Go! (programming language)

Go! is an agent-based programming language in the tradition of logic-based programming languages like Prolog.[1] It was introduced in a 2003 paper by Francis McCabe and Keith Clark.[2]

Go!	
Paradigm	Multi-paradigm: concurrent, logic, functional, imperative (object-based)
Designed by	Francis McCabe, Keith Clark
First appeared	2003
Preview release	9-30-07 / September 30, 2007
Typing discipline	strong
OS	Unix-like
License	GPLv2
Influenced by	
Prolog[1]	

Contents

Design

The authors of Go! describe it as "a multi-paradigm programming language that is oriented to the needs of programming secure, production quality and agent-based applications. It is multi-threaded, strongly typed and higher order (in the functional programming sense). It has relation, function and action procedure definitions. Threads execute action procedures, calling functions and querying relations as needed. Threads in different agents communicate and coordinate using asynchronous messages. Threads within the same agent can also use shared dynamic relations acting as Linda-style tuple stores."[2]

The authors also propose that the language is suitable for representing ontologies due to its integration of logic, functional and imperative styles of programming.[3]

Example

The following example illustrates the "ontology-oriented" type and declarations style of Go!:[3]

```
Sex ::= male | female.

person <~ {dayOfBirth:[] => day.
           age:[] => integer.
           sex:[] => Sex.
           name:[] => string.
           home:[] => string.
           lives:[string]{}}.

person:[string, day, Sex, string] $= person.

person(Nm, Born, Sx, Hm)..{
  dayOfBirth() => Born.
  age() => yearsBetween(now(), Born).
  sex() => Sx.
  name() => Nm.
  home() => Hm.
```

```
   lives(Pl) :- Pl = home().
   yearsBetween:[integer, day] => integer.
   yearsBetween(...) => ..
}.

newPerson:[string, day, Sex, string] => person.

newPerson(Nm, Born, Sx, Hm) => $person(Nm, Born, Sx, Hm).
```

The `::=` rule defines a new algebraic data type, a data type with only data constructors.

The `<~` rule defines an interface type - it indicates what properties are characteristic of a `person` and also gives type constraints on these properties. It documents that `age` is a functional property with an integer value, that `lives` is a unary relation over strings, and that `dayOfBirth` is a functional property with a value that is an object of type `day`.

The `$=` type rule indicates that there is also a theory label, with the functor `person`, for a theory that defines the characteristic properties of the `person` type - implements the `person` interface - in terms of four given parameters of types `string`, `day` , `Sex`, and `string`.

Conflict with Google

In November 2009, Google released a similarly named Go programming language (with no exclamation point). McCabe asked Google to change the name of their language as he was concerned they were "steam-rolling over us".[1][4] The issue received attention among technology news websites, with some of them characterizing Go! as "obscure".[5] The issue thread opened on the subject was closed by a Google developer on 12 October 2010 with the custom status "Unfortunate" and with the following comment: "there are many computing products and services named Go. In the 11 months since our release, there has been minimal confusion of the two languages."[6]

References

1. Claburn, Thomas (2009-11-11). "Google 'Go' Name Brings Accusations Of 'Evil' " (http://www.informationweek.com/news/software/web_services/showArticle.jhtml?articleID=221601351). *InformationWeek*. Retrieved 2009-11-14.
2. Clark, K.L.; McCabe, F.G. (2003). *Go! for multi-threaded deliberative agents*. *International Conference on Autonomous Agents (AAMAS'03)*. pp. 964–965. CiteSeerX 10.1.1.117.184 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.117.184). doi:10.1145/860575.860747 (https://doi.org/10.1145%2F860575.860747). ISBN 978-1581136838.
3. Clark, K.L.; McCabe, F.G. (2006). "Ontology oriented programming in go!". *Applied Intelligence*. **24** (3): 189–204. CiteSeerX 10.1.1.117.184 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.117.184). doi:10.1145/860575.860747 (https://doi.org/10.1145%2F860575.860747).
4. "Issue 9 - go - I have already used the name for *MY* programming language" (https://code.google.com/p/go/issues/detail?id=9). 2009-11-10. Retrieved 2009-11-14.
5. Brownlee, John (2009-11-13). "Google didn't google "Go" before naming their programming language" (http://www.geek.com/articles/news/google-didnt-google-go-before-naming-their-programming-language-20091113/). Geek.com. Retrieved 2010-01-18.
6. "I have already used the name for *MY* programming language · Issue #9 · golang/go" (https://github.com/golang/go/issues/9#issuecomment-66047478). *GitHub*. Retrieved 2019-07-04.

Further reading

- Clark, K. L.; McCabe, F. G. (2003). "Ontology Oriented Programming in Go!" (http://www.doc.ic.ac.uk/~klc/DistKR.pdf) (PDF).
- Clark, K. L.; McCabe, F. G. (2004). "Go!—A Multi-Paradigm Programming Language for Implementing Multi-Threaded Agents" (http://portal.acm.org/citation.cfm?id=998367). *Annals of Mathematics and Artificial Intelligence*. **41** (2–4): 171–206. CiteSeerX 10.1.1.133.1069 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.1069). doi:10.1023/B:AMAI.0000031195.87297.d9 (https://doi.org/10.1023%2FB%3AAMAI.0000031195.87297.d9).
- R. Bordini; et al. (2006). "A Survey of Programming Languages and Platforms for Multi-Agent Systems" (https://web.archive.org/web/20091115151833/http://www.informatica.si/vol30.htm). *Informatica*. **30**: 33–44. Archived from the original (http://www.informatica.si/vol30.htm) on 2009-11-15.
- M. Fisher; et al. (2007). "Computational Logics and Agents - A Roadmap of Current Technologies and Future Trends" (https://archive.today/20130106033209/http://www3.interscience.wiley.com/journal/118495224/issue). *Computational Intelligence*. **23** (1): 61–91. CiteSeerX 10.1.1.114.6149 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.114.6149). doi:10.1111/j.1467-8640.2007.00295.x (https://doi.org/10.1111%2Fj.1467-8640.2007.00295.x). Archived from the original (http://www3.interscience.wiley.com/journal/118495224/issue) on 2013-01-06.
- McCabe, Francis G. (February 7, 2007). *Lets Go!* (https://books.google.com/books?isbn=0975444913). Network Agent Press. ISBN 978-0-9754449-1-7.
- C. Varela; et al. (2004). *On Modelling Agent Systems with Erlang*. ACM SIGPLAN Erlang Workshop '04 (http://ftp.sunet.se/pub/lang/erlang/workshop/2004/).

External links

- Github page (https://github.com/frankmccabe/go)
- Code sample on 99-bottles-of-beer.net (http://99-bottles-of-beer.net/language-go!-289.html)