ASSOCIATION FOR AUTOMATED REASONING

NEWSLETTER

No. 36 April 1997

From the AAR President, Larry Wos...

Featured in this newsletter are two articles. The first, by Maria Paola Bonacina, discusses the relatively new area of strategy analysis. Bonacina proposes six problems that I encourage AAR researchers to solve. This is not a contest (for that, I suggest you enter one of the contests in CADE-14), but a challenge: success in answering such questions may advance our field. The second article, by William McCune, presents two programs written in Java to implement automated deduction GUIs. McCune notes that the programs may be useful to beginning logic students and to those trying to make sense of Otter proofs.

Reminder: Early CADE registration ends May 12. The CADE URL is

http://www.cs.jcu.edu.au/ cade-14/

IMPORTANT ANNOUNCEMENT: AAR Newsletter to Be Distributed via the Web

Starting with the next issue, the *AAR Newsletter* will be distributed via the Web rather than through paper copies. The URL is http://www.mcs.anl.gov/AAR/. The newsletter will be available in html form and Postscript form, for the convenience of the reader. (For those who do not have reliable Web access, we can arrange to send the newsletter by e-mail; contact pieper@mcs.anl.gov.)

As always, AAR members will continue to be eligible for a substantially reduced subscription to the *Journal of Automated Reasoning* as well as a substantial deduction in the registration fee for the (now annual) CADE conference.

We will put articles on the Web as they are accepted. Periodically (approximately four times a year) we will send out a notice by e-mail that a new "issue" (identified by number) is on the Web. Our intention here is twofold: to have the newsletter maintain currency, and to provide a means of citing a specific issue.

Having the newsletter on the Web will provide contributors with far greater flexibility and capabilities. For example, one will be able to include links to projects for more detailed description or to extended proofs or additional challenge problems. The Web will also allow AAR newsletter contributors to use color graphics and multimedia facilities.

AAR members are encouraged to contribute to this new AAR newsletter. Submissions are invited in html (standard LaTeX is also acceptable). Please send them to our editor Gail Pieper,

pieper@mcs.anl.gov.

A Note on the Analysis of Theorem-Proving Strategies

Maria Paola Bonacina
(bonacina@cs.uiowa.edu)

In recent work [3] David Plaisted proposed an approach to analyze *the complexity of theorem-proving strategies*, and applied it to *Horn propositional logic*. Most of the research in complexity of theorem proving studies the complexity of propositional proofs, while most of the work on search concentrates on the design of heuristics.[1] To my knowledge, Plaisted's approach has been the first one to treat the complexity of *searching for a proof.* This note is a comment on this ground-breaking work. The framework of [3] is a starting point to discuss open problems that need to be addressed in order to analyze theorem-proving strategies in first-order logic. These include the formalization of the search plan, the representation of contraction, the duality of forward and backward reasoning, and the definition of complexity measures for infinite search spaces.

A theorem-proving strategy is made of an inference system and a search plan. The inference system defines the possible inferences, and the search plan selects at each step of a derivation the inference rule and the premises for the next step. *Contraction-based strategies* include the strategies that integrate resolution, Knuth-Bendix completion, and term rewriting. These strategies work primarily by forward reasoning. They use *expansion rules* such as resolution to generate consequences from existing clauses. Forward-reasoning methods do not generally distinguish goal clauses from the others, seeking to obtain a contradiction from the whole set. Generated clauses are kept and used for further inferences, so that the strategy works on a *database of clauses.* A major source of redundancy for these methods is that they generate and retain clauses that do not contribute to proving the theorem. Contraction-based strategies counter this problem by using *contraction rules*, such as simplification and subsumption, to delete redundant clauses. Since the inference system features multiple inference rules, and generated clauses are kept, creating many possible choices of premises, the search plan plays an important role. Forward-reasoning strategies admit various search plans, which sort the inferences by different criteria. Contraction-based strategies employ *eager-contraction* search plans that give priority to contraction over expansion, in order to maintain the database minimal with respect to the contraction rules.

