Testing the Equivalence of Models given through Linear Atomic Representations (extended abstract)

Reinhard Pichler *
Technische Universität Wien

September 29, 1997

1 Introduction and Basic Definitions

Models may be of good use for automated theorem provers in 2 different ways: Firstly to give a hint as to why an input formula is not a theorem (e.g. by constructing a counter-model) and, secondly, to speed up the deduction process itself (e.g.: by guiding the proof search through semantic resolution refinement). In any case, an appropriate representation for models is called for. In [FL 96], *linear atomic representations* (= LAR's) of Herbrand models are used. It is shown how such models can be constructed automatically for satisfiable clause sets of a certain syntax class. Furthermore an essential property of LAR's is proven, which is absolutely necessary for any practically relevant model representation, namely: the existence of algorithms both for deciding the equivalence of models and for computing the truth value of an arbitrary clause in such a model.

What we are interested here is the efficiency of decision procedures for the equivalence of LAR's. Actually, both decision problems (i.e.: the equivalence of 2 LAR's and the evaluation of a clause to true) have been shown to be coNP-complete (cf. [Got 97]). Therefore, we cannot expect to find a polynomial algorithm without giving a positive answer to the $P = NP$-Problem. However, what we can do is find out the real "source" of complexity and make use of this theoretical result for devising an algorithm which is, in general, considerably more efficient than previously known algorithms, e.g.: the orthogonalization method in [FL 96] and the transformation into equational problems in [CP 95] followed by the solution method for equational problems described in [CL 89].

This paper is structured as follows: First we transform the original model equivalence problem into another type of problem which we shall call the *linear term tuple cover problem*, i.e.: Given a set $M = \{(t_{11}, \ldots, t_{1k}), \ldots, (t_{n1}, \ldots, t_{nk})\}$ of k-tuples of linear terms over some Herbrand universe H. Is every ground term tuple $(s_1, \ldots, s_k) \in H^k$ an instance of some tuple $(t_{i1}, \ldots, t_{ik}) \in M$?

In chapter 3 we shall give an algorithm for the solution of the linear term tuple problem along with a rough complexity estimation: Analogously to the methods of [FL 96] and [CP 95], the algorithm is in principle exponential in the total size of the input problem.

*Technische Universität Wien, Institut für Computersprachen, Abteilung für Anwendungen der Formalen Logik, Resselgasse 3/1/3, A-1040 Wien, AUSTRIA, reini@logic.tuwien.ac.at

The main results of this paper are presented in chapter 4: In theorem 4.1 a strong (and, yet, easy to check) criterion for the redundancy of term tuples is proven. By integrating this redundancy criterion into the algorithm of chapter 3 we arrive at an algorithm which is exponential in the number of term tuples (or, equivalently, in the number of atoms of the original model equivalence problem). In contrast to the algorithms in [FL 96] and [CP 95], the complexity of the expressions involved (e.g.: the arity of the predicate symbols and, in particular, the term depth of the arguments) only has polynomial influence on the overall complexity of the algorithm. Finally, these results are illustrated by a simple example.

Due to space limitations, the basic concepts and notations from [FL 96] can only be revised very briefly here without formal definitions: An *atomic representation* (= AR) of a (Herbrand) model H with signature Σ is a set $\mathcal{A} = \{A_1, \ldots, A_n\}$ of atoms over Σ with the following intended meaning: a ground atom over Σ evaluates to true, iff it is an instance of some atom $A_i \in \mathcal{A}$. In a *linear atomic representation* (= LAR), all atoms are *linear*, i.e.: they have no multiple variable occurrences. Two ARs $\mathcal{A}$ and $\mathcal{B}$ are equivalent, iff they represent the same (Herbrand) model, i.e.: the same ground atoms evaluate to true in both models. We say that a set $\mathcal{C} = \{C_1, \ldots, C_n\}$ of clauses over Σ *H-subsumes* a clause D, i.e.: $\{C_1, \ldots, C_n\} \leq_{ss}^{H} D$, iff all ground instances of D w.r.t. H are subsumed by some clause $C_i \in \mathcal{C}$. For a term t over H, we denote the set of ground instances of t by $G_H(t)$.

