

# Prolog Logic Programming

Young W. Lim

2018-02-29 Thr

- 1 Introduction
  - References
  - Getting Started

## "Logic Programming with Prolog" M. Bramer

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

# Hello World

```
$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
```

```
For help, use ?- help(Topic). or ?- apropos(Word).
```

```
?- write('Hello World'), nl, write('Welcome to prolog'), nl.
Hello World
Welcome to prolog
true.
```

```
?- statistics.  
% Started at Thu Feb 15 00:18:35 2018  
% 0.082 seconds cpu time for 136,679 inferences  
% 4,789 atoms, 3,136 functors, 2,211 predicates, 36 modules, 71,764 VM-codes  
%  
%  
%          Limit      Allocated      In use  
% Local  stack: 268,435,456      61,440      1,968 Bytes  
% Global stack: 268,435,456      61,424      8,840 Bytes  
% Trail  stack: 268,435,456      30,712      1,536 Bytes  
%  
% 1 garbage collections gained 35,344 bytes in 0.000 seconds.  
% Stack shifts: 2 local, 3 global, 3 trail in 0.000 seconds  
% 1 threads, 0 finished threads used 0.000 seconds  
true.  
  
?- halt.  
$
```

# Prolog Program 1

```
--- p1.pl -----  
dog(fid)  
cat(felix)  
animal(X) :- dog(X)  
-----
```

```
?- consult('p1.pl').  
true.
```

```
?- animal(fido).  
true.
```

```
?- animal(felix).  
false.
```

```
?- halt.
```

# Prolog Program 2

p2.pl

```
-----  
dog(fido). dog(rover).  
dog(tom). dog(henry).  
cat(harry). cat(marry).  
cat(bill). cat(steve).  
-----
```

```
?- consult('p2.pl').  
true.
```

```
?- dog(X).  
X = fido .
```

```
?- dog(X).  
X = fido ;  
X = rover ;  
X = tom ;  
X = henry.
```

```
?- cat(X).  
X = harry ;  
X = marry ;  
X = bill ;  
X = steve.
```

```
?- cat(Y).  
Y = harry ;  
Y = marry ;  
Y = bill ;  
Y = steve.
```

```
?- dog(X), cat(Y).  
X = fido,  
Y = harry ;  
X = fido,  
Y = marry .
```

```
?- dog(X), cat(X).  
false.
```

```
?- halt.
```

the data objects in Prolog are called **terms**

- Numbers
- Atoms
- Variables
- Compound Terms
- Lists
- Other Types of Terms



constants with numeric values

- integers
- numbers with a decimal point

constants that do not have numeric values

- any sequence of one or more letters, numerals and underscores beginning with lower case letter
- any sequence of characters enclosed in single quotes, including spaces and upper case letter
- any sequence of one or more special characters from a list that includes + - \* / > < = & # @

- in a query
  - a name used to stand for a term that is to be determined in query
- in a rule or fact
- any sequence of one or more letters, numerals and underscores
- beginning with uppercase letters or underscore
- anonymous variable : single underscore

# Compound Terms

- a structured data type that begins with an atom
- an atom is known as a functor
- this functor is followed by one or more arguments
- arguments in parenthesis and separated by commas
- $\text{functor}(t_1, t_2, \dots, t_n) \quad n \geq 1$