



UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Logic programming for knowledge-intensive interactive applications

Wielemaker, J.

Publication date
2009

[Link to publication](#)

Citation for published version (APA):

Wielemaker, J. (2009). *Logic programming for knowledge-intensive interactive applications*. [Thesis, fully internal, Universiteit van Amsterdam].

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

Preface

It was 1985, when I graduated from Eindhoven University. I had some options about what to do. Instead of picking one, I asked Bob Wielinga whether he had some short-term employment in the European ‘Esprit’ project that just had started. My plan was to wait for a nice opportunity to work on what I, then, considered a more fundamental artificial intelligence subject, such as machine learning. I started work on the KADS project, where Anjo Anjewierden tried to teach me the difference between hacking and programming. By studying and extending code from Anjo and Richard O’Keefe, whom I had met during a training period at Edinburgh University, I gradually discovered the beauty of proper programming. Anjo’s motto was “the ultimate documentation is the source”. For good source, handed to someone with basic knowledge of the programming language, this indeed comes close to the truth. Nevertheless, I am proud to have him as a co-author of a paper on a Prolog documentation system (chapter 8 of this thesis). The ‘art of programming’ had caught me and I abandoned my original plans doing AI research.

At least as important as the joy of striving for clear coding of large and complex systems, was the way Bob Wielinga dealt with ‘scientific programmers’. Later, I realised that in most research groups this species was treated as a semi-automated code-generation facility. Instead, Bob involved Anjo and me in the research, gave us almost unlimited freedom and was always the first to try our prototypes. His never lasting energy to modify and comment on our work was a source of inspiration. I still wonder how Bob managed to demonstrate our prototypes at project reviews without making them crash.

During this period, I had plenty of time to play the Chinese game of Go and hack around on the University computer systems on anything I liked. This resulted in SWI-Prolog. Somehow, the developers in the KADS project quickly ignored the consortium decision to use the commercial Quintus Prolog system and started using my little toy. Well, possibly it helped that I promised a bottle of cognac for every 10 reported bugs. Although this resulted in endless arguments on whether a spelling error in a warning message was a bug or not, Huub Knops managed to earn his bottle. Anja van der Hulst joined SWI, and I soon had to deal with Anja, Go and SWI-Prolog. My support in realising a prototype for her research is reminded later in this preface.

At some point, Frank van Harmelen joined SWI and pointed us to great ‘free’ software, notably Emacs and L^AT_EX. Not really knowing what to do with SWI-Prolog, we decided to join the free software community and make it available from our FTP-server. We sold academic licenses for the accompanying graphical toolkit PCE, designed by Anjo and extended a bit by me. Selling software and dealing with legal issues around licenses is not the strength of a University, as Lenie Zandvliet and Saskia van Loo will agree. Sorry for all the extra work.

SWI-Prolog/PCE became a vehicle in the department to accumulate experience on building ‘knowledge intensive interactive applications’. At some point in the early 90s, a wider academic community started to form around SWI-Prolog/PCE. An invitation to work as guest researcher for SERC by Peter Weijland allowed to initiate the ideas that lead to chapter 5 of this thesis and the research on version management systems filled a gap in my experience Anjo never conveyed to me: the use of a version management system. Robert de Hoog never thought very highly of Prolog, but he needed Anjo to work on the PIMS project and convinced me to port the system to Microsoft’s operating systems. I first tried Windows 3.1, then Windows 95 and finally succeeded on Windows NT 3.5. It was no fun, but I must admit that it was probably the step that made SWI-Prolog a success: suddenly SWI-Prolog could run for free and without any strings attached on the University Unix networks as well as on the student’s home PCs. When search engines became important on the internet, pointers from many University course-pages to SWI-Prolog greatly helped making the system popular. So, thank you, Robert!

Although SWI-Prolog was supported by a large and growing community, I started worrying. Projects tended to move to other programming languages. Although Bob managed to organise projects such that we could deploy Prolog to create ‘executable specifications’, this was a worrying development. Fortunately, a number of opportunities came along. Projects moved towards networked component-based designs, which allowed partners to use their language of choice instead of having to agree on a single language. At the ICLP in Mumbai (2003), Tom Schrijvers and Bart Demoen came with the deal where, with their help, I would provide the low-level support for constraint programming and their Leuven-based team would make their constraint systems available on SWI-Prolog. Without this cooperation SWI-Prolog might have lost contact with the logic programming community.

