Divvy: An ATP Meta-system Based on Axiom Relevance Ordering

Alex Roederer, Yury Puzis, and Geoff Sutcliffe*

University of Miami, USA

Abstract. This paper describes two syntactic relevance orderings on the axioms available for proving a given conjecture, and an ATP meta-system that uses the orderings to select axioms to use in proof attempts. The system has been evaluated, and the results show that it is effective.

1 Motivation, History, and Overview

In recent years there has been growing demand for Automated Theorem Proving (ATP) in large theories. A *large theory* is one that has many functors and predicates, has many axioms of which typically only a few are required for the proof of a theorem, and many theorems to be proved using the axioms. Examples of large theories that are in a form, or have been translated to a form, suitable for ATP include the SUMO ontology, the Cyc knowledge base, the Mizar mathematical library, the YAGO knowledge base, WordNet, and the MeSH vocabulary thesaurus.

Large theories pose challenges for ATP systems, which are different from the challenges of small theories. These include parsing and building data structures for the large numbers of formulae, loading and preprocessing the axioms only once, selecting axioms that are likely to be used for proving a given conjecture, and extracting heuristics and lemmas from proofs to improve subsequent performance. The work described in this paper addresses the issue of selecting axioms from a large theory, to obtain a proof of a conjecture. The aim is to select as few axioms as possible, but enough to obtain a proof. There have been previous efforts in this direction, including abstraction-based techniques [1], analysis of possible inference chains from the conjecture [6], axiom selection based on symbol count [10], axiom selection based on symbol overlap [5], axiom selection based on models of the formulae [7], and axiom selection based on machine learning from previous proofs [11]. In all cases, the general approach has been to order the axioms according to their *relevance* to the conjecture, and then use or select axioms with respect to the ordering. A system that selects axioms, i.e., the non-selected axioms are made unavailable, typically iterates a process of selecting axioms, executing an ATP system to try find a proof, and if the ATP system is unsuccessful (within the resource limits imposed) looping to make a new selection. A common feature of previous efforts is that they start with the

* Currently at the Automation of Logic group, Max Planck Institut für Informatik.

R.A. Schmidt (Ed.): CADE 2009, LNAI 5663, pp. 157–162, 2009.

axioms at the top of the relevance ordering, and work their way down. This is somewhat fragile, because the system performs poorly if just one necessary axiom is placed low in the ordering.

This paper describes two syntactic relevance orderings on the axioms available for proving a given conjecture, and an ATP meta-system called Divvy that selects axioms with respect to an ordering. Divvy uses a dividing approach that "starts in the middle", thus overcoming some of the fragility experienced by approaches that "start at the top". The combined system has been evaluated, and the results show that it is effective.

2 Two Syntactic Relevance Orderings

2.1 Ordering Based on Symbol Overlap

This relevance measure is based on the intuitive notion of whether or not formulae are "talking about the same things". This is measured in terms of the extent to which the formulae use the same predicate and function symbols.

First, the *contextual direct relevance* between all formulae pairs $F_1, F_2 \in S = Axioms \cup \{Conjecture\}$ is measured by

$$\frac{\sum_{s \in (sym(F_1) \cap sym(F_2))} \left(1 - \frac{|\{f : f \in S, s \in sym(f)\}|}{|S|}\right)}{|sym(F_1) \cup sym(F_2)|}$$

where $sym(F)$ is the set of function and predicate symbols occurring in F. The numerator sums the symbol weights for the symbols that occur in both F_1 and F_2, where the weight is 0 for symbols that occur in all formulae in S, and higher (approaching 1) for symbols that occur in fewer formulae. This sum is scaled by the number of unique symbols in F_1 and F_2, to produce a value in the range 0 to 1. (This a variant of the Jaccard similarity coefficient [4].) Next, the *contextual path relevance* of every path $F_a = F_1 \cdot F_2 \cdot \ldots \cdot F_n = F_c$ from an axiom F_a to the conjecture F_c is calculated as the smallest contextual direct relevance in the path, divided by the length of the path. This captures the intuition that a path is only as strong as its weakest link, and that relevance decreases with distance. Finally, the *contextual indirect relevance* between F_a and F_c is taken as the maximal contextual path relevance over all paths connecting F_a to F_c. (The latter two steps are implemented together using Dijkstra's algorithm.)

