

On the Solution of *Schubert's Steamroller* in Many Sorted Logic.

Anthony G Cohn

Department of Computer Science
University of Warwick
Coventry CV4 7AL
UK

ABSTRACT

A challenge problem for automated theorem provers posed in 1978 by Schubert has recently been solved with the Karlsruhe MKRP-system by formulating the problem in many sorted logic. It is shown here that a further improvement is possible by using a more sophisticated, *polymorphic* many sorted logic. The problem and its solution are discussed and the differences between the different formulations are analysed.

1. Introduction

In 1978 Schubert set up the following challenge problem:

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants. Therefore there is an animal that likes to eat a grain-eating animal.

An axiomatisation of this problem as given in [13] is as follows:

- (1u) $\exists w, f, b, c, s, g (W(w) \wedge F(f) \wedge B(b) \wedge C(c) \wedge S(s) \wedge G(g))$
- (2u) $\forall x ((W(x) \vee F(x) \vee B(x) \vee C(x) \vee S(x)) \rightarrow A(x))$
- (3u) $\forall x (G(x) \rightarrow P(x))$
- (4u) $\forall x (A(x) \rightarrow$
 $(\forall y (P(y) \rightarrow E(x, y)) \vee$
 $\forall y ((A(y) \wedge M(y, x) \wedge \exists z (P(z) \wedge E(y, z))) \rightarrow E(x, y))))$
- (5u) $\forall x, y (((C(x) \vee S(x)) \wedge B(y)) \rightarrow M(x, y))$
- (6u) $\forall x, y ((B(x) \wedge F(y)) \rightarrow M(x, y))$
- (7u) $\forall x, y ((F(x) \wedge W(y)) \rightarrow M(x, y))$
- (8u) $\forall x, y (((F(x) \vee G(x)) \wedge W(y)) \rightarrow \neg E(y, x))$
- (9u) $\forall x, y ((B(x) \wedge C(y)) \rightarrow E(x, y))$
- (10u) $\forall x, y ((B(x) \wedge S(y)) \rightarrow \neg E(x, y))$
- (11u) $\forall x ((C(x) \vee S(x)) \rightarrow \exists y (P(y) \wedge E(x, y)))$
- (12u) $\exists x, y (A(x) \wedge A(y) \wedge \forall z (G(z) \rightarrow (E(x, y) \wedge E(y, z))))$

* This work was partially funded by the SERC under grant GR/C/65148.

The clausal form contains 27 clauses with 65 literals and until very recently no computer generated refutation had been generated** even though a hand built proof has been found: the search space is extremely large; there are 94 potential initial resolvents and 8 potential factors and the hand computed solution requires 33 inference steps. The clausal form of the above axiomatisation is given below.

- (1cu) $\{W(w)\}$
- (2cu) $\{F(f)\}$
- (3cu) $\{B(b)\}$
- (4cu) $\{C(c)\}$
- (5cu) $\{S(s)\}$
- (6cu) $\{G(g)\}$
- (7cu) $\{\neg W(x), A(x)\}$
- (8cu) $\{\neg F(x), A(x)\}$
- (9cu) $\{\neg B(x), A(x)\}$
- (10cu) $\{\neg C(x), A(x)\}$
- (11cu) $\{\neg S(x), A(x)\}$
- (12cu) $\{\neg G(x), P(x)\}$
- (13cu) $\{\neg A(x), \neg P(y), E(x, y), \neg A(w), \neg M(w, x), \neg P(z),$
 $\neg E(w, z), E(x, w)\}$
- (14cu) $\{\neg C(x), \neg B(y), M(x, y)\}$
- (15cu) $\{\neg S(x), \neg B(y), M(x, y)\}$
- (16cu) $\{\neg B(x), \neg F(y), M(x, y)\}$
- (17cu) $\{\neg F(x), \neg W(y), M(x, y)\}$
- (18cu) $\{\neg F(x), \neg W(y), \neg E(y, x)\}$
- (19cu) $\{\neg G(x), \neg W(y), \neg E(y, x)\}$
- (20cu) $\{\neg B(x), \neg C(y), E(x, y)\}$
- (21cu) $\{\neg B(x), \neg S(y), \neg E(x, y)\}$
- (22cu) $\{\neg C(x), P(h(x))\}$
- (23cu) $\{\neg C(x), E(x, h(x))\}$
- (24cu) $\{\neg S(x), P(i(x))\}$
- (25cu) $\{\neg S(x), E(x, i(x))\}$
- (26cu) $\{\neg A(x), \neg A(y), G(j(x, y))\}$
- (27cu) $\{\neg A(x), \neg A(y), \neg E(x, y), \neg E(y, j(x, y))\}$

