THE MANAGEMENT OF HEURISTIC SEARCH IN BOOLEAN EXPERIMENTS WITH RUE RESOLUTION

Vincent J. Digricoli

Hofstra University

166-11 17th Road, Whitestone, N.Y. 11357

ABSTRACT

In assessing the power of a theorem prover, we should select a theorem difficult to prove, compare the quality of proof with the published work of mathematicians, and most important determine whether cpu time used to find the proof is economically acceptable.

In this paper we apply the above criteria to RUE resolution, equality-based binary resolution which incorporates the axioms of equality into the definition of resolution. We select a theorem in Boolean algebra, show the published proof of George and Garret Birkhoff side by side with the computer deduced proof achieved in less than 30 seconds of cpu time. The proof is quite long requiring the derivation of four lemmas and is proven by two RUE refutations of 16 and 18 steps respectively. The same refutations with the equality axioms and unification resolution are 38 and more than 40 steps. Hence, the power of RUE resolution is shown by the brevity of proof compared to using the equality axioms.

The primary pragmatic issue in theorem proving is the effective management of heuristic search to find proofs in acceptable computer time. Whether an inference system supports or obstructs this objective is a crucial property and in this paper we explain in detail the heuristics applied to find proofs. These heuristics are RUE specific and dependent, and cannot be applied in the context of unification resolution.

*This research was supported by a sabbatical grant from the IBM Systems Research Institute in New York.

I RESOLUTION BY UNIFICATION AND EQUALITY

RUE resolution is equality-based binary resolution in that it incorporates the axioms of equality into the definition of resolution, making refutations less than half as long as compared to using the equality axioms. Furthermore, it establishes a context for heuristics leading to more efficient searches for proofs. RUE resolution is based on the following rule of inference:

1. RUE Rule of Inference:

"The RUE resolvent of $P(s1,..,sn) \vee A$ and $\overline{P}(t1,..,tn) \vee B$, is $\sigma A \vee \sigma B \vee D$, where σ is a substitution and D is a disjunction of inequalities specified by a disagreement set of the complementary literals, $\sigma P(s1,..,sn)$ and $\sigma \overline{P}(t1,..,tn)$."

The above inference rule is in open form since we are free to choose the substitution and disagreement set to be used. Let us now define the notion of a disagreement set:

2. Disagreement Set of a Pair of Terms:

"If s,t are non-identical terms, the set of one element, the pair s:t, is the origin disagreement set. If s,t have the form, $f(a1,..,ak), f(b1,..,bk)$, then the set of pairs of nonidentical, corresponding arguments is the topmost disagreement set. Furthermore, if D is a disagreement set, then D' formed by replacing any member of D by the elements of its disagreement set, is also a disagreement set. If s,t are identical terms, the empty set is the sole disagreement set."

For example, the pair of terms:

$f(a,h(b,g(c))) : f(b,h(c,g(d)))$ has the

disagreement sets:

D1: { f(a,h(b,g(c))):f(b,h(c,g(d))) }

D2: { a:b, h(b,g(c)):h(c,g(d)) }

D3: { a:b, b:c, g(c):g(d) }

D4: { a:b, b:c, c:d }

We now define a. disagreement of complementary literals, P(s1, . ., sn) ,P(t1, . ., tn) *at* the union:

$$D = \bigcup_{i=1}^{n} Di$$

where Di is a disagreement set of the corresponding arguments si,ti. The topmost disagreement set of P(s1,..,sn),P(tl...tn) is the set of pairs of corresponding arguments which are not identical.

Us ing the substitution axiom for predicates, we can now state:

P(sl,..,sn) A P(tl,..,tn) - D

where D now represents a disjunction of inequalities specified by any disagreement set of P,P. in resolution by unification and equality, we can resolve P and P immediately to D. For example:

P(f(a,h(b,g(c))))

⊢ (D)

P(f(b,h(e,g(d))))

resolves in four distinct ways depending on our choice of D. We may resolve to J'(a, h(b, g(c)))≠ *f* (b, h(e,g(d))) , to a≠b v b≠c v c≠d, or to an intermediate level D. These are logically distinct deductions since an input set may imply f(a,h(b,g(c))) =f(b,h(c,g(d))) without implying a=b ^ b=c A c=d, so that the former participates in a refutation but the latter does not.

We now define a second inference rule similar to the above applying directly to an inequality:

3. NRF Rule of Inference:

"The NRF resolvent of the clause tl≠t2 v A is dA v D, where 0 is a substitution and D is a disjunction of inequalities specified by a disagreement set of 0tl,0t2."

For example we may deduce by NRF:

f(a,h(b,g(c)))≠ f(b,h(c,g(d)))

- a≠b v b≠c v c≠d

or reduce to the inequalities of any di agreement set. We call the above the Negative Reflexive Function rule.

In (Digricoli, 1983) we prove:

4. Completeness of RUE Resolution:

"Tf S is an E-unsatisfiable set of clauses, there exists an RUE-NRF deduction of the empty clause from 3."

This theorem establishes that resolving to inequalities is complete without introducing the axioms of equality(for substitution, transitivity and reflexivity) or paramodulation. Apart from symmetry, the axioms of equality are applied implicitly by the RUE-NRP rules of inference.

We describe the above as completeness in 'open form' since we have not specified either- the substitution or disagreement set to be used. In (Digricoli, 1983), we deal with this issue and define the RUE unifier and the topmost viable disagreement set as part of a completeness theory stated in strong form. however, in this paper we will use RUE resolution in open form as defined above, heuristically exploiting this form and making efficiency of proof search our primary goal.

II OUR PRIMARY EXPERIMENT

Our case study deals with proving the following theorem:

Given: the axioms of Boolean algebra:

1.	x v y = y v x	commutivity
2.	x ∧ y = y ∧ x	"
3.	x v 0 = x	
4.	x ∧ 1 = x	
5.	x v $\bar{x}$ = 1	
6.	x ∧ $\bar{x}$ = 0	
7.	x(yvz) = xy v xz	distributivity
8.	x v yz = (xvy)(xvz)	"

Prove: x v (y v z) = (x v y) v z

associativity of logical or

This is a fairly complex theorem for a human to prove and we have the following proof published in the Transactions of the American Mathematical Society [2] by George and Garret Birkhoff:

Theorem: a v (b v c) = (a v b) v c

Proof (Birkhoff) :

We first prove lemmas: L1, L2, L3 and L4.

L1: a = aa since: a=al=a(ava)=aavaa
=aavO = aa

L2: avl = 1 since: avl = (avl) l = (avl) (ava)
=av(la)=ava=l

L3: a = avab since: a=al=a(bvl)=abval
=abva=avab

L4: a(avb) = a since: a(avb) = aa v ab
= a v ab = a

From lemmas Ll,L3,L4 and axiom 7, it follows that:

a = a((avb)vc)
b = b((avb)vc)
c - c((avb)vc)

Substitute the above for a,b,c in av(bvc):

av(bvc) = a((avb)vc) v (b((avb)vc) v
c((avb)vc)) .

On the right side, factor out the expression (avb)vc to the right:

av(bvc) = (av(bvc)) ((avb)vc) .

On the right side, distribute (av(bvc)) across ((avb)vc) :

av(bvc) = ((av(bvc))a v (av(bvc))b) v
(av(bvc))c .

Applying Ll,L3,L4 and axiom 7 to each member of the right side, we obtain:

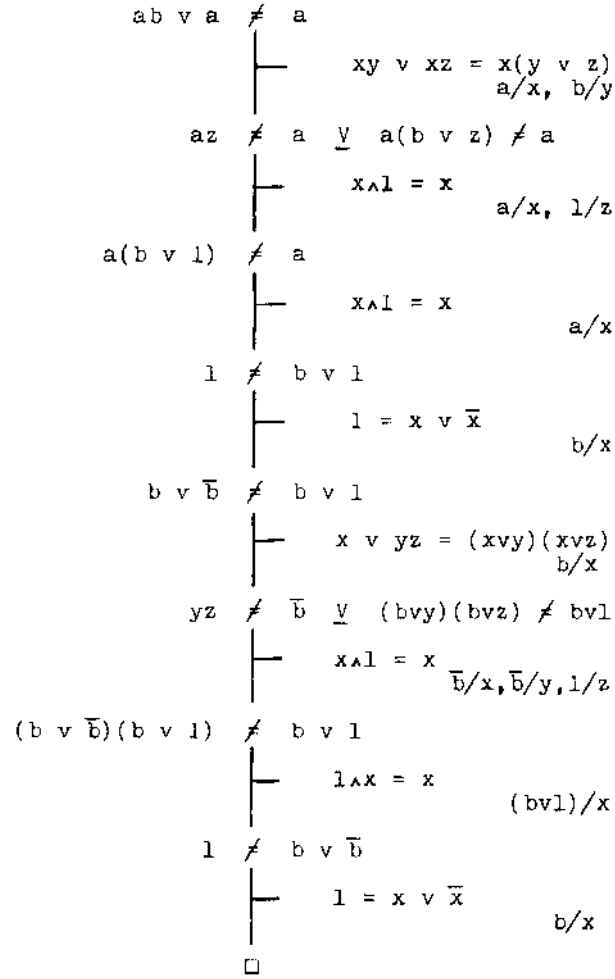
a v (bvc) (avb) v c

D

It is evident that especially the latter part of the above proof will be difficult for a human to deduce. In fact in human experiments with three mathematically astute university students, one could not prove associativity after six hours of work, a second proved associativity in nine hours and the third proved the theorem in three hours. Hence, we are asking the RUE theorem prover to prove a theorem which humans find quite difficult.

The following is the proof deduced by the RUE theorem prover in 24 seconds of cpu time (IBM 370/3081), using a total 7572 unifications in its proof search. We first ask the theorem prover to prove two lemmas, xy v x = x and (x v y)x = x, and then augment the input axiom set with these lemmas to prove associativity.

