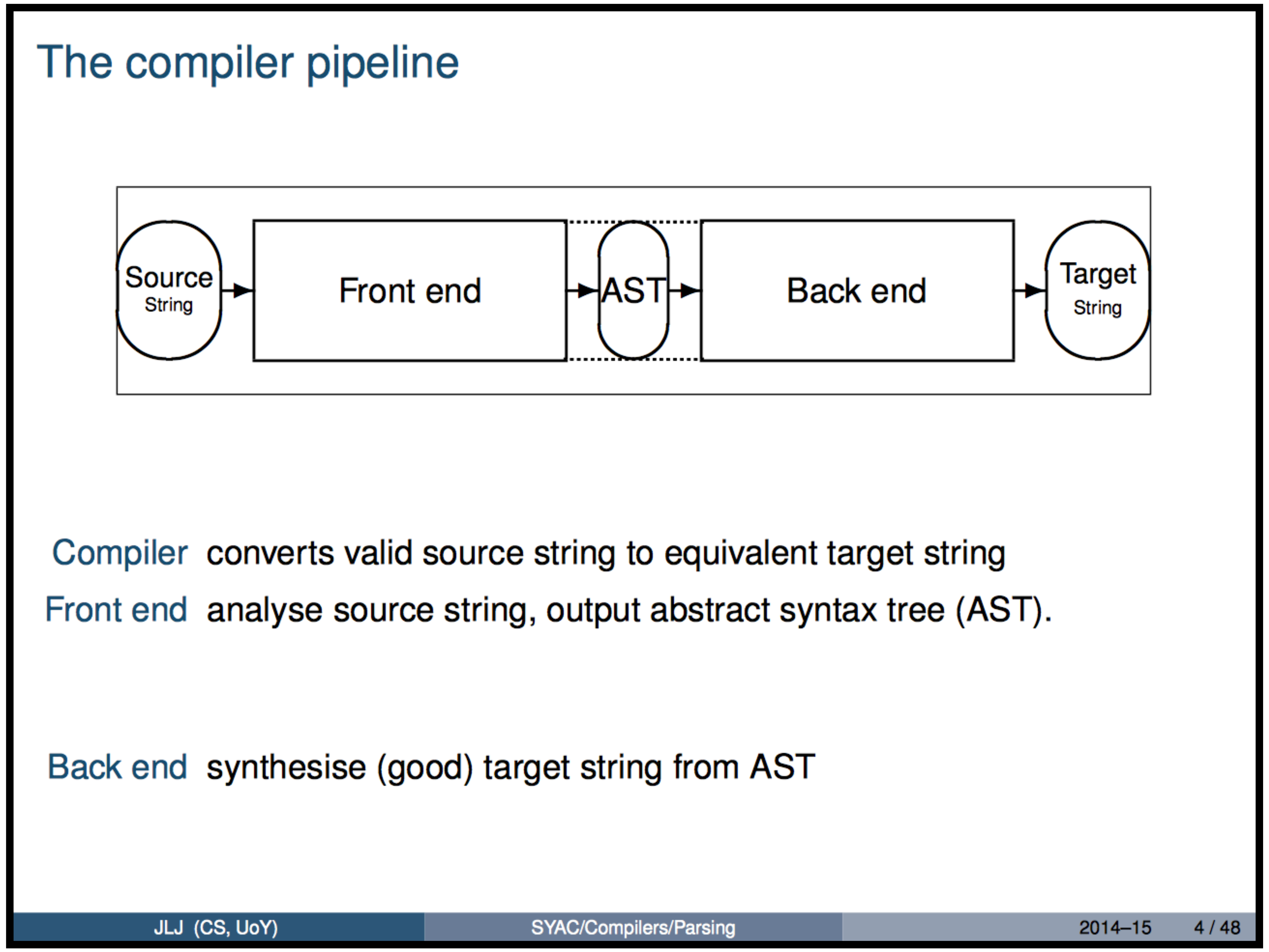
Tools: Ruby Parser

Designing and Maintaining Software (DAMS)

