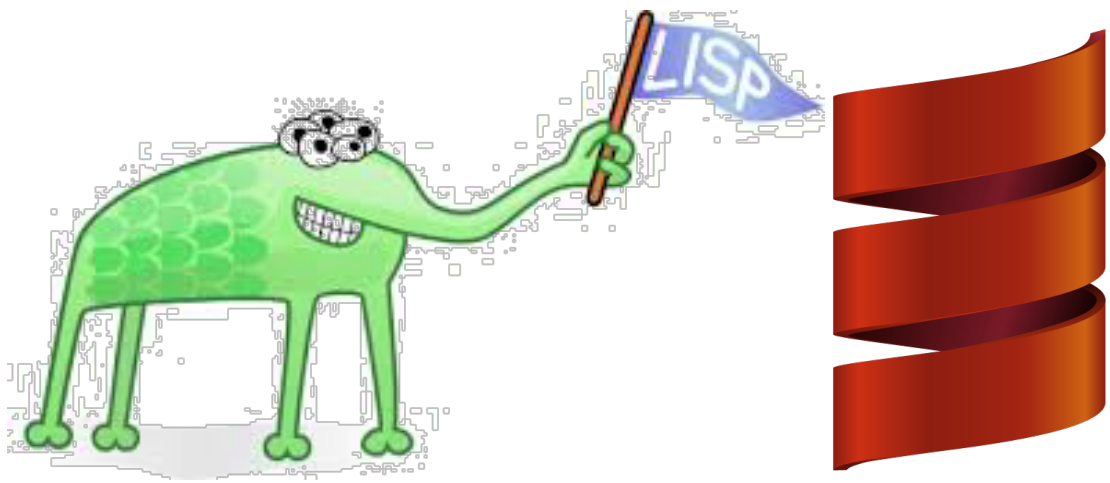
Common Lisp and Scala

Dr. Jim Newton

EPITA Research Laboratory

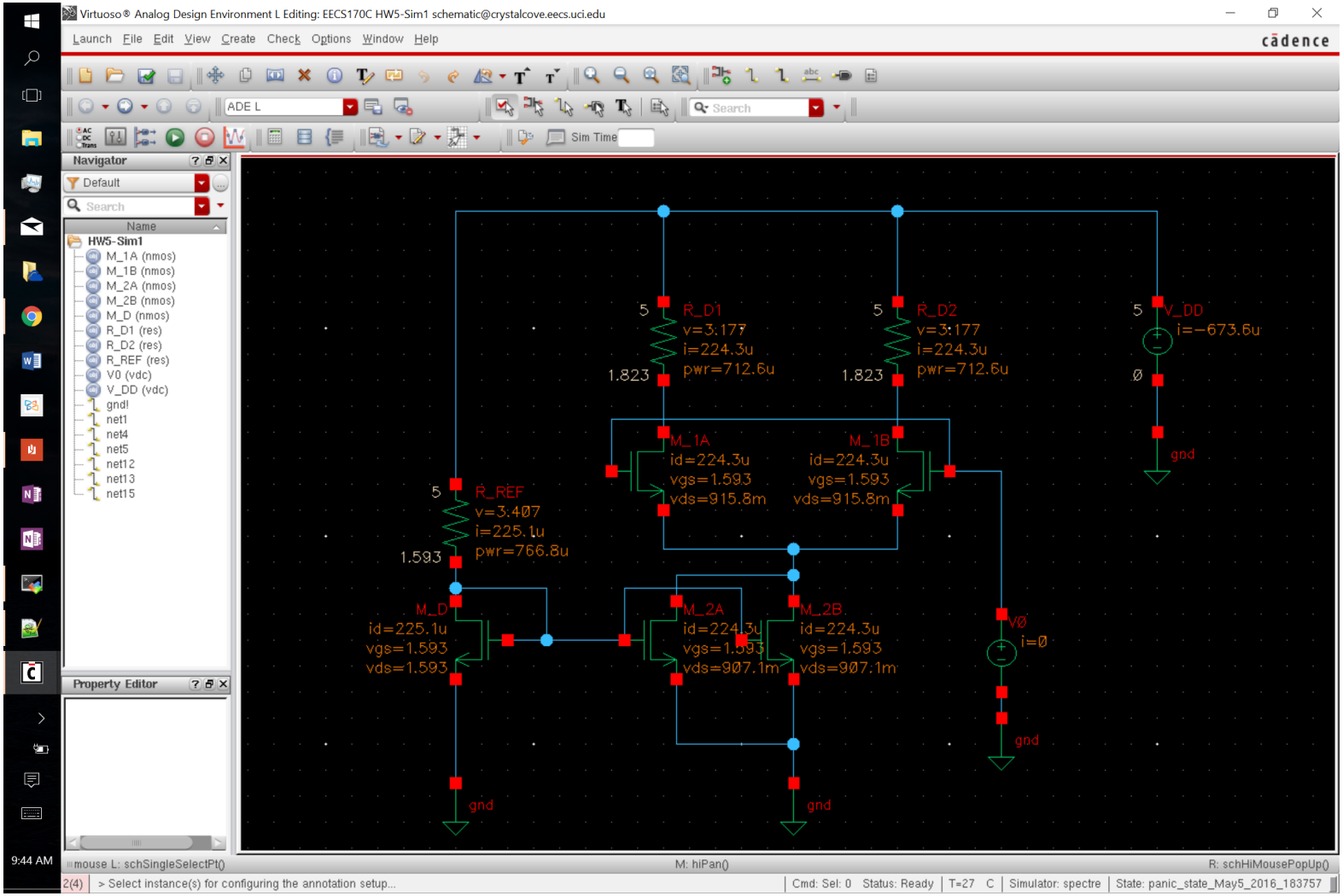
September 22, 2026



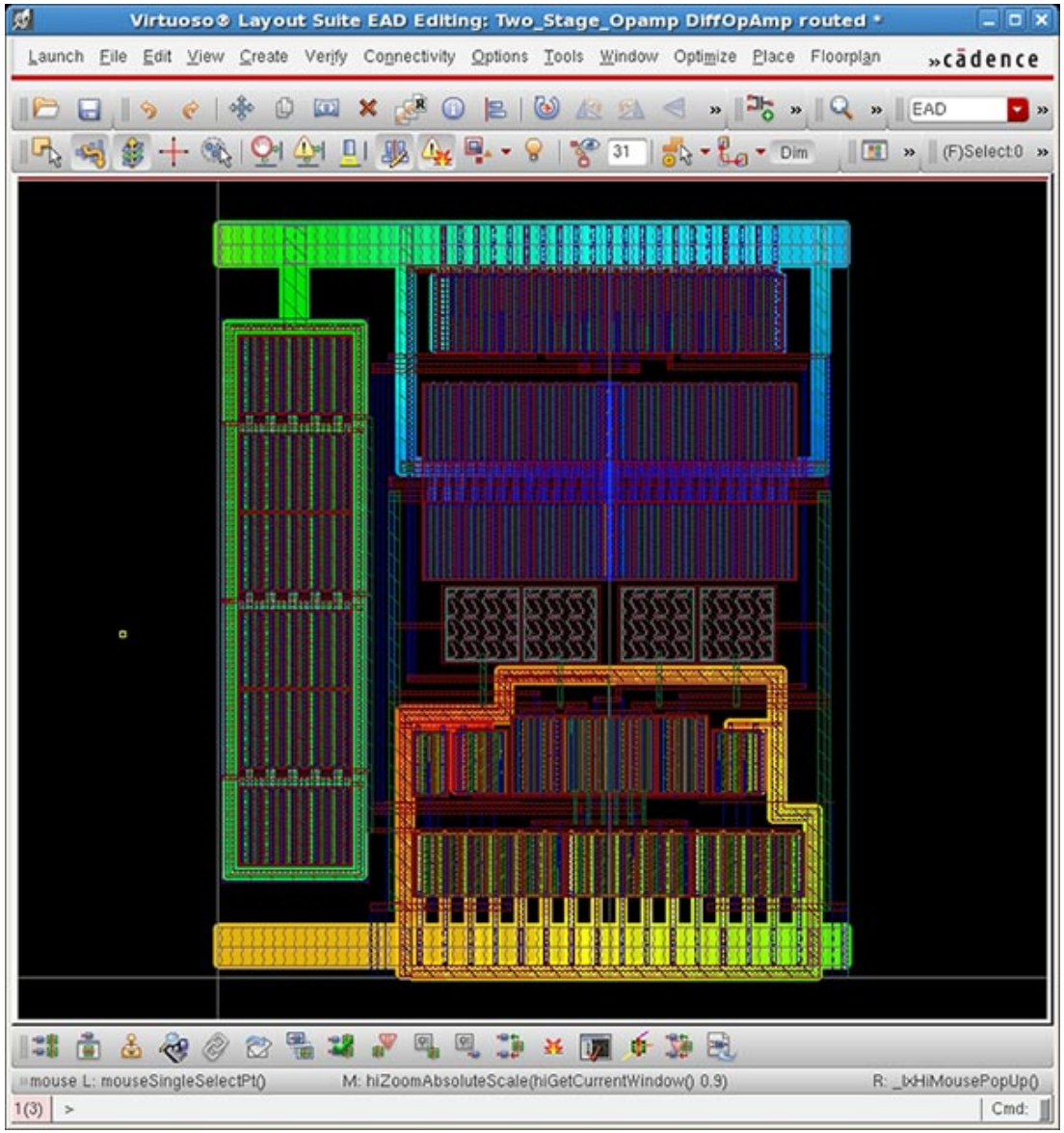
Who is Jim?

- Like most everyone, a hobby programmer nerd since childhood.
- BASIC
- Pascal, C, Postscript, 6502 Assembler
- Perl, Python, Lisp

Lisp since 1988 — Electronic Design Automation (EDA)

