Hegner: Distributivity in Incompletely Specified Type Hierarchies

Commentary by C. S. Mellish

21st July 1994

I found this a very interesting paper which interacts in some ways with issues that I have also worried about. The paper is generally well presented and is extensive in its coverage. Unfortunately, lack of time has meant that I have not been able to consider all of the material in detail. My comments will therefore mainly be of a high-level nature.

1 Motivation

Hegner's arguments about the expressivity of CUF (compared, say, to ALE) make a lot of sense[1] But there is a gap in his argument, which seems to conclude that all these systems are providing an abbreviated way of specifying *distributive* lattices.

As far as I can see, there are two main reasons for wanting to have a distributive lattice of sorts:

- It means that the semantic definition of the language can associate $\sqcap$ with intersection of sets and $\sqcup$ with union of sets. Hegner emphasises in 0.2 that for this to be appropriate there must be distributivity (are there no other properties that are required as well?). I can see that such a semantics would be conventional, but on the other hand it does not seem to be necessary.

- It means that the usual rules of Propositional Calculus can be used in an implementation. Presumably a system without the distributive law would be considerably weaker as regards establishing consistency of formulae (though I don't understand exactly what the consequences would be). But in the case of finite sort hierarchies, the exhaustive cases for $\sqcap$, $\sqcup$ and $\neg$ can be tabularised, and so there does not seem to be a need for such reasoning.

As far as I can see, it would be perfectly consistent to claim that the hierarchy shown in Figure 0.1 is indeed supposed to be the *whole* of the relevant lattice (which has some

[1] I am not sure where TFS lies on the line between these two systems though.

strange properties but nevertheless models some aspects of the world at a level of detail appropriate for some application).

In fact, if one really cares about the two reasons above, then surely one requires negation to work "properly" as well, in which case we should be looking at Boolean algebras from the start. So why the emphasis on distributivity? Is this to do with the way that in the Appendix arbitrary CUF type formulae are reduced to axioms stated entirely in terms of $\sqcap$ and $\sqcup$?

2 Incomplete Specifications

Independently of the new results, here the paper makes a most worthwhile contribution by summarising and bringing to bear mathematical results on the general problem of extensions of bounded posets with partial operations (BPPOs), which has not been adequately discussed in the Computational Linguistics literature. The result is a framework where one can discuss these problems in a general and precise way for the first time. I have personally found it very difficult in the past to know which parts of lattice theory are relevant to the sorts of problems we have, and this paper opened my eyes about some parts that I had missed.

The definitions of the different forms of extensions have a critical role in this chapter, and I think that some examples or further discussion might have been helpful as a way of fixing these in the reader's mind. I guess it is the injectiveness of extensions that prevents the kind of "collapsing" done by CUF being allowed. Presumably there is a theorem that universal extensions are injective? The fact that Theorem 1.4.7 does not guarantee injectiveness could be pointed out slightly earlier.

3 Decision Problems

In spite of my questions about the motivation, I can see that for a system like CUF that wants a traditional semantics for sorts there is a real need to check that the axioms are consistent with a distributive lattice interpretation in which distinct types are separable (this is well expressed by Hegner in 0.4). Thus for this system at least the results on computational complexity are very relevant.

Hegner shows an efficient way of determining whether a prespecification of the kind provided in a CUF program can be extended into a BPPO – this method avoids having to compute the whole BPPO, which may be exponentially larger than the original specification. But unfortunately he shows that the subsequent general problem of determining whether there is a distributive extension is NP complete. Critical in this is the problem of showing separability. This might suggest that it could be something other than distributiveness that a system like CUF should be striving for.

It is interesting that the complexity is acceptable if only one of meet and join are allowed in the prespecification, and Hegner relates this to the case in ALE, LIFE and LKB. All these systems only employ a meet operator – one wonders whether there is any value in systems with only join axioms, perhaps for machine learning.

It would be very useful at some point (probably in a subsequent paper, because this one is already very long) to survey the various systems (TFS, ALE, CUF, LKB, LIFE) that are mentioned in the paper, in terms of the expressiveness of their prespecification languages and the kinds of extensions that are assumed. For instance, TFS can work with an open or a closed world assumption – what do these options correspond to (and what are the complexities of the corresponding consistency checking procedures)?