The generalization of these concepts to term tuples is obvious and will not be formally defined here either, e.g.: By $G_H(t_1, \ldots, t_k)$ we denote the set of *ground H-instances generated by the term tuple* $(t_1, \ldots, t_k)$. Moreover, proofs will only be sketched rather than carried out in detail.

2 Transformation of the original Problem

In [FL 96], the following criterion for the equivalence of LAR's is stated:

Lemma 2.1 *Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $\mathcal{B} = \{B_1, \ldots, B_m\}$ be LAR's w.r.t. some Herbrand universe H. Then $\mathcal{A}$ and $\mathcal{B}$ are equivalent, iff*

1. $\forall j \in \{1, \ldots, m\}$: $\{A_1, \ldots, A_n\} \leq_{ss}^{H} B_j$
2. $\forall i \in \{1, \ldots, n\}$: $\{B_1, \ldots, B_m\} \leq_{ss}^{H} A_i$

This characterization of model equivalence provides the starting point for our considerations. The following theorem shows how the H-subsumption criterion can be further transformed:

Theorem 2.2 (transformation of the H-subsumption problem) *Let $B, A_1, \ldots, A_n$ be linear atoms over some Herbrand universe H. Furthermore, let $V(B) := \{x_1, \ldots, x_k\}$ denote the variables occurring in B and let Θ be defined as the set of unifiers of pairs (A_i, B), i.e.:*

$\Theta := \{\ \theta$; $(\exists i)$ *s.t. A_i and B are unifiable with* $\theta' := mgu(A_i, B)$ *and* $\theta = \theta'|_{V(B)}\}$.

Then the following equivalence holds:

$$\{A_1, \ldots, A_n\} \leq_{ss}^{H} B \text{ iff } \bigcup_{\theta \in \Theta} G_H(x_1\theta, \ldots, x_k\theta) = H^k.$$

Proof sketch: *B is H-subsumed, iff all ground instances are subsumed by some A_i. Obviously, only those instances of the A_i's play a role, which are unifiable with B, i.e.: For every ground substitution σ, $B\sigma$ must be subsumed by some $A_i\theta'_i = B\theta'_i = B\theta_i$, where $\theta'_i = mgu(A_i, B)$ and $\theta_i = \theta'_i|_{V(B)}$. But this is the case, iff for every ground substitution σ (based on H), $(x_1\sigma, \ldots, x_k\sigma)$ is a ground instance of some $(x_1\theta_i, \ldots, x_k\theta_i)$.* $\diamond$

As far as complexity is concerned, the original model equivalence problem and the resulting collection of term tuple cover problems are basically the same: Both problems are coNP-complete; the number of (term tuple cover-) subproblems and the number of term tuples within each subproblem are restricted by the number of atoms; the total length of each term tuple cover problem is restricted by the length of the original model equivalence problem; etc. Furthermore, the above transformation preserves linearity, i.e.: no term tuple in the resulting cover problems has multiple variable occurrences.

3 Solution of the Term Tuple Cover Problem

Theorem 3.1 *The procedure TERM_TUPLE_COVER given below decides the linear term tuple cover problem, i.e.: it terminates on every set of linear term tuples and it returns the value "true", iff the input set of term tuples covers all of H^k.*

