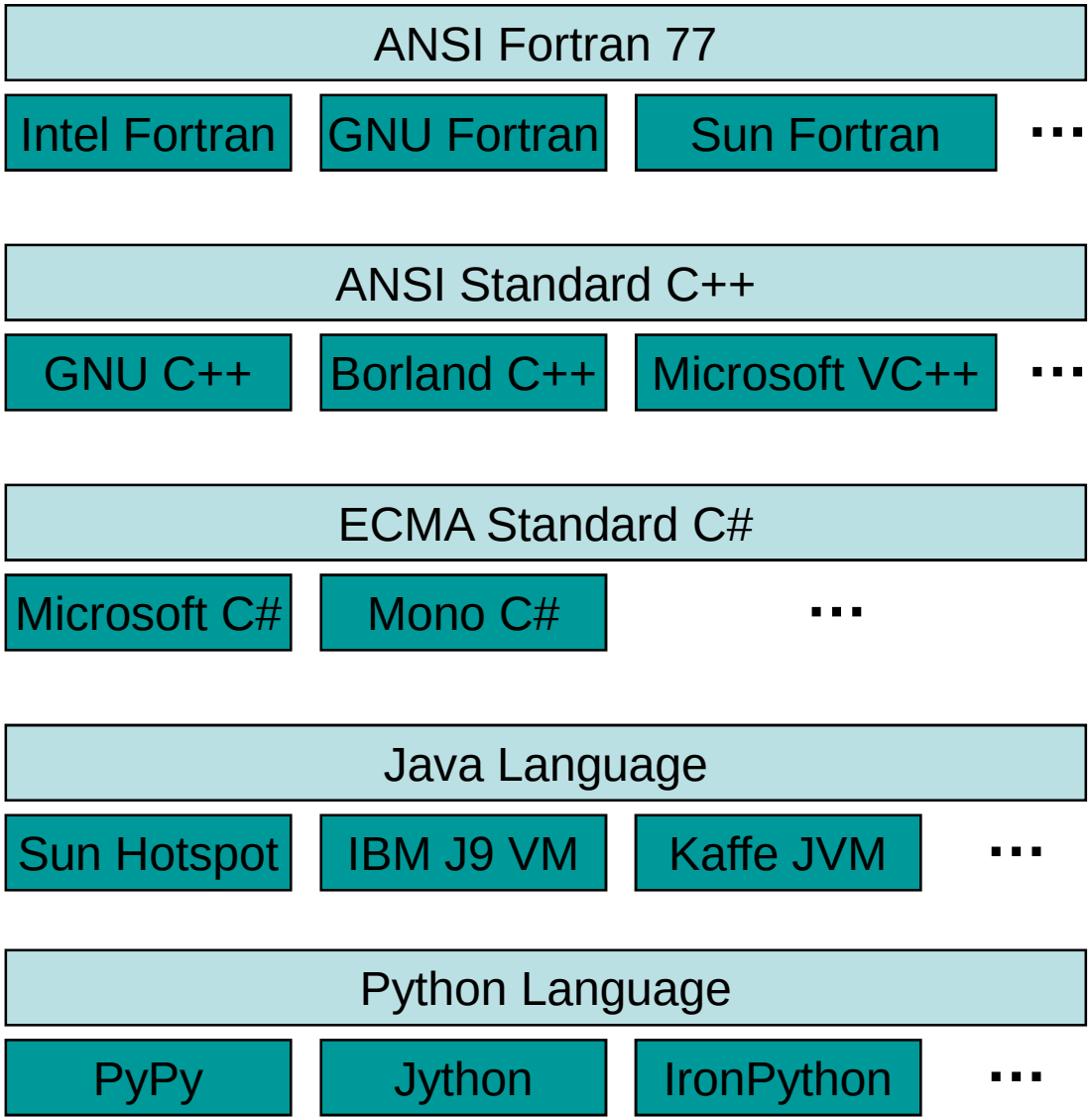
