PTTP+GLiDeS: Guiding Linear Deductions with Semantics

Marianne Brown and Geoff Sutcliffe

Computer Science
James Cook University
{marianne,geoff}@cs.jcu.edu.au

Abstract. Using semantics to guide automated theorem proving systems is an under-utilised technique. In linear deduction, semantic guidance has received only limited attention. This research is developing semantic guidance for linear deduction in the Model Elimination paradigm. Search pruning, at the possible loss of some refutation completeness, and search guidance, are being considered. This paper describes PTTP+GLiDeS, a PTTP style prover augmented with a semantic pruning mechanism, GLiDeS. PTTP+GLiDeS combines a modified version of Stickel's PTTP prover with the model generator MACE.

1 Introduction

Automated theorem proving (ATP) aims to use computer technology to solve problems that require logical reasoning. Applications for ATP systems include logic circuit design validation, software verification, mathematical and logical research [18].

Resolution [9] was developed in 1965 and has formed the basis of much of the research undertaken in the field since. Resolution uses 'proof by contradiction' in its search for a proof. Assumptions (axioms) about the problem and the negated conjecture are expressed in the clause normal form of first order logic. The naïve resolution approach takes the input clause set, S_0, and generates the set of all possible resolvents, R_0. If the empty clause is a member of R_0, a contradiction has been found and the problem solved. Otherwise, a new set S_1 is created, $S_1 = S_0 \cup R_0$, and the process continues. If set S_n contains the empty clause, S_n forms the search space for a minimal length proof. A large search space means the time taken to find a proof can be long. In order to find proofs quickly, both the size of the search space and the path the prover takes through the search space needs to be controlled in an intelligent manner.

To control the search of a resolution based system, ordering and pruning strategies are used. Ordering strategies control the order in which resolvents are generated by giving preference to certain clauses and literals. Pruning strategies prevent certain combinations of clauses and also discard clauses, preventing them from taking any further part in the deduction. While ordering strategies attempt to guide the search along paths that may be more likely to produce the empty clause, pruning strategies reduce the search space.

Search control may utilise syntactic or semantic methods. Syntactic methods use some physical feature of the clauses to determine which clauses will be resolved together and on which literals. Semantic methods use interpretations to give information about the clauses. This information is then used in choosing the clauses and literals to resolve. Semantic methods have the potential to perform much better than syntactic ones [16]. Semantic search control for forward chaining resolution strategies has been in use for some time. Set of support (SoS) [17], model resolution [6], and semantic resolution [10] are all forward chaining resolution strategies. Two systems that employ semantic guidance are CLIN-S [2] and SCOTT [11]. SCOTT is a resolution based prover that uses an interpretation to weight clauses and thus give preference to those clauses that are FALSE in the interpretation. CLIN-S is an instantiation based prover, and uses an interpretation to guide the generation of ground clauses, which are then examined for unsatisfiablity.

Incorporating semantic methods into backward chaining resolution strategies is not as easy as for forward chaining ones. Semantic guidance for linear-input resolution is well understood [1], as described in Section 3. This research aims to incorporate semantic guidance into general linear resolution [5,7], primarily considering pruning strategies.

The next section contains a brief explanation of the Model Elimination (ME) paradigm and introduces some terminology. Section 3 describes the architecture of PTTP+GLiDeS and explains the way in which semantic guidance has been incorporated into the ME based system. Implementation and performance are discussed in Sections 4 and 5 respectively. Further enhancements are being explored and these are outlined in Section 6.

2 Model Elimination

ME is a chain format linear resolution procedure for first order logic, first proposed in [4]. A chain is an ordered list of A- and B-literals, with the disjunction between the literals being implicit. Chains generated from the input clauses are called input chains and are composed entirely of B-literals. The chains that form the linear path in the refutation are called the centre chains. The input chains that are resolved with the centre chains are called the side chains. A-literals are those literals in a centre chain that have been resolved upon. A-literals are indicated by a frame, e.g., $\boxed{\text{p}}$. The first centre chain is called the top chain. One of the input chains is chosen to be the top chain; a chain generated from the negated conjecture is the usual choice. All input chains are potential side chains.