Subgoal-reduction strategies include model elimination, tableau-based methods, Prolog technology theorem proving, the MESON strategy, and the problem-reduction-format strategies. These strategies work by reducing the goal to subgoals. Each inference step involves the current goal and at most another premise. The current goal is initially a selected input clause and then the most recently generated goal. If no rule applies to the current goal, the strategy backtracks to the previous one. Since generated clauses are goals retained for backtracking, the strategy works on a *stack of goals.* Since there is no database, the application of contraction is limited, and the search plan is fixed, usually *depth-first search with iterative deepening.* A source of redundancy for these strategies is that by focusing on the most recently generated goal, they may reduce repeatedly the same subgoals. Subgoal-reduction methods counter this problem by using techniques for *lemmaizing* or *caching* that enable them to keep track of already solved or failed subgoals.

[1] See [3] and [1] for most references relevant to this note.

Plaisted's Approach to the Analysis of Strategies

The model of the search space

A theorem-proving strategy is formalized in [3] as a 5-tuple $< S, V, i, E, u >$, where S is a set of states, and E is a set of pairs of states, in such a way that $< S, E >$ forms a directed graph. The component V is a set of labels for the states in S. The component i is a mapping from the set of input clauses to S, which identifies the initial state(s). The component u is a function from S to $\{true, false\}$, where $u(s) = true$ if and only if s is a state where unsatisfiability is detected. It is assumed that u is an almost trivial test, such as recognizing that a set of clauses contains the empty clause. A more precise definition of V and u depends on the specific strategy. A function *label*, such that $label(s)$ denotes the label of state s, is also used. It is required that no two distinct edges (s_1, t_1) and (s_2, t_2) in E have $t_1 = t_2$. This implies that the directed graph is a set of *trees*. A strategy is said to be *linear* if for all states s, there is a unique state t such that $(s, t) \in E$, that is, every state has unique successor.

Given a strategy $G =< S, V, i, E, u >$ and an input set of clauses R, a state $s \in S$ is *reachable* from R if there is a path from a state in $i(R)$ to s. $S(R)$ denotes the subset of S containing all states that are reachable from R, and $E(R)$ denotes the restriction of E to $S(R)$. Then, $G(R) =< S(R), E(R) >$ is the search space for the theorem-proving problem R according to the strategy G.

For *resolution*, the labels of the states are finite sets of clauses. If R is the input set of clauses, and $s_0 \in S$ is the state with label R, the input function $i: R \rightarrow S$ is the function such that $i(\psi) = s_0$ for all $\psi \in R$. In other words, all input clause are mapped to a single initial state, whose label is the set of input clauses. It follows that $G(R)$ is *a tree* with R as the label of the root. An arc connects state s to state t if the label of t is the union of the label of s and all the resolvents in s according to the inference system. This means that each state has a unique successor, that is, the tree $G(R)$ for resolution degenerates to *a list*. Accordingly, all resolution-based strategies are *linear* in [3]. The function u is defined by $u(s) = true$ if and only if the label of s contains the empty clause.

For *model elimination*, each state is labeled by a single chain[2], which is the current goal in that state. If $R = \{\psi_1, \ldots, \psi_n\}$ is the input set of clauses, there are states $s_1, \ldots, s_n$ with labels $\{\psi_1\}, \ldots, \{\psi_n\}$, respectively, and $i(\psi_j) = s_j$, for all $1 \leq j \leq n$. In other words, there is an initial state for each input clause. This captures the fact that any input clause can be chosen as the initial chain. It follows that $G(R)$ is *a set of trees*, one for each possible initial chain. An arc connects state s to state t if the chain labeling t is generated from the chain labeling s in one ME-step (ME-extension or ME-reduction or ME-contraction). The function u is defined in the same way as for resolution. The strategies based on model elimination are *not linear*, because an arc represents a single step, and more than one step may be applicable to a chain (e.g., ME-extension steps with different input clauses).

[2] A *chain* is a sorted clause with plain literals, called B-literals, and framed literals, called A-literals for ancestors.