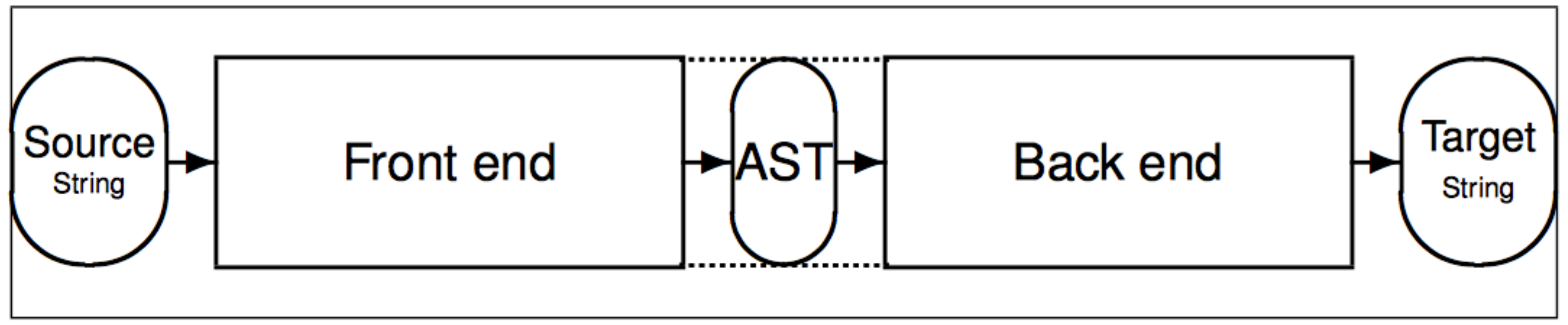
Louis Rose

Parsing recapped

Parsing (strictly speaking "lex-ing and parsing") produces an intermediate representation of a program, called an AST.



The compiler pipeline

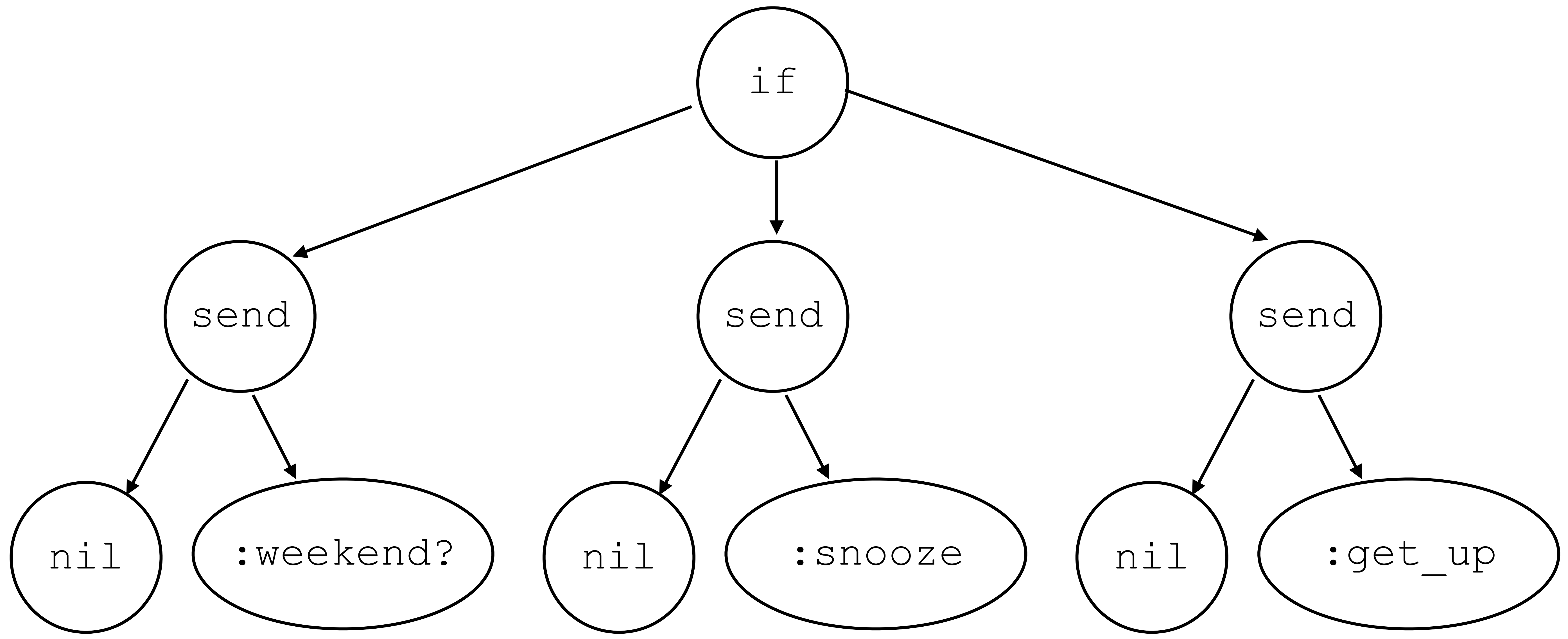


Compiler	converts valid source string to equivalent target string
Front end	analyse source string, output abstract syntax tree (AST).
Back end	synthesise (good) target string from AST

ASTs recapped

ASTs are tree data structures that can be analysed for meaning (following JLJ in SYAC 2014/15).