There are several adaptations of this basic measure, including different ways of weighting symbols, taking variables into account, and different options for treating predicate and function symbols separately. The basic and adapted measures have been implemented in C++, as the Prophet tool. It is available for use in the SystemB4TPTP interface, at `http://www.tptp.org/cgi-bin/SystemB4TPTP`.

2.2 Ordering Based on Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a technique for analysing of relationships between documents, using the terms they contain [3]. LSA has been used to

compute the relevance of axioms to a conjecture by treating the formulae as documents, and the predicate and function symbols as the terms they contain.

The computation of axiom relevance using LSA is a three step process. First, a relationship strength between every pair of symbols is computed. An initial relationship strength is computed based on the co-occurrences of the symbols in the formulae, and the total number of formulae containing the symbols (like the numerator of Prophet's contextual direct relevance). The final relationship strength is computed by repeatedly combining the existing relationship strength with the relationship strengths between each of the two symbols and each other symbol, i.e., taking into account transitive relationships between symbols. Second, a relationship strength vector is computed for each formula. The vector has an entry for each symbol. A symbol's entry is the sum, across all other symbols, of the product of the relationship strength between the two symbols, and the number of occurrences of the other symbol in the formula (so that other symbols that do not occur in the formula make no contribution to the vector entry). Finally, the relevance of each axiom to the conjecture is computed as the dot product of their symbol relationship strength vectors.

The LSA approach to computing axiom relevance has been implemented in C, as the Automated Prophesier of Relevance Incorporating Latent Semantics (APRILS) tool. It is available for use through the SystemB4TPTP interface.

APRILS has been evaluated against Prophet by comparing their relevance orderings of the axioms for 1337 TPTP problems for which EP has found a proof. Their highest ranks of an axiom used in the proof of each problem were compared - a higher rank is worse. APRILS did better than Prophet for 654 (49%) of the problems, and tied for 301 (23%) of the problems. Most of the ties were on problems containing few axioms. APRILS' methods are better suited for problems with large numbers of axioms, which contain more semantic information.

3 The Divvy ATP Meta-system

The Divvy ATP meta-system uses a relevance ordering to select subsets of the axioms available for proving a given conjecture, and attempts to prove the conjecture from the selected axioms. The basic idea is very simple - start by selecting the top half of the axioms (in the ordering), and try prove the conjecture. If the conjecture is proved then stop. If the conjecture is countersatisfiable[1] with respect to the selected axioms, then more axioms are needed. If nothing is established about the conjecture, e.g., because the proof attempt reached a resource limit, then fewer or more axioms are needed. The top quarter, and then the top three quarters, of the axioms are then considered. If neither of those produce a proof, the top eighth, three eighths, five eighths, and seven eighths, are selected. This continues, selecting an odd number of sixteenths, thirtysecondths, etc., until a granularity limit (halves, quarters, eighths, etc.), or global resource limit,

[1] *Countersatisfiable* is an SZS ontology status value [9], meaning that the conjecture is not provable from the axioms.

is reached. In all cases, selection of fewer axioms than the maximal number that has led to a countersatisfiable result is rejected.

The basic process is improved by several optimizations. First, a proof attempt using all the axioms can be made before the dividing and selecting starts. Second, the user can specify a maximal number of axioms to ever be selected, and only that number of axioms is considered from the relevance ordered list. Third, before each proof attempt, model finders can be run to try show that the conjecture is countersatisfiable with respect to the selected axioms. This aims to avoid proof attempts that fail, and to raise the maximal number that has led to a countersatisfiable result. Fourth, the proof attempts can use an ATP system that only establishes an assurance of the existence of a proof, and ultimately an ATP system that outputs a full proof is run on the conjecture and selected axioms. Fifth, the user can specify the ATP systems to be used for proof assurance, proof finding, and model finding. Sixth, and most relevant, the user can specify the tool to be used for computing the relevance ordering for the axioms.