The complexity measures to evaluate and compare the strategies

The measures of complexity of search proposed in [3] aim at measuring *the total size of the search space*. The total size of $G(R)$ is defined as $||G(R)|| = \sum_{s \in S(R)} |label(s)|$. For resolution, the labels are finite sets of clauses, and therefore $||G(R)||$ is the sum of the cardinalities of the sets of clauses labeling the nodes of $G(R)$. For model elimination, the labels are singleton sets, so that $||G(R)||$ is equal to the number of nodes in $G(R)$.

The measure $||G(R)||$ is refined into three measures, called *duplication by iteration*, *duplication by case analysis* and *duplication by combination*. Duplication by iteration is the maximum length of a path in $G(R)$. Duplication by case analysis is the maximum size of a subset of $S(R)$ no two elements of which are on the same path. Duplication by combination is the maximum cardinality of a label of a state in $S(R)$, i.e., $max_{s \in S(R)}|label(s)|$. Since $G(R)$ is a tree or a set of trees, duplication by case analysis reduces to the number of paths, and $||G(R)||$ is bounded by the product of duplication by iteration, duplication by case analysis and duplication by combination. The analysis proceeds by establishing whether these measures are exponential or linear or constant in the length of the input set R read as a single string.

The analysis of the strategies in Horn propositional logic

According to the analysis in [3], basic resolution strategies have linear duplication by iteration and exponential duplication by combination. The duplication by case analysis is trivially constant (more precisely, equal to 1), because each state has unique successor. The exponential duplication by combination captures the complexity of generating and keeping clauses, since resolvents are generated by combining in different ways the input literals. Since one of the measures is exponential, resolution strategies are regarded as inefficient. For positive hyperresolution, however, the duplication by combination is linear, because positive hyperresolvents in Horn logic are unit clauses, so that all generated clauses are unit clauses, and literals are not combined. Also, the duplication by combination of positive resolution reduces from exponential to linear if the strategy is equipped with an ordering on predicate symbols, and only the negative literals with smallest predicate are resolved upon. In summary, resolution strategies are either efficient, but not goal-sensitive (e.g., positive hyperresolution and positive resolution), or goal-sensitive, but not efficient (e.g., negative resolution), or neither efficient nor goal-sensitive (e.g., ordered resolution). These results are worst-case results. For instance, ordered resolution is efficient for some sets of clauses and orderings (e.g., all well-ordered sets[3]), but there are sets for which an ordering that makes the strategy efficient cannot be found. The same is true for goal-sensitivity.

Model elimination has exponential duplication by iteration and exponential duplication by case analysis. The duplication by combination is trivially constant (equal to 1), because each state is a singleton. The exponential duplication by iteration captures the redundancy of solving subgoals repeatedly. Duplication by case analysis is also exponential because a state may have multiple successors. Thus, resolution and model elimination are sort of dual: resolution has low

[3] A set of Horn clauses is *well-ordered* if there is a partial ordering $<$ such that if $P :- P_1, \ldots, P_n$ is a clause in the set then $P_i < P$ for all i, $1 \leq i \leq n$.

duplication by case analysis and iteration, but high duplication by combination, and vice versa for model elimination.

If model elimination is enriched with unit lemmaizing,[4] duplication by iteration becomes linear, because lemmaizing prevents solving repeatedly the same successful subgoals. Lemmatization adds a forward-reasoning character to the strategy, because lemmas are generated and retained. In the framework of [3] this means exponential duplication by combination and constant duplication by case analysis, so that model elimination with lemmaizing has the same duplication as the resolution strategies. In Horn logic model elimination can be enhanced with caching.[5] The use of caching reduces the duplication further, because not only successful subgoals (success caching), but also failed subgoals (failure caching) are not repeated. Assuming depth-first search with iterative deepening, duplication by combination becomes linear (more precisely, quadratic). Similar results apply to the other subgoal-reduction strategies. In summary, subgoal-reduction strategies are inefficient, but goal-sensitive. Subgoal-reduction strategies with caching are efficient and goal-sensitive.

