# Reasoning about types in a general purpose inference engine

Allan Ramsay
Dept of Computation,UMIST, PO BOx 88, Manchester M60 1QD, UK

**Abstract**

Many reasoning problems involve a mixture of reasoning within some tractable logic of types and more general reasoning in a general purpose logic (standard first-order logic, modal logic, default logic, intensional logic, ...). One way to tackle this issue is by trying to extend the expressive power of the logic of types without compromising its tractability too badly, as exemplified by description logics. An alternative is to embed special purpose reasoning about types inside a general purpose inference engine. The current paper shows how to embed an efficient engine for a fairly expressive logic of types within a theorem prover which has been extended to cover reasoning about equality and to deal with intensionality.

## 1 Embedding a logic of types in a general inference engine

For many tasks, the choice between a simple logic of types and a more complex general purpose logic leaves you with an awkward dilemma. Logics of types are tractable, but inexpressive, so you can't use them because you can't describe your problem properly in them. Expressive logics are intractable, so you can't use them because you can't afford to wait indefinitely for the answer to your problem. What should you do?

There seem to be two potential routes out of this impasse:

- You can try to develop a logic that is more expressive than a simple logic of types, but that is still reasonably tractable. The development of description logics (Ohlbach and Koehler, 1997; Baader and Sattler, 2001) is an attempt to solve the problem this way, by increasing the expressive power as much as possible whilst retaining the complexity of propositional logic.
- You can take a logic that has the expressive power you need, and you can try to make it tractable.

  Of course you can't hope to make an expressive logic tractable: first-order logic is semi-decidable, default logic is undecidable, fine-grained intensional logic is incomplete, and that's just how things are. But you will often find that large parts of the statement of a problem can be given without using the full power of your chosen logic, even if there are still some parts that do require it.

The current paper takes the second of these approaches. For many problems even the most refined description logics are too inexpressive to specify the knowledge base – sometimes you need axioms involving two-place relations that go beyond simple selection restrictions, or you need to talk about equality, or you need defaults or intensional axioms or other devices that simply cannot be captured in any existing description logics. At the same time, you may want to do reasoning over types, for which the full power of the more expressive logic is not actually necessary. The trick is to find ways of combining the two.

We do this by taking an existing theorem prover for an extremely expressive logic and adding a treatment of sorts to it. The existing theorem prover is an extension of Satchmo (Manthey and Bry, 1988) to cope with property theory (Turner, 1987). Property theory is an 'encoding logic' – you can use it to axiomatise the proof theory of any standard first-order logic. As such it is about as intractable as a logic can be. We use it because you need this kind of expressive power for a number of issues in natural language processing, and because it also provides a vehicle for specifying approaches to epistemic reasoning that

do not rely on possible worlds semantics, and hence are not bedevilled by issues relating to logical omniscience and logical blindness.

Reasoning in property theory is fairly heavy going. Satchmo provides quite a nice basis (Cryan and Ramsay, 1997; Ramsay, 2001), but even if you exploit the kinds of relevance testing described by (Loveland, 1991) the basic Satchmo algorithm still tends to do a lot of repetitive processing.

We therefore add a generalised logic of types to the basic engine. This logic of types allows you to construct Boolean combinations of types from a sort lattice, with the results of the type based reasoning cached in order to avoid repetition. The key properties of this logic of types are summarised as follows:

- type specifications are encoded as terms which reflect the structure of the type lattice, with only the relevant parts of the lattice instantiated in any given case. This is an extension of the use of terms to represent type hierarchies (Fall, 1990), providing considerable extra expressive power with no increase in complexity.
- Axioms whose consequents are statements in the type logic are used in the forward chaining part of the Satchmo algorithm, with the results cached for future reference. This avoids much of the repetition associated with the basic Satchmo algorithm.
- Conjunctions of constraints from the type logic are dealt with using the coroutining facilities of Sicstus Prolog. This enables the system to monitor the state of conjunctive constraints, and to react as soon as such a constraint is satisfied. This means that we do not have to keep checking the individual elements of such a constraint to see whether they are satisfied.

The fact that we have a type lattice rather than a type hierarchy means that a considerable amount of the information that we need for expressing constraints on lexical relations can be encoded in the type logic (Wedekind, 1996). The resulting system thus allows us to mix deep reasoning about intensional facets of natural language semantics with efficient reasoning about simple reasoning about sorts. The use of terms to represent types means that the two kinds of reasoning can be very smoothly integrated, since we can simply rely on the fact that the basic Satchmo algorithm uses unification to to manage existential and universal quantification, so that we do not have to add any new machinery to integrate the two kinds of reasoning.

# References


Baader, F. and Sattler, U. (2001). An overview of tableau algorithms for description logics. *Studia Logica*, 69(1).

Cryan, M. and Ramsay, A. M. (1997). A normal form for Property Theory. In *Proceedings of the 14th International Conference on Automated Deduction (CADE-14)*, volume 1249 of *Lecture Notes in Artificial Intelligence*, pages 237–251, Berlin. Springer-Verlag.

Fall, A. (1990). *Reasoning with taxonomies*. PhD thesis, Simon Fraser University.

Loveland, D. W. (1991). Near-horn Prolog and beyond. *Journal of Automated Reasoning*, 7:1–26.

Manthey, R. and Bry, F. (1988). Satchmo: a theorem prover in Prolog. In *Proceedings of the 9th International Conference on Automated Deduction (CADE-9)*, volume 310 of *Lecture Notes in Artificial Intelligence*, pages 415–434, Berlin. Springer-Verlag.

Ohlbach, H. J. and Koehler, J. (1997). Role hierarchies and number restrictions. In *Description Logics 97*, Paris.

Ramsay, A. M. (2001). Theorem proving for untyped constructive $\lambda$-calculus: implementation and application. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(1):89–106.

Turner, R. (1987). A theory of properties. *Journal of Symbolic Logic*, 52(2):455–472.

Wedekind, J. (1996). On inference-based procedures for lexical disambiguation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 980–985, Copenhagen.