Types for Ruby

Jeff Foster
University of Maryland, College Park

Joint work with Mike Furr, David An, Avik Chaudhuri, Stevie Strickland, Brianna Ren, John Toman, Mike Hicks
Types for Ruby

• Over last several years, have been working on bringing some benefits of static typing to Ruby
  ▪ Ruby = Smalltalk + Perl
  ▪ Popular for building web apps (“Ruby on Rails”)

• Goal: Make types optional but useful
  ▪ Develop a program without types (rapidly)
  ▪ Include them (later) to provide static checking where desired
  ▪ Find problems as *early* as possible (but not too early!)

*We think we finally have it right!*
Timeline

(2008)

Diamondback
Ruby

Ruby Intermediate Language

PRuby

Ruby Dust

DRails

Ruby Type Checker

RDL 1.0.0

"RDL"

Hummingbird

RDL 2.0.0
Diamondback Ruby

• How do we build a static type system that accepts “reasonable” Ruby programs?
  ▪ What idioms do Ruby programmers use?
  ▪ What typing features are needed?
  ▪ Are Ruby programs even close to statically type safe?

• Uses type inference to reduce need for annotations
Basic Types

- Type annotations specify class, method, type
  - :string is a Ruby symbol
- Ruby methods can take optional block args
  
  "a".upto("z") { |c| puts c }
Intersection and Union Types

• Intersections for overloading
  ▪ Implemented via dynamic type tests in method body
• Unions in place of interfaces

```ruby
type String, :[], "(Fixnum) → String or nil"
type String, :[], "(Fixnum, Fixnum) → String or nil"
type String, :[], "(Range or Regexp) → String or nil"
type String, :[], "(Regexp, Fixnum) → String or nil"
type String, :[], "(Regexp, String) → String or nil"
type String, :[], "(String) → String or nil"
```
Other Type Language Features

- Optional and vararg types

  ```
  type String, :chomp, "(?String) → String"
  type String, :delete, "(String, *String) → String"
  ```

- Structural types

  ```
  type IO, :puts, "(*[to_s: () → String]) → nil"
  ```

- Generics

  ```
  type Array, :[], "(Range) → Array<t>"
  type Array, :[], "(Fixnum or Float) → t"
  type Array, :[], "(Fixnum, Fixnum) → Array<t>"
  ```

- Self

  ```
  type Object, :clone, '() → self'
  ```
Tuples, Finite Hashes, and Singletons

- Array<Fixnum or String> imprecise
- Instead, use **tuple type** [Fixnum, String]
- Even better, use **singleton type** [1, String]
  - (Note strings are mutable, can’t be singletons)
- Coerce tuples to arrays as needed
  - Note need to apply coercion retroactively!
- Similar approach for finite hashes like \{a: 1, b: 2\}

```ruby
def m() [ 1, 'two' ]; end
a, b = m
# a = 1, b = 'two'
```
RIL and Diamondback Ruby

• Ruby Intermediate Language (RIL)
  ▪ GLR parser for Ruby source code
  ▪ Compact, simplified intermediate representation
  ▪ Pretty-printer that outputs valid, executable Ruby code

• Diamondback Ruby
  ▪ Static type inference for Ruby built on RIL
  ▪ Ahead-of-time constraint generation and resolution
  ▪ PRuby added profiling to handle some dyn features
Example Errors Found

• Typos in names
  ▪ `Archive::Tar::ClosedStream` instead of `Archive::Tar::MiniTar::ClosedStream`
  ▪ `Policy` instead of `Policies`

• Undefined variables

```ruby
return rule_not_found if !@values.include?(value)
```

▪ `rule_not_found` not in scope
▪ Program did include a test suite, but this path not taken
Example Errors Found (cont’d)

• Bad method arguments

```ruby
class Integer
  def to_bn
    OpenSSL::BN.new(self)
  end
end
```

- `BN.new` expects `String`, not `Integer`
- `3.to_bn` would cause a type error
Syntactic Confusion

assert_nothing_raised { @hash['a', 'b'] = 3, 4 }
...
assert_kind_of(Fixnum, @hash['a', 'b'] = 3, 4)

- First passes [3,4] to the []= method of @hash
- Second passes 3 to the []= method, passes 4 as last argument of assert_kind_of
  - Even worse, this error is suppressed at run time due to an undocumented coercion in assert_kind_of
Syntactic Confusion (cont’d)

- Programmer intended to concatenate two strings
- But here the + is parsed as a unary operator whose result is discarded

```ruby
flash[:notice] = "You do not have ..." + "..."
```

- Intention was to assign 1 to all three fields
- But this actually assigns 1 to @count, and nil to @next and @last

```ruby
@count, @next, @last = 1
```
DRuby Was Promising, But...

