The New Waldmeister Loop at Work

Jean-Marie Gaillourdet[1], Thomas Hillenbrand[2],
Bernd Löchner[1], and Hendrik Spies[1]

[1] FB Informatik, Universität Kaiserslautern, Kaiserslautern, Germany,
{jmg,loechner,spies}@informatik.uni-kl.de
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany,
hillen@mpi-sb.mpg.de

1 Introduction

We present recent developments within the equational theorem prover Waldmeister, an implementation of unfailing Knuth-Bendix completion [BDP89] with refinements towards ordered completion. The new developments rely on a novel organization of the underlying saturation-based proof procedure into a system architecture. As is known, the saturation process tends to quickly fill the memory available unless preventive measures are employed. To overcome this problem, our new "Waldmeister loop" features a highly compact representation of the search state, exploiting its inherent structure. The implementation just being available, the cost and the benefits of the concept now can exactly be measured. Indeed are our expectations met concerning the drastic cut-down of memory usage with only moderate overhead of time.

In addition it has turned out that the revealed structure of the search state paves the way to an easily implemented parallelization of the prover with modest communication requirements and rewarding speed-ups. In order to minimize communication-related latencies, we discuss some variations of the loop to maximally profit from multiple processors. All in all, these innovations allow to solve proof tasks that were previously out of reach. For a summary of the basic features of the system see [LH02]. Here we focus on new achievements that mark a milestone in the development of the system.

2 The Waldmeister loop

In recent years the architecture of automated theorem provers, also known as design of the "main loop", has gained renewed interest [RV00]. We have developed the Waldmeister loop [HL02], an instance of the *given-clause algorithm* [McC97], to pinpoint ideas that were only rudimentary present in earlier work.

The saturation is performed in a cycle working on a set $\mathcal{A}$ of *active facts* that participate in inferences and a set $\mathcal{P}$ of *passive facts* waiting to become members of $\mathcal{A}$. Initially, $\mathcal{A}$ is empty and $\mathcal{P}$ contains the input axioms. In each cycle the prover selects an equation e from $\mathcal{P}$ which has minimal weight with respect to some weighting function φ. Then, it simplifies e with $\mathcal{A}$ and discards it if it is redundant. Otherwise the prover *activates* e, i.e. inserts it into $\mathcal{A}$ and computes the set C of all critical pairs between e and $\mathcal{A}$. After simplification with $\mathcal{A}$ the nonredundant elements of C are finally inserted into $\mathcal{P}$. This cycle introduces an abstract notion of time which advances with every activation. We denote by e_i the ith activated equation and by C_i the set of critical pairs belonging to e_i. The prover stops when it can simplify a conjecture c into a tautology, which indicates that c is a logical consequence of the axioms, or otherwise when $\mathcal{P}$ is empty, in which case c is not entailed. If φ induces a *fair* heuristic, this proof procedure will show any valid conjecture in finite time [BDP89].

The major insight in the formulation of the "WALDMEISTER loop" is that the inherent structure of $\mathcal{P}$ itself can be used as starting point to dramatically cut down memory requirements. With the old loop more than 95 % of the memory is occupied with equations that stay untouched unless they get selected. Although single equations were compressed the memory usage of $\mathcal{P}$ formed the limiting factor of the whole system. The key is to condense *whole sets* of critical pairs, namely the C_i-sets, into one *entry* of constant size, essentially the number i. This changes the asymptotic space behavior of $\mathcal{P}$ over abstract time from quadratic to linear [HL02, p.493f]. To avoid excessive recomputation we employ a buffer to cache a fixed number of "promising" critical pairs individually. Their representation is also of constant size: three numbers to encode the parents and the position are sufficient. For each C_i-set the weighting function φ is used to separate the subset of individually stored or already selected equations from the rest collectively represented in the set entry. Details can be found in [HL02].

The experiments[3] presented in Table 1 show that the memory usage of WALDMEISTER is significantly reduced for nontrivial proof runs. They were performed with a cache buffer size of 40 000. The additional work necessary to reconstruct the critical pairs that are selected from $\mathcal{P}$ has only a moderate effect on the running times. In fact small speed-ups appear for some examples. We attribute this to the memory architecture of modern machines. With the new loop the processor caches are better used. The results confirm the simulations in [HL02]. Speaking in absolute terms may illustrate the difference: When attacking ROB001-1, with the old loop 1 GByte is filled within 15 minutes, whereas with the new one, it takes more than two weeks.