(1cu) to (6cu) come from (1u). (7cu) through to (12cu) come from (2u) and (3u). (13cu) comes from (4u). (14cu) through to (17cu) come from (5u), (6u) and (7u). (18cu), (19cu) and (21cu) come from (8u) and (10u). (20cu) comes from (9u). (22cu) through (25cu) are the skolemised clauses derived from (11u). Similarly (26cu) and (27cu) come from the negation of (12u).

** These solutions will be described in the last section.

Recently, Walther, [13] 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. Walther's Axiomatisation

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 *type* 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) <i>type</i> w:W | (2w) <i>type</i> f:F |
| (3w) <i>type</i> b:B | (4w) <i>type</i> c:C |
| (5w) <i>type</i> s:S | (6w) <i>type</i> g:G |
| (7w) <i>sort</i> W $\sqsubset$ A | (8w) <i>sort</i> F $\sqsubset$ A |
| (9w) <i>sort</i> B $\sqsubset$ A | (10w) <i>sort</i> C $\sqsubset$ A |
| (11w) <i>sort</i> S $\sqsubset$ A | (12w) <i>sort</i> G $\sqsubset$ P |
| (13w) $\{E(a_1, p_1), \neg M(a_2, a_1), \neg E(a_2, p_2), E(a_1, a_2)\}$ | |
| (14w) $\{M(c_1, b_1)\}$ | (15w) $\{M(s_1, b_1)\}$ |
| (16w) $\{M(b_1, f_1)\}$ | (17w) $\{M(f_1, w_1)\}$ |
| (18w) $\{\neg E(w_1, f_1)\}$ | (19w) $\{\neg E(w_1, g_1)\}$ |
| (20w) $\{E(b_1, c_1)\}$ | (21w) $\{\neg E(b_1, s_1)\}$ |
| (22w) <i>type</i> h(C):P | (23w) $\{E(c_1, i(c_1))\}$ |
| (24w) <i>type</i> i(S):P | (25w) $\{E(s_1, h(s_1))\}$ |
| (26w) <i>type</i> j(A,A):G | |
| (27w) $\{\neg E(a_1, a_2), \neg E(a_2, j(a_1, a_2))\}$ | |

