

# tuProlog 2.0: One Step Beyond

Giulio Piancastelli    Andrea Omicini  
ALMA MATER STUDIORUM—Università di Bologna, Cesena, Italy  
giulio.piancastelli@unibo.it    andrea.omicini@unibo.it

<http://tuprolog.alice.unibo.it>

## What is tuProlog

tuProlog (also known as 2P) is an Open Source Prolog engine built on top of the Java Virtual Machine, and designed to provide logic-based technology as a core ingredient for Internet application components and infrastructures. In this context, logic programming languages have already proved to be effective both as communication and as coordination tools [5], as well as in facing specific issues like security in a declarative way. To also play a key role in the development and deployment of Internet applications and infrastructures, logic-based components should be implemented accordingly to a number of engineering requirements: for this purpose, tuProlog has been designed with scalability and interoperability in mind, and to be easily deployable, light-weight, statically and dynamically configurable.

The key properties that tuProlog has been created to feature are:

**Minimality** tuProlog means to be as thin and light-weight as possible: to this end, its pure inferential core is available as a Java object containing only the most essential characteristics of a Prolog engine. Then, only the required Prolog features (e.g. I/O predicates, DCG operators) are to be added to or removed from a tuProlog engine, according to the application's contingent needs. This property is particularly relevant for use in small devices such as PDAs or mobile phones.

**Configurability** tuProlog's choice of minimality calls for a high degree of configurability as its necessary counterpart. In particular, configurability should be dynamic, so as to face the openness of most application environments such as the Internet, and enable both static and dynamic configuration of components in a uniform way. The notion of tuProlog library provides a simple yet powerful mechanism to load and unload useful predicates, functors and operators in a tuProlog engine, both statically and dynamically. Libraries can be built using either Prolog, or Java, or both languages, and can be either employed to configure a tuProlog engine when it is started up, or loaded (and then unloaded) dynamically at

any time during the engine execution. Five libraries are included in the standard `tuProlog` distribution, to provide functionalities matching the built-in predicates described in the ISO Prolog Standard; new libraries can be defined by the `tuProlog` user or developer as well.

**Deployability** Requirements for `tuProlog` installation simply amount to the presence of a standard Java Virtual Machine, and a Java invocation upon a single JAR file is everything needed to start a `tuProlog` activity.

**Interoperability** Internet standard patterns and coordination models are the two main lines along which `tuProlog`'s interoperability is developed. On the one hand, interaction is supported via TCP/IP and RMI; on the other hand, a `tuProlog` engine can also be provided as a CORBA service. However, since these approaches do not completely solve the problems of interaction coupling, and in some sense prevent more complex form of coordination other than peer-to-peer models to be enacted [3], `tuProlog` makes it possible to adopt a logic-based abstraction as a unifying interaction metaphor: components of a `tuProlog` application can be organized around Java-based tuple spaces, logic tuple spaces, and `ReSpecT` [4] tuple centres; then, `tuProlog` applications can exploit Internet infrastructures providing tuple-based coordination services, such as `TuCSon` [10] or `LuCe` [7].

Probably, the `tuProlog`'s feature that is most appealing to applications and systems developers is its full, bidirectional, easy-to-use integration scheme between the declarative/logic and the imperative/object-oriented programming paradigms to be found in the Prolog and Java worlds, respectively [6]. From the Prolog side, thanks to the `JavaLibrary` library, any Java entity (object, class, package) can be represented as a term and directly exploited: for instances, Java extensions like JDBC and Swing can be straightforwardly used from within Prolog, thus enhancing `tuProlog` with graphics and database access capabilities. From the Java side, a `tuProlog` engine can be invoked and used as a simple Java object, possibly embedded in beans or employed in a multi-threading 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.

A key requirement of the integration between Java and Prolog has been to preserve paradigms orthogonality: correspondingly, such integration has been designed so as not to mix the logic and object-oriented paradigms, thus avoiding to alter in any way the very nature of either language. This constraint has been introduced not only for conceptual cleanness, but also because only a simple, non-intrusive integration scheme could actually make `tuProlog` a practical programming framework to be used in an effective way, preserving and even promoting the power of both paradigms and technologies.