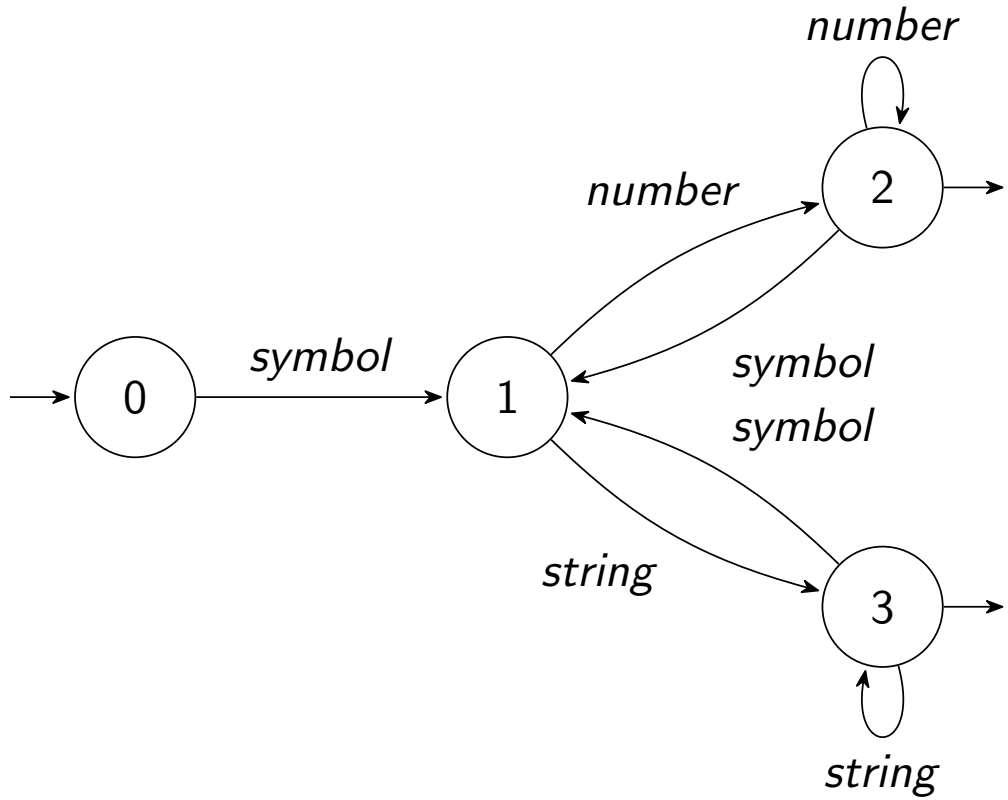


Lisp since 1988 — Electronic Design Automation (EDA)



PhD Thesis, 2015-2018

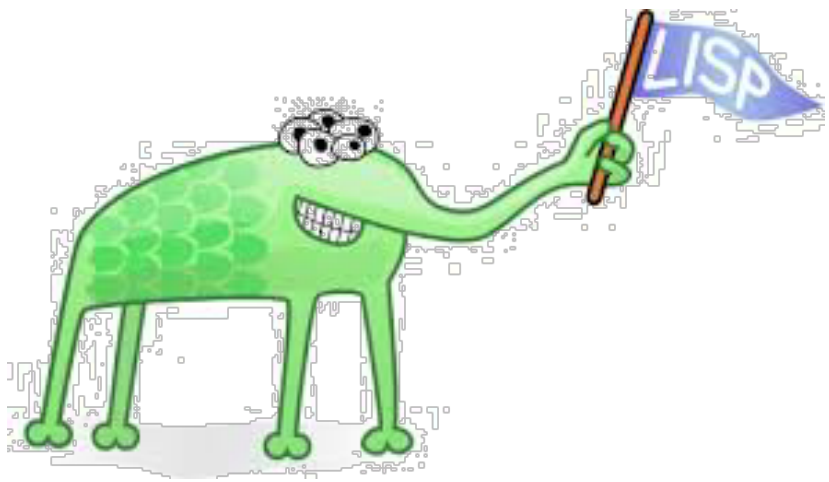
Representing and Computing with Types in Dynamically Typed Languages.

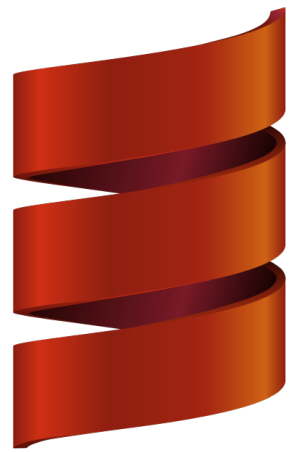


Ensignant Chercher at EPITA

Programming

- Lisp since 1988 — Common Lisp since 2003 — Scala since 2017
- Lisp 35 years — Common Lisp 20 years — Scala 6 years





Overview

1. Rational Type Expressions
2. Common Lisp Overview
   - Defining Functions
   - Mapping Functions
   - Macros
   - Object System
3. Development Flow

RTE: Rational Type Expressions

- Normal regular expression
  - code: `"a*b|c"`
  - math: $(a^* b) + c$

RTE: Rational Type Expressions

- Normal regular expression
  - code: `"a*b|c"`
  - math: $(a^* b) + c$
- Replace letters with types
  - math: $(integer^* string) + float$
  - Lisp: `(:or (:cat (:* integer) string) float)`
  - Scala: `Or( Cat( Star(Int) String) Double)`

RTE: Rational Type Expressions

- Normal regular expression
  - code: `"a*b|c"`
  - math: $(a^* b) + c$
- Replace letters with types
  - math: $(integer^* string) + float$
  - Lisp: `(:or (:cat (:* integer) string) float)`
  - Scala: `Or( Cat( Star( classOf[Int]) classOf[String]) classOf[Double])`

