Proving Harder Theorems by Axiom Reduction

Geoff Sutcliffe
Department of Computer Science
University of Miami
P.O. Box 248154, Coral Gables, FL 33124, USA
Email: geoff@cs.miami.edu

Alexander Dvorsky
Department of Mathematics
University of Miami
P.O. Box 249085, Coral Gables, FL 33124, USA
Email: dvorsky@math.miami.edu

Abstract

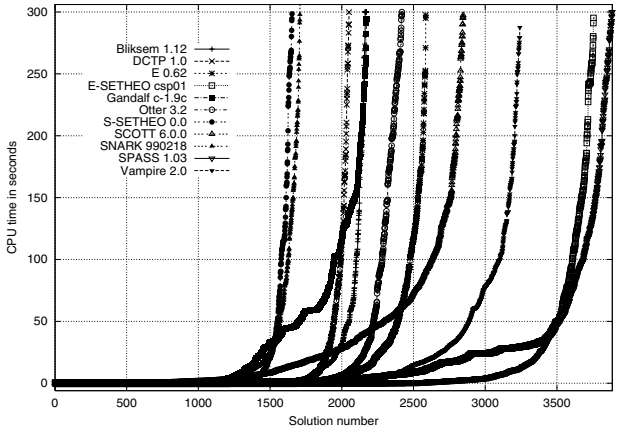
Automated Theorem Proving (ATP) problems may contain unnecessary axioms, either because some of the axiomatization of the theory is irrelevant to the particular theorem, or because the axiomatization is redundant by design. ATP systems do not have effective techniques for detecting that axioms are unnecessary (or unlikely to be necessary) to the proof of a theorem. Axiom reduction removes combinations of axioms from an ATP problem, and submits the resultant axiom-reduced problems to an object ATP system. When a combination of only unnecessary axioms is removed, the problem may be quickly solved.

Introduction

Automated Theorem Proving (ATP) is concerned with the development and use of computer programs that automate sound reasoning: the derivation of conclusions that follow inevitably from facts. In this work we are dealing with ATP for 1st order classical logic, which has well known computational properties, and henceforth all discussion is in that context. Current ATP systems are capable of solving non-trivial problems, e.g., EQP solved the Robbins problem (McCune 1997). In practice, however, the search complexity of interesting problems is often enormous, and many problems cannot currently be solved within realistic resource limits. Therefore a key concern of ATP research is the development of techniques that can be used to £nd proofs of harder theorems.

ATP systems can be designed and implemented to be (refutation) complete, i.e., they will always produce a solution if one exists. However, this guarantee is honored in general only if in£nite resources (time and memory) are provided. In reality, the performance of current state-of-the-art ATP systems has a quite distinctive form. Figure 1 plots the CPU times taken by several contemporary ATP systems to solve TPTP problems[1], for each solution found, in increasing order of time taken. The relevant feature of these plots is that each system has a point at which the time taken to £nd solutions starts to increase dramatically. This point is called the system's Peter Principle Point (PPP). Evidently a linear increase in the computational resources beyond the PPP would not lead to the solution of signi£cantly more problems. Users typically impose a realistic CPU time limit that allows the system to pass its PPP, and problems that are not solved within the limit are considered to be beyond the reach of the system. Figure 1 suggests that 300s is a realistic CPU time limit for many systems.

Figure 1: Peter Principle Points



In order to solve more problems within a realistic time limit, ATP researchers strive to improve their techniques and systems, and there has been impressive progress in the last decade (Sutcliffe, Fuchs, & Suttner 2001). The increased power of ATP systems "rolls down" the performance curves: some problems that are on the rapidly rising part of the system's performance curve are solved quickly and move down to extend the ¤atter part of the performance curve, and some problems that were considered to be beyond the reach of the system are solved within the CPU time limit.

