TeachRuby: Imperative Ruby for Introductory CS Curricula

Greg Benjamin and Ben Kirzhner

University of Maryland, College Park

Context

- Object-Oriented Programming has exploded in popularity in recent years.
- Many universities teach students OOP early in their Computer Science education.
- Java is a very popular choice for a first programming language.
  - Advanced Placement Computer Science A Exam
  - CMSC131 here at UMD

Problems with Java

- Students can't get far with OOP until they have a firm understanding of imperative programming.
- Teaching basic imperative concepts in popular OOP languages forces instructors to "hand wave" the OO parts of their programs, confusing students.

HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Echo.java

```
import java.util.Scanner;

public class Echo {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String input = s.nextLine();
        System.out.println("Input was: " + input);
    }
}
```

Let's use Ruby!

- OO scripting language like Python and Perl
- Flexible, "programmer friendly" syntax
- Top level mimics imperative programming

HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

HelloWorld.rb

```
puts "Hello, World!"
```

Echo.java

```
public class Echo {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String input = s.nextLine();
        System.out.println("Input was: " + input);
    }
}
```

Echo.rb

```
puts "Input was: " + gets
```

OOP Problems again!

- Even in Ruby, Array and String manipulation still require method calls.
- Any exceptions students encounter reveal the underlying OO aspects of the language.

Enter TeachRuby

- Imperative dialect of ruby for new programmers
- Keep the flexible syntax, restrict OOP or needlessly complex language features
- Allow for restriction of class and method calls

Language Specification - Types

- We wanted to get away from the object model and instead provide programmers with a compact yet powerful set of primatives
  - Fixnum (int)
  - Float
  - String
  - TrueClass and FalseClass (boolean)
  - NilClass (null)
  - Array
  - Hash
  - Symbol

Language Specification - Syntax

- We also wanted TeachRuby code to look as much like Ruby code as possible, so that it would be easy for students to "graduate" to Ruby
- However, we found it was necessary to make some restrictions on syntax, to keep things simple and relatively easy to understand
  - Class and module definitions
  - BEGIN and END blocks
  - Exceptions (begin / rescue / ensure)
  - Code Blocks

Language Specification - Method Calls

- We wanted to avoid object-oriented method calls and use imperative function calling instead

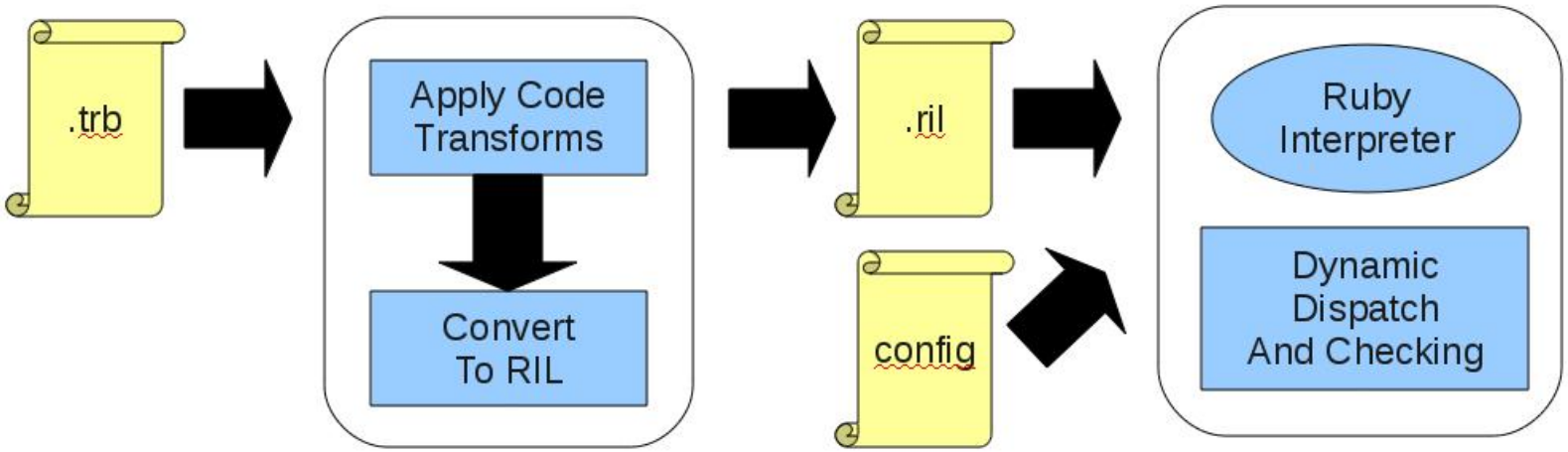
```
str = "Hello".concat " world!"

str = concat "Hello", " world!"
```