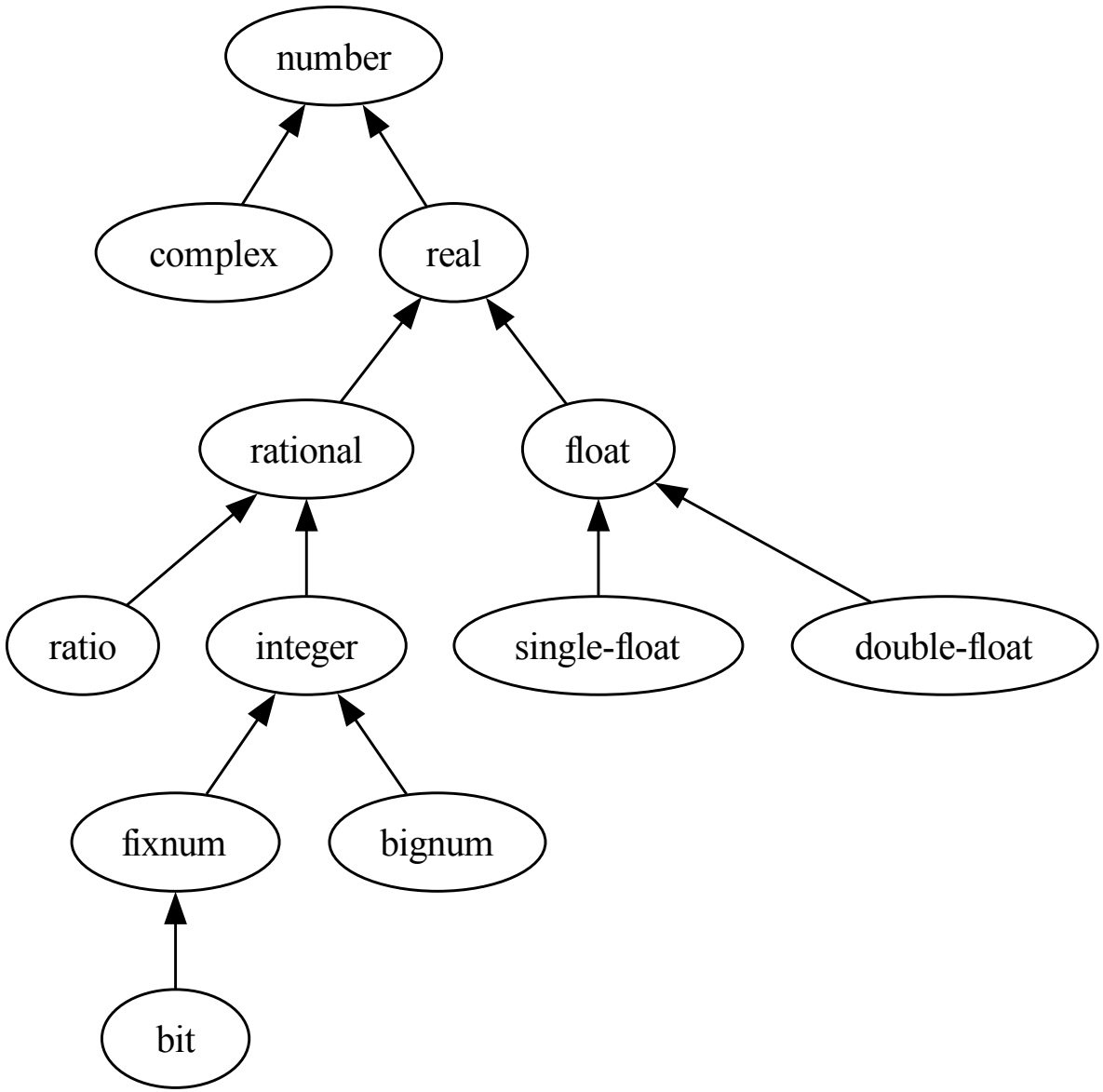
RTE: Rational Type Expressions

- Normal regular expression
  - code: `"a*b|c"`
  - math: $(a^* b) + c$
- Replace letters with types
  - math: $(integer^* string) + float$
  - Lisp: `(:or (:cat (:* integer) string) float)`
  - Scala: `Or( Cat( Star( classOf[Int]) classOf[String]) classOf[Double])`
- Support for type lattice
  - $integer \cap \overline{bignum}$
  - $integer \cup string$

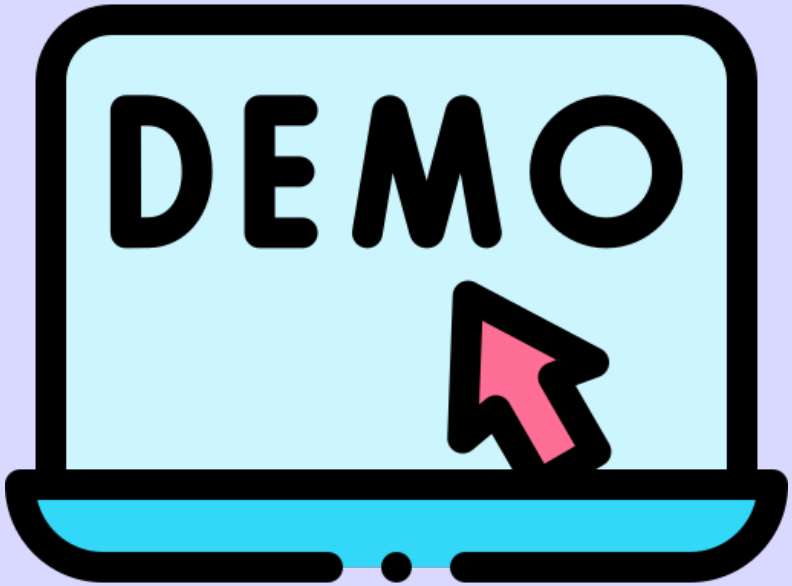
Problems/Challenges with Types

- Can overlap
  - *integer* vs *number*
- Can be empty
  - $integer \cap string$
- Can be equivalent
  - $number \cap \overline{string}$
  - $integer \cup number$
- Semantics specific to programming language.
- Subtype procedure not decidable.