In ME there are two deduction operations, extension and reduction, and one book-keeping operation, truncation. The extension operation is a binary resolution between a centre chain and a side chain. The resolution takes place between the rightmost B-literal in the centre chain and a complimentary (after unification) B-literal in the side chain. The B-literal in the centre chain then becomes an A-literal, and the B-literal in the side chain is removed. The remaining B-literals in the side chain are added to the right of the newly created A-literal in

the centre chain. A reduction operation is a unification between the rightmost B-literal in the centre chain and an A-literal. The new centre chain is formed by removing the B-literal. Reduction implements ancestor resolution and factoring. Truncation is the removal of A-literals from the right-hand end of a centre chain. See Figure 1 for an example of an ME refutation.

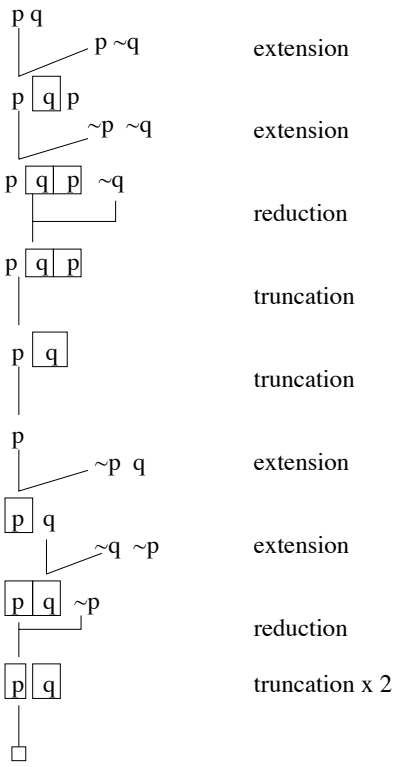

Fig. 1. An ME refutation of the set { p ∨ q, p ∨ ∼q, ∼p ∨ q, ∼p ∨ ∼q }

One method of implementing ME is using the Prolog Technology Theorem Prover (PTTP) [12] principle. The idea here is to have the theorem prover rewrite the input clause set into Prolog procedures that implement ME deduction for the clauses. The procedures are then compiled and executed on a Prolog engine (see Figure 2). Prolog is based on linear-input deduction for Horn clauses and has an incomplete search strategy and unsound unification algorithm. A PTTP style system overcomes these issues by using a bounded depth first search and unification with an occurs check.

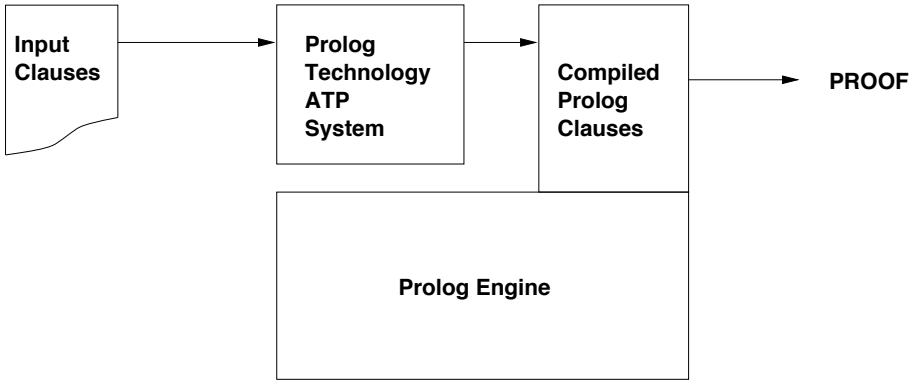

Fig. 2. Architecture of PTTP-based ATP systems

3 Architecture

The PTTP+GLiDeS semantic pruning strategy is based upon the strategy that can be imposed on linear-input deductions, as follows: If there exists a linear-input refutation, the last centre clause is the empty clause. The empty clause has the interpretation of FALSE in every interpretation. A FALSE resolvent must have one or more FALSE parents. If there is a model M of the side clauses, then this implies that the second last centre clause must be FALSE in M, and so on up to the top clause. So, if the side clauses are known and a model of them, M, can be found, then any centre clause that is TRUE in M can be rejected. A simple possibility is to choose a negative top clause from a set of Horn clauses, in which case the non-negative clauses are the potential side clauses. More sensitive analysis is also possible [3, 14]. Linear-input resolution is complete for Horn clauses only.