An alternative to relying on increased ATP system power for solving more dif£cult problems is to reformulate the problems so that they can be solved within the CPU time limit. Rather than changing an ATP system's performance curve, this approach aims to move problems into the portion of the curve below the user's CPU time limit, ideally

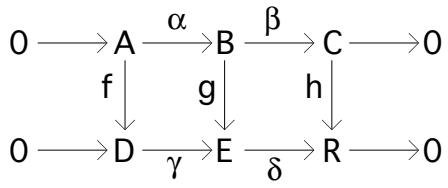
[1]The TPTP Problem Library (Sutcliffe & Suttner 1998) is the accepted standard for testing ATP systems.

down to the ¤atter portion of the curve. This ideal is not unrealistic, given the sharpness of the PPP - a small change in problem dif£culty can produce a dramatic decrease in the time required to solve it. A solution to such a reformulated problem may immediately satisfy the user's need, or the solution may be used to help produce a solution to the original problem. Examples of this approach include the magic set transformation (Stickel 1994) and iterative easing (Sutcliffe 2001). Axiom reduction is another technique in this family. Axiom reduction reformulates a problem by removing axioms that may be unnecessary for a proof of the theorem. A proof for such a reformulated problem is immediately a proof for the original problem.

A Motivating Example "Diagram Chasing" in Homological Algebra

The *Short Five Lemma* (Weibel 1994) is a standard departure point for any homological algebra course. It considers the following commutative diagram with exact rows, in an arbitrary abelian category, e.g., a category of abelian groups:

Figure 2: Short 5 Lemma diagram



All of the maps in the diagram are homomorphisms of abelian groups (in particular, they map the 0 of the domain to the 0 of the codomain). The exactness of the rows means that α and γ are injections, and that β and δ are surjections. Exactness in the middle term means that $\ker \beta = \text{Im } \alpha$ and $\ker \delta = \text{Im } \gamma$. Finally, the two squares in the diagram commute, i.e., $\gamma(f(a)) = g(\alpha(a)), a \in A$ and $\delta(g(b)) = h(\beta(b)), b \in B$. Thus eight conditions are imposed on the morphisms.

Part 1 of the short £ve lemma claims that if f and h are injections then g is an injection. Part 2 claims that if f and h are surjections then g is a surjection. Only four of the eight conditions are necessary for each of the proofs. Two of the conditions - injectivity of α and surjectivity of δ - are not needed for either proof.

These problems are easily formulated in 1st order logic, and can be submitted to an ATP system. For example, the properties of a morphism are captured by the axiom (using the FOF syntax of the TPTP problem library):

```
! [Morphism,Dom,Cod] :
  ( morphism(Morphism,Dom,Cod)
 => ( ! [El] :
        ( element(El,Dom)
       => element(apply(Morphism,El),Cod) )
    & equal(apply(Morphism,zero(Dom)),zero(Cod)) ) )
```

and the necessary properties for a surjection are captured by the axiom:

```
! [Morphism,Dom,Cod] :
  ( ( morphism(Morphism,Dom,Cod)
    & ! [ElCod] :
        ( element(ElCod,Cod)
       => ? [ElDom] :
            ( element(ElDom,Dom)
            & equal(apply(Morphism,ElDom),ElCod) ) ) )
 => surjection(Morphism) )
```

The fact that α is an injective morphism is asserted by the two formulae:

```
morphism(alpha)
injection(alpha)
```

The full axiomatization of the theory, and the conjectures of part 1 and part 2, are available online at:

```
http://www.cs.miami.edu/~tptp/ATPSystems/RedAx/
```

The axiomatization provides:

- The properties of morphisms, injections, surjections, exact sequences, and commutative squares.
- The necessary properties for injections, surjections, exact sequences, and commutative squares.
- The de£nition of subtraction in an abelian group.
- The de£nition of the diagram in Figure 2.

With the complete axiomatization, part 1 provides a reasonable challenge to contemporary ATP systems. For example, Vampire 5.0 (Riazanov & Voronkov 2002) £nds a proof[2] in 42.3s, using its automatic mode, while E 0.7 (Schulz 2002) cannot £nd a proof in 300s. All testing was done on a 450MHz SUN Ultra 80 with 1GB RAM.

The proof uses only a subset of the axiomatization:

- The properties of morphisms, injections, exact sequences, and commutative squares.
- The necessary properties for injections.
- The facts that γ is an injection, the upper row is exact, and the two squares commute.

Reformulating the problem to contain only the necessary axioms makes it much easier: Vampire £nds a proof in 0.1s, and E £nds a proof in 0.6s. Reducing the axiomatization brings the problem into the ¤atter parts of both Vampire's and E's performance curves. This success motivated the development of this technique, the removal of unnecessary axioms, for general application.