2. Refutation to Prove: xy v x = x

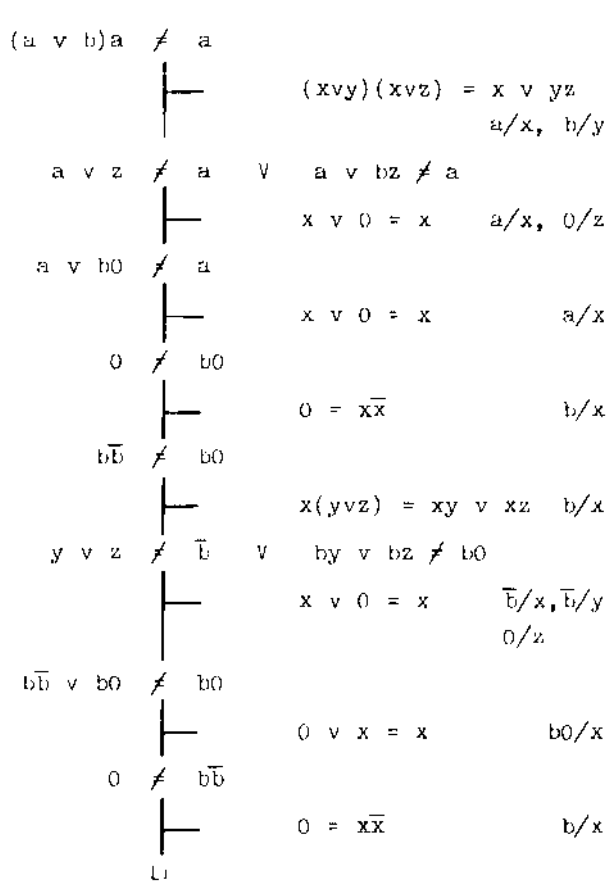


In the above refutation, we are uniformly applying the following substitution:

"In a left-to-right scan of complementary literals, first unify at the highest argument level, and then in a second scan unify at all lower levels."

The refutation is a succinct 8 step RUE proof of the lemma, xy v x =- x, and within this proof the sub-lemma, x v 1 = 1, is proven beginning at step 4. The same refutation performed with the equality axioms and standard unification resolution would be 19 steps.

3. Refutation to Prove: (x v y)x = x



We have a succinct RUE proof of 8 steps and an equivalent proof with the equality axioms and. unification resolution would be 19 steps. Note that the sublemrna 0 = XAO is proven beginning at step 4.

When the theorem prover is given the negated dual lemma:

ab *v* a ≠ a (avb)a ≠ a

it produced a. 16 step refutation proving both lemmas in 1.4 seconds with a total 1485 unifications in the proof search. This refutation is simply the concatenation of the above two refutations and counting sublemmas, four lemmas are being proven in a single run of the theorem prover. It corresponds to the work of the Birkhoff paper proving lemmas L1,L2,L3,L4 and is a substantial piece of work.

4. Refutation to Prove:

x v (y v z) = (x v y) v z

The 18 step refutation may be summarized as follows:

suppose:

a v (bvc) ≠ (avb) v c

then: (av(bvc))1 ≠ ((avb)vc)1

(av(bvc))(cv$\bar{c}$) ≠ ((avb)vc)(cv$\bar{c}$)

(av(bvc))c v (av(bvc))$\bar{c}$ ≠ *

c v (avb)$\bar{c}$ ≠ *

(Ax7,L3,L4) (Ax7,6,3)

where * is:

((avb)vc)c v ((avb)vc)$\bar{c}$

c v (avb)$\bar{c}$

(L4) (Ax7,6,3)

which completes the proof by contradiction. We see that the computer deduced proof is perhaps simpler and. more elegant than that stated in the Birkhoff paper. The actual refutation is given in Appendix T.

The theorem prover found the 18 step refutation proving associativity in 22.4 seconds using 6087 unifications in the proof search. An equivalent proof with the equality axioms and unification resolution would be more than 40 steps. Altogether 23.8 seconds with 7572 unifications were used for the entire proof. Cpu time would be substantially reduced by a more efficient implementation of the theorem prover in assembly language in place of PL/I. This is a very fine result compared to other published work (McCharen, Overbeek, Wos, 1976) and our purpose in this paper is to study the heuristic management of proof search which led to the above result.

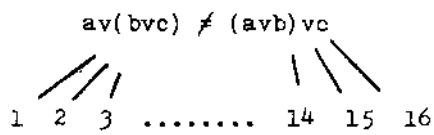
III FINDING A NEEDLE IN A HAYSTACK

Let us examine the enormity of the task of search which confronts a theorem prover seeking to find the refutations we we have stated. The difficulty of proving theorems in boolean algebra is compounded by the commutivity of the boolean functions (v,^) and the symmetry of equality, so that the axiom x v 0 = x, has 4 variants based on these properties. We can incorporate the entire effect of commutivity and symmetry by stating for the remaining boolean axioms all variants based on these properties:

1.	$x \vee 0 = x$	4 variants
2.	$x \wedge 1 = x$	4 "
3.	$x \vee \bar{x} = 1$	4 "
4.	$x \wedge \bar{x} = 0$	4 "
5.	$x(yvz) = xy \vee xz$	64 "
6.	$x \vee yz = (xvy)(xvz)$	64 "

Hence, in boolean algebra we are really dealing with an input set of 144 axioms when we drop the axioms for boolean commutivity and equality symmetry. Furthermore, the RUE inference rules implicitly incorporate the axioms of equality for substitution, transitivity and reflexivity and these axioms also do not appear in the input set. Let us assume that we will limit the input set to n clauses selected from the above variants. Let us assess the magnitude of search for the 18 step refutation which proves associativity when n is as low as 16.

The theorem prover begins with the negated theorem with skolem constants, $av(bvc) \neq (avb)vc$, and must find the refutation sequence we have stated. The refutation search is represented by a tree:



Since in RUE resolution complementary literals always resolve, each of the 16 clauses in the input set resolves immediately with the inequality of the negated theorem and in a breadthfirst search 16 resolvents appear at level one of the search tree. The complete breadthfirst expansion of the search tree to level 18 has 16-to-18th leaf nodes. The situation is actually worse if we take into account resolving to different disagreement sets making the expansion factor at a node higher than 16. Even with the input set reduced to 16 clauses, a breadthfirst search for a refutation is out of the question. The refutation we seek will be a linear input refutation represented by a path in the above tree. A miniscule number of paths from the root will culminate in the empty clause but at least one will when the input set is E-unsatisfiable.

We have stated that the refutation for associativity was actually attained by the RUE theorem prover in 6087 unifications, i.e., the theorem prover heuristically developed a subtree of 6087 nodes in place of the breadthfirst expansion. We wish now to explain precisely what heuristics were applied to define this subtree of search.

IV A HEURISTICALLY CONTROLLED PROOF SEARCH

In order to find a refutation in acceptable computer time, we must drastically prune the breadthfirst search tree and furthermore order the search in the remaining subtree. We will define the components of an evaluation function which heuristically determine which search paths to abort and what is the best leaf node to expand in the search tree. Furthermore, in applying an axiom to a node, we must heuristically select a disagreement set apt to participate in a refutation. The following principles were applied with these objectives in mind:

(1) heuristic ordering by degree of unification,

(2) selection of the lowest level disagreement set not containing an irreducible literal,

(3) heuristic substitution selection,

(4) complexity bounds relating to:

- (a) argument nesting
- (b) number of distinct variables in a clause
- (c) number of occurrences of the same constant or function symbol in a clause
- (d) maximum number of literals in a clause
- (e) maximum character length of a clause

(5) removing redundant resolvents,

(6) frequency bounds for the use of individual axioms in a ref. path.

All of the above principles are syntactic in nature and apply generically to experiments performed. (1) through (3), which were crucial in our work, are RUE specific and cannot be applied in standard unification resolution with the equality axioms. The remaining principles have been commonly used in resolution theorem proving.

A. Heuristic Ordering by Degree of Unification

If we wish to erase the literal, tl ≠ t2, in a refutation, we measure the relevancy of an axiom al = a2 by computing the degree of unification between literals as follows;

(1) Apply the mgpu (the most general partial unifier) to complementary literals to obtain tftl/tft2, tfal=tfa2.

(2) Set w=0 (unification weight).

(3) For i = 1,2:
if 0ti matches 0 ai identically,
then w = w+50,
else if σti,σai are the same function,
say: σti is f(bl,b2) and σai, f(cl,c2),
then w = w+20 and, furthermore,
w = w+15 for each matching pair of corresponding arguments.

This is a simple scheme of matching which computes a weight of 100 when the mgu of complementary literals exists, and 0 when there is no degree of unification. There is also an intermediate scoring between these extremes. We now state our first principle of heuristically ordering the expansion of the refutation search tree:

(1) Apply axioms to a negative literal in the order of higher degree of unification first and set a lower limit SDWMIN below which we suppress or postpone the application of an axiom (search directive weight minimum).

(2) Furthermore, among axioms which satisfy SDWMIN, select the first SDLIM axioms with the highest unification scores (search directive limit).

The 18 step refutation for associativity was found using SDWMIN=50, i.e., we pruned the search tree of all axioms falling below this unification score. In proving the supporting lemmas for associativity, all refutations were found using SDLIM=3. As naturally intuitive, this heuristic by degree of unification may be in RUE resolution, it, nonetheless, does not apply in unification resolution which requires the mgu at each deduction step.

B. Selecting the Lowest Level Disagreement Set not Containing an Irreducible Literal

Typically in adding the negated theorem to the input clause set, we introduce skolem constants and when it is evident that these constants are in effect arbitrary constants in respect to the input axioms, then we can conclude that inequalities on skolem constants like a=b are irreducible, i.e., we cannot deduce a=b from the axiom set.

For example, the negated theorem, av(bvc)≠(avb)vc, introduces skolem constants a,b,c which in respect to the axioms of boolean algebra are arbitrary constants and we cannot prove two of these constants equal. Thus we should never generate an inequality like a≠b in an RUE deduction. Furthermore, inequalities like (xvx)a ≠ b which demodulate to irreducible literals are also irreducible and cannot appear in a refutation. This leads to the following heuristic rule:

"In an RUE deduction, choose as resolvent inequalities, the innermost inequalities not containing an irreducible literal."

Hence, in the refutation to prove xy v x = x, we resolved:

$$l \neq b \vee l \quad \vdash \quad l = x \vee \bar{x} \qquad b/x$$

$$b \vee \bar{b} \neq b \vee l$$

since resolving to b≠l would result in an irreducible literal. This heuristic proved successful, not only in all the boolean experiments, but also in ten experiments in group and ring theory (Digricoli, 1981). The RUE theorem prover permits the user to specify as input a list of irreducible literals, and, at the end of a run, it produces a list of ground literals present at leaf nodes of the search tree which the user may scan for irreducible literals in preparation for the next run.

C. Heuristic Substitution Selection

In the completeness analysis in (Digricoli, 1983), we specify that substitutions may be unconditionally performed in variables at the first argument level, like P(x), but only in variables at lower-levels, like P(f(x)), if certain conditions are met by the input set. This leads to longer refutations requiring extensive use of the NRF rule, when the same refutations can be stated in abbreviated form without the NRF rule, if we permit immediate substitutions at lower argument levels. In fact, it occurs in experiments performed that the following maximum unification is the refutation substitution:

> "In a left-to-right scan of complementary literals, first unify at the highest argument level, and then in a second scan unify at all lower levels."