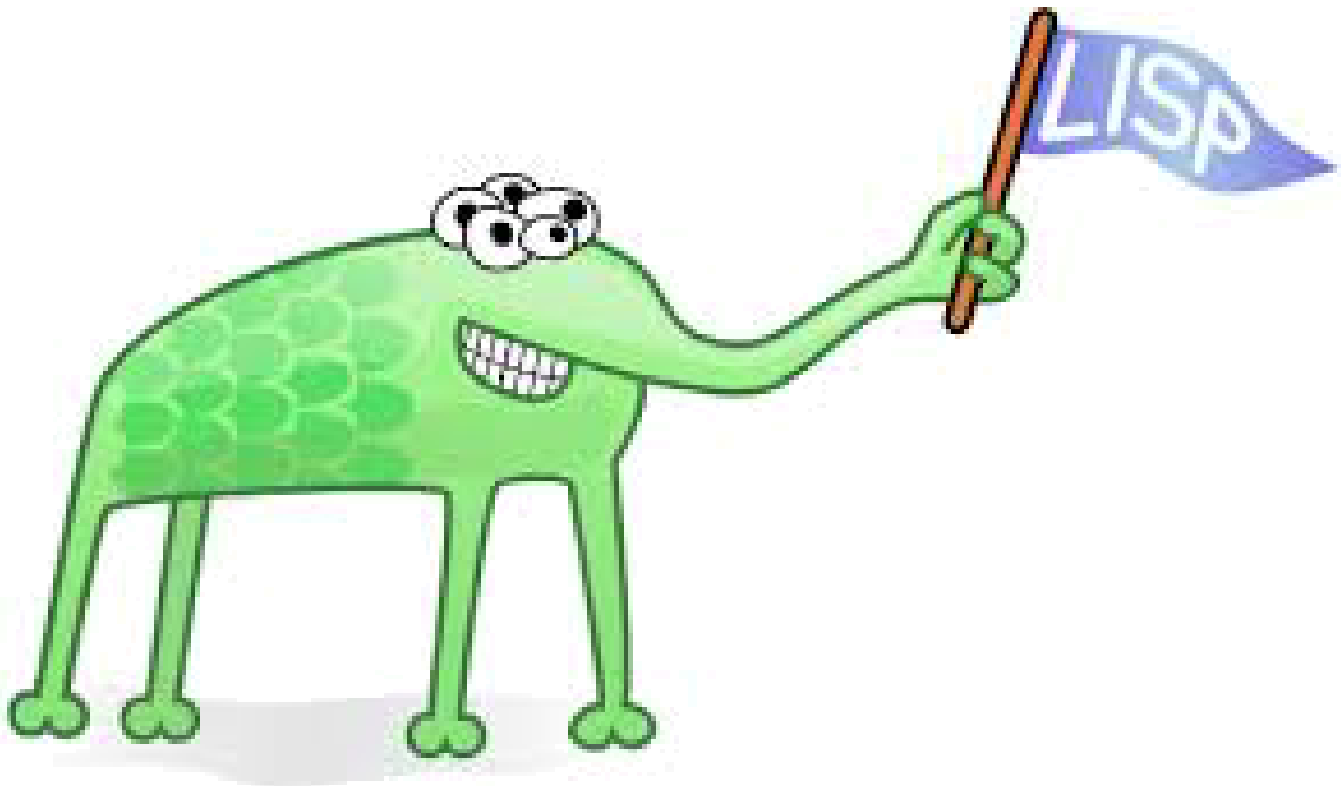
Numeric types



RTE Demo



Common Lisp a Overview



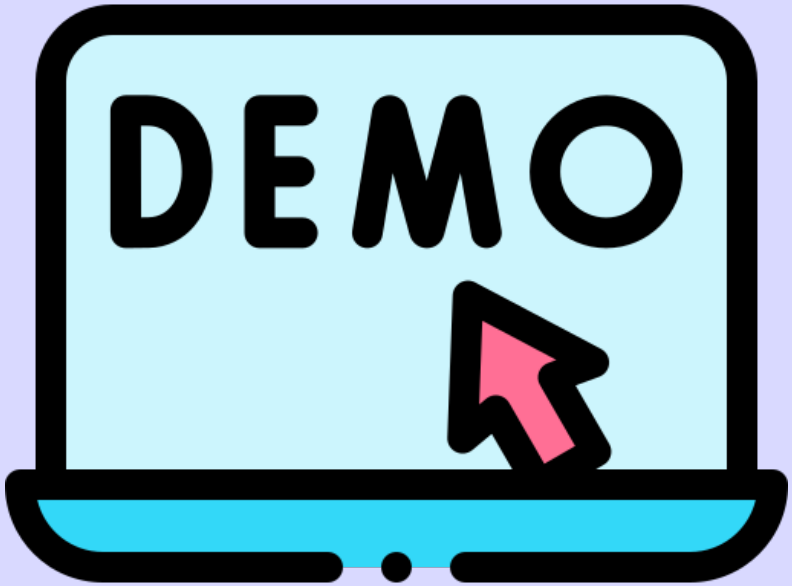
History

- Many lisps: elisp, Clojure, MIT Scheme, LeLisp, *Julia*
- Common Lisp the Language (first edition, Guy Steele) 1984
  - ANSI standard specification, 1994
  - Many implementations, SBCL, Allegro, ABCL, etc
  - Multi-paradigm, REPL, native-compiled, dynamic, dynamically typed
  - `https://lisp-lang.org`