Unfortunately, the extension of the linear-input semantic pruning strategy to linear deduction is not direct. For the non-Horn case, ancestor resolution is required for refutation-completeness. The possibility of ancestor resolutions means that centre clauses may be TRUE in a model of the side clauses. Investigation of how to allow for centre clauses that are TRUE in the model of the side clauses is a focus of this research.

In PTTP+GLiDeS, rather than placing a constraint on entire centre clauses, a semantic constraint is placed on selected literals of the centre clauses as follows: The input clauses other than the chosen top clause of a linear deduction are named the *model clauses*. In a completed linear refutation, all centre clause literals that have resolved against input clause literals are required to be FALSE in a model of the model clauses. TRUE centre clause literals must be resolved against ancestor clause literals. This leads to a semantic pruning strategy for ME deductions that at every stage requires all A-literals in the deduction so far to be FALSE in a model of the model clauses. The result is that only FALSE B-literals are extended upon, and TRUE B-literals must reduce.

The completeness of the PTTP+GLiDeS semantic pruning strategy has not yet been investigated. It is certainly possible that it is an incomplete strategy. However, the results shown in Section 5 suggest that there is not a 'large loss of completeness', while the benefits are significant.

Figure 3 shows the architecture of the PTTP+GLiDeS system. PTTP+GLiDeS uses a Prolog technology theorem prover to compile the input clauses into Prolog code, which is then run on a Prolog engine. An interpretation generator takes the model clauses from the input clause set and generates an interpretation which is also given to the Prolog engine. The Prolog code uses the interpretation to implement the semantic guidance.

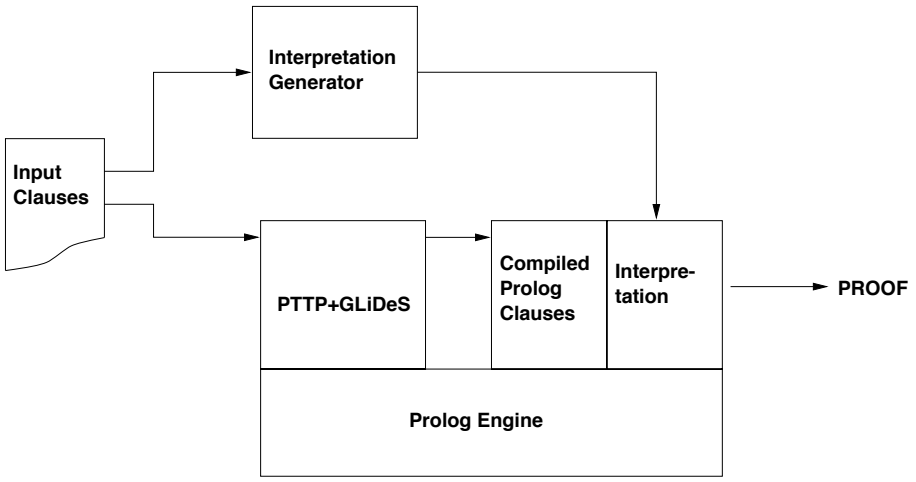

Fig. 3. Architecture of the PTTP+GLiDeS system.

4 Implementation

PTTP+GLiDeS consists of modified versions of PTTP (Prolog version 2e) [13] and MACE (v1.3.2) [8], combined by a `csh` script. PTTP+GLiDeS takes problems in TPTP [15] format as input. The `tptp2X` utility is used to transform the input problem to PTTP and MACE formats. The transformation to PTTP format selects the first conjecture clause as the top chain for the linear deduction.

A `perl` script is used to remove the first conjecture clause from the MACE format file, and MACE is called to generate a model of the remaining clauses. MACE is capable of generating many models, but in this experiment the first model generated is used. If MACE is unable to generate a model then PTTP+GLiDeS terminates. Otherwise MACE outputs its model in the form of Prolog facts, e.g.,

```
eval(functor,a,0).
```

```
eval(predicate,p(0,0),true).
```

The modified PTTP is then started.

