**KB-Teach: A Rewriting Based Automated Theorem Prover for Didactic Purposes**

M. Kreitmayr†, C. Mareco†, A. Paccanaro† 

†Laboratorio de Electrónica Digital  
Universidad Católica "Nuestra Señora de la de Asunción"  
Asunción, Paraguay  
Tel: +595-21-334650  
Fax: +595-21-310587  
e-mail: apaccana@ledip.py

†Dipartimento di Scienze dell'Informazione  
Università degli Studi di Milano  
Milano, Italy

**Abstract**

Some elementary concepts on Equational Theories and Term Rewriting Systems are introduced, followed by a brief review on the main ideas of the Knuth-Bendix completion procedure. Then KB-Teach, an automated theorem prover for Equational Theories based on the Knuth-Bendix algorithm is presented, through some examples of use. Special care was taken in designing the interface of KB-Teach since the system is meant mainly for didactic purposes. Its design was realized taking into consideration the main problems encountered by the novice student when first facing Term Rewriting Systems and so far it has proven to be very helpful on the classroom battlefield.

1. An Informal Introduction to Term Rewriting Systems.

In the following we shall present a brief, informal introduction to Term Rewriting Systems and to the Knuth-Bendix Completion Procedure, which was suggested as a mean of taking an axiomatization of an Equational theory and generating a rewrite system that can be used to decide questions of validity of identities in the theory. For a complete review on the topic see Dershowitz[1989], Hsiang et al.[1987].

Reasoning with equations is very common in every-day mathematics, since many important mathematical theories (monoids, groups, ...) can be expressed as sets of equations. For example the Group Theory can be expressed by the following set of equations:

\[
\begin{align*}
    f(e, x) &= x & \text{(left identity)} \\
    f(i(x), x) &= e & \text{(left inverse)} \\
    f(f(x, y), z) &= f(x, f(y, z)) & \text{(associativity)}
\end{align*}
\]
it exists. In the following, the reader is supposed to be acquainted with some basic concepts and terminology of mathematical logic such as term, subterm, substitution, and term unification.

The Knuth-Bendix Algorithm

To obtain a Noetherian system the Knuth-Bendix method uses an ordering relation over terms, closed under substitution, that is \( t > s \Rightarrow \forall \sigma t\sigma > s\sigma \) and chooses to transform the equation \( s = t \) in the rewrite rule \( s \rightarrow t \) only if \( s > t \). In this way, it is easy to see that a sequence of reduction steps generates a decreasing sequence of terms, and therefore its termination is granted. Ordering relations are based on assigning some static coefficients (typically called weight and index for the ordering originally proposed by Knuth and Bendix) to the function symbols which appear in the theory. The set of such coefficients for a given TRS is called Signature of the TRS. Unfortunately, all the ordering relations found so far are partial (Bachmair et al [1986], Dershowitz [1982]); thus, there exist non-comparable pairs of terms and therefore non-orientable equations. This constituted the biggest limitation of the original method, which has been partially solved by later refinements (Buchberger et al [1983], Dershowitz [1989], Hsiang et al [1987]).

Knuth and Bendix gave a condition which is necessary and sufficient for a Noetherian TRS to be confluent. First let us introduce the following definition:

**Critical Pair:** given a TRS \( R \) and two rules \( h \rightarrow r_1 \) \( l_2[r_1]\sigma \neq r_2\sigma \) and \( l_2[u] \rightarrow r_2 \) in \( R \), where \( u \) is not a variable, if there is a (most general) unifier \( \sigma \) such that \( h\sigma = u\sigma \) then \( <l_2[r_1]\sigma, r_2\sigma> \) is called a critical pair of \( h \rightarrow r_1 \) and \( l_2 \rightarrow r_2 \). A critical pair is divergent if \( l_2[r_1]\sigma \neq r_2\sigma \).

Knuth and Bendix proved the following theorem:

**Theorem:** a Noetherian TRS is confluent iff there is no divergent Critical Pair among rules.

Finally they suggested a process for transforming a non-canonical TRS into a canonical one, when it exists. This method consists in incrementing the system by converting divergent critical pairs into new rules and is called the Knuth-Bendix Completion Procedure.
Let us see a schema of the algorithm:

orient equations
repeat
\((\alpha,\beta) := \text{critical pair generated from } R; \text{ (overlapping algorithm)}\) 
\((\alpha0,\beta0) := (\alpha,\beta) \text{ reduced in normal form respect to } R;\)
if \(\alpha0>\beta0\) then
\(R := R \cup \{(\alpha0,\beta0)\}\) \{ add a new axiom \}
else if \(\alpha0<\beta0\) then
\(R := R \cup \{(\beta0,\alpha0)\}\) \{ add a new axiom \}
else if \(\alpha0=\beta0\) then
exit (failure); \{ \(R\) cannot be completed \}
foreach \((\lambda,\rho) \in R\) do \{ axiom revision loop \}
\(R := R - \{(\lambda,\rho)\};\)
\((\lambda0,\rho0) := (\lambda,\rho) \text{ reduced in normal form respect to } R;\)
if \(\lambda0>\rho0\) then
\(R := R \cup \{(\lambda0,\rho0)\}\)
else if \(\rho0>\lambda0\) then
\(R := R \cup \{(\rho0,\lambda0)\}\)
else if \(\lambda0=\rho0\) then
exit (failure); \{ \(R\) cannot be completed \}
end;
until every critical pair has been considered;

