The Logic Programming Paradigm

Jürgen Dix
Department of Computer Science
University of Koblenz
Rheinau 1, D-56075 Koblenz, Germany
dix@uni-koblenz.de

This is a detailed report about the *Logic Programming Paradigm*, an international meeting that took place in Shakertown near Lexington, Kentucky, USA from April 24 to April 27 (see URL:http://www.cs.engr.uky.edu/~inna/lpp.html). Past and current activities in Logic Programming were described and the future of the field was critically discussed.

Keywords: Future of Logic Programming, Non-Monotonic Reasoning

1. Introduction

The aim of the whole meeeting, organized by Krzysztof Apt (president of the Association of Logic Programming), Wiktor Marek, Mirek Truszczyński and David Warren, was to get a clear picture of the current activities in Logic Programming and its role in the coming years. In the organizers' words: *The aim of the meeting was*

- *to present current research directions in Logic Programming worldwide,*
- *to analyse future research directions,*
- *to boost Logic Programming in North America, and*
- *to discuss the relation between Logic Programming and related fields*

There were 13 regular talks, each scheduled for 45 minutes and followed by a fifteen-minute discussion chaired by an expert of the topic of the presentation. Sometimes very lively discussions, stimulated by the chairs who started the discussion with some comments, emerged.

In addition to the talks, there were two panels, one on the *past* and one on the *future* of Logic Programming. One evening was devoted to the idea of founding a north american chapter of the Association of Logic Programming: an idea that found enough support and will eventually lead to such a subchapter.

Among the participants in the workshop 23 were from North America, 9 from Europe, 2 from Japan and 1 from Australia, constituing a total of 35 participants. Additionally, local participants, from Lexington and Cincinnati were present.

The meeting can be viewed as a continuation of another meeting four years ago at the same place, the LPNMR retreat: see [2]. The LPNMR retreat, however, focused on a narrower domain, namely the intersection of logic programming and nonmonotonic reasoning techniques. This year it was the broad area of Logic Programming in all its appearances. *What went wrong in the past?*, *Where is the field going?*, *Where should it be going?* were the main questions hanging around.

There will be formal proceedings of the meeting, consisting of articles related to the talks (see [1]). Therefore most of the contributions mentioned below will find their ways into detailed papers in the forthcoming proceedings.

2. Talks

The first lecture was delivered by Bob Kowalski and chaired by Jack Minker: *From Logic Programming to Multi-agent systems*. Bob started with a list of successes of LP (CLP, ILP, logic grammars, knowledge representation) and discussed the case against LP (no procedural syntax, no logic for change of state or for input-output, unclear about concurrency and about treatment of objects). He then related objects and properties of LP to those in multi-agent systems: logic programs versus *single agent's beliefs*, integrity constraints versus *single agent's goals*, no operating system versus *observe-think-act* no change of state versus *beliefs, goals and environment*. He went on with describing the thinking component of an intelligent agent and viewing *beliefs* as generalized logic programs in order to make them reactive to changes that arise in the environment.

AI Communications 11 (1998) 150–157
ISSN 0921-7126 / $8.00

Bob claimed that particular example of multi-agent systems are *distributed active deductive databases*, so that disjunctive integrity constraints come in. He went on with describing an architecture for multi-agent systems. Concurrent interactions of agents are achieved by treating all interactions as interactions between an agent and a global environment. Thus concurrency is achieved in the manner of blackboard architectures and Linda-like systems and not, like in concurrent logic programming, in the manner of message passing architectures.

Gopal Gupta gave a talk on *Horn Clause Denotations and Their Applications* with Mats Carlsson as discussion leader. Gopal claimed that the use of logic programming as the notation for specifying denotational definitions of programming languages leads to many advantages:

1. both syntax and semantics are executable and can be specified in the same notation;
2. compiled code (sequential or parallel) for a program can be automatically derived from its denotation via partial evaluation;
3. program verification and model checking can also be done within the same framework.

