Comments on Logic and Knowledge Representation

Victor W. Marek

Occasions such as this one, make people look back. It is almost 44 years since I started research, in a far away land, and in an area quite different from that I pursue today.

As I look around, I see friends and collaborators. First of all Mirek Truszczynski and Jeff Remmel, friends and coauthors, each of over 20 papers. It was quite an adventure, no doubt.

It is a great day for a scientist to look around and recognize so many colleagues and coauthors, also students. A few people who are not with us today need to be mentioned. First of all late Witold Lipski, my second Ph.D. student, who quickly became a friend. Witold was a strong combinatorist, computer scientist and logician. His achievements stand even today in database theory and other areas of computer science. It was him who told me that nonmonotonic logic will be the "next big thing". It certainly was - for me and so many of you in this room. I also need to mention Anil Nerode, the first among mainstream logicians to see the need to research the nonmonotonic logic, and to incorporate it into the mainstream of foundational studies. It is relevant to this meeting to remember that it was Anil and Peter Hammer who got the idea of founding of "Annals of Mathematics and Artificial Intelligence" (I was present when they negotiated this during 1986 ORSA meeting in Miami, Fl.) and the first Symposium on Mathematics and Artificial Intelligence (it so happened that I was there, too).

Looking back, at the area of nonmonotonic logic, the one with which I am closely associated, it is necessary to say that after the pioneers (John McCarthy, Ray Reiter, Drew McDermott, Robert Moore) the next group of people who followed was led by Vlad Lifschitz, Michael Gelfond, Mel Fitting, Allen van Gelder, John Schlipf, Teodor Przymusinski, Mirek Truszczynski, Ilkka Niemelä, Jeff Remmel and others. I want to mention especially

Vladimir's work. It is hard to overestimate his contributions, many of those together with Michael Gelfond. It was, actually, Michael, who realized that autoepistemic logic allows to provide a semantics for negation in logic programming. The stable semantics followed shortly. Coming back to Vladimir's and Michael's contributions I do not only mean the introduction of stable semantics. The fundamental understanding of the role of stable semantics and its generalizations in basic problems of AI is due to them. Vladimir led the effort to understand the proof theoretic-aspects of stable semantics, an effort which is not yet (in my opinion) finished.

It took us some time to understand Default Logic of Ray Reiter, its central place in nonmonotonic reasoning, and see its relationship with the other modes of nonmonotonic reasoning. It took us even more effort to recognize that the theory of operators in lattices is the true foundation of nonmonotonic reasoning. Understanding of the role of algebraic constructions in nonmonotonic logic is fundamentally due to the work of Mel Fitting. The work of Marc Denecker, of Jeff Remmel, and of Mirek Truszczynski explained a variety of phenomena related to nonmonotonicity. All these people showed that the same methods so successfully applied to classical logic by Alfred Tarski equally well apply in nonmonotonic case.

The complexity issues for nonmonotonic reasoning were studied by many, with the significant results due to Thomas Eiter and George Gottlob. We learned that there is no "free lunch" promised to us by the pioneers, including John McCarthy. Nonmonotonic reasoning (in its many forms) turned out to be complete for various levels of polynomial hierarchy (in the propositional case) and even more complex in its predicate form (here the work of John Schlipf, of Krzysztof Apt and Howard Blair and also of Doug Cenzer and Jeff Remmel must be mentioned).

It took us almost 15 years to move from pure theorizing to the computational systems based on nonmonotonic logic. The premonitions of the computational character of nonmonotonic logic, in the hindsight present in many contributions, led to a number of systems, with long-forgotten names (like DeReS), and the still going strong systems such as *smodels* of Ilkka Niemel"a and collaborators, `dlv` of Thomas Eiter, Nicola Leone and collaborators, *ASSAT* of Fangzhen Lin and Yuting Zhao, of *aspps* of Deborah East and Mirek Truszczynski, of *cmodels* of Yuliya Lierer and Vlad Lifschitz, of *clasp* of Torsten Schaub and many others.

Nonmonotonic logic, in its computational form, *Answer Set Programming*, is going strong. The need for understanding foundations of software

engineering in ASP resulted in amazing discoveries of Vlad Lifschitz, David Pierce and Augustin Valverde, of Thomas Eiter and collaborators, and of Mirek Truszczynski of the relationships of this area to very interesting, while quite exotic, logical systems such as the modal logic S4F and also the maximal intermediate logic, the Gödel-Smetanich, or Here-and-There logic. Why is it so - was explained by the work of Alex Bochman.

The relationships between ASP and SAT, discovered by Fangzhen Lin and Yuting Zhao are not yet completely understood. The amazing progress of SAT during the past 10 years is helping us to find ways to improve ASP systems, I am convinced that we did not say the last word on these connections.

