We mentioned this previously: "Prolog arose from work on automated theorem proving"

1965: "Alan Robinson published a paper introducing the idea of theorem proving based on resolution" <-- "...the foundation of Prolog"!

– led to much work "around the world on resolution-based theorem-proving";

– BUT -- there are MANY "possible algorithms for automated inference based on [this idea of] resolution, with widely varying degrees of power and efficiency" -- "...Prolog did not arise immediately";
Prolog - a little more history, 2
[source: Webber, "Modern Programming Languages", pp. 544-546]

- "Several researchers saw the connection between automated inference and general computation,
  - and observed that the behavior of theorem provers could parallel the behavior of programming language interpreters."

- But, they tried to get these "theorem provers to prove impressively difficult things"
  - rather than on "the simple computational things that Fortran and Lisp could already do" [we're in the mid- to late-1960's now, I think]
Prolog - a little more history, 3
[source: Webber, "Modern Programming Languages", pp. 544-546]

- 1971 - Alain Colmerauer's group - Universite d'Aix Marseilles
  - "working on an artificial intelligence [AI] project:"
  - "a system to answer questions about natural language texts"

- This system needed automated deduction:
  - e.g., "if the text said that Jerry is a mouse,
  - and that mice eat cheese,
  - the system needed to answer the question, does Jerry eat cheese?"

- They were using a resolution-based technique for the automated deduction part;
Prolog - a little more history, 4

[source: Webber, "Modern Programming Languages", pp. 544-546]

• 1971 - Alain Colmerauer's group - Universite d'Aix Marseilles - continued

• They invited Robert Kowalski -- University of Edinburgh -- to visit, and he explained his resolution theorem prover
  – Kowalski's technique: SL-Resolution
  – "Philippe Roussel implemented a simplified version of SL-Resolution for the first Prolog system in 1972."

• "The name Prolog was suggested by Roussel's wife, as a derivation of programmation en logique (and we've already seen that at least some English-language references give this in English, "programming in logic")
Prolog - a little more history, 5
[source: Webber, "Modern Programming Languages", pp. 544-546]

- "Colmerauer and Roussel found that the system could be used for their **entire application**, not just for the deductive part;"
  - "It was a general-purpose programming language."
  - .... "the 1973 version looked much like modern Prolog."
- Early versions were **interpreted**, "and were extremely slow and memory intensive";
  - "In 1977, David Warren at Edinburgh developed the first Prolog **compiler**" 
  - "In 1983, he developed an important compilation technique for Prolog: the **Warren Abstract Machine**"
Warren Abstract Machine


- [Webber] The Warren Abstract Machine is "an intermediate-code target for Prolog compilation which is still used in some form by many Prolog compilers (including SWI-Prolog)."

- [Wikipedia] "The purpose of compiling Prolog code to the more low-level WAM code is to make subsequent interpretation ... more efficient"
  - "reasonably easy to translate to WAM instructions which can be more efficiently interpreted"
  - (what other language does this remind you of?)
  - can read more about the WAM in an MIT Press tutorial available on-line, "Warren's Abstract Machine", by Hassan Ait-Kaci: www.cvc.uab.es/shared/teach/a25002/WAMBOOK.PDF
Prolog - a little more history, 6

[sources: Webber, "Modern Programming Languages", pp. 544-546]

- "The availability of compiled implementations,
  - and the commercial success of various expert systems implemented in Prolog,
  - helped Prolog find a wider audience in the 1980's."

- "It remains an important language for artificial intelligence development"
Prolog - a little more history, 7

[source: Webber, "Modern Programming Languages", pp. 544-546]

- "Like Lisp and Smalltalk, Prolog is a language that follows naturally from a small set of basic elements --- in Prolog's case, resolution-based-inference."

- quote from Colmerauer and Roussel:
  
  - "Prolog is so simple that one has the sense that sooner or later someone had to discover it."

- "Certainly, the connection between theorem-proving and programming occurred to several researcher before Prolog was born;"
Prolog - a little more history, 8
[source: Webber, pp. 544-546]

- Prolog's success "is due to an important insight about how to make the connection practical."

- In resolution-based theorem proving, it is "easy it is to come up with a correct but useless variant:
  – a theorem prover that wanders around proving exponentially many true things, but none to the point."

- "The difficult thing is to find [such] an algorithm ... general enough to be the basis of a programming language ... yet can be implemented efficiently enough to be [practical]."

- "[Amongst] logic languages ... Prolog is still the most successful."
The `is` operator is an **infix** operator, which takes an unknown ... on the left, and an arithmetic expression on the right."

consider:

density(Place, Density) :-
   pop(Place, Pop),
   area(Place, Area),
   Density is Pop/Area.

beware -- is float division in swipl, but not in ALL Prologs!
In the above example, Density is unknown when the is is encountered,

- and it is up to the is to evaluate the expression,

- and let Density stand for the value."

"This means that the values of all of the variables on the right of an is must be known."
why do we need `is`? - 4

[source: Clocksin and Mellish, pp. 33-35]

- "We need the `is` operator ... to tell Prolog to evaluate the arithmetic expression."
  - "...[to Prolog,] something like `Pop/Area` ... is just an ordinary Prolog structure like `author(emily, bronte)`.

- "With arithmetic expressions, there is a special operation that can be applied ...: that of actually carrying out the ... arithmetic"
  - "This is called evaluating the arithmetic expression."

- "Clearly we cannot evaluate structures such as the `author` one..."

- "So, we have to tell Prolog when we want it to attempt to evaluate a structure."

- "This is what the predicate `is` is for."
Prolog arithmetic and comparison operators

[source: Clocksin & Mellish, pp. 33-35]

- "Depending on what computer you use, various arithmetic operators can be used on the RHS of the `is` operator."

- "All Prolog systems, however, will have:

  \[
  \begin{align*}
  X + Y & \quad /* \text{the sum of } X \text{ and } Y */ \\
  X - Y & \quad /* \text{the difference of } X \text{ and } Y */ \\
  X * Y & \quad /* \text{the product of } X \text{ and } Y */ \\
  X / Y & \quad /* \text{the quotient of } X \text{ and } Y */ \\
  X \mod Y & \quad /* \text{the remainder of } X \text{ divided by } Y */
  \end{align*}
  \]

- also has comparison operators -- only one of which is a surprise!

  \[
  < \quad > \quad <= \quad >=
  \]

  (yes, that really IS `<=` instead of `<= ...`!)
List basics

[source: no-longer-available tutorial: http://www.cse.msu.edu/~cse440/Programming1/programming1 tut.html]

• a very common data structure in Prolog: the list

• basic list syntax:
  – start and end with square brackets
  – elements within are separated by commas
  – example of a list: [a, freddie, 13.7]

• the empty list: []
"Prolog ...has a special facility to split the first part of the list (called the head) away from the rest of the list (known as the tail)."

— Yes, it's car and cdr, again...! 8-)

"We can place a special symbol | (pronounced 'bar') in the list to distinguish between the first item in the list and the remaining list."

[first, second, third] = [A|B].
A = first
B = [second, third].

[First|Rest] = [1, 2, 3, 4, 5].
First = 1,
Rest = [2, 3, 4, 5].