CAN DOMAIN SPECIFIC KNOWLEDGE BE GENERALIZED?

Alan Bundy
Department of Artificial Intelligence
University of Edinburgh
Scotland

The MECHO project (see Cl]) consists of writing a computer program which can colve a wide variety of simple mechanics problems stated in English. This program is being used as a vehicle for studying methods for guiding search in a semantically rich domain. Our methodology is to find general, justifiable, inference rules which can be combined to carry out the reasoning necessary to solve the mechanics problems. As is well known, when rules like these are run on a general inference machine the result is often a combinatorial explosion. Rules are combined in unexpected ways and the search for a solution is developed along unreasonable paths. These failures are used to debug the rules by adding to them local, domain specific control information. Finally these techniques are generalized and incorporated in the inference mechanism. We hope that this methodology will lead us to the design of a computational logic for natural reasoning.

In this paper one such transition from domain specific to general inference technique will be described. We will use this example to emphasize the importance of this generalization stage. Without it one might be led to superficial and false conclusions about the nature of natural reasoning.

Suppose we have available the following relations: Vel(Object, v, time) - (v is the velocity of object during time);
At(Object, place, moment) - (object is at place at moment); Final(period, moment) - (moment is the final moment of time interval period). We may have discovered the following domain specific information enabling us to guide the search for problem solutions along successful lines,
(i) If the program is desperate to satisfy Vel(Object, v, time), object and time being known, but all inferences have ground to a halt, then a new intermediate unknown v can be created and asserted to be the velocity of object at time,
(ii) If the program is asked to confirm that At(object, placel, moment), but it already knows that object is at some different place (place2) at moment, then the attempt to prove At(object, placel, moment) can be abandoned, because objects can only be in one place at a time,
(iii) If the final moment of period is found to be momentl, but later processing fails, it is no use backing up to recalculate Final(period, ?x), since the same answer is bound to be given.

At first glance it might seem as if these examples argued for the intervention of rich domain specific information at all control points and that a programming language which facilitated such intervention was required. But these conclusions are not justified from the examples (i)-(iii) above. In fact (i)-(iii) represent different facets of a general phenomenon. To see this notice that Vel, At and Final are all really functions: Vel is a function from objects and times to velocities; At is a function from objects and moments to places and Final is a function from periods to moments. What separates functions from other relations is that they are single valued, that is their value is guaranteed to exist and to be unique♦ (i) is an example of this existence property being used and (ii) and (iii) different uses of the uniqueness property. The generalizations of (i), (ii) and (iii) are
(i)' If the program is desperate to find a function value given its arguments, and all inferences have failed then a new entity can be created and asserted to be that value. (Note that we do not want the program to create a new entity whenever it is legal to do so as this contributes to the combinatorial explosion. The notion of being desperate for the answer can be generalized (see [21).
(ii) * If the program is trying to confirm a function value, but it already has a contradictory value stored then the confirmation attempt is to be abandoned.
(iii)' If the program has calculated a function value then it should not recalculate this on back up.

Note that (i)'-(iii)' improve on the predicate calculus representation of the existence and uniqueness of function values, by giving procedural information about when this information is to be used. They also improve on the local domain specific embodiment of (i)-(iii) by representing a large amount of such information in concise form, e.g. the existence property can now be used on At and Final and the uniqueness property can be used on Vel. All that is necessary to share the benefits of the control information embodied in (i)'-(iii)' is to specify which relations have function values.

For functions of one argument the implementation of (i)'-(iii)' can be assisted by maintaining a single slot on the property list of the argument. However, something more complicated is needed for functions of more than one argument.

Some relations may be functions in more than one way, e.g. if Timesys(period, initmom, finmom) means that initmom is the initial moment and finmom is the final moment of period then Timesys is a function in 3 ways:
(a) from period to Initmom
(b) from period to finmom
(c) from initmom and finmom to period

References

[1]l Bundy, A., Luger, G., Stone, M. and Welham, R. 1976. "MECHO: Year One", p94-103, Procs. of the 2nd AISB Conf. ed. Brady, M., Edinburgh.
[2] Bundy, A. 1977. "Will it reach the top? Prediction in the mechanics world", DAI Research Report No. 31, Edinburgh.