Language Specification - Customization

- It would be useful to allow instructors to customize which Ruby features are allowed and forbidden to better suit the needs of their students
- Also, do we allow access to the Ruby Core Library?
- But, we want instructors to still be able to supply their own libraries via either require or a config file

Details - The TeachRuby System



Details - Implementation

- We used DRuby's RIL because it was compact, had nice features for code transformation, and it allowed extensibility for static typing
- We chose to implement our modified syntax rules as a preprocessing step in the RIL parser
  - RIL uses Dypgen, a GLR parser designed to handle ambiguous parses
  - We added hooks in the firing rules to check whether syntactic constructs are allowed or not, according to a config file

Details - Implementation

- We used RIL's code transformation tools to map TeachRuby function calls to Ruby OO calls, but this was tricky:

```
str = concat "Hello", " world!"
puts str
```

- We replaced all function calls with calls to a custom Ruby wrapper to handle dispatch correctly

```
str = __p(:concat, "Hello", " world!")
__p(:puts, str)
```

Details - Implementation

```
def __procedural_function_call(method, *args, &block)
  obj, rest = args[0], args[1..-1]
  if (not obj.nil?) and (obj.respond_to? method)
    __v(obj.send(method, *rest, &block))
  elsif self.respond_to? method, true
    __v(self.send(method, *args, &block))
  else
    raise "TeachRuby: no such function " << method.to_s
  end
end

alias __p, __procedural_function_call
```

Details - Implementation

- Configuration options (core library classes, banned function, etc) are read in dynamically from a YAML config file
- We used RIL to insert "value checks" anywhere there was a hard-coded literal or a return value from a function

Details - Implementation

- Similarly, we can allow and disallow particular functions, which lets us effectively forbid metaprogramming tricks like eval

```
eval <<CODE
    class Foo
        # define some methods!
    end
CODE
```

Evaluation

- We did a case study of 5 "introductory" but nontrivial programming tasks
- We implemented each in Java and TeachRuby and tried to measure ease of programming in lines-of-code

	HelloWorld	Echo	JanTime	Strings	BubbleSort
Java	14	27	42	98	38
TeachRuby	1	10	26	78	23

Evaluation - BubbleSort

```
puts "Enter 10 numbers:"
a = []
for i in (0..9)
    a[i] = to_i gets
end

for i in (0..9)
    for j in (0..9)
        if a[i] < a[j]
            temp = a[i]

          ...
```

```
            ...

                a[i] = a[j]
                a[j] = temp
            end
        end
    end

puts "Sorted:"
for i in (0..9)
    puts a[i]
end
```

Evaluation

- TeachRuby easily outperforms Java in lines-of-code
- TeachRuby also entirely avoids OO ugliness of Java, which shows up even in these simple examples
- However, we don't necessarily trust these numbers
  - Programming tasks very easy, so small LOC numbers
  - Also, we implemented these tasks ourselves, and we are not novice coders
- We'd like to revisit this experiment in the future

Conclusions

- Ruby's flexible syntax is great for introductory programming tasks!
- With only a few changes, Ruby can be used as a simple yet powerful imperative language, without any of the ugliness of Java's object model
- Ruby allows us to enforce customizations dynamically instead of statically, which could be a very useful tool for instructors

Future Work

- We'd like to port the system to be written in Ruby to make the community more receptive to it
- Implement a nice GUI for customization
- Develop a functional dialect of Ruby
- Better error handling and reporting
- Make syntax rule customization dynamic
- Cleverly handle metaprogramming and eval
- Revisit our case studies with a fairer sample of coders