The modified PTTP produces Prolog procedures that i) maintain a list of all A-literals that have been produced in the deduction so far, and ii) call a semantic checking procedure after each extension and reduction operation. The facts produced by MACE are used to interpret the A-literals. If the semantic checking procedure finds an A-literal that is TRUE then the extension or reduction is rejected.

5 Performance

Testing has been carried out using 541 "difficult, unbiased" problems from the TPTP library v2.1.0. The testing was done on a SUN sparc20, with a CPU time limit of 600 seconds. Table 1 gives an overall summary of the results.

Total number of problems:	541	
Number of models generated:	260	
Number of problems solved from 260:	PTTP	PTTP+GLiDeS
	68	54
Number of useful models generated:	144	
Number of problems solved from 144:	PTTP	PTTP+GLiDeS
	21	19

Table 1. Summary of experimental data.

For PTTP+GLiDeS, MACE produced models for only 260 of the 541 problems, and thus PTTP+GLiDeS could attempt only those problems. Of the 260 problems for which models were generated, plain PTTP solved 68 and PTTP+GLiDeS solved 54. Altogether, there were 69 problems that had models generated and were solved by either system. Of the 260 models, only 144 proved to be useful in that they provided guidance that pruned the search space of PTTP+GLiDeS. Of these 144 problems, PTTP solved 21 and PTTP+GLiDeS solved 19. In total, there were 22 problems that had useful models generated and were solved by either system.

For the 22 problems solved, Table 2 shows the CPU times taken, the number of inferences made, and the number of inferences rejected during the search. The "CPU time" column for PTTP+GLiDeS includes the time taked for preprocessing the MACE input file to exclude the choosen top clause leaving only the model clauses, model generation and output time, writing of the Prolog procedures, and the Prolog search time. For PTTP, the CPU time includes the time for writing the Prolog procedures, and the Prolog search time. The "Inferences" columns give the total number of extension and reduction operations performed during the search for a solution. The "Rejected Inferences" are the numbers of inference operations that were rejected by the semantic pruning routine. The

"Inference Ratio" shows the number of inferences made by PTTP+GLiDeS relative to PTTP.

The number of inferences made during the search gives an indication of the search space being covered during the search. A smaller inference count on the same problem does not necessarily indicate that the proof itself was any smaller. Instead, it shows that less of the search space was covered before the proof was found.

Table 2. Results for problems where semantic guidance rejected some inferences

Problem	**PTTP**		**PTTP+GLiDeS**			**Inference**
	CPU time (sec)	Inferences	CPU time (sec)	Inferences	Rejected Inferences	**Ratio**
BOO004-1	11.0	10515	19.5	9355	365	0.89
BOO012-1	392.8	1579178	TIMEOUT			
CAT001-4	108.5	427522	549.0	376020	37716	0.89
CAT002-4	23.9	84480	87.6	80428	5220	0.95
CAT003-3	TIMEOUT		230.2	217996	34840	
CAT003-4	8.6	11077	16.7	10816	585	0.98
CAT012-3	84.5	175367	49.1	49150	4124	0.28
CAT018-1	73.0	226900	343.2	183518	19806	0.81
GRP012-3	362.8	1282139	TIMEOUT			
HEN003-3	17.2	47136	50.5	44322	1093	0.94
HEN008-1	17.7	72068	65.9	69959	803	0.97
HEN008-3	7.7	11524	17.7	10905	308	0.95
HEN012-3	25.2	85312	101.1	80481	1857	0.94
PUZ032-1	15.4	26947	19.8	18629	4427	0.69
RNG002-1	13.7	27867	36.9	27313	756	0.98
RNG003-1	14.3	31150	38.0	23867	1412	0.77
RNG040-1	9.0	563	11.9	533	67	0.95
RNG041-1	11.1	8826	16.8	4859	824	0.55
ROB016-1	8.0	4546	22.8	3738	92	0.82
SET008-1	7.4	276	9.2	370	56	1.34
SYN071-1	411.8	832600	53.3	84908	27653	0.10
SYN310-1	438.0	1476442	TIMEOUT			
Average	87.25	254354.80	91.54	71545.11	7868.83	0.82

PTTP+GLiDeS solved one problem, CAT003-3, that PTTP failed to solve, but timed out on three that PTTP did solve. In all but one case PTTP+GLiDeS took less inferences than PTTP, and in many cases significantly less. The times taken by PTTP+GLiDeS are higher than for PTTP in most cases. Two interesting cases to note are CAT012-3 and SYN071-1. These are non-Horn problems, and have the best reduction in inference counts and less CPU time than PTTP. Of the 22 problems, 7 are non-Horn and it is in these cases that PTTP+GLiDeS performs best on average for CPU time and inference counts, as shown in Table 3.

