SBR3: A Refutational Prover for Equational Theorems

Siva Anantharaman, Nirina Andrianarivelo
LIFO, Dépt. Math-Info.
Université d'Orléans
45067 ORLEANS Cedex 02
FRANCE
{siva,andria}@univ-orleans.fr

Maria Paola Bonacina, Jieh Hsiang *
Department of Computer Science
SUNY at Stony Brook
Stony Brook NY 11794-4400 USA
{bonacina, hsiang}@sbcs.sunysb.edu

May 25, 1990

1 Introduction

*SBR*3 is the latest version of a long line of term rewriting based theorem provers aimed at proving theorems in equational logic. The first of them, *Reve*, was written by P. Lescanne ([Le-83]). It evolved into the *Reve*2.*n* family, written and maintained mostly by J. Guttag's group at MIT [FG-84], and *Reve*3 for E-term rewriting at Nancy ([KK-85]). The *Reve* family is mostly concerned with completing a set of equations into a canonical system. Although well-worthy in its own right, such systems are not ideal for the purpose of proving a specific equational theorem.

In 1986, Mzali and Hsiang started to develop a new system, *SbReve*1 [HM-88], based on *Reve*2.4. The goal of *SbReve*1 was to modify the completion process in order to efficiently prove a single theorem of an equational theory rather than generating a canonical system. *SbReve*1 was overhauled into *SbReve*2 by Anantharaman at the Université d'Orléans ([AHM-89]). Much more sophisticated search mechanisms are further incorporated into its current version, *SBR*3.

In this paper we describe the functionalities and features of *SBR*3. We also present some theorems which *SBR*3 was able to prove. Some of these experimental results are beyond the capability of other existing general purpose equational theorem provers.

2 The Functionalities of SBR3

The entire *SbReve* family provers employ the overall methodology of *simplification-first*. Critical pairs are never generated as long as there is still room for simplification. Even if

*Research supported in part by NSF grants INT-8715231, CCR-8805734 and CCR-8901322. The third author is also supported by Dottorato di Ricerca in Informatica, Universitá degli Studi di Milano.

the superposition procedure is invoked, critical pairs are generated one at a time, and the simplification process is re-started as soon as a divergent critical pair is generated. This is the most significant design difference between the *SbReve* and *Reve* families.

The major difference between $SBR3$ and its predecessors is the incorporation of much more sophisticated search strategies. In the rest of this section, we describe $SBR3$'s features in two parts: its *inference* mechanisms, and its *search* mechanisms.

2.1 The Inference Mechanisms of SBR3

$SBR3$ takes as inputs an equational theory E and an equation $s = t$ and tries to prove that $s = t$ is a theorem of E. If no target equation is given, it will perform as Knuth-Bendix completion and try to generate a canonical system for E. It proves a theorem the refutational way. That is, it replaces all variables in $s = t$ by new Skolem constants and tries to find a contradiction to $E \cup \{\tilde{s} \neq \tilde{t}\})$ where $\tilde{s}$ and $\tilde{t}$ are the skolemization of s and t. Then the prover will try to reduce the inequality to an identity which yields the contradiction.

In addition to the theory and the equation, the user should also provide an ordering for ordering the terms. Usually the ordering should be a *complete simplification ordering* (a simplification ordering which is total on ground terms). In $SBR3$ the user has the choice of assigning a precedence among the operators in the theory and choose an ordering from a list implemented in the system. However, SBR3 will not check the totality for the user. The lack of totality on ground terms may actually be turned into a powerful search strategy, as we shall see later.

The backbone of $SBR3$ is the AC-UKB procedure – the AC version of the Unfailing Knuth-Bendix procedure ([AM-88]). Intuitively UKB allows an un-orientable equation to be used for reduction and superposition without getting into infinite loops. For its theory, see [HR-87] and [BDP-87]. AC-UKB incorporates AC-unification and completion ([PS-81]) into UKB and is described in detail in [AM-88].

Another feature is the inference rules for the *cancellation* axioms. An operator $*$ is *left cancellable* if $s * l = s * r$ implies $l = r$. If an operator is declared (left, right, or identity) cancellable, bigger equations can be replaced by smaller equations through complete sets of inference rules for cancellation [HRS-87]. Such inference rules are implemented in $SBR3$.

Other automatic inference mechanisms include *functional subsumption* and simplification, as well as user-defined cancellation laws for inequalities.

2.2 The Search Mechanisms of SBR3

Although the above inference mechanisms are sufficient for proving relatively simple theorems, the search space quickly grows to an unmanageable size for moderately difficult problems. The simplification-first search strategy coupled with cancellation controls the growth of the number and size of rules to some extent, but more clever means are needed.

The first problem we tackle is one of finding a shorter path to a solution. UKB, being complete, guarantees the existence of a proof through simplification and superposition should there be one. It does not, however, guarantee to provide a *short* proof. Suppose the prover can look at several different inequalities and tries to find a contradiction

simultaneously[1], then conceivably one can find a proof faster. On the other hand, one should also keep in mind not to inundate the search space with irrelevant inequalities.