If the procedure terminates successfully we obtain a complete TRS which can decide for the word problem for the theory, i.e. the problem of deciding if two terms are equal in a given equational theory. The procedure can also loop forever, due to the fact that the TRS might have an infinite number of rewrite rules.

Notice that even if the algorithm does not terminate we still have a semi-decision procedure for the word problem of the theory. This happens because if the problem \(s=t\) is true in the theory, we will eventually generate enough rules that can reduce both terms to an identical form.

Examples of canonical systems of interesting theories can be found in Hullot[1980].
2. SYSTEM PRESENTATION

The central idea of the development of KB-Teach was to build a learning tool to help students grasp the concepts of the Knuth-Bendix algorithm by showing those concepts in action in a concrete manner. The system consists of two parts: the kernel, and the Graphical Interface. Both were developed using Turbo Pascal on the PC. The kernel consists of a series of modules that perform the various functions needed by the Knuth-Bendix algorithm, such as unification, critical pair generation, rule simplification, etc. The most important part of KB-Teach is its interface, since the primary objective of the project was visibility, that is, providing a tool which would show every detail of the Knuth-Bendix procedure. This allows the user to get 'into' the process and see exactly what the program is doing at each step and hopefully, learn from it. KB-Teach's interface is a full-fledged Windows application with multiple windows, dialog boxes, and a comprehensive help system. Part of this work was based on KLab, a theorem prover developed at the Department of Computer Science of the University of Milan.

The system operates in two clearly differentiated modes: Edit and Execution.

a) Edit Mode

When the program is first executed, it enters the Edit mode. In this mode, the user is provided with facilities for editing the equational theory, and signature table, as well as loading, saving and printing the theory. An optional Message window is also present at all times, giving the user directions on what to do next and also general explanations and error messages.

Only the Equational Theory and Message windows are open at first, indicating that an Equational Theory must be entered or loaded from a file. The Equational Theory can be edited in the Equational Theory window, using the usual Copy-Paste facilities of the Windows environment. A default Signature table for the Equational Theory is also created from the contents of the Equational Theory window.

In order to develop a system useful for didactic purposes, every step of the program corresponds to a step in the Knuth-Bendix algorithm. KB-Teach tries to guide the user through these steps, explaining what needs to be done next by means of an explicit message, or by enabling only the appropriate options at the appropriate times.

The first step of the Knuth-Bendix procedure is to orient the equations in order to convert equational theory into a Term Rewriting System. This can be done by selecting Action | Orient Equations from the menu, the only item enabled at that moment. After checking the theory, the program asks whether it should use the default signature table created by the system, or let the user modify it.

If the Equational Theory can be oriented, then the corresponding TRS is displayed on the TRS window. In the following, the examples will be taken from the process of completing the group theory presented in the preceding section. At the stage just described above, the screen would appear as in fig. 1:
At this point, the program automatically changes to Execution Mode.

b) Execution Mode

This mode allows the user to do two things: complete the theory entered and simplify a term with respect to the current TRS. Each option will be discussed separately.

Completion

This process completes the theory according to the Knuth-Bendix algorithm. This can be done in two modalities: Manual, and Automatic.

Automatic Mode is the one usually found in most theorem provers: the user selects a certain number of options that are then used by the theorem prover for completing the theory without interacting with the user. The automatic completion mode implemented in KB-Teach uses a FIFO selection strategy and the ordering over terms originally proposed by Knuth and Bendix (Knuth et al. [1970]). During execution in this mode, an Output window appears giving the selected information on how the process is being carried out. The operation can be aborted at any time by choosing Action | Abort. In this case, the system goes back to Edit mode. The kernel running in automatic mode is not lightning-fast or extremely efficient, neither it was meant to be: due to the educational bias of the project, most attention was devoted to creating an intuitive, user guided manual mode.

Manual mode: In this mode, no axiom selection strategy is available, rather, it is in the user's hands. This means that he has to choose the axioms
which will be tested for overlapping and critical pair generation. Rules to be overlapped are selected by the user just by clicking on them with the mouse on the TRS window. By choosing at each step, the axioms that will generate the critical pairs, the user has complete control over the theory completion process, and can effectively direct the process from start to finish.

After the user has selected two axioms, overlapping for those axioms is tested. If no critical pairs were generated by the selected axioms, a 'beep' is played and the selection is discarded. On the other hand, if any critical pairs were generated, a 'critical pair' dialog box is displayed as shown in fig. 2.