Table 3. Results for non-Horn problems

Problem	PTTP		PTTP+GLiDeS			Inference
	CPU time (sec)	Inferences	CPU time (sec)	Inferences	Rejected Inferences	Ratio
CAT003-3	TIMEOUT		230.2	217996	34840	
CAT012-3	84.5	175367	49.1	49150	4124	0.28
PUZ032-1	15.4	26947	19.8	18629	4427	0.69
RNG040-1	9.0	563	11.9	533	67	0.95
RNG041-1	11.1	8826	16.8	4859	824	0.55
SET008-1	7.4	276	9.2	370	56	1.34
SYN071-1	411.8	832600	53.3	84908	27653	0.10
Average	89.87	174097	55.76	53778	10284	0.65

There were 15 problems out of the group of 260 problems that were solved by PTTP and not PTTP+GLiDeS. Of these, 12 were Horn problems that had models generated where the positive literals were TRUE, i.e., the models were trivial. PTTP+GLiDeS performed badly on these 12 problems as semantic checking was done when it was not going to have any effect on the search for a solution other than to slow its progress. It is a simple matter to check for this situation and omit the semantic guidance. Future implementations of PTTP+GLiDeS will have this feature. Of course, a better solution is to not generate trivial models in the first place. MACE is capable of producing many models for a given set of model clauses. In this experiment, the first model generated was used for the interpretation. A better approach may be to generate more than one model and select the 'best' one for use, or at least select a non-trivial model, as discussed in Section 7.

6 Using Multiple Models

Work is currently under way on two different multiple model versions of PTTP+GLiDeS. Version 1 generates several different models for a problem and perform the semantic checking using all models, i.e., the A-literals must be acceptable to all models before an inference operation is accepted. By using more than one model it is hoped that greater pruning will be achieved. Preliminary testing has shown that while some extra pruning is achieved, the time taken to perform the semantic checking is greatly increased. For this approach to be practical a much more efficient implementation of the semantic checking routine needs to be written.

Version 2 runs PTTP+GLiDeS in parallel with different models; the first one to find a solution kills the others. It has been observed that in some cases one model results in a timeout and another produces a solution for the same problem. By running in parallel with different models, it is hoped that one of the models will be a 'good' model and produce a solution. It may also assist in overcoming any incompleteness problems.

Table 4 shows data for PTTP+GLiDeS using the different multiple model versions and the data for PTTP. The models were hand coded rather than generated by MACE. The parallel approach of version 2 was simulated - PTTP+GLiDeS was run with each of the eight different models, then the best time was selected and multiplied by eight. The number of inferences was also multiplied by eight, but the rejected inferences count was not.

Table 4. Results for PTTP and multiple model versions of PTTP+GLiDeS

Problem	**PTTP**		**8 Models Version 1**			**8 Models** Version 2		
	CPU time (sec)	Inferences	CPU time (sec)	Inferences	Rej. Inf.	CPU time (sec)	Inferences	Rej. Inf.
LCL007-1	3.7	3	4.3	3	0	34.4	24	0
LCL010-1	4.5	1349	672.4	1321	11	374.4	10584	7
LCL118-1	4.7	1897	532.9	1392	64	327.2	11136	64

Version 1 can produce greater pruning than using a single model. However, this is at the cost of greatly increase CPU time, greater than for version 2 where the best CPU time was multiplied by 8.

Version 2 is of no benefit when all models result in solutions being found, as in the problems shown in Table 4. Its usefulness is more apparent in cases where one of the models does not produce a solution but another does. In such a case, version 1 would timeout but version 2 would not.