A caveat Working by worst-case analysis means exhibiting a set of clauses where a strategy performs poorly. Examples are the parametric sets S_n^2 and A_n that give the exponential upper bounds on duplication by combination for negative resolution and ordered resolution, respectively. Worst-case sets may display regularities of structure, such as symmetries, or recurrence relations. The worst-case sets used in [3] incorporate recurrence relations on the indices of the literals in the clauses. The recurrence relations have exponential solution, and this is used in the proofs of the exponential behavior of the strategies. It is not known whether these patterns occur in real derivations, and how often. Also, worst-case sets are hard from a combinatorial point of view for the targeted strategy, but may be easy in the practice of theorem proving. S_n^2 is satisfiable and contains no positive clauses, so that positive hyperresolution or positive resolution would not apply a single inference and establish that the set is satisfiable in the time needed to read it. Similarly, A_n is satisfiable and contains neither positive nor negative clauses, so that it is trivial also for negative resolution.

Discussion

The following comment is written having first-order logic in mind, which is also the final purpose of the approach of [3].

Infinite search spaces

The complexity of search is measured in [3] by measuring the total size of the search space in the worst-case. The total size of the search space can be measured only if the search space is *finite*, as in the propositional problems considered in [3]. Thus, the first problem is

[4] In Horn logic all lemmas generated by lemmaizing are unit lemmas.

[5] Caching is not consistent with ME-reduction, which is necessary for the completeness of model elimination in first-order logic. In first-order logic one may use lemmaizing or more complicated caching schemes.

1. Explore complexity measures and modes of analysis that are suitable for *infinite* search spaces, such as those of first-order problems, whose total size cannot be measured.

The search plan

Since the intent of [3] is to measure the total size of the search space, and the latter depends on the inference system, not on the search plan, the model of search in [3] does not represent the search plan and its application. However, the search plan is important in the practice of first-order theorem proving. A prover may find a proof with a certain search plan and run out of memory with another, or find a proof hours later. Therefore, the following questions remain open:

2. Give the mathematical definition of a search plan.

3. Represent the application of a search plan to a search space, that is, the *search process*.

4. Compare strategies with the same inference system but different search plans.

Contraction

The analysis in [3] considers first all resolution-based methods as expansion-only strategies. Results for strategies with subsumption and clausal simplification are obtained by modifying the results for the expansion-only versions of the strategies. For most strategies, the analysis of contraction in [3] consists in showing that subsumption and clausal simplification do not apply in the sets of clauses used to establish the worst-case results, e.g., S_n^2 and A_n for negative resolution and ordered resolution, respectively. For positive resolution, since in Horn logic positive clauses are unit clauses, each positive-resolution step generates a resolvent that subsumes its non-unit parent. Thus, if a positive resolution strategy applies subsumption after each resolution step, the duplication by combination reduces from exponential to linear.

Because the analysis is conducted in this way, the formalism of [3] does not include contraction inferences as a basic element. The arcs in the directed graph represent only expansion inferences, such as the addition of resolvents, or the generation of a successor chain from a chain in model elimination. The choice of representing resolution inferences by arcs that add all resolvents is an obstacle to including contraction in the model. For instance, the search space of a strategy that applies subsumption after every resolution step cannot be represented in this model, because subsumption by a resolvent may delete the parent clauses of other resolution steps that were originally enabled.

Contraction is most useful in practice in equational logic and first-order logic. For these problems the search space is *infinite*. Therefore, the approach of exhibiting a worst-case set and showing that contraction does not affect it may not apply, so that the following problem remains:

5. Provide a model of the search space and search process that includes contraction as a basic element.

Contraction-based and subgoal-reduction strategies

The search space of subgoal-reduction strategies has been traditionally represented as a tree (e.g., AND/OR-tree). The nodes are labeled by the goals (e.g. the chains of model elimination), and the input clauses are viewed as "operators" that may be applied to reduce the goals. The treatment of subgoal-reduction strategies in [3] follows this pattern. On the other hand, there is no standard for the representation of the search space of contraction-based strategies (e.g., the model of [2] is for expansion-only strategies). The search space formalism of [3] does not change this state of affairs.

