SRASS - a Semantic Relevance Axiom Selection System

Geoff Sutcliffe*
*University of Miami, USA
geoff@cs.miami.edu

Abstract

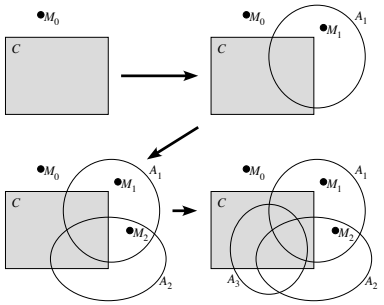
This paper describes the design, implementation, and testing of a system for selecting necessary axioms from a large set also containing superfluous axioms, to obtain a proof of a conjecture. The selection is determined by semantics of the axioms and conjecture, ordered heuristically by a syntactic relevance measure. The system is able to solve many problems that cannot be solved alone by the underlying conventional automated reasoning system.

1 Introduction

In recent years the ability of systems to reason over large theories – theories in which there are many functors and predicates, many axioms of which typically only a few are required for the proof of a theorem, and many theorems to be proved from the same set of axioms – has become more important. Large theory problems are becoming more prevalent as large knowledge bases, e.g., ontologies and large mathematical knowledge bases, are translated into forms suitable for automated reasoning, and mechanical generation of automated reasoning problems becomes more common. This work addresses the issue of selecting necessary axioms, from a large set also containing superfluous axioms, to obtain a proof of a conjecture. It is based on the idea in Petr Pudlak's PhD research, which he attributes to Jiří Vyskočil. In contrast to existing work that selects axioms based on syntactic characteristics, this work uses semantics to guide the selection.

2 Semantic Relevance Axiom Selection

The selection starts with an empty set of selected axioms. At each iteration the process looks for a model of the selected axioms and the negation of the conjecture. If no such model exists then the conjecture is a logical consequence of the selected axioms. If such a model exists then an unselected axiom that is false in the model is moved to the set of selected axioms. The newly selected axiom excludes the model (and possibly other models) from the models of the selected axioms and negated conjecture, eventually leading to the situation where there are no models of the selected axioms and the negated conjecture. The figure shows the idea. The plane represents the space of interpretations, the rectangle encompasses the models of the conjecture C, and an oval encompasses the models of the corresponding axiom A_i. In the first iteration, when the set of selected axioms is empty, the model M_0 of the negation of the conjecture, $\neg C$, is found. That leads to the selection of the axiom A_1, which is false in the model. Iteratively, the model M_1 of $\{A_1, \neg C\}$ is found, leading to the selection of A_2, the model M_2 of $\{A_1, A_2, \neg C\}$ is found, leading to the selection of A_3, at which point there is no model of $\{A_1, A_2, A_3, \neg C\}$, proving that C is a logical consequence of $\{A_1, A_2, A_3\}$. In the last part of the figure this is seen by the intersection of the axiom ovals lying within the conjecture rectangle.



Example: Consider the simple propositional problem, to prove the conjecture $C = b$ from the axioms $E_1 = a \mid b$, $E_2 = b \Rightarrow a$, $E_3 = (\neg a \;\&\; (b \mid c)) \mid (a \;\&\; \neg b \;\&\; \neg c)$. and $E_4 = b \mid (a \Leftrightarrow c)$. The conjecture C can be proved from E_3 and E_4, i.e., E_1 and E_2 are superfluous. The following table shows a possible sequence of models and selected axioms. Note how E_1, E_2, and E_3 are true in the first model, so only E_4 can be selected. E_1 and E_3 are false in the second model, but E_1 is found first. E_2 is true in every model of $\neg C$, and thus can never be selected. If the model $\{a, \neg b, c\}$ had been used in the second iteration, then E_3 would have been selected, leading to immediate success.

	Selected set	Model	Axiom
1	$\{\ \}$	$\{a, \neg b, \neg c\}$	$E_4 = b \mid (a \Leftrightarrow c)$
2	$\{E_4\}$	$\{\neg a, \neg b, \neg c\}$	$E_1 = a \mid b$
3	$\{E_1, E_4\}$	$\{a, \neg b, c\}$	$E_3 = (\neg a \;\&\; (b \mid c)) \mid (a \;\&\; \neg b \;\&\; \neg c)$
4	$\{E_1, E_3, E_4\}$	-	-