He also briefly spoke about recent advances his group has made in parallel logic programming. He stressed the importance of research in parallel logic programming, since one major complaint against logic programming languages has been their lack of efficiency. Exploiting parallelism from logic programs on today's (abundantly and cheaply available) parallel hardware is one way of improving execution efficiency. He mentioned that the ACE parallel Prolog system (`URL: http://www.cs.nmsu.edu/lldap/ace/ace.html`), designed and implemented by his group, has recently been used to parallelize Prolog applications as large as 35,000 lines long.

Gopal also stressed the importance of building logic programming based tools (e.g., compilers, abstract interpreters, partial evaluators, parallel systems, etc.) that will scale up to large applications because, otherwise, they will not get used. Finally he pointed to applications such as *software engineering* and *semantic filtering*.

The last speaker of the morning was Luc De Raedt who gave *A perspective from Inductive Logic Programming* in a session chaired by Koichi Furukawa. He gave a nice overview on the rapidly evolving area of inductive logic programming and its roots in machine learning. The success story of this branch of LP may be due to the fact that there was always a nice balance between

– *theory*,
– *techniques and systems* and
– *applications*.

Also inductive logic programming is becoming an ingredient of several commercial data mining packages. Luc considered in particular an example of nitroaromatic compounds (mutagenesis) where ILP-tools have been very successfully applied. He ended with a proposal for a relational data mining language that embeds inductive constructs within a logic programming language, which is similar to embedding constraint handling within (constraint) logic programming

The afternoon started with Danny de Schreye's talk on *Assessment of some issues in CL-theory and -program development*, chaired by Maurizio Martelli. He argued how, in the area of development and implementation, basic research in *Computational Logic* has reached significant breakthroughs, but that it is still far from its objective of formal semantics supported semi-automated software development. Danny tried to show how integration on various levels is desirable. He considered insufficient work on methodological and software engineering issues as a major weakness and pointed to some wishes for semantics-based programming tools.

In the discussion the impact of LP research on other paradigms was mentioned. There can be a more 'optimistic' view of our achievements if we strive for less *general* objectives (more focused or *modest*) and specific application fields. Also the role of *freeness* with respect to languages and sophisticated tools (such as the ones presented by Danny) has ben discussed: was it and is it a mistake to think of making money selling languages and tools instead of selling applications based on free software?

The last talk on the first day was given by Manuel Hermenegildo and chaired by Michael Maher. In the first part of his talk Manuel discussed successes, failures, and challenges in the field. He mentioned as successes the significant conceptual impact made in programming languages, databases, knowledge representation, grammars, concurrency, constraints, learning, and implementation techniques (abstract machines, global analysis, specialization, parallelization, etc.), as well as the practical systems developed (Prolog, CLP systems, ILP systems, etc.). In the area closest to him, parallelism, he pointed out the major advances made in parallel abstract machines, program analysis, automatic parallelization, scheduling, and granularity control. Regarding failures, he mentioned not meeting some expectations (that in his opinion were set too high), programming environments that are still not powerful enough, perhaps too much emphasis on declarativeness, and too little attention given to dealing with change and other practical issues. Finally, he proposed challenges such as better program development systems, better theoretical results for certain issues (e.g., change, objects, meta/higher-order), better educational material, and the development of some *killer applications* in intelligent interfaces (e.g., advanced WWW search tools), distributed agents, modeling, and software engineering.

In the second part of his talk, Manuel discussed the use of abstract interpretation in program development, rather than in the more traditional application of optimization. He presented an architecture of tools for the task and demonstrated on his laptop a program validation and diagnosis system (currently a preprocessor for the CIAO and CHIP systems), based on this architecture. He showed how the system, developed in collaboration with Cosytec, INRIA, and Linköping University, uses abstract interpretation to generate and check assertions in Prolog and CHIP/CLP(fd) programs, and integrates compile-time and run-time checking in a uniform way. He used assertions describing properties such as types, modes, non-failure, determinacy, cost, and termination to detect several bugs in programs, and showed how the system can treat modules separately and perform incremental analysis. He finished by comparing the different views on the role of types (and program assertions in general) in the logic and functional programming communities, which resulted in a very lively discussion.