function TERM_TUPLE_COVER (k, n, M): **boolean**;
/* k = dimension of the tuples */
/* n = number of tuples in M */
/* $M = \{(t_{11}, \ldots, t_{1k}), \ldots, (t_{n1}, \ldots, t_{nk})\}$ */
begin
if $n = 0$ **then return false**;
if $k = 0$ **then return true**;
if for all $i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\}$ t_{ij} is a variable
then return true;
$j := \min(\{\beta; (\exists i) t_{i\beta}\}$ is a non-variable term$\})$;
for all $f^\alpha \in FS(H)$ **begin**
/* f^α denotes a function symbol with arity α. */
/* Constant symbols are considered as function symbols with arity 0. */
$M' := \{(s_1^{(ij)}, \ldots, s_\alpha^{(ij)}, t_{i(j+1)}, \ldots, t_{ik}); t_{ij} = f(s_1^{(ij)}, \ldots, s_\alpha^{(ij)})\}$;
/* i.e.: collect all tuples, where t_{ij} has leading symbol f^α */
$M' := M' \cup \{(x_1^{(ij)}, \ldots, x_\alpha^{(ij)}, t_{i(j+1)}, \ldots, t_{ik}); t_{ij}$ is a variable $\}$;
/* i.e.: the $x_l^{(ij)}$ are fresh variables */
$k' := k - j + \alpha$;
$n' := |M'|$;
if TERM_TUPLE_COVER (k', n', M') = **false then return false**;
end { for }
return true;
end { TERM_TUPLE_COVER }.

Correctness, termination and complexity (sketch): *The correctness of the above algorithm is based on an idea, which is also central to the orthogonalization in [FL 96] and to the explosion rule for solving equational problems in [CL 89], namely:*

$$H \quad = \quad \bigcup_{f \in FS(H)} G_H\Big(f(x_1, \ldots, x_{\alpha(f)})\Big) \text{ and, therefore, also}$$

$$H^k \quad = \quad \bigcup_{f \in FS(H)} G_H\Big(f(x_1, \ldots, x_{\alpha(f)})\Big) \times H^{k-1}$$

The branching into recursive calls corresponds to this case distinction: In each recursive case, the term tuples with leading symbol different from f are deleted and the term tuples with a variable in the j-th position are restricted to those instances, where an f-term is substituted for this variable. Finally the leading symbol f may be omitted, since all term tuples share this leading symbol anyway, i.e.: the recursive calls are done with the term tuple sets $M' = \bigcup\Big\{\Big(s_1^{(i)}, \ldots, s_{\alpha(f)}^{(i)}, t_{i2}, \ldots t_{ik}\Big)\Big\}$ rather than with $M' = \bigcup\Big\{\Big(f(s_1^{(i)}, \ldots, s_\alpha^{(i)}), t_{i2}, \ldots t_{ik}\Big)\Big\}$. Furthermore, note that the deletion of leading variable positions has no influence on the correctness.

The termination is guaranteed by the fact, that the number of non-variable positions (i.e.: the positions, where in some tuple a non-variable term occurs) strictly decreases whenever TERM_TUPLE_COVER is called recursively.

As a by-product of the termination proof, we get the following estimation of the time complexity: Procedure TERM_TUPLE_COVER is exponential in the number of non-variable positions which, in general, corresponds to the total length of the input problem. In particular, if the term depth of the term tuples is increased, then the complexity of TERM_TUPLE_COVER grows exponentially. This property is shared by the algorithms in [FL 96] and [CP 95].

4 Redundancy Criterion

In the previous chapter we have seen, that the non-variable positions of a term tuple set are decisive for the complexity of procedure TERM_TUPLE_COVER. We shall now give a redundancy criterion for term tuples based on the non-variable positions. The integration of this redundancy criterion into procedure TERM_TUPLE_COVER leads to a much smaller upper bound on the time complexity.

