks that could be used by the Unification Tutor and their task features.

| Term 1 | Term 2 | Task Features |
|---|---|---|
| x(X, X) | x(a, Z) | TWO_COMPOUND_TERMS,<br>TERMS_DIFFERENT,<br>FUNCTORS_HAVE_IDENTICAL_NAMES,<br>FUNCTORS_HAVE_SAME_ARITY,<br>VARIABLES_PRESENT,<br>VARIABLES_DUPLICATED,<br>BINDINGS_ARE_CONSISTENT,<br>VARIABLE_DOES_NOT_ OPPOSE_SELF,<br>ALL_ARGUMENTS_UNIFY. |
| x(X, X) | x(Y, a(Y)) | TWO_COMPOUND_TERMS,<br>TERMS_DIFFERENT,<br>FUNCTORS_HAVE_IDENTICAL_NAMES,<br>FUNCTORS_HAVE_SAME_ARITY,<br>VARIABLES_PRESENT,<br>VARIABLES_DUPLICATED,<br>BINDINGS_ARE_CONSISTENT,<br>VARIABLE_OPPOSES_SELF,<br>NOT_ALL_ARGUMENTS_UNIFY. |

**Table 1: Some tasks and selected features from their feature sets.**

Table 2 shows some possible tasks, a student's response to each of those tasks and selected action features from the attributions that each response represents.

| Term 1 | Term 2 | Answer | Action Features |
|---|---|---|---|
| x(X, X) | x(a, Y) | {X=a, Y=a} | CORRECT,<br>UNIFY,<br>DO_NOT_ALLOW_MULTIPLE_BINDINGS |
| x(X, X) | x(a, Z) | none | INCORRECT,<br>DO_NOT_UNIFY |
| x(X, X) | x(a, b) | {X=a, X=Z} | INCORRECT,<br>UNIFY,<br>ALLOW MULTIPLE BINDINGS |
| x(X, X) | x(Y, a(Y)) | none | CORRECT,<br>DO_NOT_UNIFY |
| x(X, X) | x(Y, a(Y)) | {X=Y,X=a(Y)} | INCORRECT,<br>UNIFY,<br>ALLOW_MULTIPLE_BINDINGS |

**Table 2: Some tasks and student actions and their action features.**

The task and. action features employed by the tutor are described in Appendix A.

The Unification Tutor consists of seven sub-systems.

The *Feature Set Selector* consults the student model and selects a set of features that describes a problem suitable for the student to tackle. A strategy is employed that seeks to avoid presenting tasks which the student will find trivial as well as avoiding presenting tasks with which the student may not be able to cope. The main problem for such a strategy is that unifying compound terms is a recursive process requiring the unification of the term's arguments. Thus, a student's inability to process two compound terms may result either from an inability to process a particular form of compound term or from an inability to process one of the pairs of subterms. This creates a serious credit assignment problem for a modelling system trying to make sense of the student's failure to correctly process a pair of compound terms. The solution that has been adopted in the Unification Tutor is to only allow for use as pairs of subterms pairs of terms that there is reason to believe the student can correctly process in isolation. If this condition is sustained then any processing failure in dealing with a pair of compound terms will not be attributable to its subterms but rather will result from the outer compound terms and the contexts that they create when the student tackles the subterms. Thus, the credit for a processing error can be safely attributed to the outermost compound terms.

All valid combinations of task features for the domain are explicitly identified. Each of these is called a *feature set*. The feature sets are divided into three groups—*mastered*, *current* and *unavailable*. A feature set is placed in the mastered set if and only if its features are (correctly) associated with the action feature Correct and no subset of its features participates in an erroneous association The reasoning behind the first of these conditions is that if a feature set is associated with Correct then we have sufficient evidence to conclude that the student performs such tasks in an appropriate manner. The second condition seeks to prevent the system from concluding that a student treats a type of problem correctly when they perform appropriately for the wrong reasons.

The sub-tasks for a task involving a pair of compound terms are selected from the mastered set.

