PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT.

Finding the Best Route for Logic Programming

Hervé Gallaire
GSI
25 Bd de l'Amiral Bruix, 75782 Paris Cedex 16 France
gallaire@gsi.fr

Abstract

The panel chairman has asked us to deal with two questions relating Logic Programming (LP) to computing. They have to do with whether LP is appropriate (the most appropriate?) as a springboard for computing as a whole in the 21st century, or whether it is so only for aspects (characteristics) of computing. I do not think that there is a definite answer to these questions until one discusses the perspective from which they are asked or from which their answer is to be given. In summary, we can be very positive that LP will play an important role, but only if it migrates into other leading environments.

1 Which Perspective To Look From

We are asked to talk about directions for the future, for research as well as for development. Clearly, for me, there will not be a Yes/No answer to the questions debated on this panel. I don't shy away, but at the same time there are too many real questions behind the ones we are asked to address. Thus, I will pick the one aspects I am mostly connected to: research on deductive databases and constraint languages, experience in commercial applications development and in building architectures for such commercial developments. Whether these different perspectives lead to a coherent picture is questionable.

If I ask the question relative to computing as a whole, I can ask it from the perspective of a researcher, from that of a manufacturer, from that of a buyer of such systems and ask whether LP is the pervasive paradigm that will be the underlying foundation of computing as a whole from each of these perspectives.

If I ask the question relative to the characteristics of computing, I can look at computing from the perspective of an end-user, of an application developer (in fact from many such applications, e.g. scientific, business, CAD, decision support, teaching, office support, ...), of a system developer, of a language developer, of an architecture engineer, of a tool developer (again there are many such tools, e.g. software engineering, application analyst, etc). I can even look at it from the perspective of research in each of the domains related to the perspectives just listed, for example a researcher in user interface systems, a researcher in software engineering, in database languages, in knowledge representation, etc.

But the picture is even more complicated than it appears here; indeed I can now add a further dimension to the idea of perspective elaborated upon here. Namely I can ask whether LP is to be seen as the "real thing" or whether it is to be an abstract model essentially. For example, ask whether it is a good encompassing model for all research aspects of computing, for some of them (the perspectives), whether it is a good abstract model for computations, for information systems, for business models, even if they do not appear in this form to their users, this being asked for each type of computation carried out in a computing system.

Looking at these questions is to study whether LP should be the basis of the view of the world as manipulated at each or some of the following levels: user's level, at system level, at application designer level, at research level, ... or whether it should only be a model of it, i.e. a model in which they basic problems of the world (at that level) are studied, and that the two would match only in some occasions.

2 Global Perspective

I think we have to recognise that the world is definitely never going to be a one level world (ie providing in hardware a direct implementation of the world view); second that the world view will be made of multiple views; third we have to accept that different views will need different tools to study a version of a problem at that level; and fourth that it may be appropriate to use abstractions to get the appropriate knowledge into play. Consequently, neither LP nor any other paradigm will be the underlying foundation for computing; it is very appropriate however, for each paradigm to ask what its limits are. This is what has been my understanding of most projects around LP in the past ten to fifteen years; trying several angles, pushing to the limits. Developing hardware for example is one such worthwhile effort.

3 Model and Research Perspective

As a model of computing, from a research perspective, LP will continue to develop as the major candidate for giving a "coherent" view of the world, a seamless integration of the different needs of a computing system for which it has given good models. To come to examples, I believe that LP has made major contributions in the following areas: rule-based programming, with particularly results on deductive databases, on problem solving and AI, specific logics for time and belief, solutions to problems dealing with negation, to those dealing with constraint programming and to those dealing with concurrent programming. It will continue to do so for quite some time. In some cases it will achieve a dominant position; in others it will not, even if it remains a useful formalism. In the directions of research at ECRC, we have not attempted to get such a unified framework, even though we have tried to use whatever was understood in one area of research into the others (eg, constraints and parallelism, constraints and negation, ..). LP will not achieve the status of being the unique encompassing model adopted by everyone. Indeed, there are theoretical reasons that have to do with equivalence results and the human intelligence which makes it very unlikely that a given formalism will be accepted as the unique formalism to study. Further, there is the fact that the more we study, the more likely it is that we have to invent formalisms at the right level of abstraction for the problems at hand. Mapping this to existing formalisms is often possible but cumbersome. This has the side advantage that formalisms evolve as they target new abstractions; LP has followed that path.

