Summary of the Applications of Prolog meeting 20 April 2022

It was agreed to hold another open meeting in 2-3 weeks,
at the same time on 11 May, 17:00 in Paris, 11:00 in New York.
Here is the link:

https://us02web.zoom.us/j/83218355848?pwd=VUk3cGNPODZudE1JQWpNVis2L2U5QT09

Attended by Mats Carlson, Laurent Cervonni, Veronica Dahl, Laurent Gouzenes, Manuel Hermenegildo, Bob Kowalski, Annie Liu, Paola Mello, Enrico Pontelli, Clive Spencer, Theresa Swift, Peter Szeredi, Paul Tarau, David Scott Warren, Jan Wielemaker.

Apologies from Herve Gallaire, Maria Garcia de la Banda, Evelina Lamma.

The meeting included a summary of Progress on the Year of Prolog. Here are links to the Announcement of the Year and the Call for submissions for the Alain Colmerauer Prize:

https://prologyear.logicprogramming.org
https://prologyear.logicprogramming.org/ColmerauerPrize.html

The Prize will be awarded at a Prolog Day to be held on 10th November 2022 t Université Paris Cité, Campus Saint-Germain-des-Prés - 45, rue des Saints-Pères - 75006 PARIS.

The main part of the meeting was a brainstorming session with an introduction by Laurent Gouzenes of his applications study at:
https://www.km2-conseil.fr//PROLOG/PROLOG.htm
(If you open the link in Chrome, and right click, you can use Google translate to translate the document into English or French.)

In particular, Laurent recommended that we add to the 1995 database of Prolog applications at
https://www.km2-conseil.fr//PROLOG/DOCS/APPLICATIONS/VIEILLES%20APPLICATIONS/PROCESSING%20OLD%20APPLICATIONS%20IN%20PROLOG.htm

Many of the attendees gave examples of recent applications of Prolog.
In addition, several general observations were made.
For simplicity, I will organise the report according to my record of contributions by individual participants:

Jan: mentioned two new projects: Health care using S(CASP), and an implementation of SPARCL in Prolog competing with an alternative Javascript implementation. Jan mentioned that one of the advantages of the Prolog implementation is that it makes it easier to perform

optimisations. Jan also mentioned another application in robotics, which is secret.

Manuel: reminded us of the Applications of Prolog Conferences organised by Al Roth in the 1990s, and suggested that maybe we should consider resurrecting them. He also mentioned two classes of applications with which he has been involved: program analysis processing tools and planning industrial products.
He argued that for the database of Prolog applications it is not so important to have a large quantity, but to have a high quality.

David: agreed that it would be useful to collect information using a form. But he pointed out that much of the information might be very incomplete, because many of the applications are secret.
He mentioned that many of the XSB applications involve ontologies, extraction of information and supply chains.

Many applications are written in Prolog, and then some of the peripheral parts are implemented in more conventional languages, while the core remains in Prolog

He argued that there should be a website for applications, and that we could ask ALP to use the ALP website for that purpose.

Theresa: pointed out that there has been a lot of recent collaboration between the developers of Prolog systems, including BBN, XSB and SWISH, for example involving the inclusion of tabling in SWI.

She mentioned several recent projects, for example using a combination of NLP tools, such as SPACY, and XSB to extract information. In many of the applications, Prolog is used for orchestrating other components. Similar applications have been implemented in FLORA and Ergo-AI, based on XSB.

She also mentioned that there has been a long-standing suite of applications for US Customs over the past 20-30 years. She noted that there has been a shift from Prolog and Oracle to Java, mostly because it is hard to find Prolog programmers.

Enrico: suggested that there are applications of Prolog, but in other non-LP communities. Many applications hide their use of Prolog. And many other applications use Prolog-like rules instead of using Prolog explicitly.
Bob pointed out Oracle Policy Automation as an example:

https://documentation.custhelp.com/euf/assets/devdocs/cloud19b/PolicyAutomation/en/Default.htm#Guides/Welcome/Welcome.htm

Laurent C: asked about Constraint Prolog. Manuel and others argued that constraints, are now features of virtually all modern Prologs. Many Prologs now also include tabling and concurrency.

Paul: suggested that Prolog would be more popular if LP hadn't been removed from the ACM curriculum. But interest in Prolog is increasing, perhaps in the context of NLP applications in collaboration with machine learning.

Mats: mentioned that none of his Prolog customers have agreed to acknowledge that their applications are written in Prolog. There are two interpretations for this: Either they believe there is a stigma associated with Prolog because of the perceived failure of the Japanese Fifth Generation Project, or Prolog is a trade secret, which is too good to be acknowledged. Gopal (later) suggested that the FG Project was so long ago that its impact on the perception of Prolog is no longer significant.

Mats has assisted with the optimisation of his customers' programs, but has not been involved with their development, He has mostly worked with constraint programming.

Annie mentioned several applications at the end of the meeting, and later sent some details:

1. IBM Watson question answering system, where application in NLP helped Watson win the Jeopardy Man vs. Machine Challenge by defeating two former grand champions in 2011.

A. Lally, J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll. 2012. Question analysis: How Watson reads a clue. IBM Journal of Research and Development, 56(3/4): 2:1-2:13.

A. Lally and P. Fodor. Mar. 31 2011. Natural language processing with Prolog in the IBM Watson system. Association for Logic Programming (ALP) Issue, Featured Articles.
\url{http://www.cs.nmsu.edu/ALP/2011/03/natural-language-processing-with-prolog-in-the-ibm-watson-system/}.
Accessed April 23, 2015.