The basic process can be extended to improve performance, and to cope with practical issues that arise in implementation. **Initial Proof Attempt:** An initial proof attempt may be made using conventional automated reasoning. If this is successful then no further processing is required. **Relevance Ordering:** A syntactic relevance score is used to order the axioms in decreasing order of potential usefulness, to increase the chances of selecting a useful axiom, and selecting it early in each iteration. **Efficient Termination:** The absence of a model of the selected axioms and negated conjecture can be more efficiently established by testing for unsatisfiability (although these two are logically the same, different techniques may be used to establish the two conditions), **Greedy Termination:** Selection of the last axiom may be greedily achieved by looking for an unselected axiom whose negation can be proved from the selected axioms and negated conjecture. **Incomplete Models:** In practice, the model might be able to interpret only the symbols in the selected axioms and negated conjecture. If no unselected axiom can be evaluated as false in the model, then look for an unselected axiom that cannot be evaluated as true in the model (a *not-true* axiom). **Model Inadequacy:** In practice, it might not be possible to find a model even if one exists. If no model is found, then look for an unselected axiom that is *counter satisfiable* with respect to the selected axioms and negated conjecture. If no model is found, and no unselected axiom is counter satisfiable with respect to the selected axioms and negated conjecture, then select the axiom that has the highest syntactic relevance. **Aggressive Selection:** An axiom for which there is at least one model of the selected axioms, the conjecture, and the negation of that axiom, can be preferred over an axiom that is true in every model of the selected axioms and the conjecture. **Batch Selection:** In order to make faster progress it is possible to select multiple axioms at each iteration. **Limited Selection:** The number of unselected axioms that is considered by each method can be limited, to give higher priority to axioms with higher syntactic relevance. **Final Proof Attempt:** When the axiom selection process terminates, a final proof attempt is made using conventional automated reasoning, to build a proof.

3 Results

The process described has been implemented as the system SRASS. The various tests required by the process are all implemented using conventional automated reasoning systems. The implementation is in C, built on top of the JJParser library of TPTP compliant functions, and using the SystemOnTPTP harness for calling the component automated reasoning systems. For testing, SRASS was configured conservatively: no Initial Proof Attempts were made, Greedy Termination and Aggressive Selection were disabled, and no Batch Selection or Limited Selection was used. The automated reasoning systems used were: E 0.99 to test for provability and unsatisfiability, EP 0.99 to find proofs, FMDarwin 1.3g to build models, Paradox 2.0b to test for satisfiability and counter satisfiability, and SPASS 2.2 to further test for satisfiability and counter satisfiability. The testing was done on a 2.8GHz Intel Xeon computer with 1GB memory, and running Linux 2.6. An overall CPU limit of 600s was imposed.

Good results were produced on several problem sets, selecting only problems that E 0.99 cannot solve in 20s – 31 modal logic LCL problems, 25 Mizar SET problems, and 20 NASA SWV problems. Thus there are 76 problems (selected from the 495 in the original sets). EP finds proofs for 14 problems, and E establishes theoremhood for a further 11. SRASS finds proofs for 52 problems, and establishes theoremhood for a further 13. SRASS and EP both find proofs for 9 problems, and SRASS is faster than EP in 6 of these 9 cases. SRASS and E both establish theoremhood for 15 problems (including the cases where both find proofs), and SRASS is faster than E in 12 of these 15 cases, often by an order of magnitude. The syntactic relevance ordering is a key to the success of SRASS. Without syntactic relevance ordering SRASS solves only 42 of the problems (compared to the 65 solved with syntactic relevance ordering), and without syntactic relevance ordering the times taken are consistently much higher. Aggressive selection, on the other hand, is of less utility – SRASS solves 63 of the problems with aggressive selection enabled. However there are some cases where use of aggressive selection results in fewer axioms being selected. SRASS was also tested in the conservative configuration on three other problem sets: 13 of Art Quaife's SET problems, 9 human-style arithmetic NUM problems, and 161 software component retrieval SWC problems. These tests give less impressive results, with SRASS and EP solving similar numbers of problems, although not always the same problems.

SRASS was tested in a less conservative configuration on the MPTP Challenge problems. The challenge is divided into two divisions: the bushy division, and the chainy division, each of which has 252 problems. SRASS was configured to make an Initial Proof Attempt with a CPU limit of 30s, to Batch Select 3 axioms in each iteration, and to have a Limited Selection from the 10 most syntactically relevant axioms in each iteration. An overall CPU limit of 300s was imposed. The testing aimed only to establish theoremhood for each problem, without proofs being found. In the bushy division, E establishes theoremhood for 141 problems, and SRASS establishes theoremhood for 171. SRASS and E both establish theoremhood for 138 problems, E establishes theoremhood for 3 problems that SRASS does not, and SRASS establishes theoremhood for 33 problems for E does not. In the chainly division, E establishes theoremhood for 91 problems, and SRASS establishes theoremhood for 127. SRASS and E both establish theoremhood for 83 problems, E establishes theoremhood for 8 problems that SRASS does not, and SRASS establishes theoremhood for 44 problems for E does not.