### Equality Predicates

|  |
| --- |
| **Unification**: (=)  Succeeds if it can **unify** its arguments.  **Negation of Unification**: (\=)  Succeeds if = fails. |
| **Identity**: (==)  Succeeds if its arguments are **identical**.  **Negation of Identity**: (\==)  Succeeds if == fails. |
| **Arithmetic Equality**: (=:=)  Succeeds if its arguments **evaluate to the same integer**.  **Arithmetic Inequality**: (=\=)  Succeeds if =:= fails. |

### Definite Clause Grammars

|  |
| --- |
| **DCG Notation**:  s --> a, s, b.  a --> [a].  b --> [b]. |
| **Difference-Lists Notation**:  s(X, Y) :- a(X, A), s(A, B), b(B, Y).  a([a|W], W).  b([b|W], W). |

|  |
| --- |
| **DCG with Arguments**:  s --> as(Count), bs(Count), cs(Count).  as(0) --> [].  as(Count) --> [a], as(OldCount), {Count is OldCount+1}.  bs(0) --> [].  bs(Count) --> [b], as(OldCount), {Count is OldCount+1}.  cs(0) --> [].  cs(Count) --> [c], as(OldCount), {Count is OldCount+1}. |

### Lists

|  |
| --- |
| '.'(1, [2,3], [1,2,3]). |

### Facts & Rules

Facts:

|  |
| --- |
| happy(yolanda).  party. |

A **predicate** is a fact that is always true.

A **term** is a constant, atom, variable, or compound term.

A **compound term** consists of a function with 1 or more arguments.

|  |
| --- |
| a. % A term with functor 'a' and 0 arguments  a(b,c). % A term with functor 'a' and 2 arguments, b and c |

The **arity** of a complex term is its number of arguments.

The empty list [] is a **term**, and more-specifically, an **atom**.

A list [H|T] is represented as '.'(H,T), and is therefore a **compound term**.

A **predicate clause** is a rule.

|  |
| --- |
| f(X) :- a(X). % f(X) is the head of the predicate clause. |

|  |  |
| --- | --- |
| atom/1  integer/1  float/1  number/1  atomic/1  var/1  nonvar/1  is\_list/1 | Is an atom?  Is an integer?  Is a floating-point number?  Is an integer or floating-point number?  Is a constant?  Is an uninstantiated variable?  Is an uninstantiated variable, or another term that is not an uninstantiated variable?  Is a list? |

### Operators

:- Implication

, Conjunction

; Disjunction

### Cuts

**Green cuts** do not change the meaning of a predicate. They can be removed, and the predicate will function as normal, however it may become less efficient.

**Red cuts** affect the meaning of predicates, and should be avoided if possible.

### Arithmetic

In order to evaluate arithmetic expressions, is must be used.

|  |
| --- |
| X is 2+3.  9-4 =:= 2+3. |

Prolog uses the following **comparison operators**:

|  |
| --- |
| X < Y  X =< Y (Because screw standards?)  X =:= Y  X =\= Y  X >= Y  X > Y |