Expert systems are one area of artificial intelligence. Expert systems are computer software that have been designed in order to replicate human experts in both acts and activities, by the system’s capability to find new facts from available facts and give advice, to demonstrate and carry out intelligent tasks. In this paper, we are going to talk about the different types of expert system development tools and the differences between them. An expert system tool, is a software development environment containing the basic components of expert systems. Associated with a shell is a prescribed method for building applications by configuring and instantiating these components. The core components of expert systems are the knowledge base and the reasoning engine.

Various choices are provided when designing an expert system, the methodology, the mode of knowledge representation within the model (production rules, frames, logic programming, or some combination of these methods), the software development package, the hardware upon which the expert system is to be implemented.

There are different types of expert system tools namely:

1. Expert system shells - consists mainly of an inference engine and an editor to assist developers in building their knowledge base.
2. High-level programming languages - an expert system can be implemented using general purpose programming language. However, the programming languages such as LISP and PROLOG are the most commonly used in expert system implementation due to their high capabilities in handling symbolic data efficiently.
3. Multiple-paradigm programming environments - these systems expand the capabilities of shells in various directions. They run on engineering workstations, minicomputers, or mainframes; offer tight integration with large databases; and support the building of large expert systems.

The first tool to be discussed is JESS which is a form of Java Expert System Shell. Jess can be used in two ways namely as a rule engine or as a general-purpose programming language.

A rule engine is special kind of program that applies rules to data in a very effective way. A rule-based program can have hundreds to thousands of rules, and Jess will constantly apply them to data in the form of a knowledge base. In some domain, the rules are represented as the heuristic knowledge of a human expert, and an evolving situation will be represented by the knowledge base. In this scenario, they are said to form an expert system*.* Jess is also a general-purpose programing language, and moreover, it can directly access all Java classes and libraries. This is why, Jess is also often used as a rapid application or dynamic scripting development environment. Usually, before running a code in Java, it must first be compiled, but in Jess a line of code is immediately being executed as it is typed. This allows one to interactively analyze and experiment with Java APIs, and build up large programs step by step. Extending the Jess language is relatively easy and can be done by writing new commands in Java or Jess itself and thus, the Jess language can be customized for particular applications.

Jess being driven by lisp-style scripting language built in Java itself has some advantages as well as some disadvantages. When programming in Java, some external mechanism generates and controls the rules and in order to use Jess, the data must first be converted into text that can be handled by the interpreter.

Advantages:

* It is easier to work with the code builder using an independent scripting language.
* Jess releases a burden from the programmer, he/she no longer have to declare each rule as a set of nested class instances.
* Programming efforts in building user interface and inference engine is greatly reduce.
* Projects can be completed faster, cheaper and in a more efficient way.

Disadvantages:

* Using an independent scripting language means Java being disconnected from rule engine. Normal Java syntax cannot be used to debug syntax after external files and strings have been used to specify the rules.

Performance:

Jess’s rule engine uses an improved form of a well-known algorithm called Rete (latin for "net") to match rules against the knowledge base. Jess is actually faster than some popular expert system shells written in C, especially on large problems, where performance is dominated by algorithm quality. Note that Rete is an algorithm that explicitly trades space for speed, so Jess’ memory usage is not inconsiderable. Jess does contain some commands which will allow you to sacrifice some performance to decrease memory usage. Nevertheless, Jess’ memory usage is not ridiculous, and moderate-sized programs will fit easily into Java’s default 16M heap.

<https://www.cs.unm.edu/~luger/ai-final2/JAVA/CH%2026_Case%20Studies%20-%20JESS%20and%20other%20Expert%20System%20Shells%20in%20Java.pdf>

<http://www.jessrules.com/doc/61/intro.html>

Prolog on the other hand is considered as a high-level programming language. This program is made up of a set of clauses. A clause is either a fact or a rule which is usually used to indicate a relationship between elements. The orders of database entries are extremely important because the efficiency of the program will mainly depend on the arrangement as it can influence the number of search required to satisfy the goal. There might be situations where Prolog may not find a solution even if a solution might be easily be found from the given information because of the exact nature of the search process. The process of deriving a reasoning for Prolog programming is as follows:

* A goal is given, Prolog searches into the database form the top to the bottom for a fact that matches the goal.
* A pointer is left where the match is found and prolog instantiates the suitable variables.
* When a goal matches the head of a rule rather than a fact, the atoms within the body of the rule are treated as sub goals that must all be satisfied to prove that the head is satisfied.

Advantages:

* Prolog is a powerful language which deduces the desired supplementary facts with built in powers of deduction from statements and principles that has been set to the system which one wishes to reason.
* The programmer only has to define what is required rather than indicating how it should be computed and thus, removing the burden of worrying about implementation details from the programmers.
* Prolog encourages incremental system development and makes program tracing and debugging easier.
* With Prolog, it is easy to maintain program.
* Modular programming and testing is possible with Prolog.
* Tracing and debugging programs in Prolog is much easier.
* Enables implementation of complex data structure.

Disadvantages:

* It can be very difficult to design a database that accurately represents relationships.
* Prolog is not best suited to solving complex arithmetical computations.

<https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm>

Prolog vs Jess:

1. Prolog and Jess which is a Rete-based system are very different. Prolog is backward chaining meaning that it starts with the desired conclusion(s), work backwards to find supporting facts and is goal-directed. If one forgets the result and ask for it again, Prolog has to compute it all over again. Jess on the other hand includes both a kind of backwards chaining and a construct called defquery which lets one make direct queries of the knowledge base.
2. Prolog is meant to be used from the console whereas in Jess, the command-line is not intended for end-users.
3. Prolog focuses more on answering queries whereas Jess focusses more on acting in response to input.

One more difference is that Prolog is really meant to be used from the console; i.e., you’re actually supposed to sit down and type mortal(Socrates). In Jess, only developers do this; the command line is not intended for end-users. Prolog is really about answering queries, while Jess is about acting in response to inputs. Jess is different than some Rete-based systems in that it includes both a kind of backwards chaining and a construct called defquery which lets you make direct queries of the knowledge base. Both of these help Jess a better fit for some Prolog applications, but they don’t make Jess into a Prolog-like system. Prolog is optimized, in a sense, for space, at the cost of speed. Jess (and its Rete algorithm) is optimized for speed at the cost of space. The Rete algorithm is all about computing things -once- so they never need to be recomputed, and then reusing them. Prolog’s approach is targeted at exploring large numbers of possibilities once, while Rete is aimed at exploring medium-sized numbers of possibilities repeatedly. <http://www.iau.dtu.dk/teaching/31380/Jess/manual.pdf>