Lemmas with Matching over PTTP: Efficiency and Limitation
(A Research Memo)

K. Iwanuma, Y. Chino, H. Ashizawa

Department of Electrical Engineering and Computer Science
Yamanashi University
Takeda 4-3-11, Kofu-shi, Yamanashi, 400, Japan.
e-mail: iwanuma@esi.yamanashi.ac.jp

In this paper, we study an efficiency and a limitation of *lemmas with matching* through experiments with a simple first-order compiler implemented over PTTP (Prolog Technology Theorem Prover) technology proposed by Stickel [6, 7].

PTTP is a compilation technology which enables an extremely efficient implementation of Loveland's ME (Model Elimination) procedure [5]. It made top-down theorem provers be pretty attractive, and so far, is followed by several theorem provers (e.g. METEOR [1], SETHEO [4]). However top-down theorem provers unfortunately have a well-known disadvantage [6] that same goals may quite often be proved more than once during the search, which must cause a large amount of redundancy.

In order to avoid such a recomputation, it has been pointed out that support of *lemmas* within top-down theorem provers is very important [2, 6, 8]. Lemmas are extra clauses derivable in the proof search. Using lemmas as additional axioms makes it possible for a top-down theorem prover both to find a shorter proof and to avoid duplicate computations. However a simple straightforward use of lemmas increases the breadth of the search space, and must result in the loss of efficiency of provers in almost cases.

Caching proposed by Astrachan and Stickel [2] is a modified mechanism of lemmas, which replaces the search of solutions by cache-table lookup. It is applicable to *Horn-problems*, and succeeds in drastically reducing the search space. Schumann [8] studied a bottom-up preprocessor DELTA for combining lemmas with top-down theorem provers. DELTA generates a moderate number of easily derivable lemmas, which are later used in SETHEO as additional unit axioms. The growth of search space caused by lemmas is overcome both by a filtering of lemmas e.g., a subsumption test, in the preprocessing phase, and by some pruning operations efficiently implemented in SETHEO.

In this short paper, we investigate a simple idea, called *lemmas with matching*. It performs a restricted use of unit lemmas, dynamically generated during ME-derivations, such that a unit lemma is adopted as a input clause at each stage *only when* it *subsumes* a target goal to be solved. In such a case, the goal can be solved in one step without any instantiation for all variables appearing in the goal. It means that all other alternatives can be cut immediately if a goal is solved by such a restricted use of lemmas. The search space is never broadened as long as lemmas are used with this subsumption criterion. It implies no need of extra pruning operations which must usually be necessary in order to delete the redundancy introduced by lemmas.

The biggest concern to be confirmed is about the rate of satisfying such a rather strict criterion for lemma adoption. If this rate is low, the overhead of a newly introduced subsumption check between goals and lemmas is never compensated. We investigate this rate and the efficiency of lemmas with matching through a experiment, using slightly difficult problems picked up from TPTP problem library [9].

The kernel Prolog codes, for ME-derivations without lemmas, used in this experiment are almost same as the ones produced by a version of PTTP implemented in Prolog described in [7], except for no use of zero-cost subgoal mechanism. An additional pruning method in ours, not adopted in the PTTP, is only a cutting off alternatives of a goal, which is involved if the goal is solved by a unit clause without instantiation. [1]

The running Prolog codes must dynamically construct and maintain a table consisting of unit lemmas during the search of ME-refutation. We here consider two filtering operations, which generated lemmas must pass for its entry in the lemma-table: One is a maximal term-complexity restriction. In this experiment, we uniformly restrict the depth of each term in generated lemmas to be less than or equal to 3, because the maximal depth of terms occuring in axiom formulas attacked here is less than or equal to 2. The other filtering is a forward subsumption test over the lemma-table, while a backward subsumption test is not performed here.

The experiment was done on SUN SPARC 2 with 64M byte memory. We quitted trials at 5,000 cpu sec. The results are shown in the table.