• RIL somewhat fragile
  - Tied to Ruby 1.8.7; required fixing for Ruby 1.9, and we’re now at Ruby 2.3!

• Profiling doesn’t match well with Ruby
  - Ruby has no “compile” phase, so somewhat unnatural

• Overcoming limitations would require very complex static analysis
  - Complex type system = hard to predict = bad
Becoming Dynamic

(2008)

Diamondback
Ruby

PRuby

DRails

Ruby Intermediate Language

RubyDust

Ruby Type Checker

RDL 1.0.0

“RDL”

Hummingbird

RDL 2.0.0

(Mostly) Static Analysis

(Mostly) Static Analysis
RubyDust and RTC

• **Ruby Dynamic Unraveling of Static Types**
  - Type inference

• The **Ruby Type Checker**
  - Type checking

• RDL
  - Contract system for Ruby

• Dynamic analysis (does not examine source code)
  - Does not require Ruby front-end
  - Language features that are tricky for static analysis become easy
    - Flow-sensitivity, highly dynamic features, etc.
Method Interception

```ruby
def foo(x)
  # foo body
end

alias __rdl_foo_old, foo
def foo(x)
  # special stuff for x
  r = __rdl_foo_old(x)
  # special stuff for r
  return r
end
```

<table>
<thead>
<tr>
<th></th>
<th>x (of type t)</th>
<th>r (of type t)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RDL</strong></td>
<td>check precond</td>
<td>check postcond</td>
</tr>
<tr>
<td>RTC/RubyDust</td>
<td>err unless x.type ≤ t</td>
<td>err unless r.type ≤ t</td>
</tr>
<tr>
<td></td>
<td>x = x.wrap(t)</td>
<td>r = r.wrap(t)</td>
</tr>
</tbody>
</table>
RTC and RubyDust Wrappers

• \( x = o \text{.} \text{wrap}(t) \) returns new obj that
  ▪ Delegates (almost) all methods to \( o \)
  ▪ Associates type \( t \) with object \( o \)

■ When wrapped object \( x \) used, check/constrain \( x \)'s type
  ▪ \( y = \text{“hello”} + x \) requires \( x \text{.} \text{type} \leq \text{String} \)
  ▪ \( y = x + \text{“hello”} \) requires \( x \text{.} \text{type} \leq [+:\text{(String)}\to t'] \)
Results

• Soundness theorem for RubyDust
  ▪ If every path through each method explored, then inferred types are sound

• Some performance improvements in RTC

• But still too heavyweight!
  ▪ wrap really slows things down
  ▪ Sometimes two orders of magnitude!
What’s Next?

Diamondback Ruby

PRuby

DRails

Ruby Intermediate Language

RubyDust

Ruby Type Checker

(Redacted) Dynamic Analysis

“RDL”

Hummingbird

(Mostly) Static Analysis

RDL 2.0.0

RDL 1.0.0

(2008)
Ruby on Rails

• Very popular web app framework
  - Model - class whose instances are DB rows
  - Controller - receives HTTP requests
  - View - HTTP responses; HTML with embedded Ruby

• Large community, lots of users
  - Basecamp, GitHub, Shopify, Airbnb, Twitch, SoundCloud, Hulu, Zendesk, Square, Highrise, …

• Favors “convention over configuration”
  - Watch out for metaprogramming!
No explicit def of `owner`
- Defined via metaprogramming in call to `belongs_to`

Can’t type check in `{DRuby, PRuby, RubyDust, RTC}`!
- DRails was an attempt to handle Rails, but not robust
Putting the Right Ideas Together

Diamondback Ruby

Ruby Intermediate Language

PRuby

DRails

Ruby Type Checker

Type Language

Hummingbird

Static checking

Run code to see what it does

Experience

“RDL”