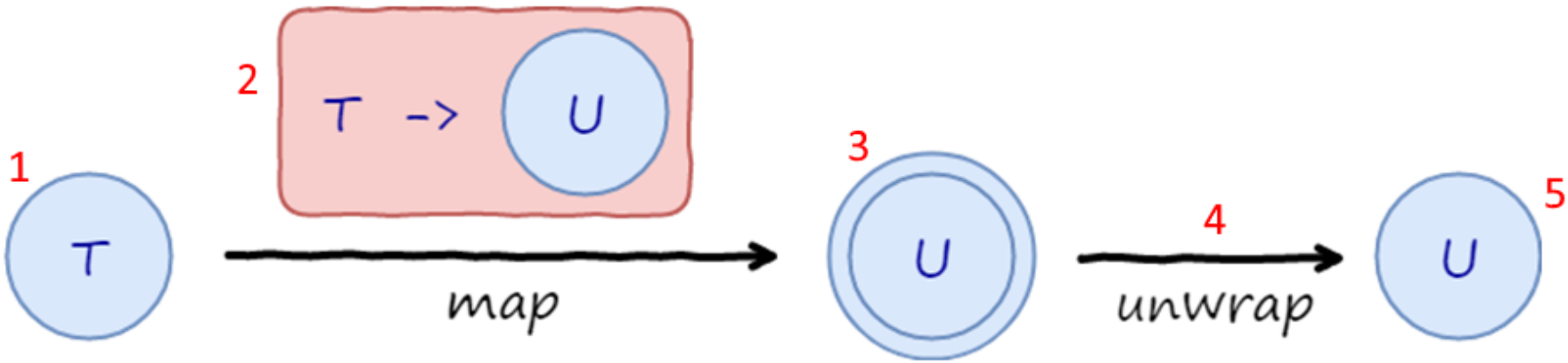
Defining Functions

- Syntax
- Evaluation
- Compilation
- Optimization
- Disassembly

Functions and Syntax Demo



Mapping Functions



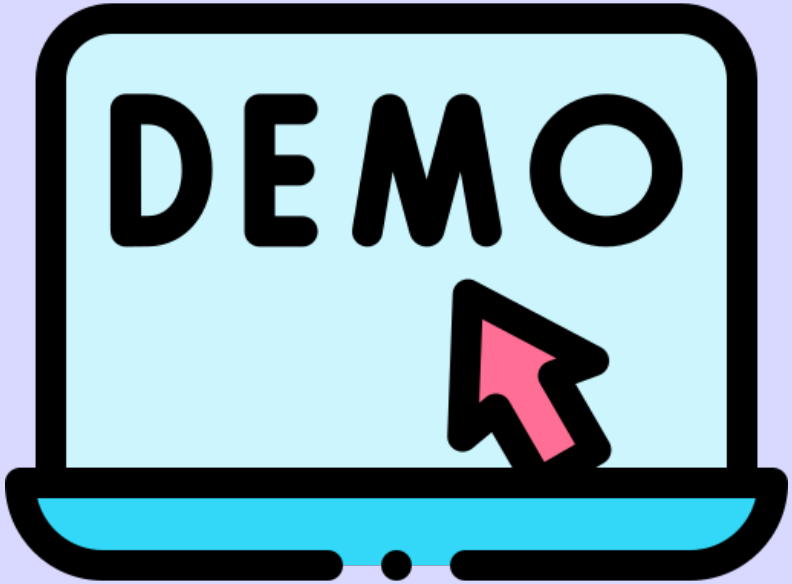
Mapping functions

function	map	by	results		
	car	cons	ignore	collect	concat
mapc	X		X		
mapcar	X			X	
mapcan	X				X
mapl		X	X		
maplist		X		X	
mapcon		X			X

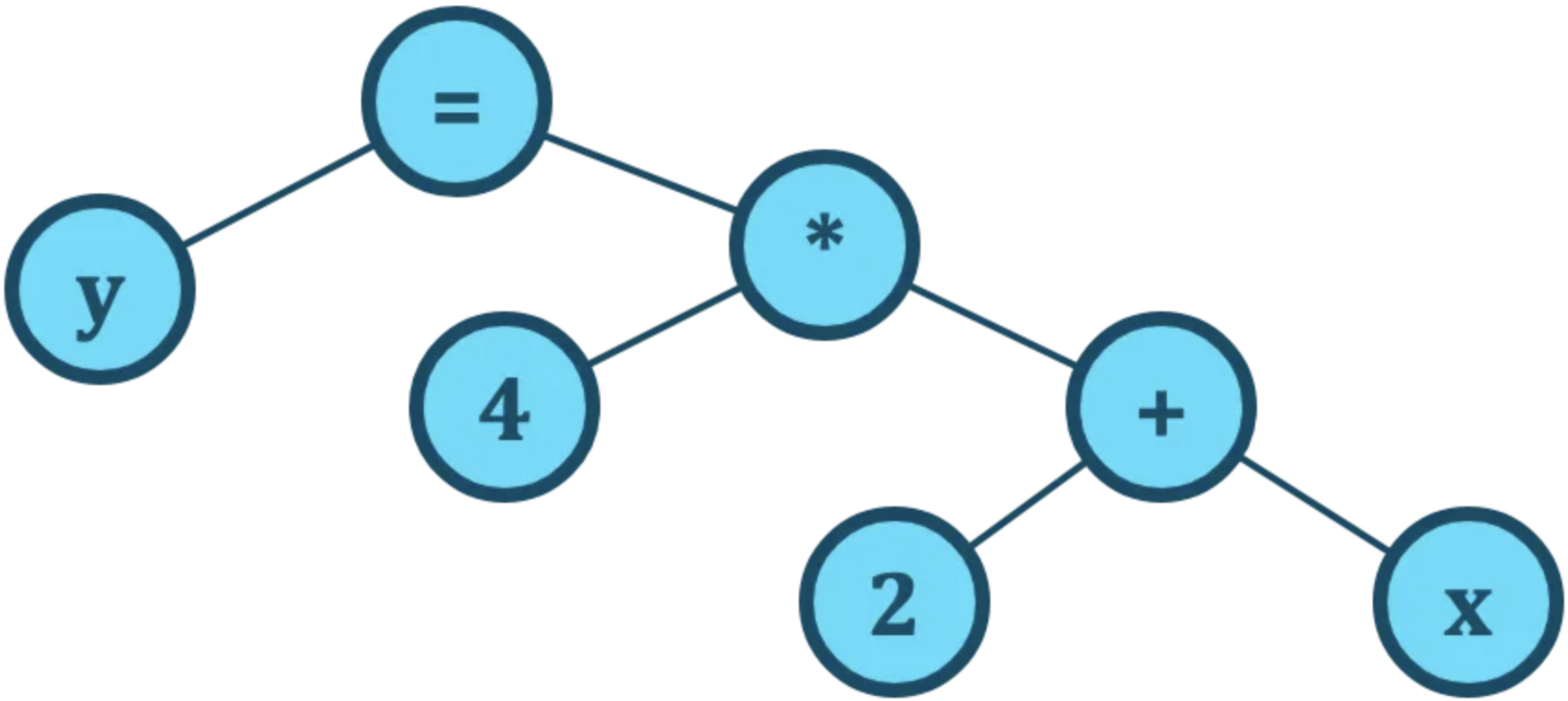
Why no `zip`?