CPU limits are imposed on the initial ATP system run using all axioms, the model finding runs, the individual ATP systems runs using selected axioms, and the overall process. The CPU limit on the individual ATP systems runs is dynamically adjusted to take into account the minimal number of axioms that must be selected (based on countersatisfiable results).

Divvy is implemented in C, and relies heavily on the TPTP world infrastructure for manipulating the formulae, running the relevance measuring tool, and running the ATP systems. It is available for use through the SystemOnTPTP interface at `http://www.tptp.org/cgi-bin/SystemOnTPTP`.

4 Evaluation

Divvy, using Prophet and APRILS, has been evaluated on the MPTP challenge problems. These are two sets of 252 problems extracted from the Mizar mathematical library by the MPTP process. The problems in the "Bushy" set have the axioms that the MPTP process determines might be necessary for a proof. While the MPTP process tries to minimize this set, each problem contains many unnecessary axioms. The problems in the "Chainy" set have all preceding Mizar knowledge as axioms, i.e., there are very many unnecessary axioms. The MPTP challenge problems, and results for some well known ATP systems, are available at `http://www.tptp.org/Challenges/MPTPChallenge/`.

Divvy was configured to use the E 1.0 ATP system [8] to establish the existence of proofs, Paradox 3.0 [2] to establish countersatisfiability, and EP 1.0 to ultimately produce the proofs. Some testing was also done using Fampire instead of E. Fampire 1.3 is the plain ATP system with the best results on the MPTP challenge. While the results using Fampire are of interest, the full evaluation was done using E because of its availability and stable high performance as a monolithic ATP system. (Fampire, in contrast, is a little known and undocumented system.) An overall CPU limit of 300s was imposed, and an initial ATYP system run using all axioms was done with a 60s CPU limit, i.e., leaving at least 240s

Table 1. Divvy results for the MPTP challenge problems, for various axiom orderings

System	Bushy			Chainy		
	Total	<60s	>60s	Total	<60s	>60s
E	141	139	2	91	80	11
Divvy(E)+Original	163	139	24	80	80	0
Divvy(E)+Reverse	151	141	10	92	82	10
Divvy(E)+Prophet-indirect	170	137	33	113	81	32
Divvy(E)+Prophet-direct	175	137	38	118	81	37
Divvy(E)+APRILS	180	140	40	117	80	37
Fampire	191	165	26	126	78	48
Divvy(F)+APRILS	186	165	21	132	68	64

for ATP system runs using selected axioms. All axioms were always available for selection, and the dividing was done down to a granularity of eighths. The testing was done on a 2.8GHz Intel Xeon computer with 1GB memory, and running Linux 2.6.

Table 1 tabulates the results. The "E" and "Fampire" rows give the results for E and Fampire alone. The "Divvy(*System*)+*Ordering*" rows give the results for Divvy using either the E or Fampire system to establish the existence of a proof, and the axioms ordered either as they are in the original problem, the reverse of the original problem, or according to the relevance values computed by Prophet or APRILS. The "<60s" and ">60s" columns give the number of problems solved in less than and more than 60s. For the "Divvy" rows this separates the problems solved in the initial ATP system run using all axioms from those that benefited from the axiom selection. Note that the number solved in less than 60s is not constant across the "Divvy(E)" rows, due to the reordering of the axioms.