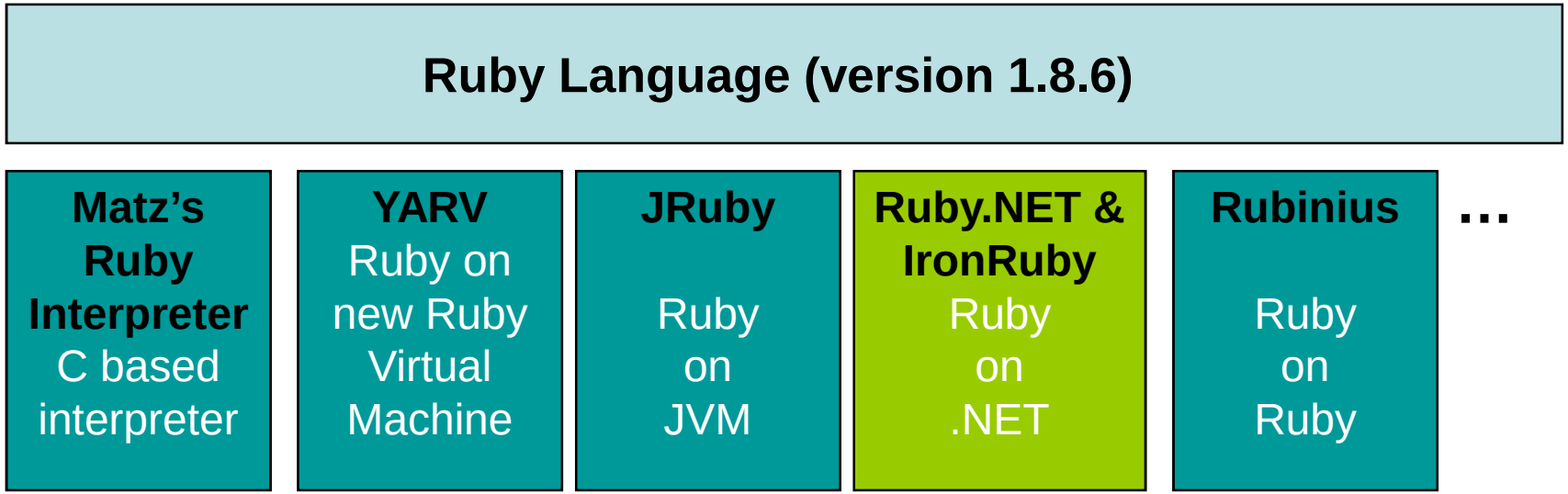
Ruby on .NET