RDL 1.0.0

Run-time interception, type rep.
Just-in-Time Type Checking

• Type annotations execute at run-time
  ▪ Calling `type` stores type [DRuby] info in global table [RTC/RDL]
  ▪ Metaprogramming [DRails] can generate types as meths created!
    - (Also inspired by PRuby)

• Statically check method body [RIL] at call
  ▪ Uses current global type table
  ▪ Memoize type checking for better performance
Type “Annotations” = Method Calls

```
type String, :[], "(Fixnum) → String or nil"
type String, :[], "(Fixnum, Fixnum) → String or nil"
...
```

- Map from `[String, :[]]` to types in `$__rdl_info`
  - Side effect: can query type info!

```
$ rdl_query String#[]
String#[]: (Fixnum, ?Fixnum) -> String
String#[]: (Regexp or Range<Fixnum>) -> String
String#[]: (Regexp, Fixnum) -> String
String#[]: (Regexp, String) -> String
String#[]: (String) -> String
```
Types and Metaprogramming

Add precondition

...which is a lambda with two args

```ruby
pre(:belongs_to) do |name,options|
  # slightly simplified
  if options
    cls_str = options[:class_name]
  else
    cls_str = name.singularize.camelize
  end
  type name.singularize, "() -> #{cls_str}"
  type "#{name.singularize} =", "(#{cls_str}) -> #{cls_str}" end
```

Zany string manipulation!

Belongs_to(:owner,
  :class_name => "User")

Type :owner, "() -> User"
Type :owner=, "(User) -> User"

Another Example

Transaction has meths:
- type, type=, ...

Transaction = Struct.new(:type, :account_name, :amount)
Transaction.add_types(String, String, String)
t = # some Transaction
name = t.account_name

Now can be type checked!

class Struct
  def self.add_types(*types)
    members.zip(types).each do |name, t|
      self.class_eval do
        type name, "() -> #{t}"
        type "#{name}=", "(#{t}) -> #{t}"
      end
    end
  end
end

For each (member, type) pair...

Create type sigs
Implementation

- Method source code from RIL (\(\rightarrow\) JSON)
- Types from RDL
- Sometimes need type casts

```ruby
module M
def foo(x) bar(x) end end
class C; include M; def bar(x) x + 1 end end
class D; include M; def bar(x) x.to_s end end
```

```ruby
type Marshal, "self.load", "(...) → Object"
var_type :@@cache, "Hash<A, B>"

r = Marshal.load(...)  
@@cache = r.rdl_cast("Hash<A, B>")
```
Experiments

• Ran on 2.3 GHz Intel Core i7 with 8GB memory
• Tests came with app or we wrote them with the goal of covering all app methods
• 6 Applications
  ▪ 3 Rails: Talks, Pubs, Boxroom
  ▪ 2 other metaprogramming: Rolify, Credit Card Transactions (CCT)
  ▪ 1 plain: Countries
### Type Checking Results: Annotations

#### Statically checked methods
- + app-specific trusted types
- + core/std lib types

#### Dynamic types
- Dyn gen’d types for metaprogramming (downcasts, generics)
- # of such used by app

#### Type casts
- (downcasts, generics)

<table>
<thead>
<tr>
<th>App</th>
<th>LoC</th>
<th>Static types</th>
<th>Dynamic types</th>
<th>Casts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Chk’d</td>
<td>App</td>
<td>All</td>
</tr>
<tr>
<td>Talks-1/4/2013</td>
<td>1,055</td>
<td>111</td>
<td>201</td>
<td>363</td>
</tr>
<tr>
<td>Boxroom-1.7.1</td>
<td>854</td>
<td>127</td>
<td>221</td>
<td>306</td>
</tr>
<tr>
<td>Pubs-1/12/2015</td>
<td>620</td>
<td>47</td>
<td>86</td>
<td>171</td>
</tr>
<tr>
<td>Rolify-4.0.0</td>
<td>84</td>
<td>14</td>
<td>24</td>
<td>71</td>
</tr>
<tr>
<td>CCT-3/23/2014</td>
<td>172</td>
<td>23</td>
<td>27</td>
<td>75</td>
</tr>
<tr>
<td>Countries-1.1.0</td>
<td>227</td>
<td>33</td>
<td>40</td>
<td>111</td>
</tr>
</tbody>
</table>
## Type Checking Results: Performance