During the discussions that followed the talks several people noted that although LP certainly has had an impact on computer science it could have had much more. Some people pointed out that LP languages are still sometimes perceived as nonserious languages by industry (wich can result in viewing LP research as obsolete). It was argued by several people that producing more fielded applications and better programming environments would help much more than just writing papers.

The first day ended with a long discussion of the pro's and contra's of establishing a North American subchapter of the Association of Logic Programming: we refer to Section 4.

The second day began with Jacques Cohen's talk on *Computational Molecular Biology: A Promising Application Using LP and CLP*, chaired by Philippe Codognet. Jacques first introduced us to the basics of molecular biology, explaining the overall process and the transformation of DNA into pre-m-RNA, m-RNA, amino acids and finally into proteins. In his talk, he considered in detail the step from pre-m-RNA into m-RNA. He considers problems in this area of molecular biology to be of great interest to the LP community and tried to attract the attention of LP researchers to this exciting new area.

His main reasons why LP should play an important role in computational molecular biology are the successes of LP in

– dealing with parsing,
– non-deterministic situations, and
– inverse computations.

He considered all these as assets that will likely play a significant role towards facilitating the solution of problems in CMB.

Saumya Debray, chaired by Paul Tarau, gave the second talk: *Link-time Optimization of Multi-Language Programs*. One of Saumya's main points was that not many real-world applications are likely to be written entirely in a single language (such as PROLOG). Thus for PROLOG to survive, the issues relating to the use of multiple languages in a single application need to be addressed. Among such issues (semantics, interface issues) he addressed in detail the performance issue and described the current direction of the work in his group: post processing of executable files to apply to them some optimisations not available at compile time.

The discussion raised the following points: (i) why not using Prolog to develop the optimiser, (ii) why should we link together binary code generated form multiple languages instead of using a more flexible component based architecture (client/server for instance), (iii) is expressiveness or performance really the key issue for the next generation of LP languages?

Philip Wadler, a well-known expert in and advocate of functional programming, ended the morning with his lecture on *How enterprises use functional languages and why they don't*, chaired by Frank Pfenning. Philip, working at Bell Labs where C and C++ were invented, admits that, compared with C it is not completely wrong to say that "no one uses functional languages". But he pointed to half a dozen uses of functional languages in the real world (see `http://www.cs.bell-labs.com/˜wadler/realworld`). The first example was

HOL resp. Isabelle, implemented in ML. They use higher-order functions and types as important ingredients. The second example was Erlang, a functional language designed by Ericsson for telecommunications applications. Erlang has primitives for spawning processes and handling messages. Compression techniques to minimise bandwidth are built-in and thus it is much better in this respect as C, C++ or Java. Other examples were Pdiff (used for phone switches), written in ML, CPL/Kleisli (a high level language for formulating queries to a database in the Human Genome project), and Natural Expert, an expert system shell.

In the second part of his talk, he analysed why functional languages are not so widely used. He claimed that a language should support inter language working, possess extensive libraries, be highly portable, have a stable and easy to install implementation, should come with debuggers and profilers and be accompanied by training courses. He also noted that *performance* is no longer a barrier to serious use of functional languages. Functional languages are within a factor of two of C, as a rough estimate. In fact Java became successful with significantly lower performance compared to C.

He ended his talk with discussing Pizza/GJ[1], an attempt to popularise functional language features by adding them to Java.

A lively discussion followed, where several mentioned that they would like to do applications, but that

- it is difficult to get academic credit for such work,
- they do not really know how to do it.

Others strongly opposed that viewpoint by saying that trying to build applications in cooperation with industry is a waste of time and would take us away from the work we should really do.

The afternoon started with Donald Loveland's talk on *Concerning the future of Automated Deduction*. The discussion was headed by Melvin Fitting. Don's talk summarized the key points of a report on the status and future directions of the area of automated deduction (AD) (available from `http://www.cs.duke.edu/AutoDedFD`). As automated deduction is an adjacent field to logic programming there may be some overlap in experiences and expectations.