Dr Wayne Kelly
Queensland University of Technology
Australia

Language vs Implementation



Ruby Implementations



RSpec

- Evolving Ruby language specification
- DSL for Behaviour Driven Development (BDD)

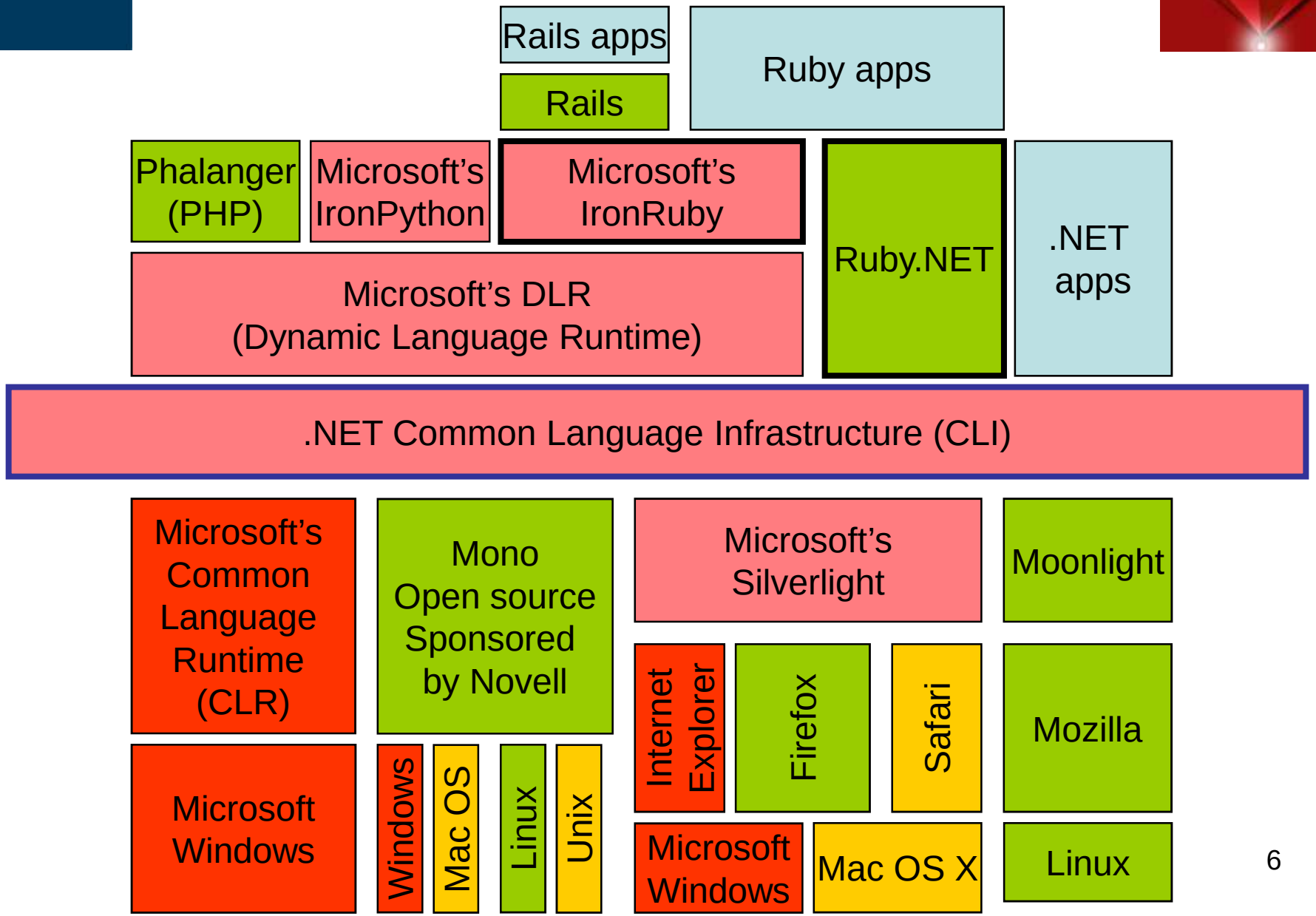
Ruby on .NET

- **Ruby.NET (2005 – 2008)**
  - Developed by Queensland University of Technology, Australia
  - Built directly on top of .NET CLI
- **Wilco Bauwer IronRuby (2006 – 2007)**
  - Dutch college student
- **Microsoft IronRuby (2007 -)**
  - Developed by Microsoft
  - Built on top of .NET Dynamic Language Runtime (DLR)

Why Ruby on .NET

- Adds another language to the .NET stable of languages
  - everyone should be able to choose their favourite language
- Provides Ruby programmers with access to .NET stuff
  - .NET system libraries such as GUI forms
  - Easy interop with other .NET components
  - .NET tools such as Visual Studio, profiling, debugging, etc.
- Where context or policy requires development to be done using .NET
  - eg requirement for fully managed and verifiable components to achieve sandboxed security.
- May provide better performance than MRI.
- New execution context possibilities (such as Silverlight).

Ruby on .NET Stack



Ruby.NET Demo

GUI Forms Design

Implementing a Compiler

Source language: Ruby

Target platform: .NET CLI

1. Create scanner and parser from grammar specification
2. Define and build Abstract Syntax Tree
3. Translate language constructs into platform instructions

Expected Result: an efficient implementation

Ruby.NET (2005 – 2008)

1. Scanning and Parsing
   - No clean simple language spec and grammar
   - Needed to create our own YACC like tool for C# (GPPG)
2. AST
   - Large, but relatively straight forward
3. Translation from Ruby constructs to CLI
   - Direct translation doesn't work due to dynamic semantics
4. Built-in classes, modules and standard libraries
   - Massive amount of porting from native C code to C#