The current set contains all feature sets that do not describe tasks containing two compound terms and which are not in the mastered set. It also contains all tasks that do contain two compound terms, which are not in the mastered set and for which there are sufficient tasks in the mastered set to provide suitable sub-tasks for the task. For example, a feature set that contains the features **NOT_ALL_ARGUMENTS_UNIFY**, **VARIABLES_DUPLICATED**, and **BIND1NGS_ARE_CONSISTENT** would require sub-tasks that do not unify (to satisfy the former feature) and sub-tasks which unify and contain variables (to satisfy the latter two features.)

The unavailable set contains all feature sets that are not in either mastered or current.

Feature sets are selected by the Feature Set Selector at random from the set of available feature sets.

The *Task Generator* takes the feature set selected by the Feature Set Selector and generates a pair of Prolog terms that represents a problem with those features. The features specify most aspects of the terms to be generated. Names are selected at random from a set of fifty pre-specified names. Tasks that represent pairs of subterms for a compound term are selected at random from the set of mastered feature sets within the constraints imposed by the features of the outer feature set.

The *Student Interface* presents the problem to the student and accepts the student's response. A help facility is available during these interactions. This help facility provides background information and describes an algorithm for unification.

This description is the only aspect of the Unification Tutor which is not viewpoint independent. However, the student is neither required to view the algorithm or to apply it. The analysis and comments provided by the tutor in no way assume that the student is using the algorithm or a variation thereof.

The *Action Analyser* takes the student's response and generates a set of action features that describe that response. The grounds on which features are identified are included in the descriptions of the features in Appendix A.

The *Task Adviser* takes the appropriate action features and the student's action features for the task and provides comments to the student. These comments are domain model driven. That is, they are based on general assumptions about the likely causes of particular actions in particular contexts. The mechanisms used are those developed for the earlier DABIS knowledge-based tutoring system (Webb, 1988a.)

The *FBM Student Modeler* takes the feature set and the student's action features and updates the student model using the FBM methodology described above.

The *Model-based Adviser* examines the student model and provides comments to the student based on that model. This is achieved by selecting an appropriate association, describing it to the student and then encouraging the student to reconsider the manner in which they tackle such tasks. An erroneous association is considered appropriate for comment if it has not previously been commented upon more than once and if it is exhibited in the student's most recent action. An association is held to be exhibited in an action if its task features are a subset of the feature set for the task on which the student was engaged and its action feature applied to the students action. These conditions seek to ensure that associations are only commented upon when they are likely to be both salient and pertinent.

The Unification tutor provides an interesting illustration of the use of multiple representations of knowledge (Brown & Burton, 1975.) The feature network describes the domain at a level suitable for instructional management and student evaluation. The Task Generator uses procedural domain knowledge to generate tasks. The Action Analyser accesses a domain problem solver that is able to apply the student's solution to the task in order to determine if the solution is a most general unifier. It also brings a considerable body of task specific domain knowledge to bear on the problem of analysing the student's actions. The Task Adviser and Model-based Adviser will both eventually make use of an inspectible problem solver. This will be able to demonstrate to the student how to solve arbitrary unification problems.

## 4.1 The Unification Tutor in the classroom

As this paper is being written the Unification Tutor is in use by Third Year Computer Science students at La Trobe University. Three versions of the system are in use - the full system as described above; a version of the system in which the global adviser is disabled and FBM is not used to revise a decision a feature set is mastered; and a version in. which the use of FBM modelling is disabled. In this latter system, the Model-based Adviser is disabled, and the Feature Set Selector does not check whether subsets of a feature set participate in erroneous association when assigning feature sets to mastered, current and unavailable.

