A General Clause Theorem Prover

Geoff Sutcliffe
Dep't of Computer Science, The University of Western Australia
Nedlands, Western Australia, 6009.
geoff%wacsvax.uwa.oz.au

Introduction

The General Clause Theorem Prover (GCTP) was designed as a test bed for semantic search control strategies in theorem proving. It is the unique combination of the Graph Construction (GC) procedure of [Shostak 1976] (which is known to out-perform many (all?) linear format systems), embedding the axioms of equality by the use of the Resolution by Unification and Equality (RUE) and Negative Reflexive Function (NRF) inference rules [Digricoli 1979]. To use RUE and NRF in the GC procedure, three modifications have been made. (i) Extension is is redefined to use RUE unification; (ii) NRF is added, and applied to negative equality B-literals as an alternative to extension; (iii) Reduction is redefined to use RUE unification. In all three cases, negative equality literals generated from the use of RUE unification are classified as B-literals and added to the right hand end of the result chain. The completeness of the combination of the GC procedure and the RUE and NRF inference rules, is to be obtained by combining their respective completeness results.

The GCTP implementation

The GCTP has been implemented in Prolog. An input clause is represented by a Prolog fact whose single argument is a list of (non-classified) literals that constitute the represented clause. Each centre chain of a derivation is represented by a list of function structures and variables. The functor of each function structure is one of a, b or c, indicating the class of the literal argument. The second argument of an A-literal function structure is a variable identical to one of the variables in the centre chain list, indicating the C-point of the A-literal. The insertion of a C-literal, predicated the truncation of an A-literal, is efficiently implemented by the instantiation of the associated C-point variable. Extension and NRF always use the rightmost B-literal of the centre chain. Although possibly not optimal, this is convenient given the list representation of chains.

The GCTP is partially self configuring. If the input clauses are all Horn clauses, an input derivation system is complete and the reduction inference rule is suppressed. If no equality literals are present in the input clauses, there is no need to embed the axioms of equality. In this case standard unification is used rather than RUE unification, and NRF is suppressed.

The selection of disagreement set, selection of substitution, and equality restrictions for RUE and NRF, have been embedded into the GCTP RUE unification and search strategy. The GCTP RUE unification unifies a pair of terms only if they are identical or one of the pair is a variable. In all other cases a disagreement pair is returned. The equality restriction for RUE has been strengthened to prevent this relaxed selection of disagreement pairs from creating an ongoing sequence of negative equality literals.

The GCTP search strategy

A modified consecutively bounded depth first search is used by the GCTP. The modified version improves on the standard version by truncating only those long inference sequences

that are not making progress towards a refutation. Compatiblity with the Set-of-Support strategy allows the GCTP to form its top chains from negative input clauses. This selection of top clauses leads to 'natural' proofs, and simplifies semantic checking in the Horn clause configuration. The GCTP has relaxed the GC procedure's specification that reduction must be done if possible, to allow extension as an alternative to reduction against a C-literal. This corrects a completeness problem of the original procedure. The GCTP uses a unit preference strategy in extensions.

C-literals at the left end of centre chains, which are proved, are removed from the centre chain and added to the set of input clauses as unit clauses. This is done so as to retain the most general unit clauses. The addition of unit clauses to the input set is particularly effective in conjunction with the unit preference strategy. In the environment of a consecutively bounded search there is an added advantage that proofs of B-literals discovered within one bound, are 'transferred' to the next bound as unit clauses.

Search pruning

In unit extensions, if the negation of the centre clause B-literal is subsumed by the input clause literal, no alternative extensions are permitted.

The GC procedure's restriction that no two non-B-literals in the centre chain may have identical atoms has been extended to check that no A- or B-literal in the centre chain is identical to any A-literal to its left. This second restriction is an extension of restriction 2 of the Model Elimination procedure. All syntactic checking is repeated in order to detect violations due to substitutions.

In the Horn clause configuration the GCTP can perform the semantic check that ensures the falsity of all past and current centre clause literals. In the GCTP the A- and B-literals in the centre chain are the literals that must be checked. A domain based approach to semantic checking [Sutcliffe 1988] is used. The semantic checking of negative equality B-literals generated from RUE unification may be interpreted in terms of term level semantic checking of input paramodulation.

Conclusion

The GCTP has been successfully tested with many of the problems given in [Pelletier 1986]. It is currently being extended to include a type checking mechanism, which will move the GCTP into the field of typed first order logic. A full description of the GCTP may be found in [Sutcliffe 1989]

References

Digricoli V.J.; Automatic Deduction and Equality; Proceedings of the 1979 Annual Conference of the ACM, 1979, pp 240-250.

Pelletier F.J.; Seventy-five Problems for Testing Automatic Theorem Provers; Journal of Automated Reasoning 2, 1986, pp 191-216.

Shostak R.E.; Refutation Graphs; Artificial Intelligence 7, 1976, pp 51-64.

Sutcliffe G.; Semantic Checking for Horn Clauses; Unpublished manuscript, 1988.

Sutcliffe G.; A General Clause Theorem Prover written in Prolog; Western Australian College of Adv. Ed., Dep't of Computer Studies, Research Report 89/2, 1989.