The approach of [3] stipulates that an arc represents the addition of all the resolvents, and the search space of resolution is a list. (If an arc represents a single resolution inference, the search space is a general graph as in [2].) This move of [3] may appear surprising. First, it makes the semantics of the formalism ambiguous, because an arc represents an inference for model elimination and all the possible inferences in the given state for resolution. Second, it has the counterintuitive effect that resolution is linear, whereas model elimination is not. A possible interpretation of this move is that the approach of [3] implicitly envisions resolution as a subgoal-reduction strategy, where the whole database of clauses is the goal (to be reduced to the empty clause) and the only operator is adding all resolvents. This interpretation is supported by the following observations:

- The tree structure, which is natural for subgoal-reduction strategies, but not for contraction-based strategies, is assumed already in the general characterization of a strategy as a tuple $< S, V, i, E, u >$. Then, the search space of subgoal-reduction strategies is represented properly as a set of trees, while the search space of resolution is reduced to a list.

- Resolution has a very high degree of non-determinism, because of all the possible selections of clauses in the database. However, it is represented as a completely deterministic strategy. On the other hand, the "degrees of freedom" of model elimination (e.g., choice of input clause for ME-extension steps and choice of initial chain) are represented (e.g., a state may have multiple successors and there is a tree for each choice of initial chain).

- The importance of the search plan is directly proportional to the degree of non-determinism of the inference system. Thus, the choices of making resolution deterministic and excluding the search plan from the representation are related. This does not affect subgoal-reduction strategies, for which depth-first search with iterative deepening is assumed. On the other hand, the search plan is fundamental for the modeling of forward-reasoning strategies.

- The positive effect of lemmaizing and caching in reducing the duplication of subgoal-reduction strategies is successfully captured. On the other hand, contraction, which counters the duplication of forward reasoning, is not included in the model.

Thus, we have the following problem:

6. Provide a common framework which is sufficiently rich to analyze subgoal-reduction, expansion-only and contraction-based strategies.

A different approach to the modeling of search and the analysis of strategies appeared in [1]. It presents the beginning of an approach to Problems 1, 2, 3, and 5. More problems remain open, as the field of strategy analysis is only at its infancy.

Acknowledgments

Thanks to David Plaisted, for answering my questions on his paper, and to Jieh Hsiang, for our discussions on Plaisted's work.

References

[1] M. P. Bonacina and J. Hsiang. On the representation of dynamic search spaces in theorem proving. In C.-S. Yang (ed.), *Proc. of the Int. Computer Symposium*, 85–94, 1996, and Technical Report, Dept. of Computer Science, Univ. of Iowa, 95-05.

[2] R. Kowalski. Search strategies for theorem proving. In B. Meltzer and D. Michie (eds.), *Machine Intelligence 5*, 181–201, Edinburgh University Press, 1969.

[3] D. A. Plaisted. The search efficiency of theorem proving strategies. In A. Bundy (ed.), *Proc. of CADE-12*, Springer Verlag, LNAI 814, 57–71, 1994, and Technical Report MPI I-94-233.

Call for Book Reviewers

We have received for review the following books related to automated reasoning and automated deduction.

1. *Goal-Directed Reasoning in Clausal Logic with Equality* by Maximilian Moser, CS Press, 1996, 141 p. (ISBN 3-931757-02-1). This is the first systematic and complete extension of a goal-driected calculus for handling equality in automated theorem proving.

2. *The Action-as-Implication Paradigm* by Bertram Fronhöfer, CS Press, 1996, 293 p. (ISBN 3-931757-91-9). This book deals with the logic underlying the linear connection method.

Interested reviewers are asked to contact Gail W. Pieper, pieper@mcs.anl.gov, for information on the type of review we are seeking. For book orders, contact Dr. Christian Suttner, CS Press, suttner@informatik.tu-muenchen.de.