This latter difference is extremely subtle in its effect. Unless the student model detects an erroneous association in a student's treatment of the domain, the performance of both versions of the system will be identical. The difference lies in the manner in which the systems determine if a combination of task features is mastered. In the full system, if an erroneous association is detected then this information is used to determine that combinations of features are not considered to be mastered for which the student provides correct answers for the wrong reasons. By contrast, in the version in which modelling is disabled, a combination of features is considered to be mastered if the student provides correct answers irrespective of reason for those answers.
Students are assigned to a treatment randomly. Their experience with these systems is being monitored and it is hoped that as a result of this process it will be possible to gain insights into the strengths and weaknesses of the FBM methodology. Preliminary results suggest that there is a significant difference between the effectiveness of the systems. During tasks the students are requested to rate their confidence in their answers. After a comment on an incorrect answer the student is asked to rate the usefulness of the comment. If, on the other hand, the answer was correct, the student is asked to rate the usefulness of (he task (as no significant comment is provided.) All ratings are provided as a number between 1 and 6, 1 indicating very low usefulness or confidence and 6 indicating very high usefulness or confidence. At the time of writing only the first two treatments have been in effect. Fifteen students have used each system. The average ratings provided by the two groups to the time of writing are shown in Table 3. These results

suggest that there is considerable benefit gained through the full use of FBM. More detailed analyses of larger numbers of students will be discussed in the presentation of this paper.

| Treatment | Mean task rating | Mean confidence rating | Mean comment rating |
|---|---|---|---|
| Full system | 4.04 | 5.40 | 3.21 |
| Modified system | 3.05 | 4.74 | 2.93 |

**Table 3: Average ratings by experimental treatments**

Further factors to be evaluated are the third treatment; answers to a questionnaire and each treatment's performance on the end of year exam. Final analysis of these factors will also be discussed when the paper is presented.

Appendix B provides a transcript of a sequence of interactions with the Unification Tutor. This has been transformed into a flat format from the interactive format used on a student's terminal screen.

## 5 Conclusion

The Unification Tutor has demonstrated that FBM can be applied successfully to a problem solving domain.

Preliminary results from our formal evaluation of the system suggest that FBM can provide significant educational benefits. If the trends apparent in these preliminary results are continued then (to the best of our knowledge) FBM will be the first approach to cognitive modelling for which a formal study has demonstrated educational benefit.

### Bibliography

[1] Amato, N. H. & Tsang, C. P. (1988). **Student Modelling in a Scale Tutoring System**. *Tech. Rep. 88/5 Department of Computer Science*, The University of Western Australia, Nedlands, W.A.

[2] Brown, J. S. & Burton, R. R. (1975). **Multiple Representations of Knowledge for Tutorial Reasoning**. In D. G. Bobrow & A. Collins (Eds.), *Representation and Understanding*, Academic Press, New York, pp. 311-349 :

[3] Brown, J. S. & Burton, R. R. (1978). **Diagnostic models for procedural bugs in basic mathematical skills.** *Cognitive Science*, 2:155-192.

[4] Quinlan, J. R. (1986). **Induction of decision trees**, *Machine Learning*, 1:81-106.

[5] Sleeman, D. H. (1982). **Assessing aspects of competence in basic algebra**. In D. H. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London, pp. 185-199.

[6] Webb, Geoffrey I. (1988a) **A knowledge-based approach to computer-aided learning**, *Int. J. Man-Machine Studies*, 29, 257-285.

[7] Webb, G.I. (1988b). **Cognitive diagnosis using student attributions**. *In Proceedings of the Sixth ASCILITE Conference*, Canberra, pp. 502-514.

[8] Webb, Geoffrey I. (1989) **Feature-Based Cognitive Diagnosis, EXCALIBUR** *Technical Report 14, La Trobe University*, Bundoora, Vic.

[9] Wenger, E. (1987) *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann, Los Altos, CA.

## Appendix A

The following is a list of the feature choices employed by the Unification Tutor. Each feature choice contains two or more features.