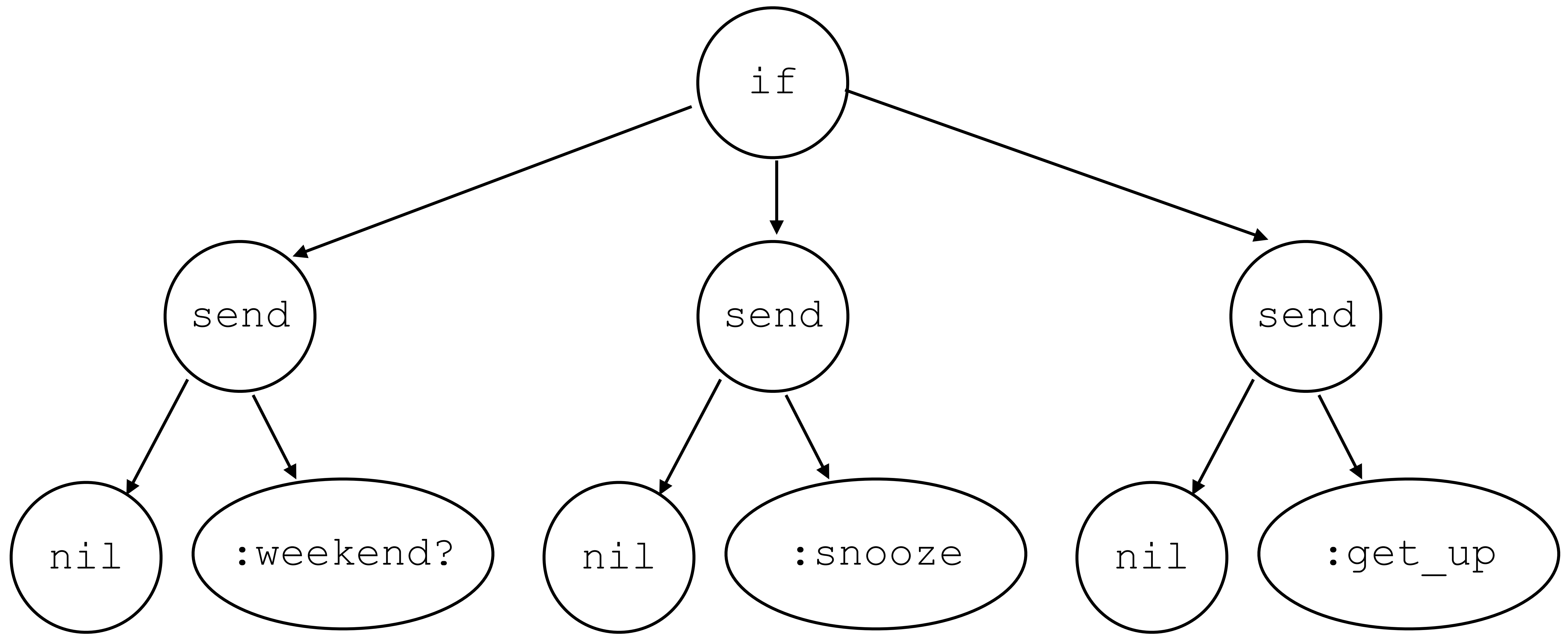
```
if weekend?
  snooze
else
  get_up
end
```



There are many ways to Ruby

Many Ruby constructs can be written with more than one concrete syntax. This does not change the abstract syntax.

```
weekend? ? snooze : get_up
```



Why do we care about parsing?

Many of the habitability factors can be approximated by using measurements of the AST.