Java GUIs for Automated Deduction

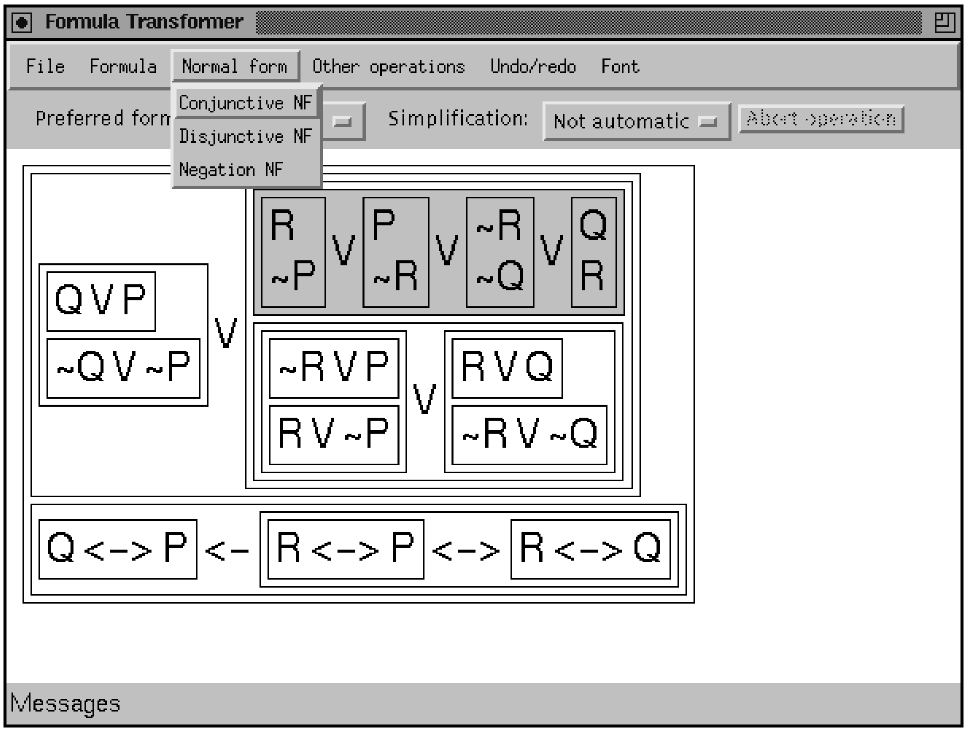
William McCune
(mccune@mcs.anl.gov)

Over the past several years I have been experimenting, off and on, with graphical user interfaces (GUIs) to various automated deduction programs. Following all of the recent hype about Java, I wished to see whether there is any more to it than annoying and expensive Web page animations, so I started a small investigation into using Java to implement platform-independent automated deduction GUIs.

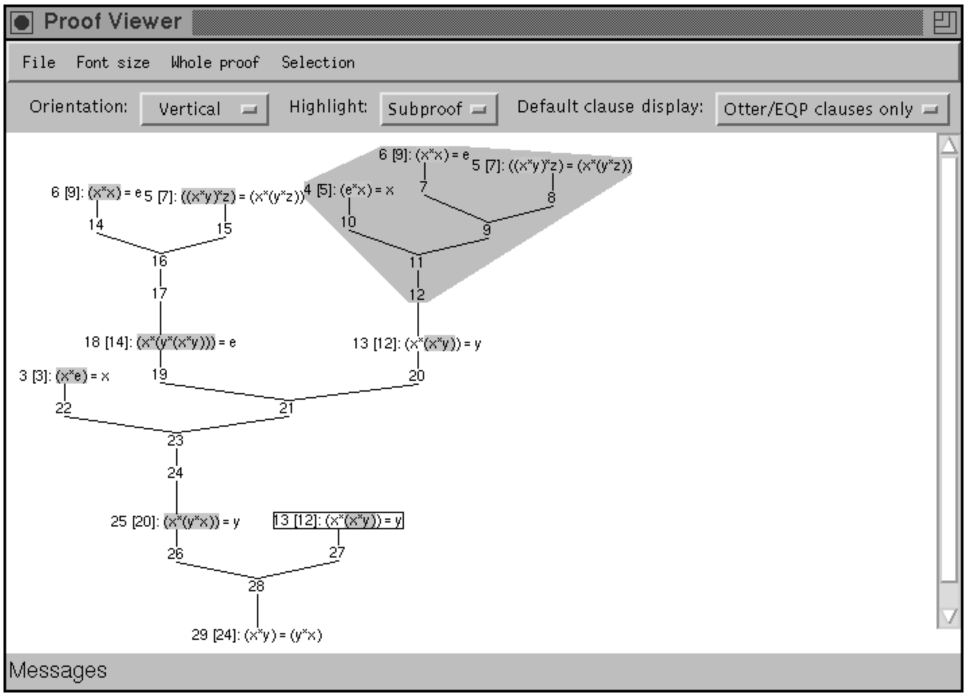
Java is a C-like, object-oriented programming language with good library functions for building GUIs. The main goal of Java is platform independence, enabling a program (including GUI functions) to run under UNIX X-windows, Macintosh, Microsoft Windows, or other operating systems.