### Task Features

**TERM_TYPES**

**TWO_VARIABLES** — The terms are both variables.

**ONE_VARIABLE_ONE_ATOM** — One term is a variable, the other an atom.

**ONE_VARIABLE_ONE_COMPOUND_TERM** — One term is a variable, the other a compound term.

**TWO_ATOMS** — Both terms are atoms.

**ONE_ATOM_ONE_COMPOUND_TERM** — One term is an atom, the other a compound term.

**TWO_COMPOUND_TERMS** — Both terms are compound terms.

**IDENTICAL_TERMS**

**TERMS_IDENTICAL** — Both terms are identical.

**TERMS_DIFFERENT** — The terms are different.

**FUNCTOR_NAMES**

This feature choice applies only to tasks containing two compound terms.

**FUNCTORS_HAVE_IDENTICAL_NAMES** — The functors have identical names.

**FUNCTORS_HAVE_DIFFERENT_NAMES** — The functors have different names.

**FUNCTOR_ARITIES**

This feature choice applies only to tasks containing two compound terms.

**FUNCTORS_HAVE_SAME-ARITY** — The functors have identical arities.

**FUNCTORS_HAVE_DIFFERENT_ARITY** — The functors have different arities.

**DUPLICATED_VARIABLES**

This feature choice applies only to tasks containing two compound terms.

**VARIABLES_NOT_DUPLICATED** — A single variable does not appear more than once in either term.

**VARIABLES_DUPLICATED** — A single variable appears more than once in at least one of the terms.

**CONSISTENT_BINDINGS**

This feature choice applies only to tasks containing two compound terms with duplicated variables.

**BINDINGS_ARE_CONSISTENT** — All of the terms opposite a single variable unify with each other.

**BINDINGS_ARE_INCONSISTENT** — Not all of the terms opposite a single variable unify with each other.

**VARIABLE_PRESENCE**

**VARIABLES_PRESENT** — At least one of the terms contains at least one variable.

**VARIABLES_NOT_PRESENT**— Neither term contains a variable.

**VARIABLE_ OPPOSITION**

This feature choice applies only to tasks containing two compound terms with duplicated variables.

**VARIABLE_OPPOSES_SELF** — A variable within a compound term appears opposite a second variable which appears again within a compound term that is opposite another occurrence of the first variable. For example a(X. X), a(Y, b(Y)).

**VARIABLE_DOES_NOT_OPPOSE_SELF** — This applies to a pair of terms if VARIABLES_DUPLICATED applies and VARIABLE_OPPOSES_SELF does not apply.

**ARGUMENTS_UNIFY**

This feature choice applies only to tasks containing two compound terms.

**ALL_ARGUMENTS_UNIFY** — All the arguments of the two terms unify.

**NOT_ALL_ARGUMENTS_UNIFY** — Not all of the arguments of the two terms unify.

## Action Features

**TERM_BINDING**

**BIND_TERM** — This feature applies to a students action if s/he places an atom or compound term on the left of a substitution pair. It is never appropriate.

**DO_NOT_BIND_TERM** — This feature applies to a student's action if s/he provides a unifier and does not place an atom or compound term on the left of a substitution pair. It is appropriate for every task that consists of terms that unify.

**NON_EXISTANT_VAR_BINDING**

**BIND_NON EXISTANT_VARIABLE** — This feature applies to a student's action if s/he includes a substitution for a variable that is not in either term. It is never appropriate.

**DO_NOT_BIND_NON_EXISTANT_VARIABLE** — This feature applies to a student's action if s/be provides a unifier and does not include a substitution for a variable that is not in either term. It is appropriate to all tasks whose terms unify.

**UNIFIES**

**UNIFY** — This applies to a student's action if s/he enters a unifier as a solution.

**DO_NOT_UNIFY** — This applies to a student's action if s/he enters 'none' as a solution.

**VARIABLES_ON_LEFT_AND_RIGHT**

**ALLOW_VARIABLES_ON_THE_LEFT_AND_THE_RIGHT**— This applies to a students action if s/he enters a unifier that contains the same variable on both the left of a pair and the right of a pair. It is not appropriate for any task.

**DO_NOT_ALLOW__ON_THE_LEFT_AND_THE_RIGHT** — This applies to a student's action if s enters a unifier that does not contain the same variable on both the left of a pair and the right of a pair. It is appropriate for every task whose terms unify.