In the meanwhile SWI-Prolog started to attract serious commercial users. One came as a surprise. Considering the shift towards networking, I had added concurrency and Unicode to SWI-Prolog. Suddenly I was bombarded by mails from Sergey Tikhonov from Solvo in St. Petersburg. Without his bug-reports and patches, multi-threaded SWI-Prolog would never have become stable and this thesis would not have existed. Steve Moyle helped to turn this into a paper (chapter 6) and his company funded the development of SWI-Prolog testing framework and documentation system. Of course I should not forget to mention his elaborations on Oxford pubs and colleges. Mike Elston from SecuritEase in New Zealand funded several projects and his colleagues Keri Harris and Matt Lilley have sent me many bug reports and fortunately more and more patches. I was pleased that he and his wife unexpectedly showed up at our doorstep a year ago. They showed me and Anja parts of

Amsterdam that were unknown to us.

The young and brilliant student Markus Triska showed up to work in Amsterdam during the summer. Unfortunately the apartment I managed to arrange for him turned out to house some companions in the form of flees, so at some point he preferred the University couch. Nevertheless, this didn't break his enthusiasm for SWI-Prolog, which resulted in various constraint programming libraries. He also managed to interest Ulrich Neumerkel. I will always admire his dedication to find bugs. His work surely contributed in realising 24×7 Prolog-based services.

Unfortunately, I cannot acknowledge everybody from the Prolog community. I will name a few and apologise to the others. Richard O'Keefe shows up again. He has followed the mailinglist for many years. Without his comments, the list would be much less informative and less fun. Then we have Paulo Moura, always trying to keep the herd of Prolog developers together. Considering that Prolog developers are more like cats than sheep, this effort cannot be left unacknowledged. Vitor Santos Costa's open mind, combined with some pressure by Tom Schrijvers has resulted in some level of cooperation between YAP and SWI-Prolog that we want to deepen in the near future. Paul Singleton connects Prolog to Java and hence allows me to survive in this world.

The most important opportunity for me was created by Guus Schreiber, who managed to participate in the MIA and MultimediaN projects on search and annotation of physical objects. These projects were run by an inspiring team from SWI, CWI and the VU. Guus' choice to use the emerging Semantic Web created a great opportunity for deploying Prolog. While I tried to satisfy their demands on the infrastructure, Michiel Hildebrand and Jacco van Ossenbruggen did most of the Prolog programming that created a price-winning demonstrator at ISWC-2006. More important for this thesis, these projects provided the opportunity and inspiration to write scientific papers. These papers, and especially the one published as chapter 10 made this thesis possible.

Writing software isn't always easy to combine with writing papers or a thesis. Somehow these two tasks require two incompatible states of mind. As there was no obvious need to write a thesis, many evaluations of my functioning at the University contained some sufficiently vague statement about a PhD, after which I and Anja had a good laugh and another year had passed. Until somewhere fall 2007, I was in the local pub with Bob Wielinga and Guus Schreiber. After some beers I ventilated some bold statements about the Semantic Web and Guus commented "you should write this down in a paper and then we can combine it with some of your recent papers and turn it into a PhD". Not completely sober, I turned home and told Anja: "They want me to write a PhD." Anja started laughing hilariously, expecting me to do the same. I didn't. Anja stared at me, confused and then slightly worried. This was not the standard act. When I asked for her support she replied: "Well, if you really want it, you should.", to continue with "As long as I don't have to read it." It was her well deserved revenge for my attitude towards reading specifications for software prototypes.

Next to work-wise, SWI has always been a great group to work in. For me, it started on the top-floor of what is now known as 'het Blauwe Huis' on the Herengracht in Amsterdam. SWI was a gathering of great people with whom I have spent many hours in the pubs nearby.

From there we moved to the top-floor of the psychology building at the Roetersstraat. I will not forget the discussions in café Solo with Frank van Harmelen, Gertjan van Heijst, Manfred Aben, Dieter Fensel and many more. I have many good memories of the colleagues with whom I shared an office. John van den Elst was very noisy, so after a month I was happy he would be in Antibes for the next month and after a month in an empty office I was happy he returned, etc. Luckily Chris van Aart, though we never shared offices, made the months without John bearable with his admiration for Java. Sharing the office with Dennis Beckers and Hedderik van Rijn was great. In that period I lived in Leiden and many Wednesdays I enjoyed a *pizza quattro formaggi* with Carolien Metselaar in Palermo before visiting the Amsterdam Go-club. Noor Christoph learned me that finishing a PhD implied there was never a dull moment.