4 Commercial Perspective

As a tool for computing in general, from a business or manufacturer's point of view LP has not achieved the status that we believed it would. Logic has found, at best, some niches where it can be seen as a potential commercial player (there are many Prolog programs embedded in several CASE tools for example, natural language tools are another example). When it comes to the industrial or commercial world things are not so different from those in the academic or research world: the resistance to new ideas is strong too, although for different reasons. Research results being very often unconclusive when it comes to their actual relevance or benefits in practical terms, only little risk is taken. Fads play an important role in that world where technical matters are secondary to financial or management matters; the object technology is a fad, but fortunately it is more than that and will bring real benefits to those adopting it. We have not explained LP in terms as easy to understand as done in the object world (modularity, encapsulation in particular). The need to keep the continuity with the so-called legacy applications is perhaps even stronger than fads. To propose a new computing paradigm to affect the daily work of any of the professionals (whether they develop new code or use it) is a very risky task. C++ is a C based language; we have no equivalent to a Cobol based logic language. And still, C++ is not a pure object oriented language. The reason why the entity-relationship modeling technique (and research) is successful in the business place is that it has been seen as an extension of the current practices (Cobol, relational) not as a rupture with them. SQL is still far from incorporating extensions that LP can already provide but has not well explained: where are the industrial examples of the recursive rules expressed in LP? what is the benefit (cost, performance, ...) of stating business rules this way as opposed to programming; and without recursion, or with limited deductive capabilities, relational systems do without logic or just borrow from it; isn't logic too powerful a formalism for many cases? LP, just like AI based technology, has not been presented as an extension of existing engines; rather it has been seen as alternatives to existing solutions, not well integrated with them; it has suffered, like AI from that situation. Are there then new areas where LP can take a major share of the solution space? In the area of constraint languages, there is no true market yet for any such language; and the need for integration to the existing environments is of a rather different nature; it may be sufficient to provide interfaces rather than integration. A not to be overlooked problem, however is that when it is embedded in logic, constraint programming needs a complex engine, that of logic; when it is embedded in C, even if it is less powerful or if it takes more to develop it (hiding its logical basis in some sense), it will appear less risky to the industrial partners who will use or build it.

5 More Efforts Needed

Let me mention three areas where success can be reached, given the current results, but where more efforts are needed. Constraint based packages for different business domains, such as transportation, job shop scheduling, personnel assignment, etc will be winners in a competitive world; but pay attention to less ambitious solutions in more traditional languages. Case tools and repositories will use heavily logic based tools, particularly the deductive database technology, when we combine it with the object based technology for what each is good at. Third and perhaps more importantly, there is a big challenge to be won. My appreciation of computing evolution is as follows: there will be new paradigms in terms of "how to get work done by a computer"; this will revolve around some simple notions: applications and systems will be packaged as objects and run as distributed object systems, communicating through messages and events; forerunners of these technologies can

already be seen on the desktop (not as distributed tools), such as AppleEvents, OLE and VisualBasic, etc and also in new operating systems or layers just above them (e.g. Chorus, CORBA, NewVave, etc). Applications will be written in whatever formalism is most appropriate for the task they have to solve, provided they offer the right interface to the communication mechanism; these mechanisms will become standardised. I believe that concurrent and constraint logic languages have a very important role to play in expressing how to combine existing applications (objects, modules). If LP had indeed the role of a conductor for distributed and parallel applications, it would be very exciting; it is possible. Following a very similar analysis, I think that LP rules, particularly when they are declaratively used, are what's needed to express business rules relating and coordinating business objects as perceived by designers and users. To demonstrate these ideas, more people, knowledgeable in LP need to work on industrial strength products, packaging LP and its extensions and not selling raw engines; then there will be more industrial interest in logic, which in the longer term will trigger and guarantee adequate research levels. This is what I have always argued was needed to be done as a follow up of ECRC research, but I have not argued convincingly. This is what is being done with much enthusiasm by start ups around Prolog, by others around constraint languages, e.g. CHIP; this is what is being started by BULL on such a deductive and object oriented system. Much more risk taking is needed.

6 Conclusion

From the above discussion the reader may be left with a somewhat mixed impression as to the future of our field; this is certainly not the intent. The future is bright, provided we understand where it lies. LP will hold its rank in the research as well as in the professional worlds. The major efforts that the 1980's have seen in this domain have played an essential role in preparing this future. The Japanese results, as well as the results obtained in Europe and in the USA, are significant in terms of research and of potential industrial impact. The major efforts, particularly the most systematic one, namely the Fifth Generation Project, may have had goals either too ambitious or not thoroughly understood by many. If we understand where to act, then there is a commercial future for logic. At any rate, research remains a necessity in this area.