## What's new in tuProlog 2.0

With the recent 2.0 release, tuProlog's internal architecture has been restructured, to allow for more ease of both extension and modification. Following sound engineering principles, such as object-oriented code structuring, reuse of established community knowledge under the form of patterns, loose coupling of composing elements, modularity, and a clear and clean separation of concerns, tuProlog's architecture has been based upon a set of managers, operating around a minimal core shaped as a Finite State Machine, and handling control of sensible parts of the engine, such as built-in primitives, predicate libraries, and logic theories. The result is regarded as a *malleable architecture*, giving rise to deeper flexibility and modifiability—two properties that, to a certain extent, have always represented a strong asset on tuProlog's appealing side.

In fact, especially during the latest years, tuProlog has been used as a basic component in a number of research projects who did benefit from its pliable nature. For instance, tuProlog has been integrated into the DCASELP environment [8] for building heterogeneous multi-agent systems, thanks to the core extendibility provided by dynamically loadable predicate libraries; the engine's unification algorithm, distributed across the classes representing the Prolog terms hierarchy, in true object-oriented fashion, has been modified to support the PRACTIONIST framework [9] for developing agents according to the Belief-Desire-Intention (BDI) model; the whole tuProlog has been tweaked to implement the AtuP argumentation engine [2] as a non-monotonic reasoning component in Internet or agent-based applications.

tuProlog is developed and maintained by the aliCE [1] research group at the ALMA MATER STUDIORUM—Università di Bologna, site of Cesena: it is built as Open Source software, and released under the LGPL license, thus allowing also for commercial derivative work. Its user community and third party contributors gather around the `tuprolog-users` mailing list and the project website hosted on SourceForge. The product website [11] also aims at collecting and delivering supplemental tuProlog-related software, such as experimental development branches, conversions on platforms other than J2SE, additional libraries and tools of interest. Currently, the site hosts not only the main tuProlog distribution, but also conversions on the J2ME and the Microsoft .NET platforms, alongside an Eclipse plug-in providing a development environment for tuProlog programmers.

## References

- [1] aliCE home page. <http://www.alice.unibo.it/>.
- [2] Daniel Bryant, Paul Krause, and Gerard Vreeswijk. Argue tuProlog: A lightweight argumentation engine for agent applications. In Paul Dunne and Trevor Bench-Capon, editors, *1st International Conference on Computational Models of Argument (COMMA06)*, pages 27–32, 2006.

- [3] Paolo Ciancarini, Andrea Omicini, and Franco Zambonelli. Coordination technologies for Internet agents. *Nordic Journal of Computing*, 6(3):215–240, Fall 1999.
- [4] Enrico Denti, Antonio Natali, and Andrea Omicini. On the expressive power of a language for programming coordination media. In *1998 ACM Symposium on Applied Computing (SAC'98)*, pages 169–177, Atlanta, GA, USA, 27 February–1 March 1998. ACM. Special Track on Coordination Models, Languages and Applications.
- [5] Enrico Denti and Andrea Omicini. Engineering multi-agent systems in LuCe. In Stephen Rochefort, Fariba Sadri, and Francesca Toni, editors, *ICLP'99 International Workshop on Multi-Agent Systems in Logic Programming (MAS'99)*, Las Cruces, NM, USA, 30 November 1999.
- [6] Enrico Denti, Andrea Omicini, and Alessandro Ricci. Multi-paradigm Java-Prolog integration in tuProlog. *Science of Computer Programming*, 57(2):217–250, August 2005.
- [7] Enrico Denti, Andrea Omicini, and Vladimiro Toschi. The LuCe coordination technology for MAS design and development on the Internet. In António Porto and Gruia-Catalin Roman, editors, *Coordination Languages and Models*, volume 1906 of *LNCS*, pages 305–310. Springer-Verlag, 2000. 4th International Conference (COORDINATION 2000), Limassol, Cyprus, 11–13 September 2000. Proceedings.
- [8] Ivana Gungui and Viviana Mascardi. Integrating tuProlog into DCaseLP to engineer heterogeneous agent systems. In Elio Panegai and Gianfranco Rossi, editors, *Italian Conference on Computational Logic (CILC-2004)*, volume 390 of *Quaderno del Dipartimento di Matematica*. Università di Parma, June 2004.
- [9] Vito Morreale, Susanna Bonura, Giuseppe Francaviglia, Fabio Centineo, Massimo Cossentino, and Salvatore Gaglio. Goal-oriented development of BDI agents: The PRACTIONIST approach. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'06)*, pages 66–72, Los Alamitos, CA, USA, 2006. IEEE Computer Society.
- [10] Andrea Omicini and Franco Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, September 1999. Special Issue: Coordination Mechanisms for Web Agents.
- [11] tuProlog home page. <http://tuprolog.alice.unibo.it/>.