The symbols a_1, f_1 etc are all typed variables with the sort of the corresponding upper case letter. (1w) 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) $\{E(a_1, p_1), \neg M(a_2, a_1), \neg E(a_2, p_2), \neg E(a_2, j(a_1, a_2))\}$ |
| 13w(4)+27w(1) |
| (29w) $\{E(a_1, p_1), \neg M(a_2, a_1), \neg E(a_2, j(a_1, a_2))\}$ |
| factor of 28w |
| (30w) $\{E(w_1, p_1), \neg E(f_1, j(w_1, f_1))\}$ |
| 17w(1)+29w(2) |
| (31w) $\{\neg E(f_1, j(w_1, f_1))\}$ |
| 19w(1)+30w(1) |
| (32w) $\{E(f_1, p_1), \neg E(b_1, j(f_1, b_1))\}$ |
| 16w(1)+29w(2) |
| (33w) $\{\neg E(b_1, j(f_1, b_1))\}$ |
| 31w(1)+32w(1) |
| (34w) $\{E(b_2, p_1), \neg M(s_1, b_2), \neg E(s_1, p_2)\}$ |
| 13w(4)+21w(1) |
| (35w) $\{\neg M(s_1, b_2), \neg E(s_1, p_2)\}$ |
| 33w(1)+34w(1) |
| (36w) $\{\neg E(s_1, p_2)\}$ |
| 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 (21w) have no resolvent because c_1 and s_1 cannot be unified. In the unsorted case the two clauses (20cu) and (21cu) do resolve to yield $\{-B(x), \neg C(y), \neg S(y)\}$. This can then be resolved with (3cu) to yield $\{\neg C(y), \neg S(y)\}$. However further inference will now yield a pure clause: either $\{\neg S(c)\}$ or $\{\neg C(s)\}$, 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 LLAMA [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 α is a *sorting function* $\ddot{\alpha}$ 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 ($\top$) element is interpreted as the universe of discourse and the bottom ($\perp$) is interpreted as the empty set. Expressions of sort $\perp$ do not denote anything and are thus nonsense; they are said to be *unsorted*. Sorts may be referred to either directly or with an expression containing least upper bound ($\sqcup$), greatest lower bound ($\sqcap$) or relative complement ($\setminus$) 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 EE. 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 (ie 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. An Axiomatisation in LLAMA.

We use all the sorts of Walther's axiomatisation and some additional ones. As will be discussed later, the sort CLIS 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

**** 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 Stickel's solution of the steamroller which is discussed later also assumes the disjointness of these sorts.

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 1.

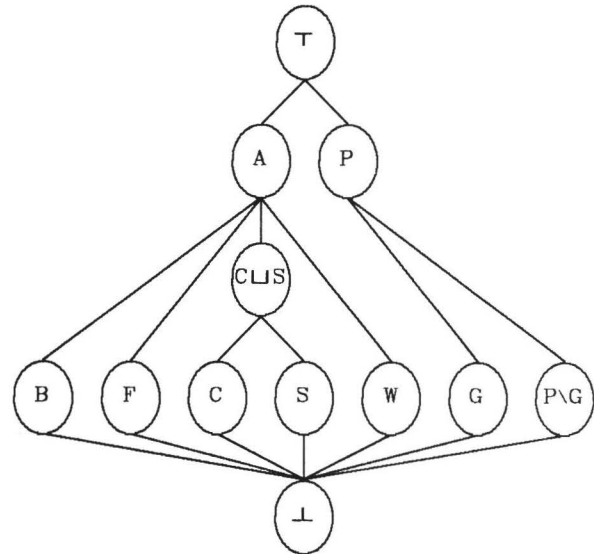


Figure 1.

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 ($A \setminus W$, $A \setminus W \setminus B$, $A \setminus W \setminus F$, $A \setminus B$, $A \setminus C$ and $A \setminus W \setminus S$) 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	$\ddot{E}(\langle W, F \rangle)$	= 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

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

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

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

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 $\ddot{M}$ and $\ddot{E}$ make use of LLAMA's polymorphism to encode much of the information about $\ddot{M}$ and $\ddot{E}$. This type of polymorphism has been called *ad hoc* polymorphism by Strachey [11] (or *overloading*) and should be distinguished from the *parametric* polymorphism to be found in, for example, [9].

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].

$$(1) \{ \neg P(p_1), \neg P(p_2), E(a_1, p_1), \neg M(a_2, a_1), \neg E(a_2, p_2), E(a_1, a_2) \}$$

	a_1	a_2	p_1	p_2
UU	$A \setminus W \setminus B$	A	P	P
UU	$A \setminus W$	$A \setminus C$	P	P
UU	$A \setminus B$	A	P	$P \setminus G$
UU	A	$A \setminus C$	P	$P \setminus G$

$$(2) \{ E(cs_1, h(cs_1)) \}$$

	cs_1
UU	$CL \setminus S$

$$(3) \{ \neg E(a_1, a_2), \neg E(a_2, j(a_1, a_2)) \}$$

	a_1	a_2
UU	$A \setminus W \setminus B$	$A \setminus W$
UU	W	$A \setminus W \setminus F$
UU	B	$A \setminus W \setminus S$