One of the important properties of a Java *applet* (a restricted kind of Java program) is that it can be fetched over the Web *automatically, as needed* and started by one's Web browser. In principle, an entire automated deduction system could be implemented as a Java applet, and almost anyone could run the system through a local Web browser. This is not currently practical for large theorem provers, but it may be useful to implement deduction and database interfaces as Java applets.

Following are screen shots of two prototype Java applets I wrote while learning about the language. The first allows one to transform propositional formulas in various ways (conjunctions are displayed vertically and disjunctions horizontally).



The second applet displays Otter proof objects in various ways. An Otter *proof object* is a LISP-like expression containing a proof (produced by our theorem prover Otter) written in fine detail with simple inference steps. In this figure, clauses that appear in the ordinary Otter proof are displayed, and clauses from the more detailed steps are abbreviated as integers.



Here is the corresponding ordinary Otter proof.

```
4,3 [] x*e=x.
6,5 [] e*x=x.
7 [] (x*y)*z=x* (y*z).
9 [] x*x=e.
12 [para_into,7.1.1.1,9.1.1,demod,6,flip.1] x* (x*y)=y.
14 [para_into,7.1.1.1,9.1.1,flip.1] x* (y* (x*y))=e.
20 [para_from,14.1.1,12.1.1.2,demod,4,flip.1] x* (y*x)=y.
24 [para_from,20.1.1,12.1.1.2] x*y=y*x.
```

I invite you to experiment with these two programs. The propositional transformer might be useful to beginning logic students, and the proof viewer can help make sense of Otter proofs. If your Web browser supports Java applets (e.g., current versions of Netscape or Microsoft's Internet Explorer), you should be able to run the programs by just pushing a few buttons. See `http://www.mcs.anl.gov/home/mccune/web-java.html` for the current links.

I found Java to be a good language for implementing GUIs. It is true that the byte-code compilers are slow (I used JDK 1.0.2 under Linux and Solaris), the GUI library routines are a little buggy, and programs don't always work the same on different platforms, but I assume that such things will improve as the language matures. It is too early to say whether Java will catch on and become a standard, but it looks promising to me. And best of all, Java's platform independence allows one to write programs with GUIs that are accessible to nearly everyone without having to enter the world of Microsoft.

Call for Papers

Workshop on Artificial Intelligence Methods in Theorem Proving

A workshop on AI methods in theorem proving will be held in connection with the 14th International Conference on Automated Deduction (CADE-14) in Townsville, Australia, July 13, 1997. This workshop solicits research papers up to 10 pages and (possibly shorter) position papers. The topics include knowledge-based control, learning techniques, and combination of reasoning systems. Please submit contribution by e-mail (preferably Postscript generated from LaTeX with Springer llncs.sty) to kohlhase@cs.uni-sb.de by May 1, 1997. For further information, see the Web site http://jswww.cs.uni-sb.de/~kohlhase/deduction/cadews-ai.html.

DL'97

The 1997 International Workshop on Description Logics will take place September 27–29, 1997, in Gif sur Yvette (Paris), France. Papers should address the foundations, extensions, integration, or use of description logics. Submissions should be short descriptions (no more than 5 LaTeX 12-point article-style pages) or a short position paper indicating interest in description logics and the workshop. They should be sent to dl97@dl.kr.org by e-mail, to arrive by May 30, 1997.

