Chapter 1

What is tuProlog

tuProlog is a light-weight Prolog framework for distributed applications and infrastructures. tuProlog is developed and maintained by the aliCE research group[1] at the Alma Mater Studiorum—Università di Bologna. It is built as an Open Source software, released under the LGPL license – thus allowing also for commercial derivative work –, and made available through the pages of the APICe web portal[2].

tuProlog is designed to be *minimal*, dynamically *configurable*, *interoperable*, straightforwardly *integrated* with Java and .NET, and easily *deployable*.

First of all, tuProlog is designed with *minimality* in mind. Accordingly, tuProlog core is a tiny Java object that contains only the essential properties of a Prolog engine. Only the required Prolog features – like, say, ISO compliance, I/O predicates, DCG operators – are then to be added to or removed from a tuProlog engine according to the contingent application needs.

The obvious counterpart of minimality is tuProlog *configurability*. In fact, a simple yet powerful mechanism based on the notion of tuProlog *library* is provided tha allows required predicates, functors and operators to be loaded and unloaded in a tuProlog engine, both statically and dynamically. Libraries can be either included in the standard tuProlog distribution, or defined *ad hoc* by the tuProlog user / developer.

A tuProlog library can be built in different ways. First of all, a tuProlog library could be straightforwardly written in Prolog. On the other hand, a tuProlog library could also be implemented using either Java or any language of .NET framework—depending on the chosen tuProlog implementation. Finally, a tuProlog library could be built by combining Prolog and Java /

[1] `http://www.alice.unibo.it`
[2] `http://tuprolog.apice.unibo.it`

.NET languages, thus paving the way for multi-language / multi-paradigm integration. Whatever the language(s) used, a tuProlog library can be either used to configure a tuProlog engine when this is started up, or loaded – and then unloaded – dynamically at any time during the engine execution.

tuProlog was first implemented upon Java, then ported upon .NET, and is now available on both platforms. *Deployability* of tuProlog owes a lot to Java and .NET. On the Java side, the requirements for tuProlog installation simply amount to the presence of a standard Java VM, and a Java invocation upon a single JAR file is everything needed to start a tuProlog activity. On the .NET side, ... ENRICO ENRICO ENRICO ENRICO

tuProlog *integration* with other languages and paradigms is kept as clean as possible, so that the components of a tuProlog application can be developed by choosing at any step the most suitable paradigm—either declarative/logic or imperative/object-oriented. On the Prolog side, thanks to the `JavaLibrary` library, any Java entity (object, class, package) can be represented as a Prolog term, and exploited from Prolog. So, for instance, Java packages like Swing and JDBC can be directly used from within Prolog, straightforwardly enhancing tuProlog with graphics and database access capabilities. In the same way, `DotNetLibrary` ... o come cavolo si chiama ENRICO ENRICO ENRICO ENRICO On the Java side, a tuProlog engine can be invoked and used as a simple Java object, possibly embedded in beans, or exploited in a multi-threaded context, according to the application needs. Also, a multiplicity of different tuProlog engines can be used from a Java program at the same time, each one configured with its own libraries and knowledge base. In the same way, sbrodolata .NET di ENRICO ENRICO ENRICO ENRICO

Interoperability misteriosa: cosa diciamo??? Ancora quanto segue???

Finally, *interoperability* is developed along two main lines: Internet standard patterns, and coordination models. So, tuProlog supports interaction via TCP/IP and RMI, and can be also provided as a CORBA service. In addition, tuProlog supports tuple-based coordination under many forms. First, components of a tuProlog application can be organised around Java-based tuple spaces, logic tuple spaces, and ReSpecT tuple centres [**?**]. Then, tuProlog applications can exploit Internet infrastructures providing tuple-based coordination services, like LuCe [**?**] and TuCSoN [**?**].