$SBR3$ provides a facility for increasing a reasonable number of inequalities to check for shorter proofs as follows. When an un-orientable equation is generated, we superimpose it into an existing inequality (say A) to create a new inequality if possible. Then the new inequality is simplified using the rest of the equations and rules into B. The inequality B is kept, without deleting A, if $A \not\leq B$ according to the ordering. We term this method the *inequality ordered-saturation strategy*. This strategy is indispensable for proving some of the more difficult problems which we experimented ([AH-90]).

Another challenge is to eliminate redundant critical pairs. This problem is especially serious in AC-rewriting due to the potentially astronomical number of AC-unifiers. In the term rewriting literature there are a handful of critical pair criteria, whose purpose is to eliminate unnecessary critical pairs. However, all of them are designed not to destroy the confluence property of any given two terms. In refutational theorem proving, on the other hand, we are only interested in the confluence of the two terms of the targeted theorem. Therefore a critical pair can be deleted or suspended as long as it does not destroy the confluence of the intended terms.

Taking advantage of this property, we employed a notion of *measure* in $SBR3$. A measure is defined syntactically on the structure of terms such as the number of occurrences of a specific operator. The measure estimates the likelihood of whether a critical pair may contribute to an eventual proof of the intended theorem. Critical pairs are ordered according to the measure which decides the next equation to be chosen to perform superposition. Certain measures even allow us to delete critical pairs if they are deemed irrelevant for producing a proof. This search strategy is called *filtration-sorted strategy* and its detail can be found in [AA-90].

Three different measures are implemented in $SBR3$.

We remark that the filtration-sorted strategy may throw away critical pairs which are useful for ensuring the *global* confluence of the system. Therefore this search strategy, once invoked, no longer guarantees the confluence of the resulting system even if the prover terminates and returns an alleged "canonical" set of rules.

3 Examples

As is clear now, $SBR3$ is not oriented towards a special domain of applications. So its efficiency is surely not optimal for every problem. All the same we give below a few examples coming from various fields, executed on a SUN 3-50, at Orléans. More detail can be found in [AH-90] and [AB-90].

The Fifth Lukasiewicz Conjecture

Lukasiewicz's many-valued logic is defined using the following four axioms:

$true \Rightarrow x == x$

$(x \Rightarrow y) \Rightarrow ((y \Rightarrow z) \Rightarrow (x \Rightarrow z)) == true$

$(x \Rightarrow y) \Rightarrow y == (y \Rightarrow x) \Rightarrow x$

[1]The basic UKB only looks at one.

$(not(x) \Rightarrow not(y)) \Rightarrow (y \Rightarrow x) == true.$

The theorem $x \Rightarrow y \vee y \Rightarrow x == true$ is known as the *fifth Lukasiewicz conjecture* [FRT-84], [TL-56]. The conjecture was given by Lukasiewicz in the 20's, as reported in [TL-56], and proved many years later [RR-58], [MA-58].

The proof by $SBR3$ is done by first deriving a few lemmas from the axioms, one of which leads to the definition of an additional operator *or*. Then $SBR3$ proves that *or* is AC. Finally, the conjecture is proved in about 2 minutes. For the final session, the inputs are

$true \Rightarrow x == x$

$x \Rightarrow x == true$

$x \Rightarrow true == true$

$(x \Rightarrow y) \Rightarrow ((y \Rightarrow z) \Rightarrow (x \Rightarrow z)) == true$

$not(not(x)) == x$

$(x \Rightarrow y) \Rightarrow y == (y \Rightarrow x) \Rightarrow x$

$or(not(x), y) == x \Rightarrow y$

$x \vee y == (x \Rightarrow y) \Rightarrow y$

Declared AC-operator: *or*.

Theorem proved: $x \Rightarrow y \vee y \Rightarrow x == true$, (24 min).

A detailed description of the experiments in Lukasiewicz logic can be found in [AB-90].

Moufang identities in alternative rings.

Alternative rings are rings with the associativity of $*$ replaced by two *alternative* axioms. The Moufang identities are a set of equational theorems of alternative rings. The Moufang identities as a challenge to theorem provers was first suggested in [Ste-87], although no automated proof was given. They were later proved automatically using a special-purpose theorem prover designed for ring theory ([Wa-87]). $SBR3$ is the first syntactic theorem prover which proved them automatically.

Alternative rings are defined by

$0 + x == x$

$0 * x == 0$

$x * 0 == 0$

$g(x) + x == 0$

$g(x + y) == g(x) + g(y)$

$g(g(x)) == x$

$x * (y + z) == (x * y) + (x * z)$

$(x + y) * z == (x * z) + (y * z)$

$(x * y) * y == x * (y * y)$

$(x * x) * y == x * (x * y)$

$g(x) * y == g(x * y)$

$$x * g(y) == g(x * y)$$
$$g(0) == 0$$
$$a(x, y, z) == ((x * y) * z) + g(x * (y * z)$$

where a is an auxiliary operator.