According to Don Loveland, the AD area is at the same time both stronger than ever and facing diminishing support. Regarding strengths, practical results are now appearing such as an automated proof of an important previously unsolved problem (Robbins Algebra) and verification of correctness of real-world networks. The diminishing support (in the U.S.) comes from a loss of defence funding for verification research and a shift in the artificial intelligence focus of research support. The reasons are that the progress in verification has been too slow and that the symbolic computation paradigm is no longer central in AI (but competes with numerical/statistical paradigms.

The U.S. funding agencies seek a focus on the near term with attention to commercial applications and applications in other domains. Don mentioned several applications, both in verification of networks and limited-domain program synthesis. Don also stated the need for more steady long-term funding to produce further progress such as is now evolving. Identifying a grand challenge problem would help provide such support. A list of technical needs and challenges was provided and included the need for more user-friendly interfaces.

The last talk, chaired by V. S. Subrahmanian, was given by Raghu Ramakrishnan: *Deductive Databases: Achievements and Challenges*. Raghu complained in his enthousiastic talk about the lack of applications of DDB but also noted that they influenced real DBMS (via optimisation techniques and some language constructs). However, entering the commercial mainstream takes place via SQL (DB2, Oracle, SQL3) with almost no recursion.

He mentioned various achievements like clear definition of *semantics* (positive, stratified, unlimited), *optimisation and evaluation* (semi-naive, magic sets, memoizing) and he emphasized that in the area of DDB, people did not stop with writing papers, they also did a lot of implementations. As essential and genuine features of DDB, Raghu considers

- *recursion*,
- *rule form*,
- *extended support for integrity constraints*.

He went on with pointing to non-trivial applications, like *Bill-of-materials*, *Flight databases*, *Analysis of Software Dependencies*, *Rendering-by-example project* and with strongly expressing that these are not enough. We need to see real applications that can benefit from the added features. In fact only broad applications using the added capabilities in meaningful ways are convincing. Raghu asked whether (1) there is no real need for negation, (2) companies are reluctant to use nonstandard tools and (3) the implementation techniques are mature enough.

[1] `http://www.cs.bell-labs.com/~wadler/pizza/`

The current status of DDB is that there are many prototypes around but yet no finished commercial products. Raghu claimed that DDB capabilities taken alone will not sustain a DBMS product but that such capabilities can be added to existing relational DBMS.

The discussion again turned around the importance of applications and that they should influence the theoretical system design. Others still opposed. VS introduced the importance of changes, updates, materialized views and uncertainty in DDB.

The last day of the meeting, Monday, saw three lectures. Vladimir Lifschitz started with *Representing causation by logic programs*, Veronica Dahl talked about *The Logic of Language* and finally Kazunori Ueda lectured on *The next 10 Years of concurrent Logic Programming*.

Vladimir Lifschitz, chaired by Teodor Przymusinski, considered in his talk two important knowledge representation tasks where LP-technology can and should be applied: *reasoning about actions* and *causal dependencies* (not involving actions). He pointed to important work done in the last 5 years. He then went on to planning problems and cited the work of Kautz/Selman: they transformed planning problems into satisfiability problems and were able to solve the latter problems by propositional theorem provers. Recently, a similar method was invented by Nebel/Köhler/Dimopoulos. They mapped planning problems into logic programs with negation in such a way, that stable models of the translation correspond to valid plans of the original problem. Moreover they used a general purpose nonmonotonic theorem prover, *smodels* from Niemelä/Simons, to successfully compute the stable models.

Vladimir then went on to introduce and discuss the action language $\mathcal{C}$, which contains fluents, actions as well as static laws of the form *"caused F if G"* and dynamic laws of the form *"caused F if G after H "*. He then explained the relation to LP: sets of laws correspond to extended logic programs. Vladimir finally suggested to consider how all this can be applied to extend the work of Nebel et. al. and to apply smodels as a nonmonotonic theorem prover.