Note that we have given priority to substituting at the first argument level as specified by the completeness theory. However, the above substitution selection which enhances the efficiency of finding proofs, is not universally compatible with RUE completeness and we show this in (Digricoli, 1983).

D. Pruning; by Complexity Bounds

An important method of pruning the search tree Is to apply complexity bounds on resolvents which are formed. Theorem provers working in areas of the search tree having little relation to a refutation tend to produce resolvents which are too complex under a variety of attributes. The RUE theorem prover permits the user to specify complexity limits which when exceeded cause the resolvent to be discarded. These limits relate to the depth of argument nesting, the character length of a clause, the number of literals in a clause, the frequency of appearance of a constant or function symbol in a clause (or literal) and the number of distinct variables in a clause. It is true that the ideal setting of complexity bounds can only be derived by Knowing a refutation. Nonetheless, it is important to know to what extent the proof search is contracted by applying complexity bounds. Experiments show that it is a good heuristic to use tight bounds to begin with (possibly derived from examining proofs in prior work with the theorem prover) and to gradually relax these bounds. The experimental results in this paper- are first stated by applying complexity limits which ar e a profile of t h e refutations derived. We then suspend all use of these bounds, to determine the degradation of search which occurs. Both sets of results turn out to be favorable .

V EXPERIMENTAL RESULTS

	Steps in RUE Proof	Run 1 - with complexity bounds		Run 2 - without complexity bounds	
		CPU Time Seconds	Total Unif.	CPU Time Seconds	Total Unif.
B1: x v 1 = 1	5(11)	0.155	183	0.508	558
B2: x ∧ 0 = 0	5(11)	0.173	220	0.607	663
B3: xy v x = x	8(19)	0.654	724	0.964	1034
B4: (xvy)x = x	8(19)	0.647	773	0.868	1020
B1∧B2∧B3∧B4	16(38)	1.401	1485	3.008	2855
B5: xv(yvz) = (xvy)vz	18(40+)	22.431	6087	70.705	18727

(1) Under length of proof, 5(11) denotes that the 5 step RUE refutation is 11 steps when restated with the equality axioms and unification resolution.

(2) In Run 1 complexity bounds were applied which perfectly fit the refutation derived, but in Run 2 no complexity bounds were applied to limit search. Apart from this distinction, both Run 1 and 2 used the same input set and heuristics, in Run 2 of B5, complexity bound were relaxed one level and applied.

(3) To prove lemmas B1 through B4, both individualy and altogether, the following-Input set was used:

1.	$x \vee 0 = x$	4 variants
2.	$x \wedge 1 = x$	4 "
3.	$x \vee \overline{x} = 1$	4 "
4.	$x \wedge \overline{x} = 0$	4 "
5.	$x(yvz) = xy \vee xz$	8 "
6.	$x \vee yz = (xvy)(xvz)$	8 variants

A complete set of variants was used for the first four axioms but only a partial set for each of thedistributive axioms. In resolving with an inequality, the theorem prover first chose from each variant set, that axiom having the highest unification score with the inequality. This reduced the applicable axioms from 32 to 6, and of these only three with the highest unification scores were applied since SDLIM was set to 3. This meant that the node expansion factor was 3 instead of 32 in the search tree and the excellence of the results in proving lemmas is due primarily to this heuristic together with avoidance of irreducible literals.

(4) In proving B5 associativity, we reduced the input set to 16 clauses and introduced variants of two proven lemmas:

1.	$x \vee 0 = x$	
2.	$x = x \vee 0$	
3.	$x \wedge 1 = x$	
4.	$x = x \wedge 1$	
5.	$x \vee \overline{x} = 1$	
6.	$1 = x \vee \overline{x}$	
7.	$x \wedge \overline{x} = 0$	
8.	$0 = x \wedge \overline{x}$	
9.	$x(yvz) = xy \vee xz$	
10.	$xy \vee xz = x(yvz)$	
11.	$(yvz)x = yx \vee zx$	
12.	$yx \vee zx = (yvz)x$	
13.	$(yvx)x = x$	lemma
14.	$x = (yvx)x$	
15.	$yx \vee x = x$	lemma
16.	$x = yx \vee x$	

In this experiment, the theorem prover in resolving with an inequality chose only those input clauses whose unification score was 50 or greater (SDWMIN=50) and 3DLIM was not used. Our next experiments will attempt to prove associativity without using lemmas and with the same input set of 32 clauses previously used. This will be an exceptionally long refutation.

REFERENCES

[1] Digricoli, V. J., "Resolution by Unification and Equality", Courant Institute, New York Univ., February 1983, (available through Univ. Microfilms).

[2] Birkhoff George and Garret, "Distributive Postulates for Systems Like Boolean Algebra", Transactions of American Mathematical Society, vol 60, July-Dec. 1946.

[3] Digricoli, V.J., "The Efficacy of RUE Resolution: Experiments and Heuristic Theory" In Proc. IJCAI-81., Vancouver, B.C., Canada, August 1981, pp. 539-547.

[4] Digricoli, V.J., "Automatic Deduction and Equality" In Proc. National ACM Conf., Detroit, Michigan, October 1979, pp. 240-250.

[5] Morris, J., "E-Resolution: an Extension of Resolution to Include the Equality Relation" In Proc. IJCAI-69.

[6] McCharen, Overbeek and Wos, "Problems and Experiments for Automated Theorem Proving", IEEE Transactions on Computers, C25-80, 1976.

L7I Robinson, J.A., "A Machine Oriented Logic Based on the Resolution Principle", JACM 12, 1965, pp.23-41.

[8] Brand, D., "Proving Theorems with the Modification Method", SIAM Journal of Computing, vol 4, no 4, 1975.

[9] Harrison, M., and Rubin, N., "Another Generalization of Resolution", JACM vol 25, no 3, 1978.

[10] Robinson, G., and Wos, L., "Paramodulation and Theorem Proving", Machine Intelligence, vol 4, 1969.

[I1] Wos, Overbeek and Henschen, "Hyperparamodulation" In Proc. 5CAD, Les Arcs, France, July 1980.

[I2] Slagle, J., "Automatic Theorem Proving with Builtin Theories Including Eq.", JACM, vol 19, no 1, January 1972.

[I3] Shostak, R., "An Algorithm for Reasoning about Equality", CACM, vol 21, No 7, July 1978.

*** Appendix I with the 18 step refutation proving associativity has been omitted due to space limitations. This refutation will be given at the conference presentation.

Definition. For a given signature SIG = (S,F,P) we say the triple (A,S,F) is an algebra of type (S,F), iff the following is satisfied.

- A is a nonempty set.
  To every sort $S \in \mathbb{S}$ is assigned a subset S^A of A such that $\top^A = A$ and for all $S,T \in \mathbb{S}$: $S \leq T \Rightarrow S^A \subseteq T^A$
  For $c \in C_S$, an element $c^A \in A$ exists, such that $c^A \in S^A$.
- For functions f, $f^A: A^n \to A$ is a mapping such that the image of $S_1{}^A \times \ldots \times S_n{}^A$ is in $S_{n+1}{}^A$ for all $(S_1, \ldots, S_{n+1}) \in SO(f)$. □

By the definition of a signature, we have that $S^A \neq \emptyset$ for every $S \in \mathbb{S}$ since every sort contains at least one constant.

Definition. A mapping $\varphi: A \to B$, which has the properties, that
a) $\varphi(S^A) \subseteq \varphi(S^B)$ for every sort $S \in \mathbb{S}$ and
b) $\varphi(f^A(a_1, \ldots, a_n)) = f^B(\varphi(a_1), \ldots, \varphi(a_n))$
is called a Σ- homomorphism. □

$(\Sigma TERM,S,F)$ is the free algebra of type (S,F) and $(\Sigma TERM_{gr},S,F)$ is the initial algebra of type (S,F), where $\Sigma TERM_{gr}$ is the set of terms without variables.

A mapping $\sigma: \Sigma TERM \to \Sigma TERM$, which is a Σ-endomorphism where $(x \in V \mid \sigma x \neq x)$ is finite, is called a Σ-substitution. Let ΣSUB be the set of all Σ-substitutions. Σ-substitutions are exactly those mappings, which are substitutions in the normal sense and have the additional property that $[\sigma(x)] \leq [x]$ for all $x \in V$.

$P(t_1, \ldots, t_n)$ is an atom, where P is a predicate symbol and the t_i s are terms such that $[t_i] \leq S_i$ where $(S_1, \ldots, S_n) = SO(P)$. A literal is a signed atom. A clause is a set of literals, i.e. an abbreviation for the disjunction of the literals, where all variables are universally quantified. A ground atom, a ground literal or a ground clause is one without variables. Instances of atoms, literals and clauses are their images under a Σ-substitution. Equality (=) is a distinguished binary predicate with domainsorts $SO(=) = (\top,\top)$.

Since a clause set CS is said to be satisfiable, if and only if a model for CS exists, there is the need for a precise definition of a model respecting polymorphic signatures.

Definiton. A ΣE-model for a clause set CS is a triple (D,SIG,R), which satisfies the following conditions:

- (D,S,F) is an algebra of type (S,F).
- For every predicate P there exists a relation $P^D \in R$ of the same arity
- All clauses in CS are valid under all Σ-homomorphisms $\varphi: \Sigma TERM \to D$, i.e.
  all clauses are valid under all assignments of values in D to variables in clauses, where sorts are respected. (A literal +P(...) is valid under φ, if it s image under φ is in the relation P^D).
- The equality predicate is represented as the identity on D.

3 Unifcation of Polymorphic Terms

For $\sigma \in \Sigma SUB$ we use the notation
$DOM(\sigma) = \{x \in V \mid \sigma x \neq x\}$, $COD(\sigma) = \{\sigma x \mid x \in DOM(\sigma)\}$ and VCOD (σ) is the set of variables in COD(σ). A unification problem for two terms s and t is denoted as ⟨s = t⟩. The set of variables of a term t is denoted as VAR(t).

For $W \subseteq V$ and $\sigma,\tau \in \Sigma SUB$, define $\sigma = \tau$ [W], iff $\sigma x = \tau x$ for all $x \in W$. Furthermore define $\sigma \sqsubseteq \tau$ [W], iff $\sigma = \lambda\tau$ [W] for some $\lambda \in \Sigma SUB$. Obviously $\sqsubseteq$ [W] is a reflexive and transitive relation on Σ-substitutions.

For a subset $U \subseteq \Sigma SUB$ we say U is separated on a set of variables W, iff DOM(σ) = W and VCOD(σ) ∩ W = ∅ for all $\sigma \in U$ and all VCOD s are disjoint. Note that such a set consists of idempotent Σ-substitutions only.