Leaner

Less **Complex**

Loosely **Coupled**

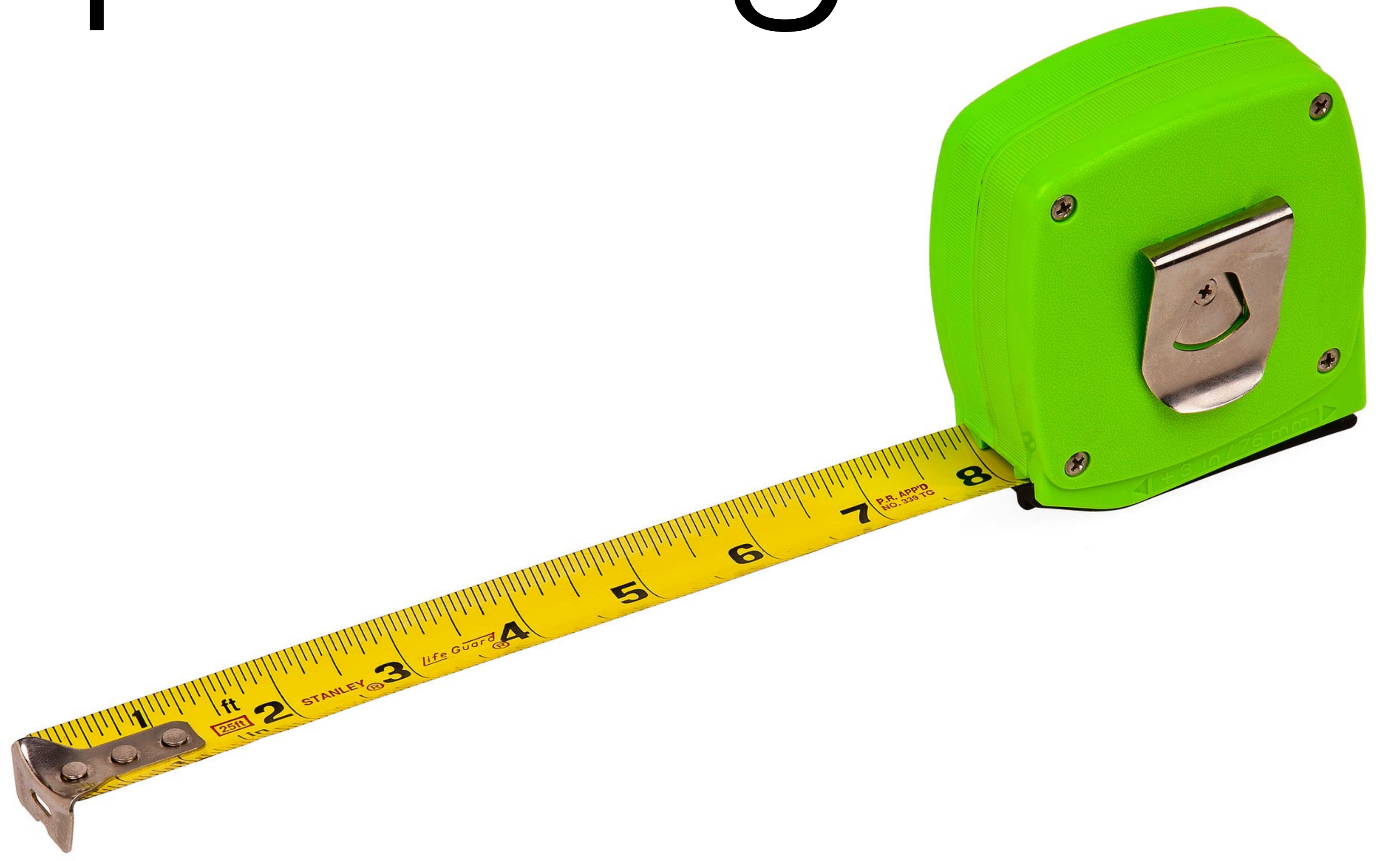
More **Cohesive**

Avoids **Duplication**

Clearer

More **Extensible**

???



Ruby’s parser gem

A Ruby implementation of a Ruby parser. Can be used to parse Ruby on the command line:

```
% gem install parser
...
2 gems installed

% ruby-parse -e "if weekend? then snooze else get_up end”
(if
  (send nil :weekend?)
  (send nil :snooze)
  (send nil :get_up))

% ruby-parse fake.rb
(begin
  (send nil :require
    (str "faker"))
  (send nil :puts
    (send
      (const
        (const nil :Faker) :Name) :name))
```

Ruby’s parser gem

Can also be used as a library from within our Ruby programs, which we’ll use heavily later in DAMS.

```
require "parser/current"

parser = Parser::CurrentRuby
ast = parser.parse("weekend? ? snooze : get_up")

ast.type                         # => :if
ast.children.first.type          # => :send
ast.children.first.children[0] # => nil
ast.children.first.children[1] # => :weekend?
```

Ruby’s parser gem

Includes an abstract class for querying / rewriting the AST.

```
require "parser/current"

class SendCounter < Parser::AST::Processor
  attr_reader :total

  def initialize
    @total = 0
  end

  def on_send(node)
    super(node)
    @total += 1
  end
end

parser = Parser::CurrentRuby
ast = parser.parse("weekend? ? snooze : get_up")
counter = SendCounter.new
counter.process(ast)
counter.total # => 3
```