$SBR3$ proved the following properties (the middle alternative law and two skew-symmetries of a) within 20 seconds:

$$(x * y) * x == x * (y * x)$$
$$a(y, x, z) == g(a(x, y, z))$$
$$a(z, y, x) == g(a(x, y, z))$$

The Moufang identities are defined as:

$$(((x * y) * x) * z) = (x * (y * (x * z))) \text{ (left Moufang)}$$
$$(((z * x) * y) * x) = (z * (x * (y * x))) \text{ (right Moufang)}$$
$$((x * y) * (z * x)) = ((x * (y * z)) * x) \text{ (middle Moufang)}$$

and they are proved in 49, 55, and 41 minutes respectively.

By adding the left and right Moufang into the input set, we are able to give a direct proof of

$$a(x * x, y, z) == ((a(x, y, z) * x) + (x * a(x, y, z)))$$

in 13 minutes. A full account of our experiments in alternative rings is given in [AH-90], although the time reported there was much more than what we are reporting here. The time was significantly improved because of the better search strategies incorporated in $SBR3$ later.

References

[AA-90] S.Anantharaman, N. Andrianarivelo, Heuristic Criteria in Refutational Theorem Proving, *Proceedings of DISCO90, 1990*

[AB-90] S.Anantharaman, M.P.Bonacina, Automated Proofs in the Logic of Lukasiewicz, *(In Preparation)*

[AH-90] S.Anantharaman, J.Hsiang, Automated Proofs of the Moufang Identities in Alternative Rings, *To appear in the J. Automated Reasoning, 1990*

[AHM-89] S.Anantharaman, J.Hsiang, J.Mzali, SbReve2: A term rewriting laboratory with (AC-)Unfailing Completion, *RTA, Springer-Verlag LNCS Vol. 355, pp533–537, 1989*

[AM-88] S.Anantharaman, J.Mzali, Unfailing Completion Modulo a set of Equations, *Research Report, no. 470, LRI-Orsay (Fr.), 1989*

[BDP-87] L.Bachmair, N.Dershowitz, D.Plaisted, Completion without failure, *Proc. Coll. on Resolution of Equations in Algebraic Structures, Lakeway, Texas, 1987*

[FRT-84] J.M.Font, A.J.Rodriguez, A.Torrens, Wajsberg algebras, *Stochastica, Vol. 8, No. 1, pp5-31, 1984*

[FG-84] R.Forgaard, J.Guttag, REVE: A term rewriting system generator with failure-resistant Knuth-Bendix, *MIT-LCS technical report, 1984*

[HM-88] J.Hsiang, J.Mzali, SbReve users guide, *Technical report, LRI, 1988*

[HR-87] J. Hsiang, M. Rusinowitch, On word problems in equational theories, *14th ICALP, Springer-Verlag LNCS Vol. 267, pp54–71, 1987*

[HRS-87] J. Hsiang, M. Rusinowitch, K. Sakai, Complete set of inference rules for the cancellation laws, *IJCAI 87, Milan, Italy, 1987*

[KK-85] C. Kirchner, H. Kirchner, Implementation of a general completion procedure parameterized by built-in theories and strategies, *EUROCAL '85, 1985*

[Le-83] P. Lescanne, Computer Experiments with the REVE term rewriting system generator, *10th POPL, pp99-108, 1983*

[MC-58] C.A.Meredith, C.C.Chang, , *Trans. Amer. Math. Soc. 87, 13, pp54,55,56, 1958*

[Mz-86] J.Mzali, Methodes de filtrage equationnel et de preuve automatique de theoremes, *Thesis, Université de Nancy, 1986*

[PS-81] G.Peterson, M.E.Stickel, Complete sets of reductions for some equational theories, *JACM Vol. 28, pp 233-264, 1981*

[RR-58] A.Rose, C.C.Rosser, , *Trans. Amer. Math. Soc. 87, 13, pp1–53, 1958*

[St-84] M.E.Stickel, A case study of theorem proving by the Knuth-Bendix method: Discovering that $x^3 = x$ implies ring commutativity, *7th CADE, Springer-Verlag, LNCS Vol 170, pp248-258, 1984*

[Ste-87] R. L. Stevens, Some Experiments in Nonassociative Ring Theory with an Automated Theorem Prover, *J. Automated Reasoning, Vol 3 no. 2, 1987*

[TL-56] A.Tarski, J.Lukasiewicz, Investigations into the sentential calculus, *Chapter 4th in A.Tarski, Logic, Semantics, Meta-mathematics, pp38-56, Clarendon Press, Oxford, 1956*

[Wa-87] T. C. Wang, Case Studies of Z-module Reasoning: Proving Benchmark Theorems from Ring Theory, *J. Automated Reasoning, Vol 3 no.4, 1987*