Run-time optimisations for reasoning with intensional logics

Allan Ramsay
Dept of Language Engineering, UMIST, PO Box 88, Manchester M60 1QD, UK
Allan.Ramsay@umist.ac.uk

Abstract. Most optimisation techniques for theorem provers for first-order logic rely on static analysis of the problem statement. For intensional logics, such as static analysis cannot be relied on, since it is impossible to predict what literals may be introduced by the intensional rules. The current paper shows how to use a dynamic (run-time) version of one well-known static optimisation, and considers its relationship to the use of 'relevance checking' in Satchmo.

1 A constructive intensional logic

We have shown elsewhere [8, 3] how to extend [6]'s theorem prover Satchmo to cope with [9]'s property theory. Property theory is a highly intensional logic which, roughly speaking, allows you to perform unconstrained quantification over propositions and properties, but places constraints on the conditions under which the Tarski biconditional

$$(\lambda xP).t \leftrightarrow P_{t/x}$$

holds. This language has numerous potential applications: I use it primarily for reasoning about natural language semantics, because that's what I'm interested in, but [10] uses a closely related language for reasoning about programs, and it can also be used as the basis of a theory of knowledge and belief which does not fall foul of the problems associated with modal treatments of these topics (logical omniscience, logical blindness).

Property theory is, clearly, highly intractable in theory – it is undecidable, rather than just semi-decidable, and incomplete (in the same sense that recursive arithmetic is incomplete, and for the same reasons). In the domain where I want to apply it, however, the problems that arise tend not be all that hard: it may be true that natural language semantics can only be captured properly if you use some language of this kind (as, for instance, situation semantics [2] uses non-well-founded set theory [1]), but most of the time you do not have to solve pathological problems in order to understand what some has said (unless, of course, they say *'I am now lying'*).

In ([8]), I showed that you could adapt Satchmo to work with property theory roughly as follows:

(i) rules with unspecified formulae in their antecedents are used only when those antecedents are ground. This means that a rule like $\forall X \forall P((believe(X, P) \& P) \rightarrow know(X, P)$[1] is turned into `know(X, P) :- believe(X, P), nonvar(P), P`.

(ii) rules with unspecified formulae in their consequents are used in the forward-chaining part of the algorithm, as though they were an extreme case of formulae with disjunctive consequents. As an example, the rule $\forall X \forall P(know(X, P) \rightarrow P$ turns into `know(X, P), nonvar(P) ==> P`, to be used by the Satchmo rule that asserts a component of the consequent of any sequent where the antecedent is currently provable but the consequent is not. The `nonvar(P)` test here is strictly redundant, since any sequent which failed to ground the consequent would be necessarily inconsistent (since it would entail *everything*, including $\perp$).

[1] I am not proposing this as a serious characterisation of the relation between knowledge and belief, just using it to illustrate the approach.

2 Optimisations

The problem with this inference engine is that although it is obtained by extending a fairly simple first-order theorem prover, it is not possible to use any optimisations which rely on a static analysis of the problem. You cannot, for instance, use [4]'s 'pure literal deletion' (where you remove a clause if it contains a positive (negative) instance of a literal which has no negative(positive) occurrences), because you do not know that the literal you are considering will remain pure. I propose a dynamic version of pure literal deletion, which makes sequents containing pure literals *temporarily* unavailable. At the point in Satchmo where a split sequent has led to the introduction of a new fact, clauses which might be *im*purified by the new fact are inspected, and reinstated if appropriate (this can lead to a cascade of reinstatements, in exact contrast to the way that pure literal deletion itself can lead to gangrene).

There are two ways to implement dynamic pure literal deletion of this kind. The weak version involves looking for antecedent literals that fail to appear in the consequent of any Horn sequent, the strong one deals with antecedent literals that fail to appear in the consequent of any sequent at all. It turns out that you cannot use the strong version with the relevance checking algorithms described in ([7, 5]), but for for the problems we are concerned with the strong version turns out to be more effective than the weak one + relevance checking in any case – Fig. 1 shows a typical set of results for a reasonably complex example (times in seconds).

	+groundedness	-groundedness
Unoptimised	1.62	1.98
Strong purification	0.38	0.42
Relevance checking	1.55	2.04
Weak purification	0.54	0.59
Relevance + weak pur.	0.52	0.59

Fig. 1. Relative effects of optimisations

References

1. P Aczel and R Lunnon. Universes and parameters. In J Barwise, J M Gawron, G Plotkin, and S Tutiya, editors, *Situation Theory and its Applications II.* CSLI Publications, 1991.
2. J Barwise and J Perry. *Situations and Attitudes.* Bradford Books, Cambridge, MA, 1983.
3. M Cryan and A M Ramsay. A normal form for Property Theory. In *Fourteenth Conference on Automated Deduction*, Townsville, Australia, 1997.
4. R Kowalski. A proof procedure using connection graphs. *JACM*, 22(4):572–595, 1975.
5. D W Loveland. Near-horn Prolog and beyond. *Journal of Automated Reasoning*, 7:1–26, 1991.
6. R Manthey and F Bry. Satchmo: a theorem prover in Prolog. In *Proc. 9th Inter. Conf. on Automated Deduction (CADE-9), LNAI 310*, pages 415–434, 1988.
7. A M Ramsay. Generating relevant models. *Journal of Automated Reasoning*, 7:359–368, 1991.
8. A M Ramsay. A theorem prover for an intensional logic. *Journal of Automated Reasoning*, 14:237–255, 1995.
9. R Turner. A theory of properties. *Journal of Symbolic Logic*, 52(2):455–472, 1987.
10. R Turner. *Constructive Foundations for Functional Languages.* McGraw Hill, London, 1991.