Definition. A set of most general unifiers $\mu U\Sigma(s,t)$ for two terms s and t is defined as a subset of ΣSUB, which is separated on W = VAR(s,t) and satisfies

- σs = σt for all $\sigma \in \mu U\Sigma(s,t)$. (correctness).
- For all $\delta \in \Sigma SUB$ with δs = δt we have $\delta \sqsubseteq \sigma$ [W] for some $\sigma \in \mu U\Sigma(s,t)$. (completeness)
- $\sigma \sqsubseteq \tau$ [W] $\Rightarrow \sigma = \tau$ for all $\sigma,\tau \in \mu U\Sigma(s,t)$. (minimality) □

Definition. A Σ-substitution is a weakening substitution (coercing) [Wa84,GM85]., iff
(i) σ substitutes variables for variables and
(ii) σ is injective on DOM (σ) and
(iii) DOM(σ) ∩ VCOD (σ) = ∅. □

The set of all weakening substitutions is called ΣWSUB. In the computation of a set of most general unifiers, weakening substitutions are used to solve unification problems like ⟨x = t⟩, where [t] is not a subsort of [x].

Definition. The set of most general weakening substitutions for t and S, $\mu W\Sigma_S(t)$ is a set of weakening substitutions which is separated on W = VAR(t) and satisfies:

- $[\sigma t] \leq S$ for all $\sigma \in \mu W\Sigma_S(t)$ (correctness)
- $\forall \delta \in \Sigma SUB$ with $[\delta t] \leq S$ there exists a $\sigma \in \mu W\Sigma_S(t)$ such that $\delta \sqsubseteq \sigma [W]$ (completeness)
- $\forall \sigma, \tau \in \mu W\Sigma_S(t)$: $\sigma \sqsubseteq \tau [W] \Rightarrow \sigma = \tau$ (minimality) □

This definition is of course only useful for terms, which are ill-sorted or for terms t where [t] is not a subsort of S and the outermost function symbol of t is polymorphic.
The following theorem is valid in the ΣRP-calculus [Wa83,Wa84] and can be extended to ΣRP'

Theorem 1. [Sch85a] Let $t \in TERM$ and $S \in \mathbf{S}$. If there exists a $\theta \in \Sigma SUB$ such that $[\theta t] \leq S$, then $\mu W\Sigma_S(t)$ exists, it is not empty and it is always finite

An immediate consequence is:

Theorem 2. For unifiable x and t the set $\mu U\Sigma(x,t)$ exists, is not empty and finite □

The proof follows from the fact ,that $\mu U\Sigma(x,t) = \{\sigma \circ \{x \leftarrow \sigma t\} \mid \sigma \in \mu W\Sigma_{[x]}(t)\}$, if $x \notin VAR(t)$

Example. The following example demonstrates, that for a unification problem $x=t$ the minimal set of most general unifiers can grow exponentially
Consider the sort structure $S = \{N,NZ,Z\}$, where N, NZ and Z have the same meaning as in the example above. Let $x \in V_{NZ}$ and $x_i \in V_N$. The signature of the function "·" (product) is
$SO(\cdot) = \{(N,N,N), (NZ,N,NZ), (N,NZ,NZ), (Z,NZ,Z), (NZ,Z,Z), (Z,NZ,Z), (NZ,Z,Z), (Z,Z,Z), (NZ,NZ,NZ)\}$. The unification problem $\langle x = (x_1 \cdot x_2 \cdot \ldots \cdot (x_{2n-1} \cdot x_{2n}) \rangle$ produces 2^n unifiers, since for every factor there are two independent solutions $\{x_{2i-1} \leftarrow y_{2i-1}\}$ and $\{x_{2i} \leftarrow y_{2i}\}$, where $[y_i] = NZ$. These solutions have to be combined independently.

For a set of variables W and a set of Σ-substitutions U, the set $MAX_{\sqsubseteq [W]}(U)$ is a set of Σ-substitutions, such that
i) $MAX_{\sqsubseteq [W]}(U) \subseteq U$ and
ii) $\forall \sigma \in MAX_{\sqsubseteq [W]}(U), \tau \in U: \sigma \sqsubseteq \tau [W] \Rightarrow \sigma = \tau$.
iii) $\forall \tau \in U\ \exists \sigma \in MAX_{\sqsubseteq [W]}(U): \tau \sqsubseteq \sigma [W]$.
Such a set exists, since $\sqsubseteq [W]$ is transitive.

The polymorphic unification algorithm ΣUNIFY, which is an extension of the algorithm for the ΣRP-calculus [Wa84], takes two terms s,t as input and produces a set of most general unifiers as output.
It is defined as follows. [Sch85a]

1) $U_{NEW} = \{\varepsilon\}$
2) REPEAT $U_{OLD} = U_{NEW}$, $U_{NEW} = \emptyset$
   For every $\sigma \in U_{OLD}$ DO
      Let (d,e) be the first disagreement pair of the two terms σs, σt.
      IF d or e is a variable, THEN DO
         $U_{NEW} = U_{NEW} \cup \{\sigma \circ \tau \mid \tau \in \mu U\Sigma(d,e)\}$
      OD.
   OD.
   UNTIL U_{NEW} is a set of Σ-unifiers for s,t or $U_{NEW} = \emptyset$
3) RETURN $MAX_{\sqsubseteq [VAR(s,t)]}(U_{NEW})$. □

We have the theorem, which is proved in [Sch85a]

Theorem 3. Let $s,t \in \Sigma TERM$ be Σ-unifiable. Then ΣUNIFY(s,t) terminates and returns a finite nonempty set, which is equivalent to $\mu U\Sigma(s,t)$. □

Definition. We say SIG is a unification unique signature, iff
- $\langle S, \leq \rangle$ is a semilattice with a unique greatest element.
- For all $f \in \mathbf{F}$ and all $S \in \mathbf{S}$, the set $\{(S_1,\ldots,S_{n+1}) \in SO(f) \mid S_{n+1} \leq S\}$ is either empty or contains a unique greatest element.

The following theorem is proved in [Sch85a]

Theorem 4. Let SIG be a unification unique signature. Then $\mu U\Sigma(s,t)$ is at most a singleton for all $s,t \in \Sigma TERM$. □

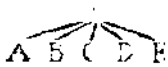
In [Sch85a] it is shown, that the condition, that $\langle S, \leq \rangle$ is a semilattice, does not restrict the expressiveness of the many-sorted calculus ΣRP'. In the case that $\langle S, \leq \rangle$ is a partially ordered set, it is possible to add sorts to the signature, such that the resulting sort structure is a semilattice and the satisfiability of clause sets is unchanged.

4 The ΣRP'-calculus.

Σ-factorization, Σ-resolution and Σ-paramodulation are defined as in [Wa83], however based on the unifier sets $\mu U\Sigma$ and $\mu P\Sigma$ respectively. The set $\mu P\Sigma$ is the set of most general unifiers for paramodulation, which is also always finite. The Σ-paramodulation rule includes the weakening rule of [Wa83]. The proofs in [WR73] can be generalized (see [Sch85a]) to the ΣRP'-calculus yielding the following result.

Theorem 5. Let CS be a clause set containing the functional reflexivity axioms. Then:
$CS \vdash_{\Sigma RP'} FALSE \iff CS$ is unsatisfiable.

Example The functional reflexvity axioms are needed for the completeness of the ∑RP-calculus
Let S -{T,A,B,C,D,E}



- Let a,b,c,d be constants of sort A,B,C,D respectively
- Let P be a unary predicate with SO(P) = (E).
- Let f ∈ F with a signature, such that [f(t$_1$,t$_2$)] = E, iff ([t$_1$] ≤ A and [t$_2$] ≤ C) or ([t$_1$] ≤ B and [t$_2$] ≤ D), otherwise [f(t$_1$,t$_2$)] = S.
- Let CS be the clause set { {a=b}, {c≠d}, {P(f(a,c))}, {¬P(f(b,d))} }.

This clause set is unsatisfiable, but E-paramodulation is not possible, since [f(b,c)] = [f(a,d)] = S. There is no deduction of the empty clause in ERP*. If the functional reflexivity axioms are present, then f(a,c)≠f(b,d) can be deduced from the functional reflexivity axioms and the clauses in CS. Hence the empty clause is then deducable. □
This result is somewhat theoretical In practical applications, the functional reflexivity axioms are not used, since (i) they increase the search space enormously and (ii) the subsumption rule would delete all of them but the axiom x≠x Clearly, this omission may be the reason for incompleteness

5 The Sort-Theorem. For a polymorphic signature SIG and a related clause set CS, we define an unsorted relativization CS_{rel} [Wa83, Ob62] as follows
For every sort S use a unary predicate P_S
The set of sort axioms A^{Σ} consists of
$\forall x \neg P_S(x) \vee P_T(x)$ if S≤T for S,T ∈ S.
$\forall x\ P_S(x)$ for the greatest sort S of S.
$\forall x_1 \ldots x_n \neg P_{S_1}(x_1) \vee \ldots \vee \neg P_{S_n}(x_n) \vee P_{S_{n+1}}(f(x_1, \ldots x_n))$
for every f ∈ F and every $(S_1,\ldots,S_n) \in SO(f)$.
The relativization C_{rel} of a clause C is the clause itself, where the sorts of the variables are ignored and for every variable x of C a literal $\neg P_{[x]}(x)$ is added to the clause. We have $CS_{rel} = A^{\Sigma} \cup \{C_{rel} \mid C \in CS\}$

The following theorem, which is proved in [Sch85a], and which is proved for other signatures in [Ob62] and [Wa83], states the equivalence of the sorted and the unsorted version of a clause set:

Theorem 6. (Sortensatz) Let CS be t SIG-sorted clause set Then
CS is unsatisfiable iff CS_{rel} is unsatiafiable D

Example This example it taken from [SM78], which appears to be a goldmine for theorem proving examples During a course on automated theorem proving IN the last semester our students had to translate these puzzles into first order predicate logic and to solve them with our theorem *prover* (Markgraf karl Refutation Procedure) [KM84] One of these problems (Problem 471 reads as follows
When Alice entered the forest of furgetfulness, she did not forget everything, onry certain things She often forgot her name and the most likely to forget was the day of the week Now the boo and the unicorn were frequent visitors to this forest These two are strange creatures The lion lies on Mondays, Tuesdays and Wednesdays and tells the truth on the other days of the week The unicorn, on the other hand lies on Thursdays, Fridays and Saturdays, but tells the truth on the other days of the week
One day Alice met the lion and the unicorn resting under a tree They made the following statements
Lion Yesterday was one of my lying days
Unicorn Yesterday was one of my tying days
Prom these statements Alice who was a bright girl, was able to deduce the day of the week What was it?