TPTP problems	time (sec)	ME inferences (unit res. no inst.)	matched lemmas	stored lemmas	generated lemmas (term depth. ≤ 3)
apabhp	348.5	325,481 (810)	0	2	212 (4)
ex4	394.3	97,077 (0)	139	5	29 (26)
ex5	55.7	52,160 (109)	538	7	47 (8)
nonob	7.0	8,554 (5)	146	17	141 (141)
wos4	1,402.0	785,064 (4,651)	6,351	834	141,934 (113,162)
wos15 (H)	943.4	346,341 (1,601)	5,171	1,406	142,587 (127,773)
wos21 (H)	4,562.1	1,415,662 (19,736)	21,913	1,334	737,745 (616,702)
wos22 (H)	2,136.1	784,231 (5,954)	2,363	1,791	266,313 (248,691)
wos31	NON [33,442.5]	[26,318,754 (128,477)]	[14,579]	[5]	[45,079 (45,079)]
wos33	NON [39,631.5]	[48,249,367 (1,526,526)]	[1,561,448]	[19]	[1,830,499 (1,830,499)]
ls36 (H)	1435.0	653,283 (3,490)	1,773	789	239,424 (208,736)
ls37 (H)	300.5	120,565 (3,103)	1,853	465	57,036 (45,628)
ls112	NON [8,261.6]	[6,561,957 (0)]	[0]	[38]	[24,719 (24,719)]
ls118	NON [7,978.7]	[6,011,852 (0)]	[0]	[14]	[11,525 (11,525)]
ls121	NON [7,183.0]	[5,437,684 (0)]	[0]	[6]	[10,560 (10,560)]
EXQ2	NON [36,283.0]	[31,618,788 (131,569)]	[28,829]	[1]	[64 (64)]

The simple use of lemmas through matching can, by itself, solve some difficult problems, although the required absolute cpu time is not so small, in comparison with other

[1]The reason is that this pruning operation is quite common in literature, for instance, which is also used in a earlier version of PTTP in [6]. It can be performed with quite little over-head if lemmas with matching has already been installed in advance.

advanced theorem provers. This indicates that the rate of a lemmas subsuming a goal is not so low for a class of problems. However, this rate falls down extremely for problems involving many variables but containing few constants or function symbols, such as Ls112, Ls118, Ls121. This shows the system has a sufficient value as an assistant in a system fully supporting lemmas with unification. It seems to never lose its own value even in such a complete lemmaizing system.

In order to increase the hit rate for those problems, an usual use of lemmas with unification is, of course, valuable, but also a mechanism for lazy matching with lemma and A-literals in ME-derivations seems to be worth while to study. An important point is that such a use of lemmas also never enlarges the search space. A trial, with a lazy lemma matching only with direct ancestor A-literals, has shown a good performance for wos33, but unfortunately, the caused over-head can not be ignored for the other problems. We are to investigate in detail this method in near future.

Also, a huge amount of latent redundancy in the search is pointed out by the large number of lemmas dropped out by the forward subsumption test. This number is nearly equal to the number of worthless duplicate computation. This redundancy can be decreased not only by a pruning operation, such as anti-lemma recently developed by SETHEO group [4], but also be reduced by lemmas with unification. Once lemmas with unification is allowed, a new stronger pruning method can be weaved into a system. This is also a future work.

References

[1] O.L. Astrachan and D.W. Loveland, METEORs: high performance theorem provers using model elimination, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, (Kluwer Academic Publishers, Netherlands, 1991).

[2] O.L. Astrachan and M.E. Stickel, Caching and lemmaizing in model elimination theorem provers, *Proceedings 11th Inter. Conf on Automated Deduction (CADE-11), LNAI* **607**, USA (1992) 778-782.

[3] S. Fleisig, D. Loveland, A.K. Smiley III and D.L. Yarmush, An implementation of the model elimination proof procedure. *J. ACM* **21** (1) (1974) 124-139.

[4] Chr. Goller, R. Letz, K. Mayr and J. Schumann, SETHEO V3.2: recent developments – system abstract–, *Proceedings 12th Inter. Conf on Automated Deduction (CADE-12), LNAI* **814**, Nancy, France (1994) 778-782.

[5] D.W. Loveland, A simplified format for the model elimination theorem-proving procedure, *J. ACM* **16** (3) (1969) 349-363.

[6] M.E. Stickel, A prolog technology theorem prover: implementation by an extended prolog compiler, *J. Automated Reasoning* **4** (1988) 353-380.

[7] M.E. Stickel, A prolog technology theorem prover: a new exposition and implementation in prolog, *Theoret. Comput. Sci.* **104** (1992) 109-128.

[8] J.M.Ph. Schumann, A bottom-up preprocessor for top-down theorem provers –system abstract–, *Proceedings 12th Inter. Conf on Automated Deduction (CADE-12), LNAI* **814**, Nancy, France (1994) 774-777.

[9] G. Sutcliffe, C. Suttner and T. Yemenis, The TPTP problem library, *Proceedings 12th Inter. Conf on Automated Deduction (CADE-12), LNAI* **814**, Nancy, France (1994) 252-266.