With the new architecture it is now possible to solve problems that were previously out of reach. So WALDMEISTER is now able to prove with simple standard heuristics that a ring with $x^5 = x$ is commutative (RNG036-7), or that the Robbins Axiom together with the second Winker Lemma entails Boolean (ROB007-1). For these problems the size of $\mathcal{A}$ exceeds 70 000, and more than 500 million equations are inserted into $\mathcal{P}$. For such a large search state 200 MByte suffice, most of which are occupied by $\mathcal{A}$. To our knowledge the only other system that can solve ROB007-1 is McCune's prover EQP [McC97]. The EQP proof

[3] All experiments were performed on machines with 1 GHz Pentium III processors and 4 GByte RAM connected with 100 MBit Ethernet. Most proof tasks are taken from TPTP v. 2.5.0 [SS99]. They are selected as representatives of different domains to cover a wide range. The exception X6 states that a ring with $x^6 = x$ is commutative.

Table 1. Comparison of resource utilization (examples are taken from [HL02]).

	time [s]		size [MByte]			time [s]		size [MByte]	
	old	new	old	new		old	new	old	new
GRP164-1	67	68	71	13	RNG027-5	516	676	313	82
LAT038-1	69	74	100	19	ROB006-1	624	592	1012	21
X6	141	137	213	13	ROB001-1*	2940	3065	1115	23

* The proof attempt of ROB001-1 was aborted when $|\mathcal{A}| = 7\,000$.

has been found with built-in AC-axioms and an incomplete strategy. Clauses exceeding a weight limit are discarded. It has been noted that too simplistic discarding heuristics are harmful to the success of the proof search. So some tuning knowledge concerning weight or clause limits is required, and it may be specific for problem domains or even individual problems. For the Waldmeister loop, however, such knowledge is not required.

3 Waldmeister in parallel

To use multiple processors for automated theorem proving very different approaches have been investigated (cf. [Bon00] for a recent overview). It turns out that communication can be the major bottleneck: communication latencies may introduce delays in the computation, whereas communication bandwidth limits the amount of data transmitted. Variations in machine loads and in communication delays can further influence the proof search. Because the saturation process is unstable to small disturbances our main design requirement is that the prover shall remain deterministic in order to ensure that proof runs are reproducible. This is in our experience of utmost importance for user acceptance.

From literature two main directions are of interest for us. The first one tries to speed up a sequential prover by parallelizing the most time-consuming subtasks (e.g. [BGKW98]). Because of the typically high communication demands this works best on shared memory machines. The second one is an instance of multi-agent search. Several instances of theorem provers cooperate by exchanging information (e.g. [DF99]). Some of these approaches are specifically designed to run on networked workstations or PCs.

We focus here on parallelizing Waldmeister because some kind of multi-agent search can be built on top later on. As the design focus of the system was on high-performance and not on easy parallelization this is somewhat challenging. But it turns out that the Waldmeister loop offers a new starting point.

Waldmeister spends most of its time in the computation, normalization, and heuristic assessment of critical pairs before they are inserted into $\mathcal{P}$. Depending on the example, this amounts from about 80 % to over 90 % of the total running time, the longer the task runs the more. The compact representation of critical pairs makes a good basis for parallelizing this part of the prover, for it offers a compact interchange format. So the communication bandwidth is not a limiting factor anymore. To achieve a deterministic behavior, we use an *asymmetric* form of parallelization: one process, the *master*, performs the saturation, whereas several *slaves* compute, normalize, and assess the individual critical pairs which constitute the smallest unit of parallelization. Thus, the parallelization happens above the inference level and therefore basic routines such as unification or simplification are unaffected. The master informs the slaves about the changes of $\mathcal{A}$, whereas the slaves send back small, constant-size entries describing the nontrivial critical pairs and their weights.

In a first attempt we want to change the overall loop only minimally. When the master has activated an equation e_i, the n slaves process C_i. The work can be partitioned evenly among the slaves with some simple modulo scheme which gives a good load balance. The main problem with this approach is that it is very sensitive to communication delays. The master has to wait for all messages of the slaves before it can proceed, as it has to integrate all critical pairs belonging to equation e_i before it can select equation e_{i+1}. The whole saturation process is therefore stopped when one message is delayed. This limits the achievable speed-ups considerably in practice.

