**Prolog Vocabulary**

**Arity** – The number of arguments in a predicate. human(socrates) has an arity of one.

**Arguments** – The terms in parentheses after a functor. In human(socrates) the argument is Socrates.

**Assert** – State that something is true. One can assert that all humans are mortal. mortal(X) :- human(X), or that Socrates was a human. human(socrates). A Prolog program is made up of a collection of logical assertions. Those assertions are called clauses in a Prolog program.

**Backtracking** – The search algorithm that Prolog uses to prove or disprove goals. It tries clauses in each appropriate predicate in the order found and if failing to find a proof trying some, will backtrack to try the next ones until a solution is either found or all possible combinations of clauses have been used.

**Bind (bound)** – The process of giving a value to a variable through unification. If the goal is mortal(X), then unification will bind X to socrates.

**Clause** – One assertion of a predicate. A predicate can have multiple clauses. Clauses can be dynamic, which means they can be changed during execution, or static which means they were compiled into a faster, unchangeable, internal form.

**Compiler** – A tool for creating byte codes that run on a WAM, thus providing compiled code performance.

**Consult** – The process of reading a source file and asserting each of the clauses into the dynamic database, where they can be used by the theorem proving engine.

**Debugger** – A tool for tracing the execution of a Prolog proof.

**Dynamic Database** – the collection of clauses in memory as a Prolog program runs. These clauses are typically consulted from the source code.

**Functor** – The main portion of a structure and head of a clause. In mortal(X), mortal is the functor.

**Goal** – A pattern that can be unified with the heads of clauses to find clauses that can be used to prove or disprove the goal, unifying any variables that need unification.

**Listener** – The tool that lets one interact with the dynamic database, providing goals as queries and seeing the results. It’s where one could ask if Socrates was mortal.

**Predicate** – A collection of similar clauses identified by a functor and arity. For example the human/1 predicate might have multiple clauses of the form human(socrates).

**Recursion** – Programming technique where a function can call itself in a process of reducing a problem, step by step. In Prolog it refers to a predicate that refers to itself. For example, the ancestor(X, Z) :- parent(X, Y), ancestor(Y, Z).

**Structure** – A complex term with a functor and arguments.

**Term** – A fundamental building block of a Prolog program that might be a number, or an atom or a more complex structure.

**Theorem Prover** – A software engine that, given a collection of logical assertions, can prove or disprove various goals, maybe unifying variables that make the assertion true. For example, given the example in the definition of assert, the question mortal(X) would be true if X = socrates.

**Unification** – The algorithm used to determine the least restrictive variable bindings between two terms so that the terms are equal.

**Variable** – A logical variable, which, unlike variables in conventional languages, does not get values assigned, but rather gets values as the result of the unification algorithm.

**WAM** – A Warren Abstract Machine. The theorem proving engine design used by most Prolog implementations for compiled code, developed by David Warren.