

Interoperability in Programming Languages

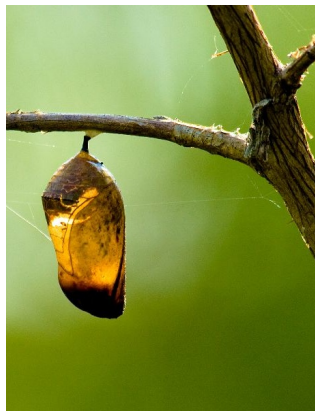
Todd Owen Malone

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

28 April 2014
Senior Seminar

What is Interop?

- Interoperability or Interop
- The ability for a system to use parts from another system
- In programming languages: The interaction of
-



Bluedrakon

<http://tr.im/pWUi>

Why is Interop Important?

Developer time and effort

- Existing and working code is easier to use as-is.
- Third-party systems: source code is unavailable
- Legacy systems: extensive or little-understood code base.

Language Purpose:

- Low-level memory access (C)
- Parallel or distributed systems (Erlang, Clojure)
- Statistics (R)

Outline

- 1 Tools used in achieving Interoperability
- 2 Difficulties in Interop and approaches to dealing with them
- 3 Conclusions

Outline

- 1 Tools used in achieving Interoperability
 - Virtual Machines
 - Markup Languages
- 2 Difficulties in Interop and approaches to dealing with them
- 3 Conclusions

Virtual Machines

- Virtual Machines (VMs) are a runtime environment for a program
- High-level languages compile to an intermediate language
- Intermediate language: Java bytecode or Common Intermediate Language
-

Markup Languages



Outline

1 Tools used in achieving Interoperability

2 Difficulties in Interop and approaches to dealing with them

- Overview
- Metadata
- Standards

3 Conclusions

Some common difficulties in interop

-
-
-
-
-

Metadata and type conversion

Metadata: Data about data

```
(def mylist [1, 2, 3, 4])  
(with-meta mylist {:length 4, :type Integer})
```

In Clojure:

- lists are untyped; can contain entries of different types.
- metadata, added as above, is all user-controlled.

Why Metadata?

- Decontextualized data can carry context with it
- Data transfer between languages with different type strictness.

The importance of Standards



Outline

- 1 Tools used in achieving Interoperability
- 2 Difficulties in Interop and approaches to dealing with them
- 3 Conclusions**

Conclusions

-
-
-
-
-

The End!

Contact:

■ `malone153@morris.umn.edu`

Questions?

References



N. F. McPhee, E. Crane, S. Lahr, and R. Poli.

Developmental Plasticity in Linear Genetic Programming.

In Günther Raidl, *et al*, editors, *GECCO '09*, pages 1019–1026, Montréal, Québec, Canada, 2009.



R. Poli and N. McPhee.

A linear estimation-of-distribution GP system.

In M. O'Neill, *et al*, editors, *EuroGP 2008*, volume 4971 of *LNCS*, pages 206–217, Naples, 26-28 Mar. 2008. Springer.

See the GECCO '09 paper for additional references.