In the discussion, Teodor pointed that a particular planning problem discussed can be solved faster by brute force than by using the proposed computational mechanism. Some people also distinguished between planning from first principles (corresponding to clean theory) and planning from second principles (constituting real applications): these approaches have different underlying methodologies.

Veronica Dahl was chaired by Dale Miller and she argued in her talk that logic is an inseparable part of language and can play an important role in natural language processing. She first examined families of formalisms from linguistics and computational linguistics, drawing their connections to logic. She also considered a linguist's wishlist in terms of programming language features that would make his/her task straightforward (*scalability, debugging, avoid left recursion, easy ways to change parsing strategies*). In the last part of her talk, she asked how to regain the NLP market beyond prototyping and how LP can benefit from this. She suggested to speed Prolog up, to incorporate features from the wishlist and she strongly emphasized to go out into industrial applications. Although programming in C is now much more fashionable, maintaining large C-systems can perhaps turn out to be very difficult.

Veronica finally discussed *Assumptive Logic Programming*—LP augmented with linear, intuitionistic and timeless assumptions, and presented a few of its recent applications, such as controlling virtual worlds and robots through natural language.

In the discussion Frank Pfenning and Dale Miller argued to use linear logic: changing states can be nicely represented within linear logic.

Kazunori Ueda gave the very last talk of the meeting and was chaired by Catuscia Palamidessi. He started with giving a review of the last 13 years of concurrent logic and concurrent constraint programming and its many achievements. He argued that the power and the simplicity of concurrent languages comes from *logical variables as communication channels*. Among the failures and problems in this area he mentioned the absence of good textbooks and the difficult and tedious way to learn the foundations.

His own view of concurrent LP is that of *LP + directionality* (or *LP + embedded concurrency control*) rather than LP–completeness. There seems to be an impression in the LP community that concurrent LP has not much in common with the original LP, but he stressed that this was not the case and both could benefit from each other technically. He also compared the roles of CCP with/without static analysis to the roles of untyped/typed lambda-calculus. Among the great challenges in this field, Kazunori emphasized to

– find a *lambda-calculus* which serves as a common platform for non-conventional computations (parallel, distributed embedded, real-time, mobile ...), and
– develop *type systems* (in the broadest sense) for both logical and physical properties.

3. Panels

The two panels were scheduled for Sunday, Panel 1 before dinner, Panel 2 just after it. Panel 1, *Logic Programming and Software Engineering: has anything gone wrong?* was chaired by Krzysztof Apt and consisted of 10-15 minutes contributions of Howard Blair, Leora Morgenstern, Michael Gelfond and Alan Robinson. The fifth panelist, Ken Bowen, had to cancel his participation but sent his comments per email.

Howard started the panel with some contributions of Ken Bowen. Ken argued that Logic Programming has failed to carry through on its logical promise. The direct basis upon logic promised development of powerful programming systems supporting the evolution of powerful tools. The development stopped short of real systems. Very few powerful and promising analytical tools that have been described in papers and theses have evolved into actual programs in use on a daily basis. Ken also complained about the disappointing attempts of an ISO PROLOG standard: many people were unwilling to contribute and instead invented whole new language systems not previously even implemented.

Howard argued that powerful algorithms work because of nontrivial mathematical organizing principles, e.g. *Simplex-Algorithm*, *Rivest/Shamir/Adleman* or *Fast Fourier Transform*. Twenty years ago there was an expectation that LP would draw deeply from classical mathematical logic to import methods that would enable not only powerful programming tools, but powerful new algorithms. Some partial success for LP has come about through drawing connections to operations research: *constraint logic programming*, *applications of linear-* (*and integer-*) *programming techniques*. LP has also helped drive increasing understanding of nonmonotonic reasoning in ways that draw upon classical logic. Still, after resolution, which sits at the foundation, LP has not yielded an algorithm in the same league as RSA.