<table>
<thead>
<tr>
<th>App</th>
<th>Running time (s)</th>
<th></th>
<th></th>
<th>Or. Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Orig</td>
<td>No$</td>
<td>Hum</td>
<td></td>
</tr>
<tr>
<td><strong>Talks-1/4/2013</strong></td>
<td>162</td>
<td>1,590</td>
<td>256</td>
<td>1.6×</td>
</tr>
<tr>
<td><strong>Boxroom-1.7.1</strong></td>
<td>263</td>
<td>705</td>
<td>327</td>
<td>1.2×</td>
</tr>
<tr>
<td><strong>Pubs-1/12/2015</strong></td>
<td>72.0</td>
<td>4,470</td>
<td>217</td>
<td>3.0×</td>
</tr>
<tr>
<td><strong>Rolify-4.0.0</strong></td>
<td>5.63</td>
<td>7.79</td>
<td>6.71</td>
<td>1.2×</td>
</tr>
<tr>
<td><strong>CCT-3/23/2014</strong></td>
<td>3.06</td>
<td>78.2</td>
<td>17.4</td>
<td>5.7×</td>
</tr>
<tr>
<td><strong>Countries-1.1.0</strong></td>
<td>1.02</td>
<td>18.1</td>
<td>4.62</td>
<td>4.5×</td>
</tr>
</tbody>
</table>
# Type Errors in Talks

<table>
<thead>
<tr>
<th>version</th>
<th>code</th>
<th>bug</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8/12-4</td>
<td><code>copute_edit_fields</code></td>
<td>misspelled method</td>
</tr>
<tr>
<td>1/7/12-5</td>
<td>`@list.talks.upcoming {</td>
<td>a, b</td>
</tr>
<tr>
<td>1/26/12-3</td>
<td><code>subscribed_talks(true)</code></td>
<td>wrong arg type</td>
</tr>
<tr>
<td>1/28/12</td>
<td><code>handler.object</code></td>
<td>undefined method</td>
</tr>
</tbody>
</table>
RDL 2.0.0

Diamondback
Ruby

Ruby
Intermediate
Language

PRuby

DRails

RubyDust

Ruby Type
Checker

RDL 1.0.0

"RDL"

Hummingbird

RDL 2.0.0

(2008)
Switching Front-Ends

• RIL is written in OCaml
  ■ Might reduce user base if require ocaml installation!

• RDL Hummingbird impl. uses different parser
  ■ Someone else keeps it up to date with latest Ruby!
  ■ Side effect: great error reporting!

```
typecheck.rb:149:in `error': (RDL::Typecheck::StaticTypeError)
a.rb:7:3: error: got type `String' where return type `Fixnum' expected
a.rb:7:   return 'forty-two'
a.rb:7:     ^~~~~~~~~~~~~~~~~~
```
Occurrence Typing

- Needed to type check methods with \( \cap \) types

```ruby
# Type "(Fixnum) → Fixnum"
# Type "(String) → String"
def m(x)
  if x.is_a? Fixnum
    ...
  elsif x.is_a? String
    ...
  else
    raise "Error!"
  end
end
```

- Many ways to introspect on types
- Singleton types very useful here
Performance

• Still too many dynamic checks

```ruby
type "(Fixnum) -> Fixnum"
def m(x)
    return p(x) # no need to dyn check p's arg type!
end

type "(Fixnum) -> Fixnum"
def p(y)
    return y
end
```

- Will require code rewriting
Usability

• Will anyone want to use this?
  ▪ Some interest from Ruby community
  ▪ But still remains to be seen

• Advertising plans
  ▪ ruby-talk mailing list
  ▪ Some existing contacts in industry, core Ruby devs
  ▪ Rubyconf 2016
  ▪ …?
Conclusions

• The answer: Just-in-time static type checking
  ▪ A long time to figure out, but now seems obvious!

• RDL 2.0.1 is available!
  ▪ [https://github.com/plum-umd/rdl](https://github.com/plum-umd/rdl)
  ▪ [https://rubygems.org/gems/rdl](https://rubygems.org/gems/rdl)
  ▪ v1.0.0 released Dec 18, 2015
  ▪ 4,957 downloads as of Dec 5, 2016

• Lots of future research work
  ▪ Back to type inference
  ▪ Dependent/refinement types
  ▪ …?