- Map across multiple lists in parallel.

Mapping Functions Demo



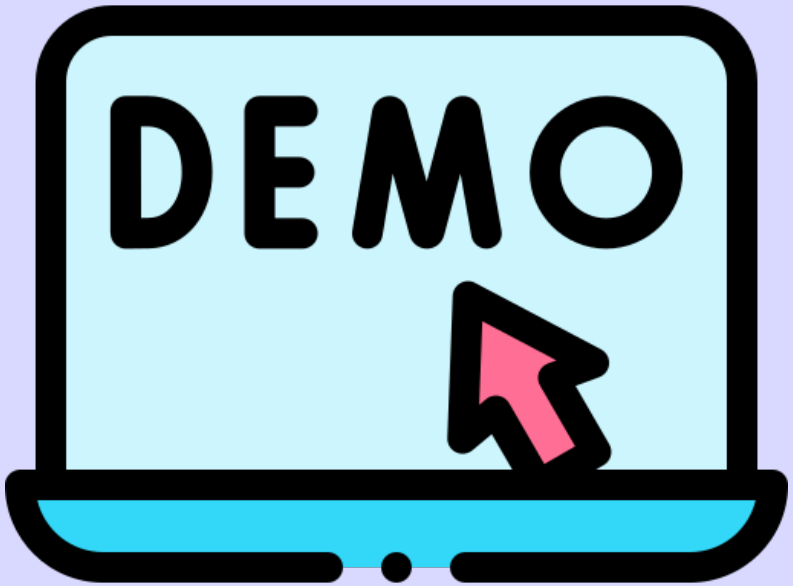
Macros, code that writes code.



