

Interoperability in Programming Languages

Todd 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 ability of a language to call on code from another language



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

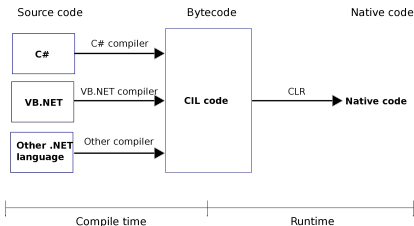
- 1 Tools used in achieving interoperability
- 2 Common difficulties in interop
- 3 Concepts in overcoming difficulties
- 4 Conclusions

Outline

- 1 Tools used in achieving interoperability
 - Virtual Machines
 - Markup Languages
- 2 Common difficulties in interop
- 3 Concepts in overcoming difficulties
- 4 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



Wikipedia

https://en.wikipedia.org/wiki/Common_Intermediate_Language_Runtime

High-level vs Bytecode

```
public class Fib{
    public int fibonacci(int n) {
        if(n == 0){
            return 0;
        }else if(n == 1){
            return 1;
        }else{
            return fibonacci(n - 1) + fibonacci(n - 2);
        }
    }
}

public class Fib {
    public Fib();
    Code:
        0: aload_0
        1: invokespecial #1           // Method java/lang/Object."<init>":()V
        4: return

    public int fibonacci(int);
    Code:
        0: iload_1
        1: ifne          6
        4: iconst_0
        5: ireturn
        6: iload_1
        7: iconst_1
        8: if_icmpne     13
        11: iconst_1
        12: ireturn
        13: aload_0
        14: iload_1
        15: iconst_1
        16: isub
        17: invokevirtual #2           // Method fibonacci:(I)I
        20: aload_0
        21: iload_1
        22: iadd
```

Markup Languages

- Markup languages are a way of modeling data.
- XML and JSON can model data like objects.
- Markup languages are independent of programming languages.

```
<Person>
  <name> Cliff </name>
  <birthdate> 4/16/1978 </birthdate>
  <height> 74 </height>
  <weight> 212 </weight>
</Person>
```

```
{
  "name": "Cliff",
  "birthdate": "4/16/1978",
  "height": "74",
  "weight": "212";
}
```


Outline

1 Tools used in achieving interoperability

2 Common difficulties in interop

- Type systems
- Data structures
- Data processing

3 Concepts in overcoming difficulties

4 Conclusions

Differences in type systems

- Languages represent data in different ways



Mismatched structures

- Untyped lists can contain different types.
- Strongly typed lists can only contain the type given by the list.
- A language might not have an analogous structure
- A data structure in one language

```
[23, v, "hello", True]
```

An untyped list

```
{:name "Cliff", :age 32}
```

A map

Handling data

- Languages act on data in different ways



Outline

1 Tools used in achieving interoperability

2 Common difficulties in interop

3 Concepts in overcoming difficulties

- Metadata
- Standards

4 Conclusions

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

- Standards ensure agreement on what metadata means.



Outline

- 1 Tools used in achieving interoperability
- 2 Common difficulties in interop
- 3 Concepts in overcoming difficulties
- 4 Conclusions**

Conclusions

-
-
-
-
-

Thank you for listening!

Contact: `malone153@morris.umn.edu`

Questions?

References

See the GECCO '09 paper for additional references.