2. Microsoft Research's work on network verification. I learned from Andrey Rybalchenko, a few years ago, that he mainly uses SICStus

Prolog, and their program could run a month to do verification.

Andrey Rybalchenko
https://www.microsoft.com/en-us/research/people/rybal/

He has various papers, and I found this one mentions SICStus Prolog:
Gleissenthall KV, Köpf B, Rybalchenko A. Symbolic polytopes for quantitative interpolation and verification. In International Conference on Computer Aided Verification 2015 Jul 18 (pp. 178-194). Springer, Cham.
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/main-2.pdf

(He had also papers before joining MSR, e.g.
Applying Prolog to Develop Distributed Systems
Nuno P. Lopes, Juan A. Navarro, Andrey Rybalchenko, Atul Singh
Theory Pract. Log. Program. 10(4-6): 691-707 (2010)
https://arxiv.org/abs/1007.3835)

3. Google's Logica last year (after it's Yeda several years back)
https://opensource.googleblog.com/2021/04/logica-organizing-your-data-queries.html

https://logica.dev/
"Logica compiles to SQL and gives you access to the power of Google BigQuery engine with the convenience of logic programming syntax. This is useful because BigQuery is magnitudes more powerful than state of the art native logic programming engines."

4. Erlang was originally developed at Ericsson but has been used in many other places after it became open source. It supports super-fast concurrency.
https://en.wikipedia.org/wiki/Erlang_(programming_language)

It started from Prolog. https://www.erlang.org/faq/academic.html
"10.3 Where does Erlang syntax come from?
Mostly from prolog. Erlang started life as a modified prolog. ! as the send-message operator comes from CSP. Eripascal was probably responsible for , and ; being separators and not terminators."
"10.6 How did the first Erlang compiler get written?
(or: how was Erlang bootstrapped?) In Joe's words:
First I designed an abstract machine to execute Erlang. This was called the JAM machine; JAM = Joe's Abstract Machine.
Then I wrote a compiler from Erlang to JAM and an emulator to see if the machine worked. Both these were written in prolog.

At the same time Mike Williams wrote a C emulator for the JAM. Then I rewrote the erlang-to-jam compiler in Erlang and used the prolog compiler to compile it. The resultant object code was run in the C emulator. Then we threw away prolog."

Clive: sent an email after the meeting:

It's not just Prolog that is suffering as a programming language: https://www.quora.com/Why-isnt-Smalltalk-widely-used-anymore

Lots of our customers recode out of Prolog as the hard bit is often early on --- i.e. what are we really trying to do -- one you have that figured the rest is easy?

Is there anything unique about Prolog? As in something which can only be done using Prolog? No, as I understand, all languages are about the same in terms of what they can do, it's just a lot easier in a high-level type-free declarative language.

Most applications I've seen are using Prolog because it's convenient; but really Prolog is a logical reasoning language. So why do SemWeb people use OWL? What about AllegroGraph and the Prolog reasoner? They have a huge following:
https://allegrograph.com/new-york-times-article-is-there-a-smarter-path-to-artificial-intelligence-some-experts-hope-so/

Prolog can support meta-level programming, i.e. you can build a new language to code rules about a specific domain.

Biggest Prolog application? Watson - not so fashionable nowadays, but made IBM a billion dollar business division. Maybe try and get Dave Ferrucci to say something?
Kyndi (what's happening there - they raised 20m USD a while back?):
https://venturebeat.com/2019/07/10/kyndi-raises-20-million-for-explainable-ai-that-derives-insights-from-documents/

"Kyndi's secret sauce is a decades-old programming language called Prolog that was designed with probabilistic and logical computational reasoning in mind."

Can we talk about GraphStax -- I presume they use Prolog?
https://graphstax.ai/our-company.html

I come across lots and lots of bits of Prolog -- but nothing spectacular.

As Bob mentioned - some successful products have been reworked out of Prolog but still retain some of the hallmarks (like syntax or semantics)

Between 200 and 2013, we sold DealBuilder/ContractExpress to thousands of lawyers and generated hundreds of thousands of contacts; but Phil Vasey was told to learn C++ and recode it for political reasons (lack of programming resources being the main one).

MS told us that CE was one of the most stable and big-free applications they'd ever come across.

Now, we focus on VisiRule (at LPA); and yet some of our customers having done the hard bit (which is to tease our what the solution is and what the problem is), recode for various reasons:
https://www.visirule.co.uk/lpa-customers

Yes, Prolog was slow and unable to work with normal computing 'things' -- but those days are long gone. Our current implementation is run some 10 thousand times faster than when I joined LPA -- which makes me wonder how the hell did we ever build the early fault finding tool for BT or the very early version of XiPlus.

So what --- we want to somehow let new programmers now that Prolog is 'good' or at least 'alive'

What about PDC Prolog - I don't know if they qualify --- their scheduling software is used worldwide by most airlines. What about Datalog? I guess they're not Turing complete?

Also, you find many many papers in formal specification and verification where they have rules which adopt a Prolog syntax but are implemented using Java or C etc

And then finally what about our VisiRule users? They may dabble inadvertently with bits of Prolog (say like is/2), but for the most part they simply draw their logic and we generate the executable rules in the form of Flex (our expert system language), which in turns compiles into Prolog.

The future -- sure neuro-symbolic hybrid reasoning -- but everyone is rushing there so it's hard to position in the herd.

PS: we got David Hovel to come and present at PAP 25 years ago and describe how Prolog was used in Windows NT : https://web.archive.org/web/20040603192757/research.microsoft.com/research/dtg/davidhov/pap.htm