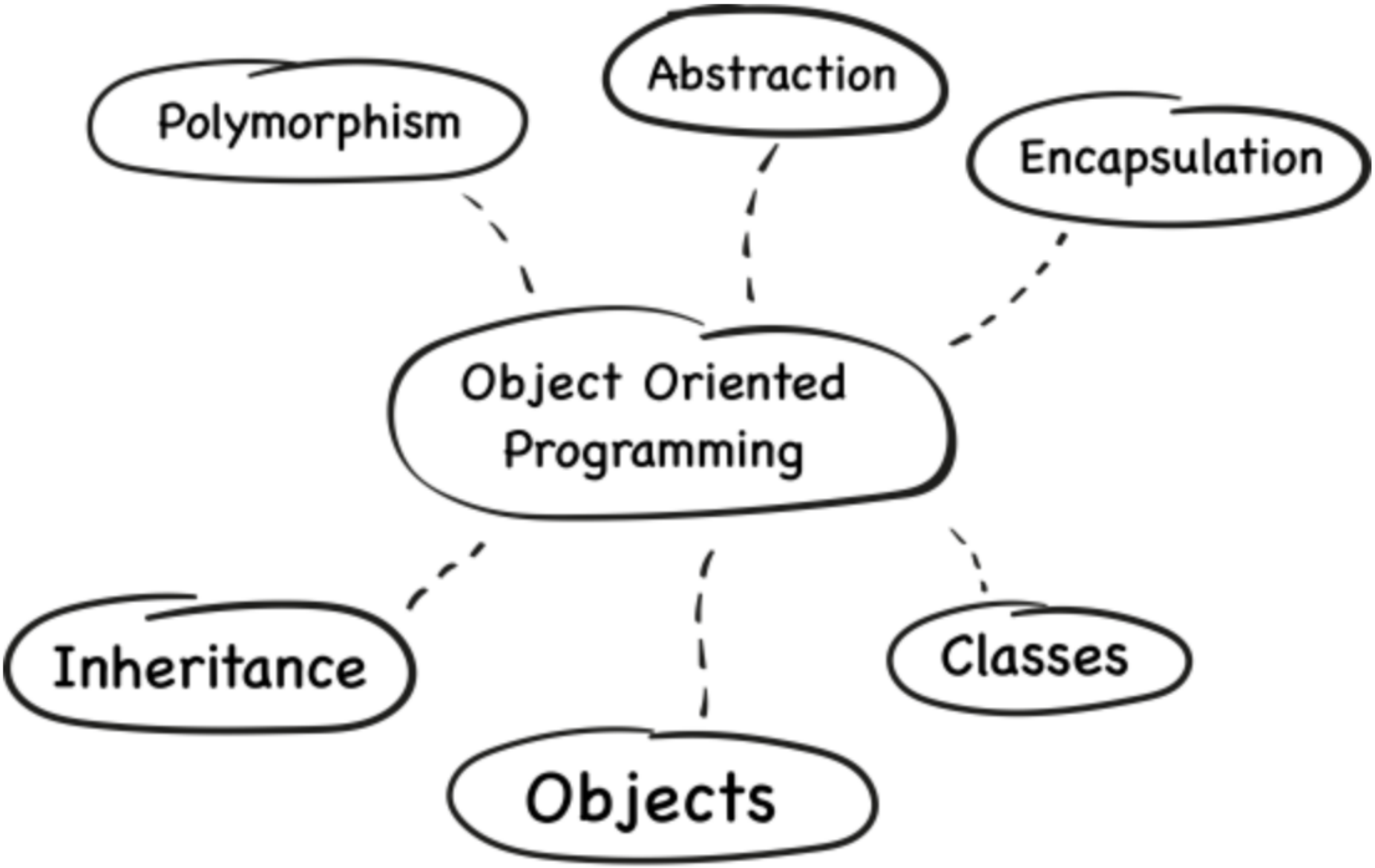
Macros

- Syntactical Abstraction
- Homoiconicity
- Partial Predicate Problem

Macros Demo



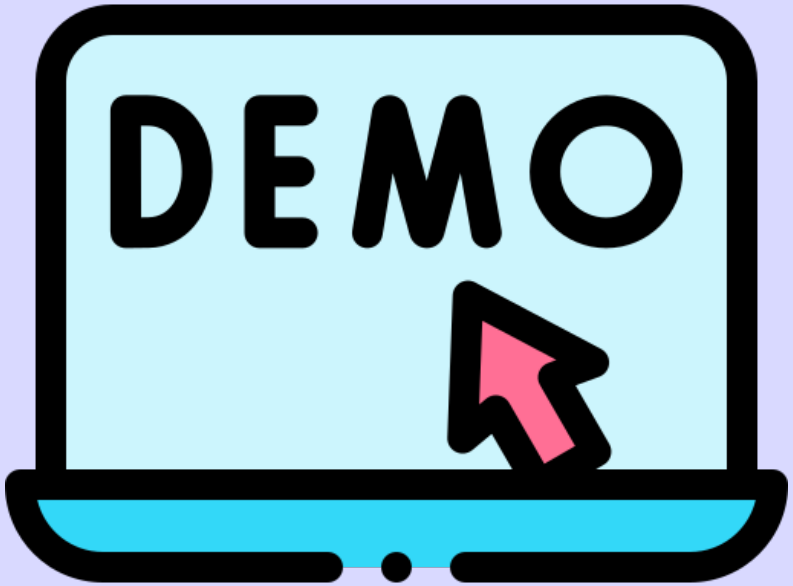
Object Orientation



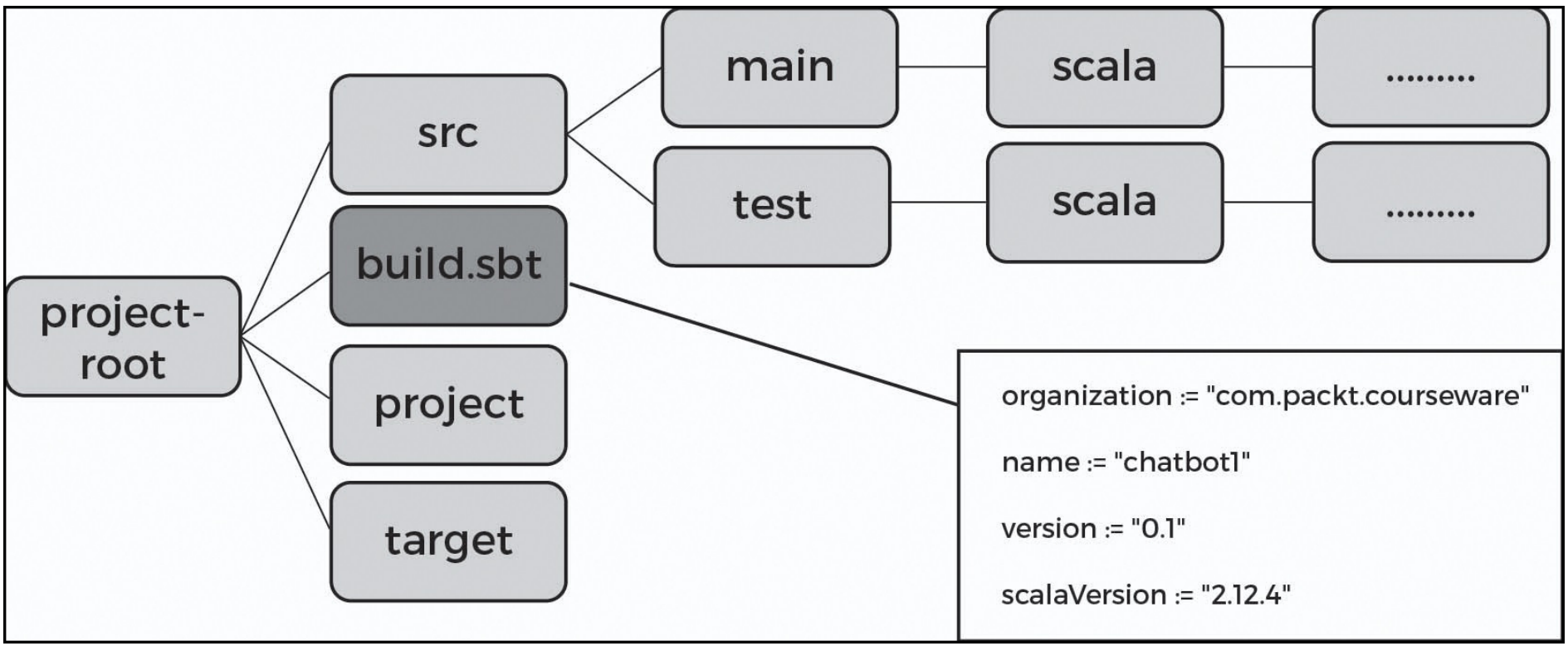
Object System: CLOS

- Classes
- Generic Functions
- Type hierarchy / lattice
- `subtypep`

Object Orientation Demo



Development Flow and Infrastructure



Development Flow

- Interactive
- Compile as you go
- Test broken/unfinished code
- System definition. *e.g.* `rte-test.asd` `research.asd`
- Code structure should not match directory structure

Conclusion

- Common Lisp is a historically interesting language with many interesting ideas.
- It strives to optimize expressivity.
- There are many similarities to Scala, but some fundamental differences in the way to approach certain problems.
- Common Lisp and Scala are both *weird*.
- Weird things make life interesting.

Q/A

Questions?