A second life for Prolog

What went wrong and how we fixed it

Jan Wielemaker
J.Wielemaker@cwi.nl
Overview

• Now: invited talk
  • WWW: Why Prolog, Why not and Why again
• Afternoon (17:50 – 19:10) Tutorial 1
  • Introducing Prolog, the simple stuff, beyond SLD

• Tomorrow morning (08:00 – 10:00) Tutorial 2
  • Handling data, interface to the outside world
Why Prolog for language?

- DCG: A powerful grammar formalism
  - Unlimited look-ahead
  - Non-deterministic (can provide multiple parses)
- We can capture the semantics of language in logic
  - This allows us to reason about language
  - Translate, ...
- [https://swish.swi-prolog.org/example/grammar.pl](https://swish.swi-prolog.org/example/grammar.pl)
Does it work?

- To some extend
  - Artificial languages (document formats, computer languages)
  - Controlled natural language (e.g., ACE)
  - Natural language in limited domains (e.g., Watson)
Real natural language?

- We all know it doesn’t. Why not?
  - Top-down parsing comes with too many choicepoints (slow)
  - Long sentences produce too many possible parses (choose)
  - Languages with free word ordering are hard to express (expressivity)

- Or does it?
  - Alpino (Dutch parser) is still one of the best parsers for Dutch.
  - Hybrid: A Prolog representation is compiled into a finite state machine and a statistical model is used for disambiguation. Overall control is again in Prolog.
Graph exploration

daughter(Daughter, Parent) :-
  parent(Parent, Daughter),
  female(Daughter).
Great!

- Concise description
- Works in all directions:
  - Create a table of all daughters and their parents
  - Find the daughters of a parent
  - Find the parents of a daughter
  - Verify a specific daughter is the daughter of a specific parent
- Is pretty fast
But ...

- Now we do travel planning, traditionally by railway!
- You can use a connection in two directions
- You can travel around in circles without ever reaching your destination
- The number of connected tracks is pretty huge

⇒ Prolog looses its declarative beauty!
What to do?

- Prolog is a programming language, so we can code a proper solution!
- **Extend** the inference mechanism of Prolog, so we can still use the declarative version!
- Restrict ourselves to domains that do not suffer too much from this issue (**special purpose language**)
Coding using SLD resolution

```prolog
travel(S1, S2, Route) :-
    travel_bf(S2, [S1-[S1]], Route).

travel_bf(To, [To-Route|_], Route).
travel_bf(To, [S-Route0|T], Route) :-
    findall(S1-[S1|Route0], (adjacent(S,S1), \+member(S1,T)), New),
    append(T, New, Agenda),
    travel_bf(To, Agenda, Route).

adjacent(S1, S2) :- connected(S1, S2).
adjacent(S1, S2) :- connected(S2, S1).

connected('Warshau', 'Poznań').
```

Break cycle
Coding using SLD resolution

- Can implement any algorithm
- Is typically still compact compared to alternatives

(Debugging)
- We can retry (time machine)
- Harder to follow control flow

✗ Steep learning curve if you come from an imperative background
Beyond SLD

- SLG (Tabling)
  - Terminates provided finite data structures are used
  - In some sense comparable to DataLog
- Constraint Logic Programming
  - Use domain knowledge to reorder search and be smarter than generate-and-test for finding possible values

➡ Declarative islands
Prolog as a special-purpose language?

- Can solve isolated, relatively small and simple problems
- For many of these, there are subsystems in other languages
  - Parser generators
  - Rule subsystems
  - ...
- Embedding Prolog suffers from the **relational impedance mismatch** that also complicates using relational databases from many languages.

➔ Still, **Amzi!** targets this
Use Prolog as a specification language

- Explore using interpretation
- DSL for problem domain
- Compile
- Interface
- FSA
- Graph lib
- Neural net
- C/C++
- DB
- Prolog
Specification language

- Flexible syntax that is targeted at data
- Grammars are great for generating code

Examples
- Alpino (we have seen)
- Weather prediction (university Leiden)
- Natural language understanding (Kyndi)
- Business rule management (SecuritEase)
- ...
Using Prolog as glue

- As we have seen
  - Prolog can accommodate _declarative islands_
  - Prolog can be used to _generate_ problem specific _code_
- Prolog has a natural fit with
  - Relational data (RDF and RDBMS)
  - Hierarchical data (XML, JSON, etc)
But ...

- Traditional Prolog is a little autistic
- Only file I/O
- Poor representation for text
- Poor representation for arrays
- Often painful embedding support
SWI-Prolog

- Language
  - Scalable support for multi-core hardware
  - Unicode support, unlimited length atoms, volatile compact strings
  - Unbounded arity for terms provides arrays
  - Dicts (key-value objects)
  - Scalable dynamic database with lazy indexing
  - Security and garbage collection (atoms, clauses, stack)

- Connections
  - Strong web server and client libraries
  - Connections to languages and databases
  - Parse and write document formats (RDF, XML, HTML, JSON,...)
Take home

✗ Classical Prolog as a declarative language has limited value

✔ Modern Prolog offers more powerful declarative subsystems that can be used as declarative islands

✔ Prolog is a great data representation and specification language

✔ Prolog is great in providing a unifying framework for a hybrid technology stack.