Axiom Reduction

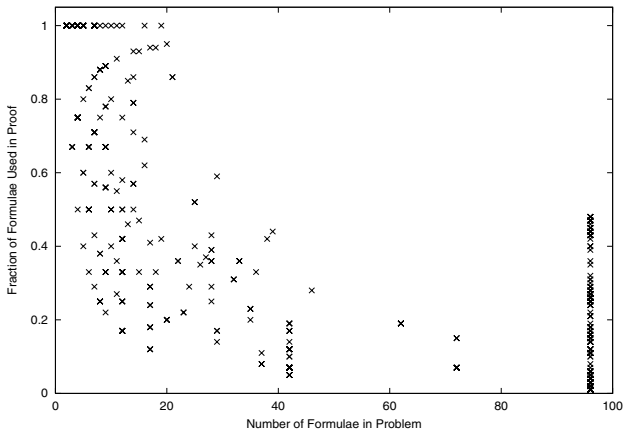
There are two situations in which unnecessary axioms may be present in the formulation of a problem. First, as in part 1 of the short £ve lemma, some of the axiomatization may be irrelevant to the theorem. Irrelevant axioms increase the search space but cannot contribute to a proof. Second, the axiomatization may be redundant by design. For example, in group theory a minimal axiomatization contains only the left or right identity axiom - the other is dependent, but commonly both are provided (ditto for the inverse axioms). Although dependent axioms may be helpful, e.g., for £nding shorter proofs, they also increase the search space.

A survey of proofs of FOF problems from the TPTP con£rms that many of the problems contain unnecessary axioms. There are 1135 FOF problems in TPTP v2.5.0 that

[2] Actually, Vampire produces a refutation of the clause normal form of the problem.

are believed to be theorems. Of these, 939 have axiom formulae, and hence may have unnecessary axioms. SPASS 2.0 (Weidenbach & et al. 2002), a system recognized for its ability on FOF problems due to its excellent FOF to CNF converter, can solve 513 of the 939, with a 600s CPU time limit. (Again, all testing was done on the 450MHz SUN Ultra 80 with 1GB RAM.) Figure 3 shows the fraction of formulae in SPASS' proofs vs. the number of formulae in those 513 problems. Evidently many of the problems contain unnecessary axioms. Note that the 600s CPU time limit is well beyond SPASS' PPP.

Figure 3: Fractions of formulae used in SPASS proofs



Human theorem provers are often capable of recognizing that certain axioms are unnecessary (or unlikely to be necessary) to the proof of a theorem, and hence avoid introducing those axioms into their proof attempt. However, human theorem provers have to be careful about which axioms they avoid, because their work rate is very low and they cannot afford too many proof attempts without all the necessary axioms. In contrast to human theorem provers, ATP systems do not have effective techniques for detecting that axioms are unnecessary (or unlikely to be necessary) to the proof of a theorem. Subsumption style techniques (Wos, Overbeek, & Lusk 1993; Benanav 1992) determine that a formula may be discarded because a "better" formula is available, but do not detect that a formula is in principle unnecessary for proof. Data management techniques such as Vampire's limited resource strategy (Riazanov & Voronkov 2002) and Otter's `max_weight` setting (McCune 1994) remove formulae that are not expected to be processed due to resource limits, but make no judgement regarding their potential to contribute to a proof. Relevancy testing techniques - see (Plaisted & Yahya To appear) for an example and further references - seem to require signi£cant machinery in order to be effective. There are only a few simple techniques that remove formulae that oviously cannot contribute to a proof, based on syntactic or semantic arguments, e.g., pure-literal deletion and tautology deletion. When equality is present it is dif£cult to use syntactic methods to determine that a formula cannot contribute to a proof. Alongside, and partially caused by, the inability to detect which axioms are unnecessary for a proof, the performance curves in Figure 1 show that ATP systems are more effective at quickly £nding proofs for easy problems than they are at taking a long time to £nd proofs for harder problems.

The combination of unnecessary axioms, ATP systems' inability to carefully detect unnecessary axioms, and ATP systems' better ability on easy problems, suggest trying to make a problem easier for ATP systems to solve by "quickly and carelessly" removing combinations of axioms, in the hope that some combination removes only unnecessary axioms and the theorem can be quickly proved from the remainder. This is the technique of axiom reduction. Axiom reduction removes combinations of axioms from a problem, and submits the resultant *axiom-reduced* problem to an object ATP system. A short CPU time limit is imposed on the object ATP system so that if the axiom-reduced problem is not easily solved, then the proof attempt is abandoned and another axiom-reduced problem can be generated. Axiom reduction aims to generate an axiom-reduced problem that falls into the ¤atter portion of the object systems performance curve, where a short time limit is enough to £nd a proof.