He then pointed to *resolution* which represents a culmination of a line of research that stretched back to Skolem, and even to prior work. There is a thread that can be seen from Skolem through Herbrand, Gödel, Henkin, Prawitz, Davis/Putnam, and finally reaching its fulfillment with Robinson. There is here represented a focused persistence that stretched through 50 years. Computer Science in general lacks that spirit in the current era. Howard's proposal is to strengthen the connections to mathematics and operations research in particular. Such methods establish a conduit between LP and the mathematical fixed-point community, which had its world congress in Athens in 1996 (with excess of 1000 attendees) many of whom know a lot about very powerful algorithms. He also pointed to interesting applications such as the design and use of *Hybrid systems*.

The second speaker was Leora Morgenstern of IBM Research, who applies nonmonotonic reasoning techniques to commercial problems. She has developed an expert system for benefits inquiry in insurance. The system extends the nonmonotonic inheritance networks of Horty et al. (1987) by allowing the attachment of wellformed formulae at nodes. This allows more general nonmonotonic reasoning with nice computational behavior.

Leora discussed why researchers and industry are so far apart. (1) academics and industry often focus on different problems: toy problems like Tweety vs. specific business problems. (2) there is often a gap between a prototype developed in academe and industrial-strength tools. (3) solving business problems may require solving hard problems outside of NMR. For example, formulating business rules runs into the KR bottleneck. (4) logic is hard—for students to learn, and for people in business to understand. (5) academics have not been enterprising in marketing and public relations.

Leora emphasized that toy problems are a necessary part of research, but should be augmented by some commercial problems. She emphasized that there is no point in working on boring commercial problems. Rather, one should choose commercial problems where the relationship between academe and industry is mutually beneficial: academe offers a better way of solving the problem; while examining the problem in industry opens up an interesting area of inquiry, and leads to new results.

Michael Gelfond went on and asked *how to build knowledge bases containing recursive definitions, defaults and epistemic operators*. Modern software systems often need to combine the ability to represent knowledge about the system's domain with the ability to implement various procedural algorithms which require access to this knowledge. In the last ten years Michael claimed that substantial progress in making logic programming suitable for this task evolved. There now exist expressive logical languages with clearly defined non-monotonic entailments, solid mathematical foundations which allow to prove correctness, termination, and other important properties of logic programs. These languages are capable of representing *recursive definitions*, *defaults with exceptions*, *epistemic statements* and other sophisticated forms of knowledge.

Some progress was made also in developing methodology for representing knowledge but more work is needed in this direction. Careful analysis of simple but representative examples will be very useful in this respect. We need to find elegant ways for accomplishing simple tasks, e.g. counting the numbers of elements in the database satisfying some property P. At the moment we have neither clear semantics nor efficient implementation of $setof$ or other aggregates used for this type of problems. Michael complained that not enough attention was paid in the past to a clear separation between procedural and declarative aspects of logic programming: systems which cleanly combine negations, constraint satisfaction algorithms, abduction, etc. are needed.

Michael believes however that a comparatively small group of people working in logic programming and deductive databases can be proud of its achievements.

Finally Alan Robinson, in his well-known entertaining and amusing style, gave us his own perception of the development of programming languages and of his personal use of them in the last 40 (!!) years. He started in 1956 with assembly languages, switched to Fortran in 1958, to Algol in 1964 and (with a small intermezzo of Pop2) into Lisp in 1968. McCarthy convinced him to use Lisp by implementing his resolution procedure in 2 hours. Alan outed himself by admitting that he has never written a Prolog program (but he added that "neither has Bob Kowalski", which led Bob to contradict). He concluded with saying that for him, personally, at present *nothing* has gone wrong. But he immediately relativized this statement by saying that once, when he told people that he is a strong believer of freedom and therefore does not like types (λ-Prolog), they stated "but you can use them to fix your errors". His immediate answer: "Which errors?"

