# A flexible Prolog interpreter in Python

Carl Friedrich Bolz

April 15, 2007

Efficient virtual machines for the Prolog programming language (and also for other dynamic high-level languages) are usually large and intricate pieces of software implemented in a low-level language such as C that are extensively optimized over many years and are therefore difficult to change in many respects. Due to their inherent complexity it is hard to develop them further, to maintain them or to use them to experiment with language extensions or new built-ins.

Early implementation decisions, such as choice of garbage collector, are extremely hard to change afterwards. In addition, it is also much harder to perform advanced optimizations or transformations (such as partial evaluation) on the virtual machine itself due to the low level of the implementation language. Furthermore it is (by definition) impossible to port them to a non-C platform such as the Java Virtual Machine or the .NET framework Therefore the numerous implementations of a Prolog virtual machine integrated into one of these environments are all new implementations which have to be maintained themselves and therefore all have different levels of maturity and slightly different sets of built-ins available.

Using a higher-level language than C addresses many of these problems. It allows the language implementor to work at a higher level of abstraction, which eases implementation, maintenance and extensions of the virtual machine and reduces the size of the code base. Furthermore, advanced optimizations and transformations can be performed. To regain usable performance the implementation needs then to be translated to a low-level language again.

The goal of the PyPy project[1] is to support the implementation of interpreters in the high-level language Python. Python is an object oriented dynamic language which is becoming more and more popular for scripting use as well as for "real" applications. PyPy provides a translation toolchain that helps to translate interpreters written in Python to a low-level language such as C and the .NET Intermediate Language. To make this translation work, these interpreters have to be written in RPython, which is a proper subset of Python chosen in such a way that it is amenable to analysis.

In this talk we give an quick overview of the PyPy project and present a Prolog interpreter written in RPython, exploring the possibilities of implementing a well-known declarative language in a high-level language (as opposed to C). One of the main goals of the implementation was to keep it as simple and as extensible as possible (even at the cost of performance). The PyPy tool suite is used to reach a reasonable level of performance and to ensure portability to various platforms.

[1] http://codespeak.net/pypy