Axiom reduction has been implemented in the RedAx (**Red**uce **Ax**ioms) meta-ATP system. RedAx is parameterized by the object ATP system, and a time limit for each of the object system's proof attempts. RedAx preemptively removes equality axioms that are not needed by the object ATP system, e.g., if the object system is SPASS, all equality axioms are removed. The remaining axioms are weighted according to the number of symbols in the formula and the depth of nesting of functions. Axioms are then removed in a simple combinatorial fashion, with a preference for removing heavy axioms. Successively, all combinations of $0, 1, 2, \ldots$ axioms are removed from the problem, and the resultant axiom-reduced problem is submitted to the object ATP system through the SystemOnTPTP package (Sutcliffe 2000). RedAx is implemented in `perl`.

Testing Results

RedAx has been tested on problems from TPTP v2.5.0, using SPASS 2.0 and Vampire 5.0 as the object ATP systems. All testing was done on the 450MHz SUN Ultra 80 with 1GB RAM.

The £rst round of testing used SPASS 2.0 as the object ATP system, on 60 TPTP FOF problems randomly selected from the 426 problems with axioms that SPASS could not solve alone within 600s. A simpli£ed variant of the problem selection technique used in the CADE ATP System Competition (Sutcliffe, Suttner, & Pelletier 2002) was used to ensure a broad selection of problem types. A 3600s CPU time limit was imposed, with a 60s limit for each axiom-reduced problem. Additionally, for comparison, SPASS 2.0 was run alone on the problems with a 3600s time limit. Table 1 shown the results of this testing. RedAx with SPASS found proofs for 14 problems, and SPASS alone found proofs for 7 problems, with an overlap of 6 problems. The remaining

45 problems were not solved by either system con£guration within the 3600s time limit.

The TPTP rating in the second column of Table 1 measures the dif£culty of the problem with respect to the current state-of-the-art in ATP (Sutcliffe & Suttner 2001). Problems rated 0.00 are easy, and problems rated 1.00 have not been solved by any ATP system in regular testing (such testing typically imposes a realistic CPU time limit of 300s). Seven of the problems solved by only RedAx have a rating of 0.50 or higher. The results thus show that axiom reduction effectively extends the capability of SPASS on hard problems.

Problem	TPTP Rating	SPASS 2.0 CPU time	RedAx CPU time	# of Axioms	# Used
GEO083+1	0.67	Timeout	2059.9	16	14
GEO084+1	0.50	3041.8	2038.4	16	14
GEO093+1	0.67	2245.6	173.7	16	15
MGT005+2	0.33	Timeout	2438.0	12	10
MGT042+1	0.83	3500.4	249.6	9	8
MGT043+1	0.33	1380.0	267.1	9	8
MGT047+1	0.50	Timeout	819.6	15	14
MGT051+1	0.50	Timeout	818.5	15	14
MGT064+1	0.83	1559.2	642.8	19	18
SET011+3	0.57	Timeout	1447.6	7	5
SET593+3	0.71	Timeout	244.8	6	5
SET594+3	0.14	2591.9	1624.6	8	6
SET595+3	0.71	1788.8	Timeout	8	-
SET351+4	0.86	Timeout	1864.2	11	9
SWC021+1	0.67	Timeout	383.1	95	94

Table 1: Results for RedAx and SPASS 2.0

The second round of testing used Vampire 5.0 as the object ATP system, on the 73 geometry and set theory FOF problems with a TPTP dif£culty rating of 1.00. Geometry and set theory were chosen because they have rich axiomatizations, and it is realistic to a priori believe that not all axioms will be necessary for the proofs of all theorems. Vampire was chosen because it is a very powerful system for problems with the characteristics of the geometry and set theory problems. They are non-Horn problems with some equality, and Vampire 5.0 was the best performing system for this category of problems in the CADE-18 ATP system competition. A 3600s CPU time limit was imposed, with a 60s limit for each axiom-reduced problem. Additionally, for comparison, Vampire 5.0 was run alone on the problems with a 3600s time limit. Table 2 shown the results of this testing. RedAx with Vampire found proofs for 4 problems, and Vampire alone found no proofs.

