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: The ability for two systems to interact.
- Often shortened to interop.
- In programming languages: The ability of a language to call on code from another language.
Why is interop important?

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

Language Strength:
- Explicit memory access (C)
- Parallel or distributed systems (Clojure, Erlang)
- Statistics (R)
Outline

1. Common difficulties in interop
2. Concepts in interoperability
3. Tools used in achieving interoperability
4. Conclusions
1. Common difficulties in interop
   - Type systems
   - Data structures
   - Data processing

2. Concepts in interoperability

3. Tools used in achieving interoperability

4. Conclusions
Differences in type systems

- Languages represent data in different ways
- Statically-typed languages assign types as soon as data is collected.
- Dynamically-typed languages only deal with types when evaluating data.

```java
Class Person
    string name = "Cliff"
    date dateOfBirth = 4/16/1978
    int height = 74
    double weight = 212
end
statically-typed person
```

```java
Class Person
    var name = "Cliff"
    var dateOfBirth = 4/16/1978
    var height = 74
    var weight = 212
end
dynamically-typed person
```
Types in data structures

- Untyped lists can contain different types,

- Strongly typed lists can only contain the type given by the list.

[23, v, "hello", True]
An untyped list

[1, 53, 13, 100]
a typed list

Object[] = [?, ?, ?, ?]
A Java list of Objects
A data structure in one language may be absent in another.

Any language can build any data structure, but it may be more difficult in certain languages.

Building a non-native data structure takes time and effort.

{:name "Cliff", :age 32}

Maps are common data structures, but absent in C.
Handling data

- Languages act on data in different ways.
- Handling NULL or NIL objects.

Result = [Disha, null, Vinita]

Images based on Shetty and Vadivel[2]
Outline

1. Common difficulties in interop
2. Concepts in interoperability
   - Metadata
   - Standards
3. Tools used in achieving interoperability
4. Conclusions
Metadata: Data about data 
or: Information beyond what the data itself can convey

(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 use and checking is up to the programmer.
Metadata and type conversion

no metadata

[1, 2, 3, 4]  ➔  [?, ?, ?, ?]

{:type Integer}

[1, 2, 3, 4]  ➔  [1, 2, 3, 4]
Metadata and standards

{type number} [1, 2, 3, 4] → {type Integer} [?, ?, ?, ?]

{type number} [1, 2, 3, 4] → {type number} [1, 2, 3, 4]
The importance of standards

Standards are meant to ensure:

- Agreement on what metadata is being used, and how.
- All involved parties know how data will be represented.
- Future parties will know how data is represented.
- In general, that correct communication happens.

```json
{
    "name": "Person",
    "properties": {
        "name": {
            "type": "string"
        },
        "birthdate": {
            "type": "date"
        },
        "height": {
            "type": "number"
        },
        "weight": {
            "type": "number"
        }
    }
}
```
Outline

1. Common difficulties in interop
2. Concepts in interoperability
3. Tools used in achieving interoperability
   - Virtual Machines
   - Markup Languages
4. Conclusions
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.
public class Fib{
    public Fib();
    Code:
    0: aload_0
    1: invokespecial #1
    4: return

    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: iload_1
    1: ifne 6
    0: iload_1
    1: invokevirtual #2
    16: isub
    17: invokevirtual #2
    20: aload_0
    21: iload_1
    22: add
    23: return
}

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

}
Interoperability with virtual machines

- Usually some overhead associated with calling other languages.
- Overhead can be lessened when all languages are on one VM.
- High-level languages can have conventions to call other high-level languages on the same VM.
- Common language ensures common syntax and behavior.

A Java method of object cliff:
```java
ciff.getAge();
```

Clojure calling Java:
```clojure
(. getAge cliff)
```

JRuby calling Java:
```ruby
require 'java'
ciff.getAge()
```
Markup languages

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

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

XML model of a person

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

JSON model of a person
Schema and standardization

- Schema provide both standardization and additional metadata.
- Libraries exist to check incoming data against a schema.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Person">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="birthdate" type="xs:date" />
        <xs:element name="height" type="xs:double" />
        <xs:element name="weight" type="xs:double" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
Client/server interop with markup languages

Communication across a network using XML

Client

network

Server

```xml
<Person>
  <name>Cliff</name>
  ...
</Person>

Person
var name = Cliff
...

Person
String name = Cliff
...
```
Outline

1. Common difficulties in interop
2. Concepts in interoperability
3. Tools used in achieving interoperability
4. Conclusions
Interoperability

Conclusions

- Interop allows programmers to extend existing systems without requiring them to know the original language.
- Also allows programmers access to the strengths of languages other than the main system language.
- Metadata and standards allow programmers to reason about interoperability, and to communicate how their system handles interop.
- Virtual machines and markup languages make use of these concepts to enable interop.
Thank you for listening!

Questions?

Contact: malone153@morris.umn.edu
Interoperability

References