These axioms are derived* from the unsorted axioms 4u, 11u and the negation of 12u 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 $P(p_1)$ may always be evaluated immediately without losing completeness). Thus 1 could be preprocessed to:

$$(1') \{ E(a_1, p_1), \neg E(a_2, p_2), E(a_1, a_2) \}$$

	a_1	a_2	p_1	p_2
UU	B	S	P	P
UU	F	B	P	P
UU	W	F	P	$P \setminus G$

Thus there are now only 6 literals overall. There are four possible resolvents and three possible evaluations (on 1'(1), 1'(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.

$$(4) \{ E(a_1, p_1), \neg E(a_2, p_2), \neg E(a_2, j(a_1, a_2)) \}$$

$$1'(3) + 3(1)$$

	a_1	a_2	p_1	p_2
UU	F	B	P	P

$$(5) \{ E(a_1, p_1), \neg E(a_2, j(a_1, a_2)) \}$$

$$\text{factor of 4}$$

	a_1	a_2	p_1
UU	F	B	P

$$(6) \{ E(a_3, p_1), \neg E(a_2, p_2), E(a_1, a_2) \}$$

$$5(2) + 1'(1)$$

	a_1	a_2	a_3	p_1
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':

$$(6') \{ E(a_3, p_1), \neg E(a_2, p_2) \}$$

	a_2	a_3	p_1
UU	S	F	P

$$(7) \{ E(a_3, p_1) \}$$

$$6'(2) + 2(1)$$

	a_3	p_1
UU	F	P

$$(8) \{ E(a_1, p_1), E(a_1, a_2) \}$$

$$7(1) + 1'(2)$$

	a_1	a_2	p_1
UU	W	F	$P \setminus G$
FF	W	F	G

We have now derived a clause with an environment for which it is FF ie 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 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 CLJS

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

saves one unit clause in the axiomatisation of the problem; this reduces the search space slightly (the MKRP system could also take advantage of this) but does not reduce the proof length since (23w) is not used in the proof.

Part of the saving comes from the ability to represent information such as "caterpillars are much smaller than birds" not as unit clauses as Walther has to (14w) 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 performed without having to restrict the SA are entirely 'free'. (They are called *elementary evaluations* in [4]). They do not increase the search space because the old clause can be deleted. An example is the evaluation performed in producing clause 6'. Also, it has already been argued that it is beneficial to evaluate pure (ie not resolvable) literals as soon as possible, as was done in producing clause 1'. However even a non elementary evaluation may not actually increase the search space since the parent clause may be 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), 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 are unlikely to be realised are when there is little sortal structure in the problem. In this case the SAs will be very large (ie 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 LLAMA 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 30w, 32w 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 a_1 and a_2 have been irrevocably chosen to be of sort W and F respectively.

The point is of course that instead of choosing to resolve (29w) against (17w) an inference engine might just as well have chosen any of the other three unit clauses involving M, for example (14w). In this case a_1 and a_2 would now be of sort B and C respectively. However this resolution cannot lead to a refutation (which is why the environment $\langle B, C, P, P \rangle$ does not appear in the SA for (1'): 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. It would also hold even if the environment $\langle B, C, P, P \rangle$ were still in the SA for (1') because in the LLAMA proof the decision about the sorts of a_1 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 checking 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 1' 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'.

both cases special purpose reasoning systems were employed. The ITP system [7] has a system of "qualifier" literals which can be used to simulate a many sorted logic to a limited extent. The success of the ITP system on the Steamroller problem is reported in [10] but no details of the proof were given so no detailed comparison is possible. Stickel has also solved the steamroller with his non clausal logic which "builds in theories". *Theory resolution* removes the need to resolve upon axioms in the theory by utilising a decision procedure that can determine the unsatisfiability of a set of clauses using predicates in the theory. The statistics in the table of figure 3 show the results of Stickel's system for (a) no knowledge built in (b) 2u-3u built in (c) 2u-3u and 5u-11u built in**.