Theorem 4.1 (redundancy criterion based on non-variable positions) *Let $M = \{T_1, \ldots, T_n\}$ be a set of linear term k-tuples w.r.t. some Herbrand universe H. Furthermore, let $1 \leq p \leq k$ be an index within the k-tuples (i.e.: p is a position of M which has no subpositions).*
Now suppose that at position p, there exist non-variable terms but some function symbol $f \in FS(H)$ does not occur as leading symbol of any of these terms, i.e.: $(\exists i)$ s.t. $[T_i|p]$ is a non-variable term but $(\nexists i)$ s.t. $[T_i|p]$ is a term with leading symbol f. Then every term tuple $T \in M$ with a non-variable term at position p is redundant *and may, therefore, be deleted, i.e.: Let $M' := \{T \in M; [T|p]$ is a non-variable term$\}$. Then M covers all of H^k, iff $M - M'$ does.*
Proof sketch: *Suppose that the set $M = \{T_1, \ldots, T_n\}$ covers all of H^k. Furthermore let T be an arbitrary ground term tuple which is covered by some T_i with a non-variable term at position p. Then $[T|p]$ is a term t with leading symbol g for some $g \in FS(H)$ s.t. $g \neq f$.*
Now let us look at the term tuple S which results from substituting an arbitrary term s with leading symbol f for position p in T, i.e.: $[S|p] = s$ and S and T coincide everywhere else. This term tuple S can obviously not be covered by any of the term tuples with a non-variable term at position p, since any such non-variable term has a leading symbol g different from f. Hence S is subsumed by some term tuple T_j, which has a variable x at

position p, i.e.: $S = T_j\sigma$, for some ground substitution σ, s.t. $x\sigma = s$.
Now define the ground substitution τ, where $x\tau = t$ and $y\tau = y\sigma$ for all other variables. Then the equality $T = T_j\tau$ holds and, therefore, T is also a ground instance of T_j. $\diamond$

Note that (for a fixed Herbrand universe H) the redundancy criterion of theorem 4.1 can be easily tested in polynomial time. Nevertheless, it is, in general, even more powerful than H-subsumption: An instance of some term tuple $T \in M$ is *redundant*, either if it is H-subsumed by the other term tuples in M or if the term tuple cover problem M is unsolvable anyway.

The following theorem puts this redundancy criterion to work by combining it with the procedure TERM_TUPLE_COVER from chapter 3.

Theorem 4.2 (application of the redundancy criterion) *The linear term tuple cover problem can be decided in time $O(2^n + pol(N))$, where n denotes the number of term tuples and $pol(N)$ is some polynomial function in the total length N of an input problem instance.*
Proof sketch: *We modify procedure TERM_TUPLE_COVER in the following way: Before the* **for all** *f^α - loop is entered, the redundancy criterion of theorem 4.1 is applied to position j. If some function symbol $f \in FS(H)$ is missing as leading symbol of some term at position j, then all term tuples with a non-variable term at position j are deleted and TERM_TUPLE_COVER is called recursively with the remaining term tuples. Otherwise the* **for all** *f^α - loop is entered and executed as usual:*

If $|FS(H)| = 1$, then the original procedure TERM_TUPLE_COVER does not branch at all. Hence, it naturally has polynomial time complexity in the length of the input problem.

If $|FS(H)| > 1$, then the redundancy check immediately before the **for all** *f^α - loop has the following effect: In case that some function symbol f is missing as a leading symbol at the non-variable position j, then at least 1 term tuple is deleted in the recursive call of TERM_TUPLE_COVER. If on the other hand, all function symbols f actually do occur at position j, then the sets of term tuples in the recursive calls within the loop contain at most $(|M| - |FS(H)| + 1)$ term tuples (since in the recursive call for each function symbol f, the term tuples which have any other leading function symbol at position p are deleted). The worst case occurs, when FS(H) contains only 2 function symbols: In this case, the procedure branches into 2 recursive calls with at most $M - 1$ term tuples.* $\diamond$

The following example illustrates the whole algorithm for the model equivalence problem of LARs.

