WIKIPEDIA

Logtalk

Logtalk is an object-oriented logic programming language that extends and leverages the Prolog language with a feature set suitable for programming in the large.[1] It provides support for encapsulation and data hiding, separation of concerns and enhanced code reuse.[1] Logtalk uses standard Prolog syntax with the addition of a few operators and directives.

The Logtalk language implementation is distributed under an open source license and can run using a Prolog implementation (compliant with official and de facto standards)[1] as the back-end compiler.

Logtalk

Paradigm	Logic programming, object-oriented programming, prototype-based programming
Designed by	Paulo Moura
First appeared	1998
Stable release	3.48.0 / 6 July 2021
OS	Cross-platform
License	Artistic License 2.0 (2.x) / Apache License 2.0 (3.01.x)
Website	logtalk.org (https://logtalk.org)
Influenced by	
Prolog, Smalltalk, Objective-C	

Contents

Features

Logtalk aims to bring together the advantages of object-oriented programming and logic programming.[1] Object-orientation emphasizes developing discrete, reusable units of software, while logic programming emphasizes representing the knowledge of each object in a declarative way.

As an object-oriented programming language, Logtalk's major features include support for both classes (with optional metaclasses) and prototypes, parametric objects,[2] protocols (interfaces), categories (components, aspects, hot patching), multiple inheritance, public/protected/private inheritance, event-driven programming, high-level multi-threading programming,[3] reflection, and automatic generation of documentation.

For Prolog programmers, Logtalk provides wide portability, featuring predicate namespaces (supporting both static and dynamic objects), public/protected/private object predicates, coinductive predicates, separation between interface and implementation, simple and intuitive meta-predicate semantics, lambda expressions, definite clause grammars, term-expansion mechanism, and conditional compilation. It also provides a module system based on de facto standard core module functionality (internally, modules are compiled as prototypes).

Examples

Logtalk's syntax is based on Prolog:

```
?- write('Hello world'), nl.
Hello world
true.
```

Defining an object:

```
:- object(my_first_object).

    :- initialization((write('Hello world'), nl)).

    :- public(p1/0).
    p1 :- write('This is a public predicate'), nl.

    :- private(p2/0).
    p2 :- write('This is a private predicate'), nl.

:- end_object.
```

Using the object, assuming is saved in a my_first_object.lgt file:

```
?- logtalk_load(my_first_object).
Hello world
true.

?- my_first_object::p1.
This is a public predicate
true.
```

Trying to access the private predicate gives an error:

```
?- my_first_object::p2.
ERROR: error(permission_error(access, private_predicate, p2), my_first_object::p2, user)
```

Anonymous functions

Logtalk uses the following syntax for anonymous predicates (lambda expressions):

```
{FreeVar1, FreeVar2, ...}/[LambdaParameter1, LambdaParameter2, ...]>>Goal
```

A simple example with no free variables and using a list mapping predicate is:

```
| ?- meta::map([X,Y]>>(Y is 2*X), [1,2,3], Ys).
Ys = [2,4,6]
yes
```

Currying is also supported. The above example can be written as:

```
| ?- meta::map([X]>>([Y]>>(Y is 2*X)), [1,2,3], Ys).
Ys = [2,4,6]
yes
```

Prolog back-end compatibility

Supported back-end Prolog compilers include B-Prolog, Ciao Prolog, CxProlog (http://ctp.di.fct.unl.pt/~amd/cxprolog/), ECLiPSe, GNU Prolog, JIProlog (http://www.jiprolog.com/), Quintus Prolog (https://quintus.sics.se/), Scryer Prolog (https://github.com/mthom/scryer-prolog/), SICStus Prolog, SWI-Prolog, Tau Prolog (http://tau-prolog.org/), Trealla Prolog (https://github.com/infradig/trealla), XSB, and YAP.[4] Logtalk allows use of back-end Prolog compiler libraries from within object and categories.

Developer tools

Logtalk features on-line help, a documenting tool (that can generate PDF and HTML files), an entity diagram generator tool, a built-in debugger (based on an extended version of the traditional Procedure Box model found on most Prolog compilers), a unit test framework with code coverage analysis, and is also compatible with selected back-end Prolog profilers and graphical tracers.[5]

Applications

Logtalk has been used to process STEP data models used to exchange product manufacturing information.[6] It has also been used to implement a reasoning system that allows preference reasoning and constraint solving.[7]

See also

- Mercury (programming language)
- Oz (programming language)
- Prolog++
- Visual Prolog

References

1. Paulo Moura (2003). Logtalk: Design of an Object-Oriented Logic Programming Language. PhD thesis. Universidade da Beira Interior
2. Moura, Paulo (2011). *Programming Patterns for Logtalk Parametric Objects*. Applications of Declarative Programming and Knowledge Management. Lecture Notes in Computer Science. **6547**. doi:10.1007/978-3-642-20589-7_4 (https://doi.org/10.1007%2F978-3-642-20589-7_4). ISBN 978-3-642-20588-0.
3. "Practical Aspects of Declarative Languages". Lecture Notes in Computer Science. **4902**. 2008. doi:10.1007/978-3-540-77442-6 (https://doi.org/10.1007%2F978-3-540-77442-6). ISBN 978-3-540-77441-9.
4. "Logtalk compatibility" (https://logtalk.org/compatibility.html). Logtalk.org. 2016-10-10. Retrieved 2021-07-06.

5. / (2013-02-12). "Developer Tools – LogtalkDotOrg/logtalk3 Wiki – GitHub" (https://github.com/LogtalkDotOrg/logtalk3/wiki/Developer-Tools). Github.com. Retrieved 2013-08-19.
6. "Logic Programming". Lecture Notes in Computer Science. **4079**. 2006. doi:10.1007/11799573 (https://doi.org/10.1007%2F11799573). ISBN 978-3-540-36635-5.
7. Victor Noël; Antonis Kakas (2009). *Gorgias-C: Extending Argumentation with Constraint Solving* (ftp://ftp.irit.fr/IRIT/SMAC/DOCUMENTS/PUBLIS/lpnmr-09_noel-kakas.pdf) (PDF). Logic Programming and Nonmonotonic Reasoning. Lecture Notes in Computer Science. **5753**. pp. 535–541.

External links

- Official website (https://logtalk.org)
- Logtalking blog (https://logtalk.org/blog.html)
- From Plain Prolog to Logtalk Objects: Effective Code Encapsulation and Reuse (Invited Talk). Paulo Moura. Proceedings of the 25th International Conference on Logic Programming (ICLP), July 2009. LNCS 5649. Springer-Verlag Berlin Heidelberg". (Slides (https://logtalk.org/papers/iclp2009/logtalk_iclp2009.pdf))