We use the predicates MO(x), TU(x), ... , SO(x) for saying that x is a Monday. Tuesday etc. Furthermore we need the binary predicate MEMB, indicating set Membership and a 3-ary predicate LA LA(x y z) is true if x says at day y that he lies at day z. LDAYS(x denotes the set of lying days of x The remaining symbols are self explaining One-character symbols like u,x,y,z are regarded as universally quantified variables

Axiomization of the days of the week:
MO(x) ⇔ ¬(TU(x)∨WE(x)∨TH(x)∨FR(x)∨SA(x)∨SU(x))
TU(x) ⇔ ¬(WE(x)∨TH(x)∨FR(x)∨SA(x)∨SU(x)∨MO(x))
WE(x) ⇔ ¬(TH(x)∨FR(x)∨SA(x)∨SU(x)∨MO(x)∨TU(x))
TH(x) ⇔ ¬(FR(x)∨SA(x)∨SU(x)∨MO(x)∨TU(x)∨WE(x))
FR(x) ⇔ ¬(SA(x)∨SU(x)∨MO(x)∨TU(x)∨WE(x)∨TH(x))
SA(x) ⇔ ¬(SU(x)∨MO(x)∨TU(x)∨WE(x)∨TH(x)∨FR(x))
SU(x) ⇔ ¬(MO(x)∨TU(x)∨WE(x)∨TH(x)∨FR(x)∨SA(x))

Axiomization of the function yesterday:
MO(yesterday(x)) ⇔ TU(x)
TU(yesterday(x)) ⇔ WE(x)

WE(yesterday(x)) ⇔ TH(x)
TH(yesterday(x)) ⇔ FR(x)
FR(yesterday(x)) ⇔ SA(x)
SA(yesterday(x)) ⇔ SU(x)
SU(yesterday(x)) ⇔ MO(x)

Axiomization of the function LDAYS:
MEMB(x LDAYS(lion)) ⇔ MO(x)∨TU(x)∨WE(x)
MEMB(x LDAYS(unicorn)) ⇔ TH(x)∨FR(x)∨SA(x)

Axiomization of the predicate LA:
¬MEMB(x LDAYS(u)) ∧LA(u x y) ⇒ MEMB(y LDAYS(u))
¬MEMB(x LDAYS(u)) ∧ ¬LA(u x y) ⇒
¬MEMB(y LDAYS(u))
MEMB(x LDAYS(u)) ∧LA(u x y) ⇒ ¬MEMB(y LDAYS(u))
MEMB(x LDAYS(u)) ∧¬LA(u x y) ⇒ MEMB(y LDAYS(u))

Theorem:
∃x LA(lion x yesterday(x)) ∧
LA(unicorn x yesterday(x))

The MKRP proof procedure at kaisersiautern found t proof for this unsorted version after 183 resolution steps, among them 81 unnecessary steps, hence the final proof was 102 steps long This proof contains a lot of trivial steps corresponding to common sense reasoning (like if today is Monday, it is not Tuesday etc)

Later the sort structure and the signature of the problem at hand was generated automatically by a translator module which accepts an unsorted clause set as input and products the equivalent many -sorted version together with the corresponding signature [Sch8 5b]

The sort structure and the signature contain all the relevant information about the relationship of unary predicates (like our days) and the domain-rangesort relation of functions The sort structure of the subsorts of DATS in our example is equivalent to the lattice of subsets of (Mo, Tu, We, Th, Fr, Sa, Su) without the empty set, ordered by the subset order Hence there are 127 (-2^7 1) sorts. The function yesterday is a polymorphic function with 127 domain sort relations liie yesterday ((MO, WE)) -(SU, TU)

The unification algorithm exploits this information and produces only unifiers, which respect the sort relations, I e (x <- t) is syntactically correct, if and only if the sort of the term t is less or equal the sort of the variable I We give an example for unification

the unifier of x:SO∘TU and yesterday(y.MO∘TU) is ($x \leftarrow$ yesterday(y_1: MO) , $y \leftarrow y_1$:MO). The MKRP theorem-proving system [KM84] has proved the theorem in the sorted version immediately without any unnecessary steps. The length of the proof is 6 As the protocol shows, the final substitution into the theorem clause was x ← y:Th. Thus the ATP has found the answer. Thursday. in a very straight forward and humanlike way Here is a proof protocol:

C1 All x:Mo MEMB (x LDAYS(lion))
C2 All x:Tu MEMB (x LDAYS(lion))
C3 All x:We MEMB (x LDAYS(lion))
C4 All x:Th MEMB (x LDAYS(unicorn))
C5 All x:Fr MEMB (x LDAYS(unicorn))
C6 All x:Sa MEMB (x LDAYS(unicorn))
C7 All x,y:Days u:Animal
MEMB(x LDAYS(u)) ¬LA(u x y)
MEMB (y LDAYS(u))
C8 All x,y:Days u:Animal
MEMB(x LDAYS(u)) LA(u x y)
¬MEMB(y LDAYS(u))
C9 All x,y:Days u:Animal
¬MEMB(x LDAYS(u)) ¬LA(u x y)
¬MEMB(y LDAYS(u))
C10 All x,y:Days u:Animal
¬MEMB(x LDAYS(u)) LA(u x y)
MEMB(y LDAYS(u))
C11 All x:Th∘Fr∘Sa∘Su ¬MEMB(x LDAYS(lion))
C12 All x:Tu∘We∘Su∘Mo ¬MEMB(x LDAYS(unicorn))
Th All x:Days
¬LA(lion x yesterday(x))
¬LA(unicorn x yesterday(x))

Proof.

C4,1 & C10,1 → R1:
All x:Th y:Days LA(unicorn x y)
MEMB(y LDAYS(unicorn))
R1,2 & C12,1 → R2:
All x:Th y:Tu∘We∘Su∘Mo LA(unicorn x y)
C3,1 & C8,3 → R3
All x:Days y:We MEMB(x LDAYS(lion))
LA(lion x y)
R3,1 & C11,1 → R4:
All x:Th∘Fr∘Sa∘Su y:We LA(lion x y)
R4,1 & Th,1 → R5
All x:Th ¬LA(unicorn x yesterday(x))
R5,1 & R2,1 → R6. □

Rec ently, Walther, [13J has derived a computer solution to the problem by reaxiomatising it in a many sorted logic before giving it to the MKRP-system at Karlsruhe [12] . His many sorted formulation has a significantly reduced search space only 12 clauses with 16 literals with an initial search space of 12 possible inferences and a total of 10 new clauses in the deduced proof

The purpose of this paper is to show that further efficiency is possible by recasting the problem in a different many sorted logic [3,4] which possesses a number of additional features which increase its expressiveness as compared to Walther's' logic [12] and the other many sorted logics to be found in the literature, eg [8,6,2,5] A table comparing the formulation of the problem in the different logics may be found towards the end of this paper in figure 2

2. HaJther's Axiornatisation

For convenience, Walther's many sorted axiomatisation is repeated here in order to better compare the formulations in the two different many sorted logics

In his logic *typv* declarations define the sort of function and constant symbols (his predicate symbols may also be typed but this feature is not needed in this example). *Sort* declarations define the sort hierarchy Animals, plants, grains, wolves, foxes, birds, caterpillars and snails are all sorts, named A, P, G, W, F, B, C and S respectively.

(1w) *type* w:W

(2w) *type* f:F

(3w) *type* b:B

(4w) *type* c:C

(5w) *type* s:S

(6w) *type* g:G

(7w) *sort* W ⊏ A

(8w) *sort* F ⊏ A

(9w) *sort* B ⊏ A

(10w) *sort* C ⊏ A

(11w) *sort* S ⊏ A

(12w) *sort* G ⊏ P

(13w) {E(*a1*,*p1*), -M(*a2*,*a1*), -E(*a2*,*p2*),E(*a1*,*a2*)}

(14w) {M(*c1*,*b1*)}

(15w) {M(*s1*,*b1*)}

(16w) {M(*b1*,*f1*)}

(17w) {M(*f1*,*w1*)}

(18w) { -E(*w1*,*f1*)}

(19w) { -E(*w1*,*g1*)}

(20w) {E(*b1*,*c1*)}

(21w) { -E(*b1*,*s1*)}

(22w) *type* h(C):P

(23w) {E(*c1*,h(*c1*))}

(24w) *type* i(S):P

(25w) {E(*s1*,h(*s1*))}

(26w) *type* j(A,A):G

(27w) { -E(*a1*,*a2*), -E(*a2*,j(*a1*,*a2*))}

The symbols *a1, f1* etc are all typed variables with the sort of the corresponding upper case letter (lw) to (6w) define a signature and play no part in the proof.

As already mentioned, Walther's proof is 10 steps long (9 resolutions and a factorisation). It is reproduced below (in a slightly altered form to show the factorisation explicitly).

(28w) {E(*a1*,*p1*), -M(*a2*,*a1*), -E(*a2*,*p2*), -E(*a2*,j(*a1*,*a2*))}
13w(4)+27w(1)

(29w) {E(*a1*,*p1*), -M(*a2*,*a1*), -E(*a2*,j(*a1*,*a2*))}
factor of 28w

(30w) {E(*w1*,*p1*), -E(*f1*,j(*w1*,*f1*))}
17w(1)+29w(2)

(31w) { -E(*f1*,j(*w1*,*f1*))}
19w(1)+30w(1)

(32w) {E(*f1*,*p1*), -E(*b1*,j(*f1*,*b1*))}
16w(1)+29w(2)

(33w) { -E(*b1*,j(*f1*,*b1*))}
31w(1)+32w(1)

(34w) {E(*b2*,*p1*), -M(*s1*,*b2*), -E(*s1*,*p2*)}
13w(4)+21w(1)

(35w) { -M(*s1*,*b2*), -E(*s1*,*p2*)}
33w(1)+34w(1)

(36w) { -E(*s1*,*p2*)}
15w(1)+35w(1)

(37w) {}
25w(1)+36w(1)

He attributes the success of his system in finding a proof to the significant reduction in the clause set and to the restriction on unification preventing the matching of variables with incompatible sorts For example clauses (20w) and (21 w) have no resolvent because *c1* and *s1* cannot be unified. In the unsorted case the two clauses (20cu) and (21 cu) do resolve to yield { B(x), -*C(y),* -S(y)}. This can then be resolved with (3cu) to yield { *C(y), 3(y)*\ However further inference will now yield a pure clause either { S(c)j or |-C(s)j. where c and s are skolem constants. The dead end is detected much earlier in the many sorted logic

3. A Brief Description of LLAMA***

The many sorted logic LIJAMA [4,3] is unusual in that the quantifiers are unsorted; the restriction on the range of a quantified variable derives from the argument positions of the function and predicate symbols that it occupies, associated with every non-logical symbol is a *sorting function a* of the same arity which describes how its sort varies with the sorts of its inputs; polymorphic functions and predicates are thus easily expressible and statements usually requiring several assertions may be compactly expressed by a single assertion. The sort structure itself is a complete boolean lattice. The top (T) element is interpreted as the universe of discourse and the bottom (JL) is interpreted as the empty set. Expressions of sort _L do not denote anything and are thus nonsense; they are said to be ill*sorted.* Sorts may be referred to either directly or with an expression containing least upper bound (U), greatest lower bound (n) or relative complement (\) operators.