Example 4.3 (summary of the model equivalence solution method) *Let $\Sigma = \{P^2, Q^1, f^2, a^0\}$ be the signature which underlies the LARs $\mathcal{A}$ and $\mathcal{B}$, with*

$$\begin{aligned}\mathcal{A} &= \{P(f(a,f(a,x),f(y,z)),\ P(f(x,f(a,a)),f(y,z)),\ Q(f(a,y)),\\ &\quad\ P(f(x,f(a,f(y,z))),a),\ Q(f(b,y)),\ P(f(x,f(y,a)),z)\} \text{ and}\\ \mathcal{B} &= \{P(f(u,f(a,v)),w),\ Q(f(x,y)),\ P(f(u,a),f(v,w))\}\end{aligned}$$

According to lemma 2.1, the model equivalence problem of $\mathcal{A}$ and $\mathcal{B}$ can be expressed by 9 H-subsumption problems, each of which can be further transformed into a term tuple cover problem. We only work out the first one here: By theorem 2.2 we know, that $\mathcal{A} \leq^H_{ss} P(f(u,f(a,v)),w)$ holds, iff $M = \{(a,x,f(y,z)),(x,a,f(y,z)),(x,f(y,z),a),(x,a,z)\}$ covers all of H^3. The actual parameters and the actions carried out by the calls of procedure TERM_TUPLE_COVER (combined with the redundancy criterion of theorem 4.1) are given below:

original call: $M = \{(a, x, f(y,z)), (x, a, f(y,z)), (x, f(y,z), a), (x, a, z)\}$

M is reduced (by theorem 4.1) to: $M = \{(x, a, f(y,z)), (x, f(y,z), a), (x, a, z)\}$.

Column 1 (consisting of variables only) is ignored.

1st recursive call (for $a^0 \in FS(H)$ *in column 2):* $M_1 = \{(f(y,z)), (z)\}$

The first tuple is deleted by theorem 4.1.

The procedure returns **true** *since the second tuple consists of variabels only.*

2nd recursive call (for $f^2 \in FS(H)$*):* $M_2 = \{(y, z, a), \}$

Columns 1 and 2 (consisting of variables only) are ignored.

The only remaining tuple is deleted (by application of theorem 4.1 to column 3).

The next recursive call is done with the empty set and, hence, returns **false**.

The overall result of the original procedure call is **false**, *i.e.: M does not cover H^3. Therefore, $\mathcal{A}$ does not H-subsume $P(f(u, f(a,v), w)$ and, hence, the LARs $\mathcal{A}$ and $\mathcal{B}$ are not equivalent.*

5 Concluding Remarks and Future Work

The theoretical insight into the real "source" of complexity of the model equivalence problem of LAR's (namely the number of atoms rather than the total length of the input problem) ultimately led to an algorithm which is, in general, considerably more efficient than previously known ones. A similar result for clause evaluation would be desirable. To this end, 2 kinds of generalization are necessary, namely from atoms to clauses (containing also negative literals) and from linear to non-linear expressions. Both directions of generalization are non-trivial, e.g.: The problem transformation in theorem 2.2 cannot easily be extended so as to cope with negative literals. Furthermore, the redundancy criterion in theorem 4.1 is no longer correct for non-linear term tuples. Hence, the investigation of the *clause evaluation problem* must be left for future work.

Another aspect which deserves further investigation arises from the close relationship between models and constraint solving: In chapter 1 it was already mentioned that the problems of model equivalence and clause evaluation can be first transformed into equational problems and then tackled by constraint solving methods. In other words, algorithms developed in the field of constraint solving can be directly applied to the problems presented here. On the other hand, it would be interesting to find out in what way the ideas presented in this paper can be applied to *constraint solving*.

References

[CL 89] H. Comon, P. Lescanne: Equational Problems and Disunification, Journal of Symbolic Computation, Vol 7, pp. 371-425 (1989).

[CP 95] R.Caferra, N.Peltier: Extending semantic Resolution via automated Model Building: applications, Proceedings of IJCAI'95, Morgan Kaufmann (1995).

[FL 96] C.Fermüller, A.Leitsch: Hyperresolution and Automated Model Building, Journal of Logic and Computation, Vol 6 No 2, pp.173-230 (1996).

[Got 97] G.Gottlob: The Equivalence Problem for Herbrand Interpretations, unpublished note (1997).