The key idea for hiding such delays is to *postpone* the integration of critical pairs by some fixed amount p, $p > 1$. That means that the critical pairs of equation i are integrated into $\mathcal{P}$ just before equation e_{i+p} is selected. This changes of course the search behavior of the prover, but it changes it *deterministically* for a fixed value of p. This scheme has the further advantage that it can compensate variations in processing times of subsequent C_i-sets to some degree and even smooth out performance differences of the slaves. It is thus possible to simplify the approach, each C_i is handled only by one slave, the number of messages per set is cut down from n to two.

Experiments show that for long-lasting runs a moderate value of p does not change the search process significantly. For example for the proof task ROB020-1 the size of $\mathcal{A}$ increases from 36 976 to 37 135 when $p = 30$. In effect, as can be seen from Table 2, the parallel Waldmeister gains nice, nearly linear speed-ups. Note that this is for a high-performance, originally sequential prover and runs on standard desktop PCs that are interconnected by standard Ethernet.

Exemplified by our system, the Waldmeister loop architecture shows a way to significantly push the limits saturation-based provers have been afflicted with. Less constricted by memory and accelerated by the parallelization the prover is ready for new, more demanding applications.

4 Ongoing work and future prospects

Complementing the reported work on increasing the raw inference power of the prover, we are continuing our research line on integrating more intelligence into the proof search, and develop practically rewarding redundancy elimination techniques based on constraint techniques. We hope that the analysis of the newly found proofs will give insights for the construction of refined strategies.

Table 2. Experiments with parallel WALDMEISTER

	ROB020-1				ROB007-1			
number of processors	1	3	5	7	1	3	5	7
cpu time of master [h]	8.4	0.9	0.9	0.9	43.6	4.4	4.5	4.4
average cpu time of slaves [h]	—	3.6	1.8	1.2	—	18.7	9.3	6.1
wall clock time (wct) [h]	8.4	3.7	1.9	1.4	43.8	21.0	10.1	7.5
speed-up (w.r.t. wct)	1.0	2.3	4.4	5.9	1.0	2.1	4.3	5.8

Other work is more on the implementation side: As a spin-off of the COMPIT initiative, the indexing implementation is right now being revised to improve its time and space efficiency. It seems promising to enhance the parallelization approach and to dynamically adapt the number of slaves depending on the computational needs.

The WALDMEISTER system, originally developed for SOLARIS and LINUX, has just recently been ported to run under the new MACOS X operating system, and is also available for WINDOWS users via the CYGWIN platform.

Please feel invited to visit the thoroughly rewritten WALDMEISTER Web pages at `http://www-avenhaus.informatik.uni-kl.de/waldmeister` .

References

[BDP89] L. Bachmair, N. Dershowitz, and D. A. Plaisted. Completion without failure. In *Resolution of Equations in Algebraic Structures*, volume 2, pp. 1–30. Academic Press, 1989.

[BGKW98] R. Bündgen, M. Göbel, W. Küchlin, and A. Weber. Parallel term rewriting with PaReDuX. In W. Bibel and P. H. Schmitt, eds., *Automated Deduction—A Basis for Applications, Volume II: Systems and Implementation Techniques*, pp. 231–260. Kluwer Academic Publishers, 1998.

[Bon00] M. P. Bonacina. A taxonomy of parallel strategies for deduction. *Annals of Mathematics and Artificial Intelligence*, 29(1–4):223–257, 2000.

[DF99] J. Denzinger and D. Fuchs. Cooperation of heterogeneous provers. In T. Dean, ed., *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 10–15. Morgan Kaufmann, 1999.

[HL02] Th. Hillenbrand and B. Löchner. The next WALDMEISTER loop. In A. Voronkov, ed., *Proceedings of the 18th International Conference on Automated Deduction*, LNAI, pp. 486–500. Springer-Verlag, 2002.

[LH02] B. Löchner and Th. Hillenbrand. A phytography of WALDMEISTER. *AI Communications*, 15(2,3):127–133, 2002.

[McC97] W. McCune. 33 basic test problems: a practical evaluation of some paramodulation strategies. In R. Veroff, ed., *Automated Reasoning and its Applications: Essays in Honor of Larry Wos*, pp. 71–114. MIT Press, 1997.

[RV00] A. Riazanov and A. Voronkov. Limited resource strategy in resolution theorem proving. Tech. Report CSPP-7, University of Manchester, 2000.

[SS99] C. B. Suttner and G. Sutcliffe. The TPTP problem library (TPTP v2.2.0). Tech. Report 99/02, James Cook University, Townsville, 1999.