*** Logic Lacking A Meaningful Acronym. Having sought a suitable name for the logic for a long time, I am indebted to Graeme Ritchie for this suggestion.

Furthermore, by specifying the result sort of predicates to be one of four special boolean sorts TT, *FF.* UU, EE (representing 'true', 'false', 'either true or false', and 'nonsense'), it is sometimes possible to detect that a formula is contradictory or tautologous without resort to general inference rules Expressiveness can be further improved by allowing the sort of a term to be a more general sort than the sort of the argument position it occupies However this feature is not needed for the current problem.

Associated with every formula in the logic is a *Sort Array (SA)* which is a mapping from sort environments to boolean sorts. A sort environment is a mapping from variables to sorts Thus a SA records what the sort of a formula is, depending on what sorts its constituent variables are regarded as taking A good way to view a SA is as an n dimensional array where n is the number of variables in the formula. Each dimension is indexed by the different sorts and each position in the array will contain one of the four boolean sorts *FF,* TT, UU or FIE. If all entries are EE then the formula is nonsense (or illsorted) and can be deleted. Entries which are TT will be ignored by the system (effectively treated as though they were EE) since they cannot lead to a refutation In the case of a clausal logic, if any of the entries are FF then the formula is a contradiction since all variables are universally quantified.

Inference in the logic includes ordinary resolution but there are some new inference rules In particular it is sometimes possible to *evaluate* literals because they are always of sort FF in the possible environments of the SA of the clause Evaluated literals may be deleted without having to resolve them away It is sometimes advantageous to restrict the SA of a clause (le restrict the set of possible sort environments in the SA, by changing some of the entries in the SA to EE) in order to evaluate a literal. Examples of such inferences will be found later on in the paper

4. AD Axiomatisation in LIAMA

We use all the sorts of Walther's axiomatisation and some additional ones As will be discussed later, the sort CUS allows the axiomatisation to be one clause smaller than it would otherwise be, and the sort P\G is added because, since G is a strict subsort**** of P, there must be a sort which is interpreted as all those plants which are not. grains The use of P\G will also be discussed later. Part of the hierarchy is depicted in figure I

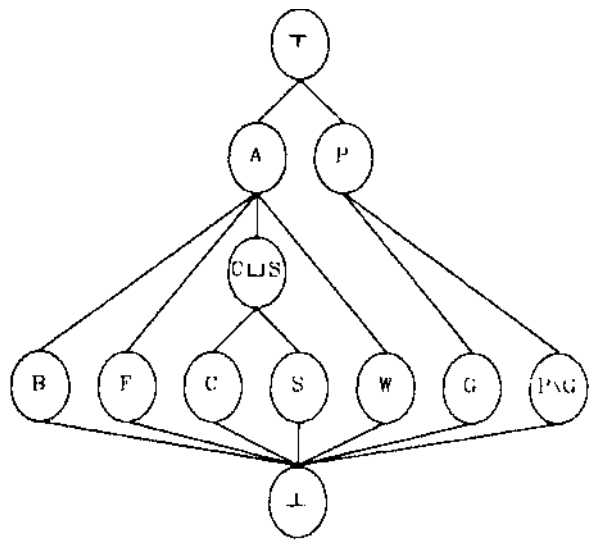


Figure 1.

Although LLAMA actually requires the sort lattice to be a complete boolean lattice with (in this case) 2^7 sorts, for simplicity only those needed to define the sorting functions for the non logical symbols are shown. Six further sorts (A\W, A\W\B, A\W\F, A\B, A\C and A\W\S) are used during the proof (see clauses 1 and 3 below)

The sorting functions for the various non-logical symbols (including skolem constants) are as follows Entries not given are assumed to be EE (ie undefined) unless otherwise inferable

$\ddot{M}(\langle A,A\rangle)$	=	UU	$\ddot{E}(\langle A,T\rangle)$	=	UU
$\ddot{M}(\langle C,B\rangle)$	=	TT	$\ddot{E}(\langle W,G\rangle)$	=	FF
$\ddot{M}(\langle S,B\rangle)$	=	TT	$\ddot{E}(\langle W,P\rangle)$	=	FF
$\ddot{M}(\langle B,F\rangle)$	=	TT	$\ddot{E}(\langle B,S\rangle)$	=	FF
$\ddot{M}(\langle F,W\rangle)$	=	TT	$\ddot{E}(\langle B,C\rangle)$	=	TT

$\ddot{h}(\langle CUS\rangle)$ = P $\quad$ $\ddot{j}(\langle A,A\rangle)$ = G

The sorting function for a *characteristic* (or sort) predicate τ is

$\ddot{\tau}(\langle\tau\rangle)$ = TT $\quad$ $\ddot{\tau}(\langle T\backslash\tau\rangle)$ = FF

Eg, the sorting function for the predicate P is:

$\ddot{P}(\langle P\rangle)$ = TT $\quad$ $\ddot{P}(\langle A\rangle)$ = FF

Note how the sorting functions for both M and E make use of LLAMA's polymorphism to encode much of the information about M and E. This type of polymorphism has been called *adhoc* polymorphism by Strachey [11] (or *overloading*) and should be distinguished from the *parametric* polymorphism to be found in, for example, [9].

**** Actually it does not follow that G is a strict subsort of P if the sort structure is derived from the unsorted axiomatisation. although it could be argued that the original English statement of the problem does implicitly imply this is in fact so. Walther's axiomatisation also specifies G to be a strict subsort of P. In any case this detail is not crucial to the problem

It should also be noted that in LLAMA the sorts B, F, C, S, W, G, and P\G are all *disjoint:* their interpretations are non-overlapping sets (this is because, as currently formulated, LLAMA requires complete knowledge about the sort structure). Again, this disjointness is not present in the unsorted axiomatisation nor is it present in Walther's formulation, nor is it stated explicitly in the English statement of the problem, although it is true in the real world. However, this disjointness information is only used to reduce the search space and will not be used in the proof itself since there are no positive sort literals. It may be noted that Shekel's solution of the steamroller which is discussed later also assumes the disjointness of these sorts

The clausal form of the axioms comes out as just the following three clauses. The sort environments for which the corresponding SA is not EE or TT are listed next to each clause (the first column gives the result sort). Environments for which a clause is TT can be ignored during a refutation because the clause is tautologous in these environments and thus they can be deleted for the same reason as ordinary tautologous clauses can. These SAs are not input with the axioms but are derived using the sorting functions by an algorithm to be found in *[4]*

(1) { ¬P(*p1*), ¬P(*p2*),E(*a1*,*p1*), M(*a2*,*a1*), ¬E(*a2*,*p2*),E(*a1*,*a2*)}

	a1	*a2*	*p1*	*p2*
UU	A\W\B	A	P	P
UU	A\W	A\C	P	P
UU	A\B	A	P	P\G
UU	A	A\C	P	P\G

(2) {E(*cs1*,h(*cs1*))}

	cs1
UU	C|S

(3) { ¬E(*a1*,*a2*), ¬E(*a2*,j(*a1*,*a2*))}

	a1	*a2*
UU	A\W\B	A\W
UU	W	A\W\F
UU	B	A\W\S

These axioms are derived* from the unsorted axioms 4u, 11u and the negation of 1 2u respectively. All the other unsorted axioms have been subsumed by the sort lattice and the sorting functions.

Thus this axiomatisation has just 3 clauses and 9 literals! There are only 4 possible resolvents initially There are also some other possible inferences initially because certain literals are evaluable In particular literals 1(1), 1(2), 1(4) can only be evaluated since there are no positive occurrences of P or M. (In fact characteristic literals such as *P(p1)* may always be evaluated immediately without losing completeness). Thus 1 could be preprocessed to.

(1') {E(*a1*,*p1*), ¬E(*a2*,*p2*),E(*a1*,*a2*)}

	a1	*a2*	*p1*	*p2*
UU	B	S	P	P
UU	F	B	P	P
UU	W	F	P	P\G

Thus there are now only 6 literals overall. There are four possible resolvents and three possible evaluations (on l'(l). l'(3). and 3(1)) in this modified input clause set. None of the three clauses can be factored

* It should be quite possible to derive this axiomatisation (including the sorting functions) automatically from Walther's formulation or perhaps-even from the unsorted formulation.

A proof** goes like this.

(4) {E(*a1*,*p1*), ¬E(*a2*,*p2*), ¬E(*a2*,j(*a1*,*a2*))}
1'(3)+3(1)

	a1	*a2*	*p1*	*p2*
UU	F	B	P	P

(5) {E(*a1*,*p1*), E(*a2*,j(*a1*,*a2*))}
factor of 4

	a1	*a2*	*p1*
UU	F	B	P

(6) {E(*a3*,*p1*), ¬E(*a2*,*p2*),E(*a1*,*a2*)}
5(2)+1'(1)

	a1	*a2*	*a3*	*p1*
UU	B	S	F	P

Since literal 6(3) is always false in the environments of clause 6 it can be evaluated and deleted from the clause without changing the SA or the meaning of the clause to produce a simpler clause 6';

(6') {E(*a3*,*p1*), ¬E(*a2*,*p2*)}

	a2	*a3*	*p1*
UU	S	F	P

(7) {E(*a3*,*p1*)}
6'(2)+2(:)

	a3	*p1*
UU	F	P

(8) {E(*a1*,*p1*),E(*a1*,*a2*)}
7(:)+1'(2)

	a1	*a2*	*p1*
UU	W	F	P\G
FF	W	F	G

We have now derived a clause with an environment for which it is *FF* le false. Since variables are universally quantified such a clause is a contradiction and we have our desired refutation, In an ordinary clausal logic, only a null clause indicates a falsehood but in this calculus it is possible for a non null clause to be contradictory by virtue of sortal information.