![Critical Pair Dialog Box](image)

**Figure 2**

This dialog box shows both axioms, highlighting the overlapping subterm as well as the critical pair. When the Simplify button is pressed, the simplified critical pair is shown. If more than one critical pair exists (as indicated by Cp # 1 / N, where N is the total number of critical pairs), the user can move through them using the << (previous) and >> (next) buttons. By showing details such as the overlapping subterm and the substitution, the user can actually see the inner workings of the algorithm, and at the same time, check whether he obtains the results he expected. This is an important help for a student who is just starting to study the subject, and a way for him to verify his understanding of it.

The next step, according to the Knuth-Bendix algorithm, is that, if any axioms were added to the theory, it is necessary to check their effect on the TRS. Therefore another dialog box is presented to the user, showing the previous state of the TRS, along with the new axioms generated. (fig. 3)
This dialog box tells the user that it is necessary to simplify the new theory, and asks him whether he would like to see the simplification of the theory step by step. If the user presses the Simplify button, the program automatically simplifies the TRS all at once and the new TRS is presented. If the user presses the No button, the simplification is done anyway, but the result is not shown. If Yes is selected, however, yet another dialog box appears showing the theory (fig. 4).

Here the user can select an axiom and see the outcome of its simplification respect to the current theory. Also, the dialog informs whether the simplified axiom changed or not, and if it was then inserted or deleted from the theory. This provides a way to see the effect of the new axiom(s) on the theory by showing which ones are deleted, inserted, or changed. After the theory has been simplified, the whole process is repeated, that is, two axioms are selected, then the critical pairs dialog is displayed and so on.
Simplification

At any stage of the execution mode, the user can simplify a term respect to the current TRS. This activity can again be performed in two ways: manual, and automatic. In the latter, the user enters a term and the system simplifies it respect to the whole theory automatically, showing the resulting term. In manual mode, however, it is the user who must select the axiom with which the system will try to simplify the term. When the user selects Action\Term Simplification|Manual, the dialog box shown in figure 5 appears.

![Figure 5](image)

In this dialog, the user must enter a term and then select an axiom by double-clicking on it. If the LHS of the axiom in question does not match the term, a beep is played, just like the manual completion mode. If, however, the term is simplified by the axiom, the dialog box of figure 6, showing the overlapping subterm, and the simplified term is displayed.

![Figure 6](image)

At this point, the user may want to continue reducing the term, which he can do by pressing the Keep reduction button. This will cause the Term simplification dialog box to reappear. This time, however, its Term field will contain the previously reduced term \( f[X,Y] \) in the example shown. In this
way, the user can reduce it even further, if that is possible, by selecting another axiom. The manual option was provided to let the user test his own paths of simplification, and experiment with them.

3. FUTURE ENHANCEMENTS - THE HISTORY WINDOWS.

Future versions of the system will provide a History window, both for completion and simplification mode.

The completion history window will show a tree, where each node will represent a TRS derived from its parent by choosing the pair of axioms shown as labels of the child. The starting TRS is located at the root. Let us see how the History window should appear (fig. 7):

![Figure 7](image)

In the last figure, the original TRS is at the root. After selecting the axioms 3 and 3, a new TRS is derived from the original one, and it is represented as the node labeled 3-3. We then could have chosen the axioms 3 and 2, generating the 3-2 node and so on. At this point we could change our minds and decide to try another path. To do that, we just click on another node (the root in this example), and select the axioms 3-2, then 1-4 and 5-2 to get to the state shown in the figure. The reason for keeping a history of the changes in the TRS is to be able to identify the different paths taken to get to a given TRS, and in general, learn to direct the search for proving a theorem and/or deriving a certain rule. At any moment, the user may backtrack simply by choosing from the History window a node and continue the process from that point in a different direction.

Similarly, the Simplification mode will provide its own History window, which works very much like the one used in the completion process, that is, the original term is the root of a tree, and each child is the result of simplifying its parent respect to the axiom that appears as label of the child. Here again, the user can select a node at any moment, and continue from that point in the direction he wants.
4. CONCLUSIONS

In this paper, we presented KB-Teach, a rewriting based automated theorem prover for didactic purposes. Special emphasis was given to the didactic features of the system, and the reasons for their inclusion. KB-Teach is currently being used by students of the Artificial Intelligence course to reinforce their theoretical concepts of term rewriting systems. So far, the results have been encouraging, and due to feedback from the students, the system undergoes periodical revisions and enhancements.

REFERENCES


N. Dershowitz [1982], "Orderings for Term Rewriting Systems", J. of Theoretical Computer Science 17, pp. 279-301.

N. Dershowitz [1989], "Completion and its applications", in Resolution of equations in algebraic structures, pp. 31-48.

J. Hsiang, M. Rusinovitch [1987], "On Word Problems in Equational Theories", 14th Int. Conf. on Automata Languages and Programming, Karlsrhue