7 Conclusion

The preliminary experiments are encouraging. In the cases where both PTTP and PTTP+GLiDeS find a solution, PTTP+GLiDeS makes fewer inferences on average. This indicates that the semantic guidance is successfully pruning the search space. A side effect of the pruning may be a loss of refutation completeness. Further work needs to be done to assess the extent of this. The time taken by PTTP+GLiDeS is greater than PTTP in the majority of cases where both systems find a solution. It may be possible to improve the time taken by PTTP+GLiDeS by making the semantic checking code more efficient.

Currently the `tptp2X` utility chooses the first conjecture clause in the problem as the top centre chain for PTTP. The failure to generate models in some cases is due to this unintelligent selection of the top chain. The performance of both PTTP and PTTP+GLiDeS are likely to be improved if this selection is done more intelligently. In particular, it is hoped that MACE will be able to produce models for many more problems, hence giving PTTP+GLiDeS an opportunity to attempt more problems.

Trivial model generation is another problem that needs to be overcome. It is possible to examine the model generated by MACE and determine if it is unlikely to be of use, as in the case of a trivial model, and reject it. It would be preferable to prevent generation of such a model in the first place. Further examination of this issue is needed.

Using multiple models should enable PTTP+GLiDeS to achieve greater pruning of the search space. Preliminary results show that this is the case but the increase in CPU time is currently too high.

References

1. A. Bundy. *The Computer Modelling of Mathematical Reasoning.* Academic Press, 1983.
2. H. Chu. *CLIN-S User's Manual.* Chapel Hill, USA, 1994.
3. D.A. de Waal and J.P. Gallagher. The Applicability of Logic Programming Analysis and Transformation to Theorem Proving. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, number 814 in Lecture Notes in Artificial Intelligence, pages 207–221. Springer-Verlag, 1994.
4. D.W. Loveland. Mechanical Theorem Proving by Model Elimination. *Journal of the ACM*, 15(2):236–251, 1968.
5. D.W. Loveland. A Linear Format for Resolution. In Laudet M. et al., editor, *Proceedings of the IRIA Symposium on Automatic Demonstration*, pages 147–162. Springer-Verlag, 1970.
6. D. Luckham. Some Tree-paring Strategies for Theorem Proving. *Machine Intelligence*, 3:95–112, 1968.
7. D. Luckham. Refinement Theorems in Resolution Theory. In Laudet M. et al., editor, *Proceedings of the Symposium on Automatic Demonstration*, pages 163–190. Springer-Verlag, 1970.
8. W.W. McCune. A Davis-Putnam Program and its Application to Finite First-Order Model Search: Quasigroup Existence Problems. Technical Report Technical Report ANL/MCS-TM-194, Argonne National Laboratory, Argonne, USA, 1994.
9. J.A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, January 1965.
10. J.R. Slagle. Automatic Theorem Proving with Renamable and Sematic Resolution. *Journal of the ACM*, 14:687–697, October 1967.
11. J.K. Slaney. SCOTT: A Model-Guided Theorem Prover. In R. Bajcsy, editor, *Proceedings of the 13th International Conference on Artificial Intelligence*, pages 109–114. Morgan-Kaufman, 1993.
12. M.E. Stickel. A Prolog Technology Theorem Prover. In *Proceedings of the 1st International Symposium on Logic Programming*, pages 211–217. IEEE Computer Society Press, 1984.
13. M.E. Stickel. A Prolog Technology Theorem Prover: A New Exposition and Implementation in Prolog. Technical Report Technical Note 464, SRI International, Menlo Park, USA, 1989.
14. G. Sutcliffe. Linear-Input Subset Analysis. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction*, number 607 in Lecture Notes in Artifical Intelligence, pages 268–280, Saratoga Springs, NY, USA, June 1992. Springer-Verlag.

15. G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.
16. L. Wos. *Automated Reasoning - 33 Basic Research Problems.* Prentice-Hall, 1988.
17. L. Wos, D. Carson, and G.A. Robinson. The Unit Preference Strategy in Theorem Proving. In *Proceedings of the AFIPS 1964 Fall Joint Computer Conference* , pages 615–621. Spartan Books, 1964.
18. L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning Introduction and Applications.* Prentice-Hall, Englewood Cliffs, New Jersey, 1984.