Note that this proof is linear, neither the unsorted hand produced proof of Schubert nor the many sorted proof of Walther are linear. Linearising a non-linear proof often increases its length. The total number number of inference steps (not counting preprocessing and simplifications) is only 5 steps. This compares with 10 steps in [14] The inclusion of the extra sort CUS

** This proof is hand produced However, an implementation of the logic will be finished in the near future.

saves one unit clause in the axiornafisation of the problem, this reduces the' search space slightly (the MKRP system could also take advantage of this) but dues not reduce the proof length since (23w) is not used in the proof

Part of the saving conies from the ability to represent information such as "caterpillars are much smaller than birds" not as unit clauses as Walther lias to (14 w) but as sorting functions Because this information can be represented without using a clause a reduction in the search space occurs The sorting functions for M and E are then used to advantage during the proof process to evaluate certain literals Evaluations which can be perfornied without having to restrict the SA are entirely 'free' (They are called elementary *a valuations* in I4I) They do not increase the search space because the old clause can be deleted An example is the evaluation perfornied in producing clause 6' Also, it has already been argued that it is beneficial to evaluate pure (i e not resolvable) literals as soon as possible, as was done in producing clause 2 ' However even a non elementary evaluation may not actually increase the search space since the parent clause may he restricted to exclude the sort environment used to produce the evaluation without losing completeness No non elementary evaluations were performed in the proof above though they might be used in a different branch of the search space (for example we have already noted above that three non elementary evaluations are possible initially (1.1). (3) and 3(1)) '

Obviously the overhead of computing and maintaining the SAs for the clauses has to be taken into consideration but this should be overshadowed by the reduction of the search space which accrues through use of the many sorted logic in most cases the cost of the SA operations appears to be some polynomial*** of the number of sorts and the number of variables in a clause, whilst the size of the search space is in general exponential in the number of clauses. The occasions when savings arc unlikely to be realised arc when there is little sortall structure in the problem In this ease the SAs will be very large (i e will be UU in most environments), few evaluations will be possible and inference will be no better than in an unsorted logic with the additional burden of having to maintain the SAs.

A second reason why the 1,1AW A proof is shorter is because of its polymorphism Inspection of Walther's proof (or indeed of the unsorted proof) shows that a literal with predicate symbol M is resolved away three times during the proof (in producing clauses 30 w, 32 w and 36w) Each time a different unit clause is involved These steps are all combined in the evaluation of the polymorphic clause (1) to produce clause ('1'). The important point is that (1 ') is still polymorphic**** The literal involving M has been deleted but without having to make such a commitment as to the precise sorts of the variables as in a non-polymorphic logic In a non-polymorphic many sorted logic once that literal has been resolved away then a commitment is made as to which precise sorts the variables involved in the resolution should be For example, (30w) is a result of a resolution involving M. But now al and *as* have been irrevocably chosen to be of sort Wand I' respectively.

The point is of course that instead of choosing to resolve (29w) against (1 7w) an inference engine might just as well have chosen any of the other three unit clauses involving M, for example (14 w) In this case *al* and a2 would now be of sort B and C respectively However this resolution cannot lead to a refutation (which is why the environment <B.C,P,P> does not appear in the hA for (!'): it is tautologous and thus will not be used in a refutation) Thus any further inference involving this resolvent would be totally wasted The advantage of a polymorphic logic thus is that .generality is retained and choices involving the precise sorts of variables can be delayed. This argument would hold even if M had been resolved away conventionally rather than by evaluation ft would also hold even if the environment <B,C,P,P> were still in the SA for (12) because in the LLAMA proof the decision about the sorts of a; and *a.2* is made as needed when resolving to produce clauses 4, 6 and 8.

Thus adding polymorphism to a many sorted logic in this way does seem to add genuine extra power One could imagine that a logic such as Walther's could perhaps be simulated with an unsorted logic by running the sort axioms intelligently (eg never resolving on an uninstantiated characteristic literal, but choosing instantiated characteristic literals as top priority when available and also cheeking characteristic literals in the same clause for inconsistency by testing whether the clause is subsumed by any sort, lattice axiom) but it is not obvious how LLAMA could be directly simulated in such a manner with a clausal unsorted logic.

5. Final Remarks

Some of the statistics appertaining to the three axiomatisations of the Steamroller are summarised in the table of figure 2.

This example demonstrates the advantage not only of using a many sorted logic, but also the value of polymorphism* and of using the four boolean sorts for describing the sortal behaviour of predicates

Schubert's Steamroller has also recently been solved by two other automated reasoning systems, in

	Unsorted logic	Walther's logic	LLAMA
No of clauses initially	27	12	3
No. of literals initially	65	16	9
No of possible inferences initially	102	12	7
length of proof	33	10	5

Figure 2

*** Analysing the complexity of the SA operations is an area for further research

♦♦** Note that I' would have to be represented as three clauses in Walther's logic, each one corresponding to one of the sort environments of 1 '. Thus polymorphism can allow one to obtain some "non clausal" effects

* Note that the implicit restriction on the quantification on variables from the arguments places they occur in is essential to exploiting LLAMA's polymorphism since variables with unique sorts exclude the possibility of polymorphic clauses such as 1'

(2) constant symbol C_k, for each $k \in K$; (3) simple variables $x_1, \cdots, x_m, \cdots$, and a.v.'s $x_1^{\Sigma P_i}, \cdots, x_n^{\Sigma P_i}, \cdots$, for each $i \in I$, where P_i is a unary predicate symbol; (4) a $\lambda(j)$-ary predicate symbol R_j, for each $j \in J$; (5) a $\xi(l)$-ary function symbol F_l, for each $l \in L$; (6) logical connectives $\neg$, and $\rightarrow$; and (7) a universal quantifier $\forall$.

Based on this language, the *terms* of L_Σ^1 are defined as usual except that each variable is now either a simple variable or an a.v.. The set, $Form(L_\Sigma^1)$, of *well-formed formulas* of L_Σ^1, is defined as usual. The definable syntactic objects $\cup$, $\cap$, $\subseteq$ and $\exists$, and the standard notions such as *sentences* are also introduced in the usual way.

2.2. Interpretation of L_Σ^1

A structure is needed to interpret each formula in L_Σ^1. Let OS_a be a structure for L_Σ^1. Then $OS_a = < \Omega, \{P_i\}_{i\in I}, \{R_j\}_{j\in J}, \{F_l\}_{l\in L}, \{C_k\}_{k\in K} >$* where Ω is the universe of OS_a; P_i is a unary relation $P_i \subseteq \Omega$; R_j is a $\lambda(j)$-ary relation $R_j \subseteq \Omega^{\lambda(j)}$; F_l is a $\xi(l)$-ary function $F_l : \Omega^{\xi(l)} \rightarrow \Omega$; and a distinguished element C_k is an element of Ω, for each $k \in K$. The interpretation of a formula in the structure OS_a then requires a variable assignment function s as follows:

Definition 2.2 For a set V of variables of L_Σ^1 and the universe Ω of structure OS_a, s is an assignment function $s : V \rightarrow \Omega$ such that for a simple variable x, $s(x) = a$, where $a \in \Omega$; and for an aggregate variable $x^{\Sigma P_i}$, $s(x^{\Sigma P_i}) = a$, where $a \in P_i$.

Assignment function for the terms of L_Σ^1 is defined as usual. For notational convenience symbol s is also used for the assignment for the terms. The validity of each formula is determined by the following interpretation rules.

Definition 2.3 For $R_j(t_0, \ldots, t_{\lambda(j)})$, $\psi_1, \psi_2 \in Form(L_\Sigma^1)$, where t_i's are terms, the satisfaction of the formulas with respect to s in OS_a is defined by,

(1) $\models_{OS_a} R_j(t_0, \ldots, t_{\lambda(j)})[s]$ iff $< s(t_0), \ldots, s(t_{\lambda(j)}) > \in R_j$,

(2) $\models_{OS_a} \neg \psi_1[s]$ iff $\not\models_{OS_a} \psi_1[s]$,

(3) $\models_{OS_a} \psi_1 \rightarrow \psi_2[s]$ iff if $\models_{OS_a} \psi_1[s]$ then $\models_{OS_a} \psi_2[s]$,

(4) For a simple variable x, $\models_{OS_a} \forall x\ \psi_1[s]$ iff for any $a \in \Omega$, $\models_{OS_a} \psi_1[s\{x \mid a\}]$,

(5) For an aggregate variable $x^{\Sigma P_i}$, $\models_{OS_a} \forall x^{\Sigma P_i} \psi_1[s]$ iff for any $a \in P_i$, $\models_{OS_a} \psi_1[s\{x^{\Sigma P_i} \mid a\}]$,

where for variables v_m and v_k,

$$s(v_m \mid a)(v_k) = \begin{cases} s(v_k) & \text{if } v_m \neq v_k \\ a & \text{if } v_m = v_k \end{cases}.$$

As a corollary to the definition, the interpretations of $\cup$, $\cap$, $\subseteq$ and $\exists$ can also be easily defined.

2.3. Σ-Extensibility of L_Σ^1

L_Σ^1 is as convenient as L_m in abbreviating the relativized expressions in a one-sorted language into more compact forms. If a formula σ_m in L_m is of the form

$$\sigma_m = \forall x_i \ \_\_\_\_\_\ x_i \ \_\_\_\_\_ ,$$

where x_i is a sort variable belonging to some sort S_i, then σ_m can be syntactically translated into σ_Σ^t in L_Σ^1

$$\sigma_\Sigma^t = \forall x^{\Sigma P_i} \ \_\_\_\_\_\ x^{\Sigma P_i} \ \_\_\_\_\_ ,$$

where the unary relation intended by P_i is identical to the sort S_i. During the translation, L_Σ^1 is augmented with P_i and accordingly the structure for L_Σ^1, say $OS_a(L_\Sigma^1)^t$, can be constructed from the many-sorted structure for L_m, say $MS(L_m)$. The following theorem can be shown (proofs for the theorems given in this paper can be found in [Shin, 1985]):

Theorem 2.1 A sentence σ_m in L_m is true in $MS(L_m)$ iff σ_Σ^t in L_Σ^1 is true in $OS_a(L_\Sigma^1)^t$.