It is tempting to compare the situation in ASP with that of SAT. After all, these formalisms solve the same class of problems (at least when ASP is limited to SLP). It is also revealing for it shows clearly that some aspects of ASP need the community attention.

The first, and most obvious question that still lacks an answer is the issue of proof systems associated with ASP. After all computations (of stable models, say) are proofs of literals. Let us recall that the lower bounds in SAT are obtained by looking at proof systems. As of today, the proof systems for ASP are two: one is the system inherited from the logic HT (Gödel-Smetanich maximal non-classical logic) and another is the idea of proof schemes (and M-proofs). I doubt that these are the last words in proof-theory of ASP. From the point of view of logic we need more such systems. The quest for proof systems for ASP started with the work of Vlad Lifschitz in the middle of 90ies. We need more of this kind of work.

The algorithms for ASP also need our attention. DPLL-like algorithms are not the only way to compute stable models. The introduction of the idea of learning was the great leap forward of SAT. But learning is partial closure under resolution and is a kind of hybrid approach between Quine and Davis-Putnam approaches. Learning was tried in the context of ASP by John Schlipf. I am not claiming that this is the path to be taken in computation of stable models, but certainly *something* needs to be done to speed up computation.

To make the task of programming in ASP easier Ilkka Niemelä and collaborators (but later also other system designers) introduced *grounders*. Next, considerable effort went into understanding the nature of stable semantics for programs admitting constraints. It is all about making the life of programmers easier with more concise programs and more elaborate program

constructs. Starting with Ilkka Niemelä and collaborators, then Jeff Remmel, and more recently Mirek Truszczynski and collaborators we gained great insights into programs with constraints and their stable semantics. But we do not have an authoritative, general consensus approach to the stable semantics of programs with constraints. Worse, we do not even know which constraints are truly important in Knowledge Representation.

We know that SLP which, after all, is most important part of ASP captures the same class of problems as SAT. But we do not know where ASP is better than SAT, that is we do not have reasonable guidelines which would allow for the use of ASP in tandem with SAT in portfolio arrangement. This, despite the fact that Michael Gelfond and Nicola Leone looked at the Knowledge Representation with ASP, and despite the fact that Chitta Baral wrote a book on ASP. We may need a kind of "How to solve it with ASP" book of examples and techniques. While Marc Denecker attempted the "grand synthesis" of ASP and SAT with his ID-Logic, we do not know whether it is the only way to do so.

We do know that ASP may be used in model checking (Eugenia Ternovska), but we do not know if ASP can serve as a glue for decision procedures and other tasks, in a way analogical to SAT (this is known as SAT modulo theories, SMT.)

Let me list a number of steps that need to be taken if we ever want to make ASP "industrial strength". Let me prequalify this list with the statement that some of these features, in some form, are already present, especially in `dlv`. First of all, we need to be able to handle data types such as strings and associated operations such as concatenation and regular expressions evaluation. We need to be able to connect to remote databases both for reading the data and writing results. We need to be able to call imperative languages from within the ASP programs and specify interfaces for such programming languages when those need to use ASP systems for search. Systems such as `dlv`, *smodels*, and *aspps* have some of these features. But we need a consensus for standards for all these features. We need to be able to use data described by means of XML documents to be translated into extensional databases, and we need tools for outputting the collections of stable models as XML documents. We need to be able to tie Semantic Web with ASP. The work of Thomas Eiter and collaborators offers a step in this direction.

I am not aware of *programming environments* that would make possible real programming in ASP. There were proposals of *some* tools (viz. Marina

De Vos). Yet, we did not do the real effort in this area. Anyone who wrote even a simplest program in imperative language knows that the current situation of programming in ASP is a major block in the development of ASP as a viable mainstream programming tool.

I limited my remarks to ASP, because it is what I know. I was present as it emerged from the widely understood Knowledge Representation area and tried to help in its development.

But widely understood Logic and more generally Foundations are not limited to ASP, SAT and their relatives. While logicians, especially those with mainstream mathematics continue their research, logic systematically spawns various smaller applicable areas (for instance automata theory, automated theorem proving, applications of model-theoretic techniques to electronic design automation). It is a triumph of logic that so many of its subspecialties have applications. Even the most elusive areas (recursion theory, model theory, proof theory) found important applications. What a satisfaction!

At the same time the center of gravity of Foundations shifted. So many of us either moved from mathematics into computer science or did not even try in mathematics. It is not an accident that foundations with the computer science twist are strongly represented in such places as IAS in Princeton.

Time to complete these chaotic remarks. I want to thank all of you who contributed to this session. There is nothing more gratifying for a scientist to see that the effort was not in vain.