**MULTIPLE_BINDINGS**

**ALLOW_MULTIPLE_BINDINGS** — This feature applies to a students action 1ff s/he enters a unifier that contains two or more substitution pairs with the same variable on the left. It is never appropriate.

**DO_NOT_ALLOW_MULTIPLE_BINDINGS** — This feature applies to a student's action 1ff s/he enters a unifier and it does not contain two or more substitution pairs with the same variable on the left. It is always appropriate for tasks whose terms unify.

**OVERSPECIALIZATION**

**OVERSPECIALIZE_UNIFIER** — This feature applies to a student's action if no other error is detected and s/he provides more substitutions than are contained in the MGU for the task. It is never appropriate.

**DO_NOT_OVERSPECIALIZE_UNIFIER** — This feature applies to a students action s/he provides a unifier and OVERSPECIALIZE_UNIFIER does not apply to the action. It is always appropriate for tasks whose terms unify.

**CORRECTNESS**

This feature choice applies to all tasks. Its function is to record whether the student has treated the task correctly. Thus, CORRECT is appropriate for all tasks.

**CORRECT** — The student has responded to the task correctly.

**INCORRECT** — The student has not responded to the task correctly.

## Appendix B

The following is a transcript of a short session with the Unification Tutor. Student input is underlined.

Comments generated by the Model-based Adviser are enclosed in boxes.

---

```
Consider the following two terms

W

W

Enter the most general unifier for these terms or type none, help or exit.

=>{}

Well done!


Press space to continue.
```

---

```
Consider the following two terms

Result
W

Enter the most general unifier for these terms or type none, help or exit,

=>none

These terms do unify. Two terms unity if there is a unifier for them, A unifier
for two terms is a substitution that when applied to each term provides the same
result. If the substitution (W=Result) is applied to either of the two terms it
results in Result.


Press space to continue.
```

---

```
Consider the following two terms

k(SubList, SubTerm)

a

Enter the most general unifier for these terms or type none, help or exit.
=>none

Yes, well done!


Press space to continue.
```

---

```
Consider the following two terms

Z

Lst

Enter the most general unifier for these terms or type none, help or exit.
=>none

These terms do unify. Two terms unify if there is a unifier for them. A unifier
for two terms is a substitution that when applied to each term provides the same
result. If the substitution {Lst = Z} is applied to either of the two terms it
results in Z.


Press space to continue.
```

```
Consider the following two terms

W
x(Elt, lst)

Enter the most general unifier for these terms or type none, help or exit.
=>none

Excellent!


Press space to continue.
```

---

```
Consider the following two terms

Result

Parent

Enter the most general unifier for these terms or type none, help or exit.

=>none

These terms do unify. Two terms unify if there is a unifier for them. A unifier
for two terms is a substitution that when applied to each term provides the same
result. If the substitution {Parent = Result} is applied to either of the two
terms it results in Result.

It appears to me that whenever you examine two terms that are different you
state that the terms do not unify.

Perhaps you should reconsider how you tackle such problems.


Press space to continue.
```

---

```
Consider the following two terms

sandy

ancestor

Enter the most general unifier for these terms or type none, help or exit.
=>none

Excellent!


Press space to continue.
```

---

```
Consider the following two terms

m

Result

Enter the most general unifier for these terms or type none, help or exit.

=>{m=Result}

Your answer contains a substitution pair which does not have a variable on the
left m = Result. A substitution pair indicates a term that should be substituted
for a variable. The variable must appear on the left of the pair.


Press space to continue.
```

---

```
Consider the following two terms

Result

Member

Enter the most general unifier for these terms or type none, help or exit.

=>none

These terms do unity. Two terms unify if there is a unifier for them. A unifier
for two terms is a substitution that when applied to each term provides the same
result. If the substitution {Result=Member} is applied to either of the two
terms it results in Member.

It still appears to me that when you examine two terms that are different you
state that the terms do not unify.

You really should think more carefully about how you answer such problems.


Press space to continue.
```