The power of L_Σ^1 over L_m lies in the fact that whereas in the latter a sort variable with new sort may not be introduced, in the former *a variable whose range is restricted to a subset of the universe Ω can be introduced as needed in its extension.* Let T_Σ be a theory in L_Σ^1 and let a formula $\phi_\Sigma \in T_\Sigma$ be of the form

$$\phi_\Sigma = \forall x\ (S_1(x) \cap S_2(x) \rightarrow \psi(x)).$$

In L_Σ^1, in order to introduce a variable ranging over $S_1 \cap S_2$, all that must be done is to add a new unary predicate symbol S_k to L_Σ^1, abbreviate ϕ_Σ by $\forall x^{\Sigma S_k}\ \psi(x^{\Sigma S_k})$, and augment T_Σ by the defining axiom $\forall x\ (S_k(x) \subseteq S_1(x) \cap S_2(x))$. The extended language $L_\Sigma^{1\prime}$ is called a Σ-*extension* of L_Σ^1 and the augmented theory T_Σ', a Σ-*extension* of T_Σ. For the semantics of the new predicate symbols in $L_\Sigma^{1\prime}$, it can be shown that there is a unique expansion by definition of OS_a, say OS_a', which is a model of T_Σ'. OS_a' is called *an expansion by Σ definition* of OS_a. This characteristic of L_Σ^1 is called Σ-*extensibility*. The following theorem establishes the validity of Σ extensibility of L_Σ^1:

Theorem 2.2 For any $\psi' \in T_\Sigma'$, there is a $\psi \in T_\Sigma$ such that for any assignment function s, $\models_{OS_a'} \psi'[s] \iff \models_{OS_a} \psi[s]$.

3. UWR-resolution

The problem identified in Section 1, namely, the generation of useless resolvents that lead to dead ends, occurs only when a many-sorted theory of a certain class is refuted by a resolution scheme. For instance, for the many-sorted theories with tree structure stated in [Walther, 1984a], this problem never occurs. When the tree constraint is lifted, however, this problem may appear. The conditions under which the problem may arise are explained below, this time in terms of L_Σ^1.

Let $Ran(t)$ stand for the range associated with a term t. Given a set of unary relations P, let a set of immediate predecessor of a relation $P_i \in P$, denoted by $IM(P_i)$, be defined by $IM(P_i) \stackrel{d}{=} \{P_j \mid P_j \in P,\ P_i \subset P_j,$ and if $P_j \subseteq P_k \subseteq P_i$ then either $P_j = P_k$ or $P_k = P_i\}$. A situation may occur in which two variables v_i and v_j have $|IM(Ran(v_i)) \cap IM(Ran(v_j))| > 1$. Hence there are possibly more than one range over which v_i can be unified with v_j. In fact, if x_k is a variable such that $Ran(x_k) = P_k$ and $P_k \in IM(Ran(v_i)) \cap IM(Ran(v_j))$, then any substitution $\theta = \{x_k/v_i, x_k/v_j\}$ is a legitimate mgu of $\{v_i, v_j\}$, since $\theta v_i = \theta v_j$. There are, therefore, as many mgus for

* From next section on, "Σ" on a symbol is omitted as long as the meaning of the symbol is unambiguous.

$\{ v_i , v_j \}$ as $| IM(Ran(v_i)) \cap IM(Ran(v_j)) |$. Not all the resolvents generated using each of these mgus are useful.

A way to remedy the situation is proposed in the following. First, a few new notions are introduced. A ***wr-substitution component*** is any expression of the form t/v, where v is a variable and t is a term different from v satisfying $Ran(t) \subseteq Ran(v)$. A ***wr-subpair*** is a set of wr-substitution components $\{t/v_i , t/v_j\}$ satisfying, (1) $Ran(v_i) \not\subseteq Ran(v_j)$ and $Ran(v_j) \not\subseteq Ran(v_i)$, (2) $| IM(Ran(v_i)) \cap IM(Ran(v_j)) | > 1$, and (3) $Ran(t) = Ran(v_i) \cap Ran(v_j)$. A ***wr-substitution*** is a set of wr-substitution components which possibly contains one or more wr-subpairs. A ***wr-resolvent*** is a resolvent which is generated by using a wr-substitution as a unifier.

The Σ-extensibility of L_Σ^1 plays the central role in introducing wr-subpairs. From the definition of a wr-subpair, it is clear that wr-resolvents can only be expressed in an extended language of L_Σ^1. To be more specific, let a theory in L_Σ^1 be formalized by an ordered pair $< OA , T_\Sigma >$ where OA is a set of ordering axioms expressed with the symbol " $\subset$ " such that $P_i \subset P_j$ means $\forall x (P_i(x) \rightarrow P_j(x))$ and T_Σ is a set of the nonlogical axioms of the theory expressed in L_Σ^1. The following is an example:

Example 3.1 Consider the following $< OA , T_\Sigma >$:

OA : (1) $D \subset B$, $D \subset C$

(2) $E \subset B$, $E \subset C$

T_Σ : (3) $\forall x^{\Sigma B} (P(x^{\Sigma B}) \cup Q(x^{\Sigma B} , f^4(x^{\Sigma B})))$

(4) $\forall x^{\Sigma C} \neg P(x^{\Sigma C})$

(5) $\forall x^{\Sigma B} \forall x^{\Sigma C} \neg Q(x^{\Sigma B} , x^{\Sigma C})$

The following is done: For $x^{\Sigma B}$ of $P(x^{\Sigma B})$ in (3) and $x^{\Sigma C}$ of $\neg P(x^{\Sigma C})$ in (4), a wr-subpair $\{ x^{\Sigma K}/x^{\Sigma B} , x^{\Sigma K}/x^{\Sigma C} \}$ is introduced with L_Σ^1 being extended by a unary predicate symbol, say K, where $\forall x (K(x) \leftrightarrows B(x) \cap C(x))$. The extension of L_Σ^1 requires $< OA , T_\Sigma >$ to be also extended, i.e., OA is augmented to OA^+ by the ordering axioms such as (2^+) below. (6) is derived as the wr-resolvent of (3) and (4). Finally, (6) is resolved with (5) resulting in $\square$ as follows:

(2^+) $K \subset B$, $K \subset C$, $D \subset K$, $E \subset K$

(6) $Q(x^{\Sigma K} , f^4(x^{\Sigma K}))$ (3)+(4)

(7) $\square$ (5)+(6)

It is not difficult to see that the range of t in a wr-subpair $\{t/v_i , t/v_j\}$, i.e., $Ran(t) = Ran(v_i) \cap Ran(v_j)$, is the weakest range over which $\{v_i , v_j\}$ can be unified -- weakest in the sense that if $P_w = Ran(t)$, then there is no P_i such that $P_w \subset P_i$ and v_i and v_j are still unifiable over P_i. For this reason, the unification stated above is called **unification over the weakest range** and the resolution involving such unification is called **UWR-resolution.** The idea behind UWR-resolution is therefore to subsume all the possible unifications by one unification over the weakest possible range.

The completeness of UWR-resolution must be proved. Given $< OA , T_\Sigma >$, let $R_W(T_\Sigma)$ be the set of all clauses consisting of members of T_Σ and the resolvents (including wr-resolvents) of members of T_Σ (similar to Robinson's resolution operator [Robinson, 1965]). Also let $R_W^n(T_\Sigma)$ be defined so that for each $n \geq 0$, $R_W^0(T_\Sigma) = T_\Sigma$ and $R_W^{n+1}(T_\Sigma) = R_W(R_W^n(T_\Sigma))$. Completeness theorem states:

Theorem 3.1 A $< OA , T_\Sigma >$ is unsatisfiable if and only if $R_W^n(T_\Sigma)$ contains $\square$, for some $n \geq 0$.

In fact, when the UWR-resolution is applied, $< OA , T_\Sigma >$ is refuted not in L_Σ^1 but in its extended language $L_\Sigma^{1'}$. Therefore it further must be justified that when $R_W^n(T_\Sigma)$ contains $\square$, for some $n \geq 0$, whether the unsatisfiability of $< OA , T_\Sigma >$ can be truly shown in L_Σ^1 without extending its vocabulary. Let $R_\Sigma(\cdot)$ stand for an identical situation as $R_W(\cdot)$ except that in the former no wr-resolvents are generated. In this way the unsatisfiability of $< OA , T_\Sigma >$ can be shown without extending L_Σ^1. The following theorem can be shown:

Theorem 3.2 Given $< OA , T_\Sigma >$, $R_W^n(T_\Sigma)$ contains $\square$, for some $n \geq 0$, if and only if $R_\Sigma^m(T_\Sigma)$ contains $\square$, for some $m \geq 0$.

4. Efficiency of UWR-resolution

One way to discuss the efficiency of UWR-resolution is to compare $R_W(\cdot)$ and $R_\Sigma(\cdot)$ defined previously. The following facts show the efficiency of UWR-resolution:

Lemma 4.1 Given $< OA , T_\Sigma >$, if n is the smallest non-negative integer for which $R_W^n(T_\Sigma)$ contains $\square$ and m is the smallest non-negative integer for which $R_\Sigma^m(T_\Sigma)$ contains $\square$, then $n = m$.

Lemma 4.2 Given $< OA , T_\Sigma >$, for each $i \geq 0$ $| R_W^{i+1}(T_\Sigma) - R_W^i(T_\Sigma) | \leq | R_\Sigma^{i+1}(T_\Sigma) - R_\Sigma^i(T_\Sigma) |$.

Finally, the following theorem can be proved:

Theorem 4.3 Given $< OA , T_\Sigma >$, if n is the smallest non-negative integer for which $R_W^n(T_\Sigma)$ and $R_\Sigma^n(T_\Sigma)$ both contain $\square$, then $| R_W^n(T_\Sigma) | \leq | R_\Sigma^n(T_\Sigma) |$.

5. Conclusion

The use of aggregate variables have been demonstrated in a new approach, namely, UWR-resolution, for proving theorems. It has been shown how such an approach avoids the problems encountered in the use of a many-sorted language.

REFERENCES

[1] Cohn, A. G. "Improving the Expressiveness of Many-Sorted Logic", *Proc. of the 3rd NCAI*, Washington, 1983.

[2] Henschen, L. J. "N-Sorted Logic for Automatic Theorem Proving in Higher-Order Logic", *Proc. of ACM Conference*, Boston, 1972.

[3] Robinson, J. A. "A Machine-Oriented Logic Based on the Resolution Principle", *JACM*, Vol. 12, No. 1, 1965.

[4] Shin, D. G. "An Extension of a First-Order Language and Its Applications" Ph.D. Dissertation, University of Michigan, Ann Arbor, 1985.

[5] Shin, D. G. and Irani, K. B. "Knowledge Representation using an Extension of a Many-sorted Language", *Proc. of the 1st CAIA*, Denver, 1984.

[6] Walther, C. "A Many-sorted Calculus Based on Resolution and Paramodulation", *Proc. of the 8th IJCAI*, Karlsruhe, 1983.

[7] Walther, C. "Unification in Many-sorted Theories", *Proc. of the 6th ECAI*, Pisa, 1984a.

[8] Walther, C. "A Mechanical Solution of Schubert's Streamroller by Many-Sorted Resolution", *Proc. of the 4th NCAI*, Austin, 1984b.