FTP97

The 1997 International Workshop on First-Order Theorem Proving will take place at Schloss Hagenberg, near Linz (Austria), on October 27–28, 1997. The focus will be on on first-order theorem proving as a core theme of automated deduction, and to provide a forum for presentation of very recent work and discussion of research in progress. The workshop welcomes original contributions on theorem proving in first-order logics; equational reasoning and term-rewriting systems; constraint-based reasoning; unification algorithms for first-order theories; specialized decision procedures; propositional logic; abstraction; first-order constraints; complexity of theorem proving procedures; and applications of first-order theorem provers to problems in artificial intelligence, verification, and mathematics. Papers that bridge the areas of theorem proving and constraints (e.g., in the areas of equational reasoning, term rewriting systems, and satisfiability problems) are especially welcome.

The technical program will include presentations of the accepted papers and an invited talk by Bruno Buchberger on the Theorema Project.

The workshop is held right before the International Conference on Constraint Programming. The organization of the two events in sequence represents an opportunity for cross-fertilization of ideas between the two neighboring fields. Authors are invited to submit an extended abstract of up to 5 pages, by electronic mail (uuencoded gzipped Postscript files, preferably produced by LaTeX) to ftp97@cs.uiowa.edu by August 27, 1997. The extended abstracts that are accepted will be collected in a technical report of RISC-Linz, which will be distributed at the workshop. A special issue of the *Journal of Symbolic Computation* is planned. The authors of accepted extended abstracts will have the possibility of submitting a full version of their papers to the special issue. More information on the two conferences can be found at http://www.logic.tuwien.ac.at/FTP97/ and http://www.mpi-sb.mpg.de/conferences/CP97/.

EMCSR 1998

The Austrian Society for Cybernetic Studies has issued a call for papers for the Fourteenth European Meeting on Cybernetics and Systems Research (EMCSR 98), to be held at the University of Vienna, Austria, on April 14–17, 1998. Sessions are planned in such areas as theory and applications of artificial intelligence, fuzzy systems, knowledge-based systems, and approximate reasoning.

For further information, contact the program chairman Robert Trappl, Department of Med. Cybernetics and Artificial Intelligence, University of Vienna, Freyung 6/2, A-1010 Vienna, Austria; e-mail sec@ai.univie.ac.at, or see the Web site at http://www.ai.univie.ac.at/emcsr/.

KR'98

The Sixth International Conference on Principles of Knowledge Representation and Reasoning will take place in Trento, Italy, on June 2–5, 1998. KR'98 is intended to be a place for the exchange of news, issues, and results among the community of researchers in this area.

Authors are encouraged to submit papers that present substantial new results while clearly showing the applicability of those results to implemented or implementable AI systems. Also encouraged are "reports from the field" of applications, experiments, developments, and tests.

The following are some of the topics of interest: representational formalisms, reasoning techniques, implemented KR&R systems, significant applications, implications for machine learning and decision theory, parallel and distributed implementations, efficiency measures and complexity, databases, and software engineering. Extended abstracts (up to 12 pages) are due Dec. 1, 1997. For further information, see the Web page http://www.kr.org/kr/kr98.

LICS'98

The Thirteenth Annual IEEE Symposium on Logic in Computer Science will be held at the Indiana University Conference Center in Indianapolis, June 21-June 24, 1998. Meeting rooms and accommodations have been reserved for pre- and post-conference workshops at the conference center for June 19–20 and June 25–26.

Researchers and practitioners are invited to submit proposals for workshops on topics relating logic, broadly understood, to computer science. Of particular interest are meetings that would foster innovation, deepening, or expansion of practical applications of logic in computer science.

Proposals should consist of two parts: (1) a short scientific justification of the proposed topic, its significance, and the particular benefits of a workshop; and (2) an organizational part including contact information about organizers, proposed format and agenda, procedures for selecting papers and participants, and duration (half a day to two days) and preferred period (pre- or post-LICS). Additional organizational plans may include potential invited speakers, demo sessions, and plans for proceedings or other publications. Proposals are due April 1, 1997.

Proposals may be submitted either electronically or in hard copy, and should be addressed to Daniel Leivant, Department of Computer Science, Indiana University, Bloomington, IN 47405, USA; leivant@cs.indiana.edu.