Result: Often slower than MRI

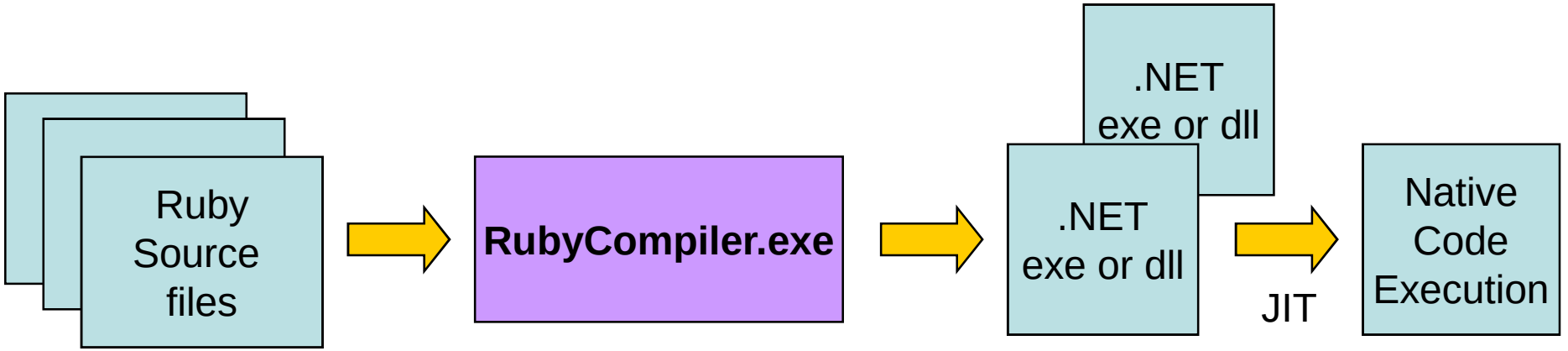
IronRuby (2007 -)

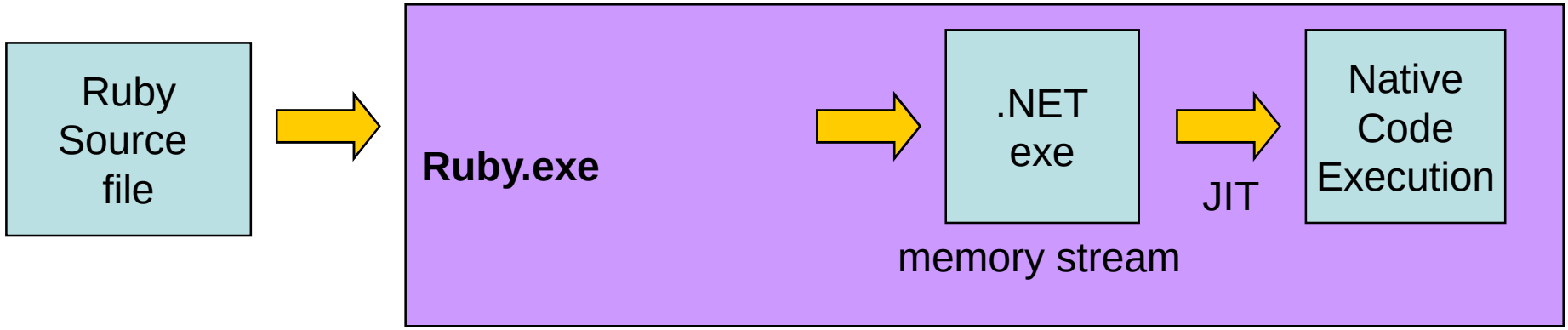
1. Scanning and Parsing
   - Licensed the Ruby.NET scanner and parser
2. AST
   - Created their own (similar) AST
3. Translation from Ruby constructs to CLI
   - Strongly leveraged the Dynamic Language Runtime (DLR).
4. Built-in classes, modules and standard libraries
   - Implementation based on RSpec
   - Optimized for use with DLR.