	(a)	(b)	(c)
No. of formulae initially	18	12	8
Length of proof	33	10	6
Time taken(h:mm)	2:53	0:20	0:01.6

Figure 3

These figures are taken directly from [10]. No statistics on the original size of the search space were given. The times refer to a system running on a Symbolics 3600. These results show a dramatic improvement in the size of the search space by building in axioms. A solution is found about 100 times faster with (c) compared to (a)! Thus although a proof can be found with no special purpose inference machinery for handling taxonomic information it is very expensive to compute***. However the proof length is still longer than in either of the many sorted solutions. Theory resolution can handle characteristic literals more efficiently than a standard unsorted logic but cannot eliminate them altogether as is possible in many sorted logic. However theory resolution is more general than a many sorted logic: for example paramodulation can be viewed as an instance of theory resolution.

In this problem, sorts have managed to reduce the amount of normal inference required to almost nothing, but of course in larger problems there will still be a non trivial amount of such inference required, for it is not claimed that a many sorted logic is a general panacea. However it does seem to be a useful weapon in attacking large search spaces.

6. Acknowledgements

My special thanks go to Christoph Walther for his helpful comments on a draft of this paper. I have also had useful discussions with Alan Frisch, Pat Hayes and Bob Kowalski.

** For comparison, LLAMA "builds in" 1u-3u and 5u-10u while Walther's many sorted formulation "builds in" 1u-3u.

*** Stickel also reports a personal communication from Walther that the MKRP system found an unsorted solution too. However this took about 10 times longer than with the sorted formulation of the problem and relied heavily on the TERMINATOR module [1] (a fast procedure for finding unit refutations).

7. References

- [1] G Antoniou and H J Ohlback, "TERMINATOR," in *Proc IJCAI 8*, Karlsruhe (1983).
- [2] D de Champeaux, "A Theorem Prover Dating a Semantic Network," in *Proc AISB/GI Conf on AI*, Hamburg (1978).
- [3] A G Cohn, "Improving the Expressiveness of Many Sorted Logic," in *Proc AAAI*, Washington DC (1983).
- [4] A G Cohn, "Mechanising a Particularly Expressive Many Sorted Logic," PhD Thesis, University of Essex (1983).
- [5] H B Enderton, *A Mathematical Introduction to Logic*, Academic Press (1972).
- [6] L J Henschen, "N-Sorted Logic for Automatic Theorem Proving in Higher Order Logic," in *Proc. ACM Conference*, Boston (1972).
- [7] E Lusk and R A Overbeek, "The Automated Reasoning System ITP," Technical Report ANL 84 27, Argonne National Laboratory (1984).
- [8] J R McSkimin and J Minker, "The Use of a Semantic Network in a Deductive Question Answering System," in *Proc IJCAI 5*, Cambridge (1977).
- [9] A Mycroft and R A O'Keefe, "A Polymorphic Type System for Prolog," *Artificial Intelligence* 23(1984).
- [10] M E Stickel, "Automated Deduction by Theory Resolution," Draft Report, SRI International (1984).
- [11] C Strachey, "Fundamental Concepts in Programming Languages," *Unpublished notes for a NATO Summer School*, (1967).
- [12] C Walther, "A Many Sorted Calculus Based on Resolution and Paramodulation," in *IJCAI 8*, Karlsruhe (1983).
- [13] C Walther, *Schubert's Steamroller - A Case Study in Many Sorted Resolution*, Universitat Karlsruhe (1984).
- [14] C Walther, "A Mechanical Solution of Schubert's Steamroller by Many Sorted Resolution," in *Proc AAAI*, Austin, Texas (1984).