In the discussion a lack of public domain standard Prolog was pointed out as a main stumbling block in using Prolog for commercial projects. Further, lack of work on modules and types in the logic programming framework was mentioned (though refuted by some, for example by pointing out the work of Dale Miller). Others pointed out that speed considerations are nowadays hardly an issue, citing Java that is some 10 times slower than C++. Paul Tarau asked if we need a new Paradigm-Shift. Frank Pfenning argued that programming in the large has not been addressed in LP and that Methodologies of programming are needed, instead of new incremental add-ons of current systems.

After the dinner, the second panel on the future of LP took place. The speakers Jürgen Dix, Maarten van Emden, Ray Reiter and Koichi Furukawa were chaired by Wiktor Marek.

Jürgen Dix started by complaining about the lack of real applications and the lack of serious systems. He pointed to a similar meeting at the same place, four years back, where the same arguments were discussed, but nothing spectacular happened since. In fact the home page of LP and NMR related implemented systems (`URL:http://www.uni-koblenz.de/ag-ki/LP/`), established two years ago, still contains only a dozen systems among which only 2 are serious systems usable in real applications. Jürgen pointed to two particular promising applications namely *planning* and *diagnosis*. In recent work it has been showed that clever encodings of (i) planning problems into normal logic programs and (ii) diagnosis problems into positive disjunctive logic programs exist in such a way that general purpose nonmonotonic provers ("smodels" of Niemelä/Simons for (i) and "Protein" of the Koblenz group) can compete with dedicated planning resp. diagnosis systems.

He also pointed to applications of LP for intelligent Web-querying: while the system kernel could be Prolog, mediators can be implemented as extended logic programs (V. S. Subrahmanian used in his HERMES system annotated logic programs, others do it with disjunctive logic programs). A recent application of the "smodels" prover is in program verification, where "smodels" is used as a fixpoint engine for model checker based on μ-calculus.

Maarten van Emden recalled the expectation of logic programmers in the late seventies, namely that logic programming could unify computer science. This expectation was founded on the observation that logic programming is a good framework for topics as diverse as: *formal grammars*, *relational databases*, *program verification*, *term rewriting*, *procedure-oriented programming*, *functional programming*, *dataflow programming*. An advantage of doing this is that these diverse areas would acquire a common language, theory, theorems, techniques. However, this has not become true.

Maarten argued that the opposite seems to have happened with logic programming. At that time any two logic programmers had plenty of interests in common. Now, logic programming has branched out into applications such as *machine learning* (via inductive logic programming), *knowledge representation* (via abductive logic programming), *natural-language process-*

ing (via logic grammars), *compilation* (via abstract interpretation), *operations research* (via constraint logic programming), *numerical analysis* (via interval constraints).

Nowadays, any two logic programmers are likely to be in different applications. Their only point of contact is logic programming. As this part is more or less mature, it is of less research interest than the application. So two arbitrarily picked logic programmers are less likely to have a common research interest than twenty years ago.

In this seemingly fragmented state, it is worth asking: do we still have a common approach to these diverse application areas? Maarten argued that the answer is yes. What sets LP-people apart is a preference for a more principled and fundamental approach. Perhaps we should say: a more *logical* approach to the subject at hand. This commonality is worth identifying and cultivating. It will be a worthy *Son of Logic Programming*.

Koichi Furukawa considered *Inductive Logic Programming* as an important and promising research area for Logic Programming. After giving a historical overview of the main highlights achieved in the last 7 years, he illustrated how ILP emphasizes the *inductive* aspect of the logic rather than *deductive* aspect. From a computational point of view, the new school of ILP, as represented by Stephen Muggleton, developed a procedure for performing induction by deductive computation plus very much simplified inductive computation. Furukawa-san argued that future research will and should extends in broad fields of Logic Programming. For example ILP should include non-monotonic reasoning, constraint logic programming, and try to integrate abductive logic programming and the event calculus into ILP.