The results show that the basic Divvy idea, despite its simplicity, takes effective advantage of axiom ordering. The fact that Divvy improves on E without any axiom ordering indicates that the original order of the axioms in the MPTP challenge problems is better than random - more relevant axioms already occur earlier in the problems, thanks to the nature of the MPTP process. This is confirmed by the reduced performance using the axioms in reverse order. The successive improvements obtained by adding explicit axiom orderings shows that Prophet and APRILS do compute meaningful axiom relevance. The improvement from the original ordering to APRILS' ordering is 10% - a non-trivial improvement for these challenging problems. When Fampire is used as the underlying ATP system, the reduced performance of Divvy on the Bushy problems is believed to be an artifact of Fampire's strategy scheduling, which works effectively only with a reasonably large CPU limit (e.g., around 300s). When Fampire is used in Divvy, with Divvy being given an overall CPU limit of 300s, the individual ATP system runs receive relatively small CPU limits, typically less than 50s. The significantly larger number of unnecessary axioms in the Chainy problems makes the relevance ordering and axiom selection more valuable for Fampire.

5 Conclusion

This work and implemented system have shown that axiom ordering in conjunction with axiom selection is an effective technique for proving theorems in large theories. The syntactic orderings presented in this paper have been shown to be meaningful, and are available for use in other contexts. The Divvy ATP meta-system has shown that "starting in the middle" is a useful way of selecting axioms, and can be used with any meaningful axiom ordering. Current work is focussing on optimizing the accuracy and runtime performance of APRILS.

References

1. Barker-Plummer, D.: Gazing: An Approach to the Problem of Definition and Lemma Use. Journal of Automated Reasoning 8(3), 311–344 (1992)
2. Claessen, K., Sorensson, N.: New Techniques that Improve MACE-style Finite Model Finding. In: Baumgartner, P., Fermueller, C. (eds.) Proceedings of the Workshop on Model Computation - Principles, Algorithms, Applications (2003)
3. Deerwester, S., Dumais, S., Furnas, G., Landauer, K., Harschman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
4. Jaccard, P.: Étude Comparative de la Distribution Florale Dans une Portion des Alpes et des Jura. Bulletin del la Société Vaudoise des Sciences Naturelles 37, 547–579 (1901)
5. Meng, J., Paulson, L.: Lightweight Relevance Filtering for Machine-Generated Resolution Problems. In: Sutcliffe, G., Schmidt, R., Schulz, S. (eds.) Proceedings of the FLoC 2006 Workshop on Empirically Successful Computerized Reasoning. CEUR Workshop Proceedings, vol. 192, pp. 53–69 (2006)
6. Plaisted, D.A., Yahya, A.: A Relevance Restriction Strategy for Automated Deduction. Artificial Intelligence 144(1-2), 59–93 (2003)
7. Pudlak, P.: Semantic Selection of Premisses for Automated Theorem Proving. In: Urban, J., Sutcliffe, G., Schulz, S. (eds.) Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories. CEUR Workshop Proceedings, vol. 257, pp. 27–44 (2007)
8. Schulz, S.: E: A Brainiac Theorem Prover. AI Communications 15(2-3), 111–126 (2002)
9. Sutcliffe, G.: The SZS Ontologies for Automated Reasoning Software. In: Sutcliffe, G., Rudnicki, P., Schmidt, R., Konev, B., Schulz, S. (eds.) Proceedings of the Workshop or Knowledge Exchange: Automated Provers and Proof Assistants. CEUR Workshop Proceedings, vol. 418, pp. 38–49 (2008)
10. Sutcliffe, G., Dvorsky, A.: Proving Harder Theorems by Axiom Reduction. In: Russell, I., Haller, S. (eds.) Proceedings of the 16th International FLAIRS Conference, pp. 108–112. AAAI Press, Menlo Park (2003)
11. Urban, J., Sutcliffe, G., Pudlak, P., Vyskocil, J.: MaLARea SG1: Machine Learner for Automated Reasoning with Semantic Guidance. In: Baumgartner, P., Armando, A., Gilles, D. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 441–456. Springer, Heidelberg (2008)