Problem	TPTP Rating	Vampire 5.0 CPU time	RedAx CPU time	# of Axioms	# Used
GEO098+1	1.00	Timeout	1080.8	16	14
GEO110+1	1.00	Timeout	1145.1	17	15
GEO116+1	1.00	Timeout	473.7	17	16
SET674+3	1.00	Timeout	1609.7	40	38

Table 2: Results for RedAx and Vampire 5.0

The success of RedAx with Vampire on these problems is signi£cant because it shows that axiom reduction can be used not only to extend the capability of a particular ATP system, but can also be used to push the frontier of what ATP systems can deal with at all.

Analysis and an Application

The simple combinatorial removal of axioms means that very many axiom-reduced problems can be formed and submitted to the object ATP system. If there are N axioms then there are $2^N - 1$ possible axiom-reduced problems. This $O(E)$ search space has to be viewed in the context of the super-exponential growth of the search space of ATP systems, which is $O(N^{2^{searchdepth}})$. Axiom reduction helps the object ATP system by reducing N, so that it has a smaller search space. Additionally, by removing successively more axioms, RedAx may £nd a proof with some unnecessary axioms left in the axiom-reduced problem, i.e., without having to search through to an axiom-reduced problem containing a minimal axiomatization for the theorem. This possibility is borne out by the testing results, in which proofs were found typically with only one or two axioms removed. It seems that a "few rotten eggs can spoil the basket" for ATP systems, and that axiom reduction can be used to remove them.

The short CPU time limit imposed on the object ATP system is founded in the hope that an axiom-reduced problem will fall into the ¤atter portion of the system's performance curve. Within the short time limit the object ATP system develops less of its search space, and in turn this may enable the system to make better local decisions about the direction of its search. In the context of the $O(E)$ RedAx search space, the short time limit is necessary in order to work through more of the possible axiom-reduced problems. A reasonably long overall CPU time limit is appropriate for RedAx, £rstly to allow more axiom-reduced problems to be generated and attempted, and secondly because each axiom-reduced problem provides new hope for a proof. This contrasts with giving an ATP system enough time to go past its PPP, where there is little increased hope for £nding a proof.

An Application in Homological Algebra

Axiom reduction has been used in a piecewise construction of a proof of part 2 of the short £ve lemma. Recall that part 2 claims that if f and h are surjections then g is a surjection. Even with the minimal axiomatization required for the proof, this problem appears to be well beyond the reach of contemporary ATP systems, with a realistic or even unrealistic CPU time limit. In order to construct the full proof, three lemmas were used as intermediate steps:

IL1 For every element $e \in E$ there exist elements $b \in B$ and $e' \in E$ such that $e' = g(b) - e$ and $\delta(e') = 0_R$.

IL2 IL1 extended to claim that there exists an element $a \in A$ such that $e' = \gamma(f(a)) = g(\alpha(a))$.

IL3 For every element $e \in E$ there exists an element $b \in B$ such that $e = g(b)$. (It is surprising that this lemma about surjection is necessary, but without it ATP systems seem unable to make the £nal step in the overall proof.)

The ATP problems were therefore:

P1 Prove IL1 from the axiomatization.

P2 Prove IL2 from the axiomatization and IL1.

P3 Prove IL3 from the axiomatization and IL2.

P4 Prove part 2 of the short £ve lemma from the axiomatization and IL3.

Only P4 can be solved directly by Vampire 5.0 with a 7200s time limit, in 0.6s. RedAx with Vampire 5.0 fares no better, solving only P4 in 4.2s (note the overhead of the RedAx problem preparation). A version of Vampire manually con£gured by the developer for these problems can solve P1, P3, and P4, in 2.1s, 30.4s, and 4.4s, respectively. The manually con£gured Vampire cannot solve P2 with a 15000s CPU time limit. However, using the manually con£gured Vampire in RedAx, with a 60s limit for each axiom-reduced problem, produces a proof for P2 in 12265s. The axiom-reduced problem has 29 of the 31 axioms, omitting the definition of a morphism and the property that subtraction produces an element in the same set. It is interesting that a proof is possible without the de£nition of a morphism; here the remaining axioms are suf£cient to derive the instance of the de£nition used in the proof. The proof is found from the 202nd axiom-reduced problem, in only 4s. Thus without the two axioms a proof can be found very easily (setting the CPU time limit for each axiom-reduced problem to 4s would give an overall CPU time of only 808s!). With all the component proofs available, a proof for part 2 of the short £ve lemma can be constructed