Goals and Priorities

1. Semantic compatibility with MRI
   - run existing Ruby applications correctly without change
2. .NET Interoperability
   - use other .NET components from Ruby
   - use Ruby components from .NET
3. Performance

Ruby.NET Approach





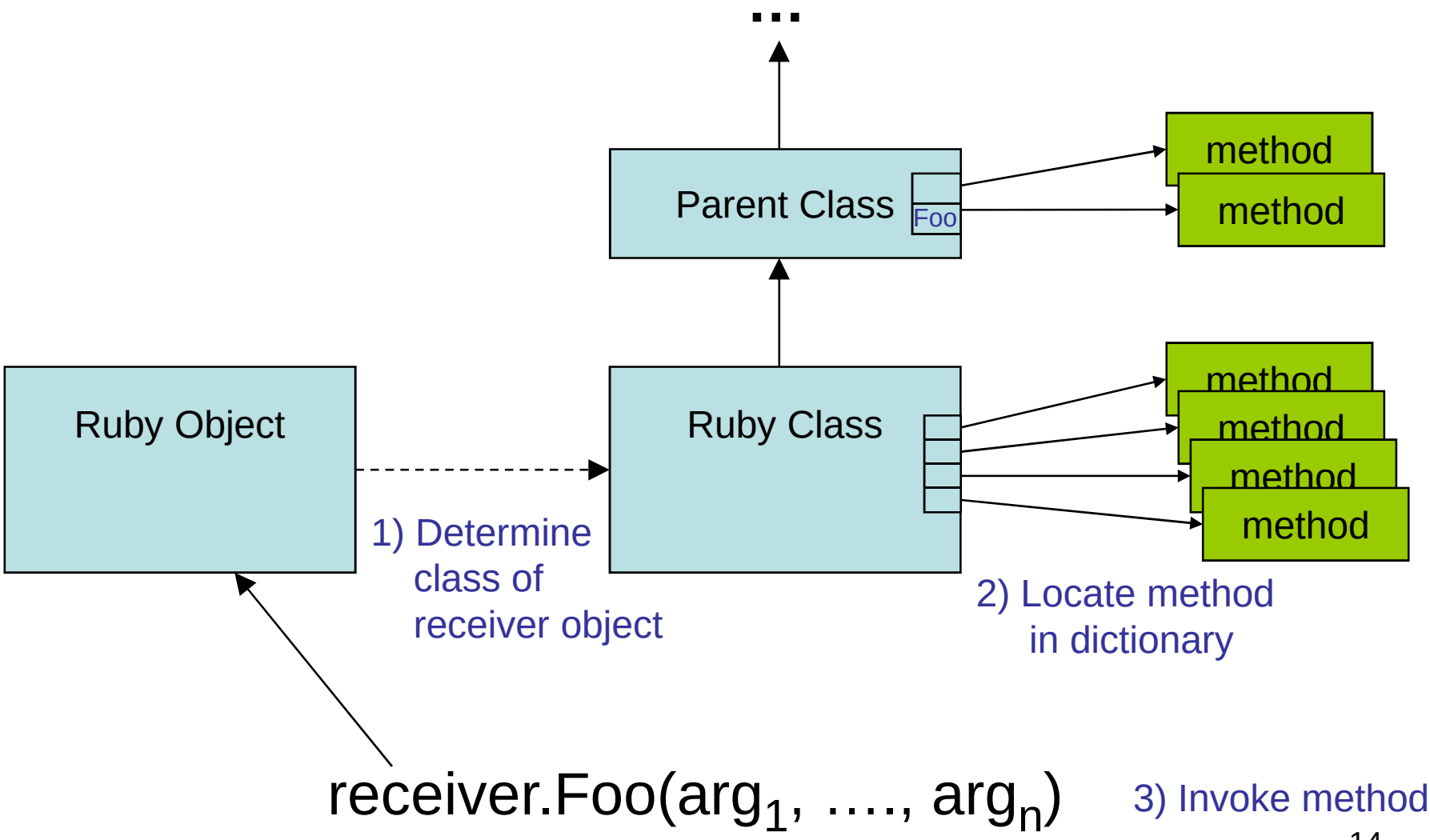
Calling Ruby Methods

- Everything in Ruby is a method call, even:

`x+1`

  - unfortunately, doesn't translate into native addition
  - if x is a Fixnum, calls Fixnum.+
  - we don't generally know the type of x at compile time
  - the standard Fixnum.+ can be replaced
  - the standard Fixnum.+ is non trivial

Ruby Method Dispatch



Fixnum.+ (Ruby.NET)

```
public object fix_plus(Class last_class, object recv, Frame caller, Proc block, object param0)
{
    if (param0 is int)
    {
        try
        {
            return checked((int)recv + (int)param0);
        }
        catch (System.OverflowException)
        {
            return Bignum.NormaliseUsing(IronMath.integer.make((int)recv) + (int)param0);
        }
    }

    if (param0 is Float)
    {
        return new Float((double)(int)recv + ((Float)param0).value);
    }

    return Numeric.rb_num_coerce_bin(recv, param0, "+", caller);
}
```

Fixnum.+ (IronRuby)

```
[RubyMethod("+")]
public static object Add(int self, int other) {
    try {
        return checked(self + other);
    }
    catch(OverflowException) {
        return BigInteger.Add(self, other);
    }
}

[RubyMethod("+")]
public static double Add(int self, double other) {
    return (double)self + other;
}

[RubyMethod("+")]
public static object Add(CodeContext/*!*/ context, object self, object other) {
    return Protocols.CoerceAndCall(context, self, other, LibrarySites.Add);
}
```

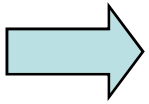
Dynamic Call Sites (with DLR)

```
x = 42

...

   x+1
```

- One DynamicSite object per call site.
- In this case, we know second argument is always Fixnum
- After first call, we *expect* x to be a Fixnum subsequently.
- Optimize call site to simply test that x is Fixnum and then call `Fixnum.Add(int, int)`
- If test(s) fails, call UpdateBindingAndInvoke to dynamically generate new *lightweight* code with new tests
- Self updating call sites– dynamically optimized.
- Note: also need to check that class hasn't been modified.

.NET → Ruby Interop (Ruby.NET)

```ruby
class Foo < Bar
    def print(a, b)
        ...
    end
    if (Verson < 2.9)
        def optimize(x)
            ...
        end
    end
end

x = Foo.new
```



```
public class Foo: Bar
{
  public object print(object a, object b)
  {
    return Eval.Call(this, "print", a, b);
  }
}

public class Foo_print: MethodBody
{
  public override object Call(...) {
    ...
  }
}

public class Foo_optimize: MethodBody
{
  public override object Call(...) {
    ...
  }
}

object x = Eval.Call(Foo, "new");
```

Ruby → .NET Interop

Syntax: `require 'Fred'`

- May load:
  - Fred.rb (Ruby source code), or
  - Fred.so (Native Ruby extension library), or
  - Fred.dll (a normal .NET component).
- Loading a .NET component causes Ruby classes to be created and populated using .NET reflection.
- .NET classes and methods can then be used like normal Ruby classes and methods.
- .NET method overloading requires runtime resolving
- ref and out parameters also pose a challenge.
- To what extent should we do automatic coercion?

Project Status and Future

- I am no longer working on Ruby.NET
- IronRuby:
  - Involved as an external contributor
  - Still in prototype stage
    - you can try it out today, but not yet suitable for production use.
    - currently supports most language features
      (still missing continuations, green threads and eval)
    - currently supports most built-in classes and modules and some native standard libraries (seeking external contributions)
  - Next major goal is to support Gems and Rails
    - hope to have Hello World rails app working by RailsConf.

Links & Questions?

Wayne Kelly:

- Queensland University of Technology, Australia
- w.kelly@qut.edu.au

Ruby.NET:

- http://code.google.com/p/rubydotnetcompiler/
- http://rubydotnet.googlegroups.com/web/Home.htm

IronRuby:

- http://www.ironruby.net/
- http://rubyforge.org/projects/ironruby
- http://rubyforge.org/mailman/listinfo/ironruby-core