Ray Reiter, finally made some heuristic observations about how nonmonotonic logics might contribute to improving the efficiency of commonsense reasoning. One of the early arguments in favour of nonmonotonic logics was that they allow one to fill in gaps in a knowledge base by making default assumptions, and that such augmented knowledge bases make reasoning more efficient (e.g. by reducing the need for proofs by cases). But all nonmonotonic formalisms have very high complexity, so there seems to be something wrong with this argument. However, in most cases in the literature, whenever a nonmonotonic theory can be *compiled* into an equivalent, monotonic logic (e.g. by solving the circumscription axiom), the result is a definition. Normally, definitions support efficient reasoning. So it seems that in many cases, nonmonotonic reasoning does lead to efficient commonsense reasoning.

The discussion after the panel fell short because both the panelists as well as the audience were quite tired. In addition David Pearce purchased some bottles of wine …

4. Miscellenea

The evening of the first day of the meeting was devoted to discussing the possibility of forming some kind of organization of North American logic programmers. The question of such an organization arose with the demise of the *International Logic Programming Symposium* (*ILPS*), which had previously served as a focal point for logic programming in North America. There was feeling that some organization was important to try to build relationships, research and otherwise, among North American LP researchers. The goals included increased mutual knowledge of research and research directions of groups in North America to allow for joint funding efforts, student exchange, etc. There was also some feeling that yet another organization was not necessary and we would better spend our time reaching out to other communities. The conclusion was that David S. Warren would form and coordinate a steering committee that would try to put together a proposal for how to proceed. Also we would organize a day or two informal workshop, probably to be scheduled around ICLP'99, at which North American researchers would provide broad overviews of their research directions and goals.

David Pearce, the network coordinator of Compulog Net (`URL:http://www.compulog.org/`) mentioned that many theoreticians from Compulog Net's academic nodes, have little exposure to the vast number of powerful and significant practical contributions the technology currently makes in business, industry and science. He drew attention to some of the current success stories and volunteered to collect pointers to these systems. In particular, Compulog Net is organising a one day industrial conference during JICSLP 98 in Manchester ((`URL:http://www.cs.man.ac.uk/~kung-kiu/jicslp98.html`) which will describe some of the best work being done in LP, CLP and ILP applications. The day should provide a vibrant mix of case-studies from some of the most promising application areas. There is also the following site `URL:http://www.compulog.org/net-www/forum.html`, providing information about the 2nd Strategic Planning Workshop to boost *Computational Logic*.

In addition Ken Bowen has recently compiled a list of industrial LP applications: please visit URL: http://www.als.com/nalp.html and click on *Applications*. A similar list consisting of applications of *functional progrmming* is maintained by Philip Wadler: URL:http://www.cs.bell-labs.com/~wadler/realworld.

Given the successes and the unique infrastructural role of *Compulog Net*, Gopal Gupta is working on and trying to put forward *Compulog Americas*, a *Compulog Net Europe* equivalent like structure. Interested people are referred to URL:http://www.cs.nmsu.edu/~complog.

5. Conclusion

Like four years ago, the workshop site was still very charming: a large outdoor museum embedded in the gentle hills of the countryside near the Kentucky and Dix rivers. The Shaker's erstwhile craftmanship is always nice to admire and there are not many possibilities to lodge in such a unique 19 century environment.

Wiktor Marek, Mirek Truszczyński and in particular Deborah East, Matt Levy and Inna Pivkina, did a wonderful job in organizing the event—the only drop of bitterness being the cold and rainy wheather during the last two days.

Postscriptum

Some two dozens pictures taken during the workshop can be accessed through COMPULOG Americas home page located at URL:http://www.cs.nmsu.edu/~complog (click on *Events*). Many thanks to Jacques Cohen and his digital camera (and to Enrico Pontelli for putting the pictures on the web)!

References

[1] K. R. Apt, V. Marek, M. Truszczynski, and D. S. Warren, editors. *The Logic Programming Paradigm: Current Trends and Future Directions*, LNAI, Berlin, 1999. Springer.

[2] Jürgen Dix. Detailed Report on the First LP & NMR retreat. *AI Communications*, Vol. 8, No. 1:39–43, 1995. Also in Computational Logic, Vol. 2, 1995, pages 94–97.