Conclusion

Axiom reduction differs from much of the current development of ATP systems and techniques, which aims to solve both a broader range of problems and some harder problems. Axiom reduction aims only to £nd proofs of harder theorems. As well as being directly useful in a production environment, £nding such proofs provides valuable information to both users and developers of ATP systems. For users, axiom reduction identi£es unnecessary axioms. Finding small or minimal sets of axioms suf£cient for the proof of a theorem is of scienti£c interest. For developers, having access to a proof provides insights that can lead to further development of their ATP systems, bringing the problem and other problems within the reach of the systems (with a realistic time limit).

The current simple implementation of axiom reduction in RedAx has already been successful and useful. The success is founded on insights into the performance characteristics of ATP systems, acknowledgement of the existence of unnecessary axioms, and adaptation of the human ability to focus on relevant axioms. Immediate future work is to more carefully order the axioms for removal, based on the expectation that axioms that share predicate and function symbols with the conjecture are more likely to be necessary than those that do not. This will be implemented by heuristically evaluating axioms: axioms that share symbols with the conjecture will have weight 1, and recursively, unweighted axioms that share symbols with axioms of weight W will have weight $W+1$ (this is an intuitive relaxation of the technique used in (Plaisted & Yahya To appear)). This weighting function will be combined with the current symbol and depth measures. Further experimentation is also planned. In particular, it is planned to investigate the ef£cacy of removing all combinations of $N-1, N-2, \ldots$ axioms from the axiomatization, i.e., removing more axioms before less. One possible outcome is to detect minimal subsets of axioms suf£cient for a proof.

References

Benanav, D. 1992. Recognising Unnecessary Clauses in Resolution Based Systems. *Journal of Automated Reasoning* 9(1):43–76.

McCune, W. 1994. Otter 3.0 Reference Manual and Guide. Technical Report ANL-94/6, Argonne National Laboratory, Argonne, USA.

McCune, W. 1997. Solution of the Robbins Problem. *Journal of Automated Reasoning* 19(3):263–276.

Plaisted, D., and Yahya, A. To appear. A Relevance Restriction Strategy for Automated Deduction. *Arti£cial Intelligence*.

Riazanov, A., and Voronkov, A. 2002. The Design and Implementation of Vampire. *AI Communications* 15(2-3):91–110.

Schulz, S. 2002. E: A Brainiac Theorem Prover. *AI Communications* 15(2-3):111–126.

Stickel, M. 1994. Upside-Down Meta-Interpretation of the Model Elimination Theorem-Proving Procedure for Deduction and Abduction. *Journal of Automated Reasoning* 13(2):189–210.

Sutcliffe, G., and Suttner, C. 1998. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning* 21(2):177–203.

Sutcliffe, G., and Suttner, C. 2001. Evaluating General Purpose Automated Theorem Proving Systems. *Arti£cial Intelligence* 131(1-2):39–54.

Sutcliffe, G.; Fuchs, M.; and Suttner, C. 2001. Progress in Automated Theorem Proving, 1997-1999. In Hoos, H., and Stützle, T., eds., *Proceedings of the IJCAI'01 Workshop on Empirical Methods in Arti£cial Intelligence*, 53–60.

Sutcliffe, G.; Suttner, C.; and Pelletier, F. 2002. The IJCAR ATP System Competition. *Journal of Automated Reasoning* 28(3):307–320.

Sutcliffe, G. 2000. SystemOnTPTP. In McAllester, D., ed., *Proceedings of the 17th International Conference on Automated Deduction*, number 1831 in Lecture Notes in Arti£cial Intelligence, 406–410. Springer-Verlag.

Sutcliffe, G. 2001. The Design and Implementation of a Compositional Competition-Cooperation Parallel ATP System. In *Proceedings of the 2nd International Workshop on the Implementation of Logics*, 92–102.

Weibel, C. 1994. *An Introduction to Homological Algebra*. Cambridge University Press.

Weidenbach, C., and et al. 2002. SPASS 2.0. In Voronkov, A., ed., *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in Lecture Notes in Arti£cial Intelligence, 275–279. Springer-Verlag.

Wos, L.; Overbeek, R.; and Lusk, E. 1993. Subsumption, a Sometimes Undervalued Procedure. Technical Report MCS-P93-0789, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, USA.