

A detailed oil painting by Pieter Bruegel the Elder depicting the Tower of Babel. The central focus is a massive, multi-tiered stone structure rising from a rocky hill. The tower is covered in various architectural styles and has multiple levels with arched windows and doors. In the foreground, a group of people in period clothing are gathered around a stone wall, looking up at the tower. The background shows a vast landscape with rolling hills and a distant city. The sky is filled with clouds.

MULTI LANGUAGE PROGRAMMER

I PROBLEM

5 LANGUAGES

N + I SOLUTIONS

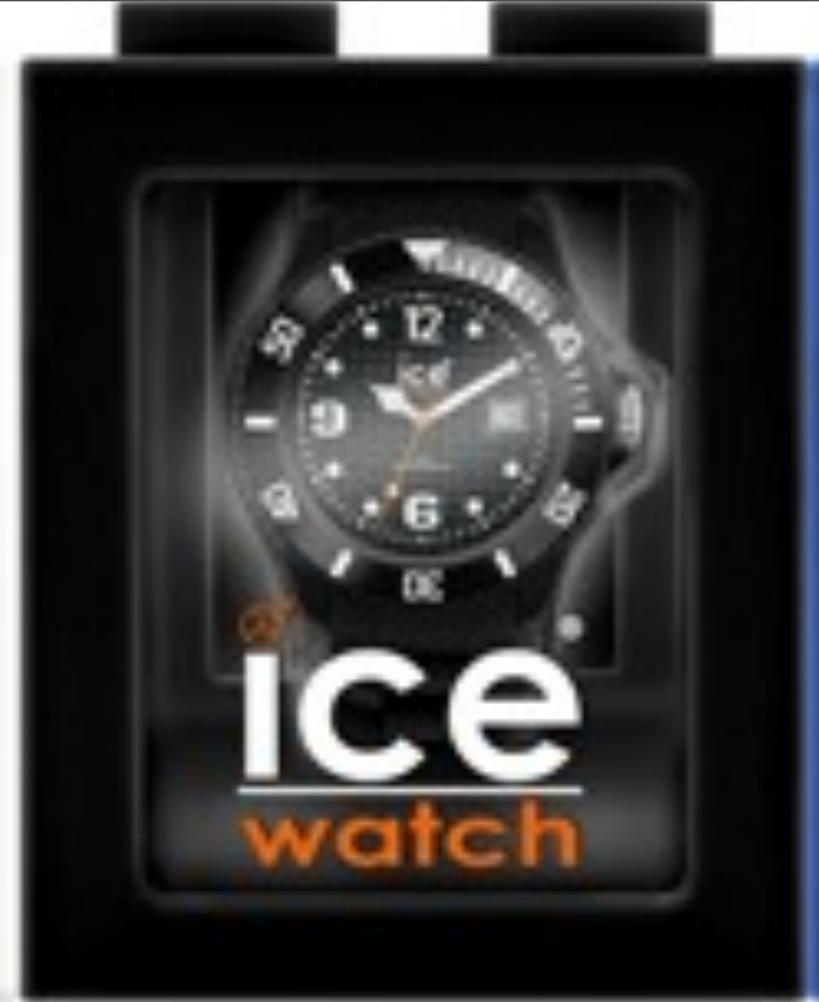
THE PROBLEM IS ABOUT...





vrijdag 25 november 11

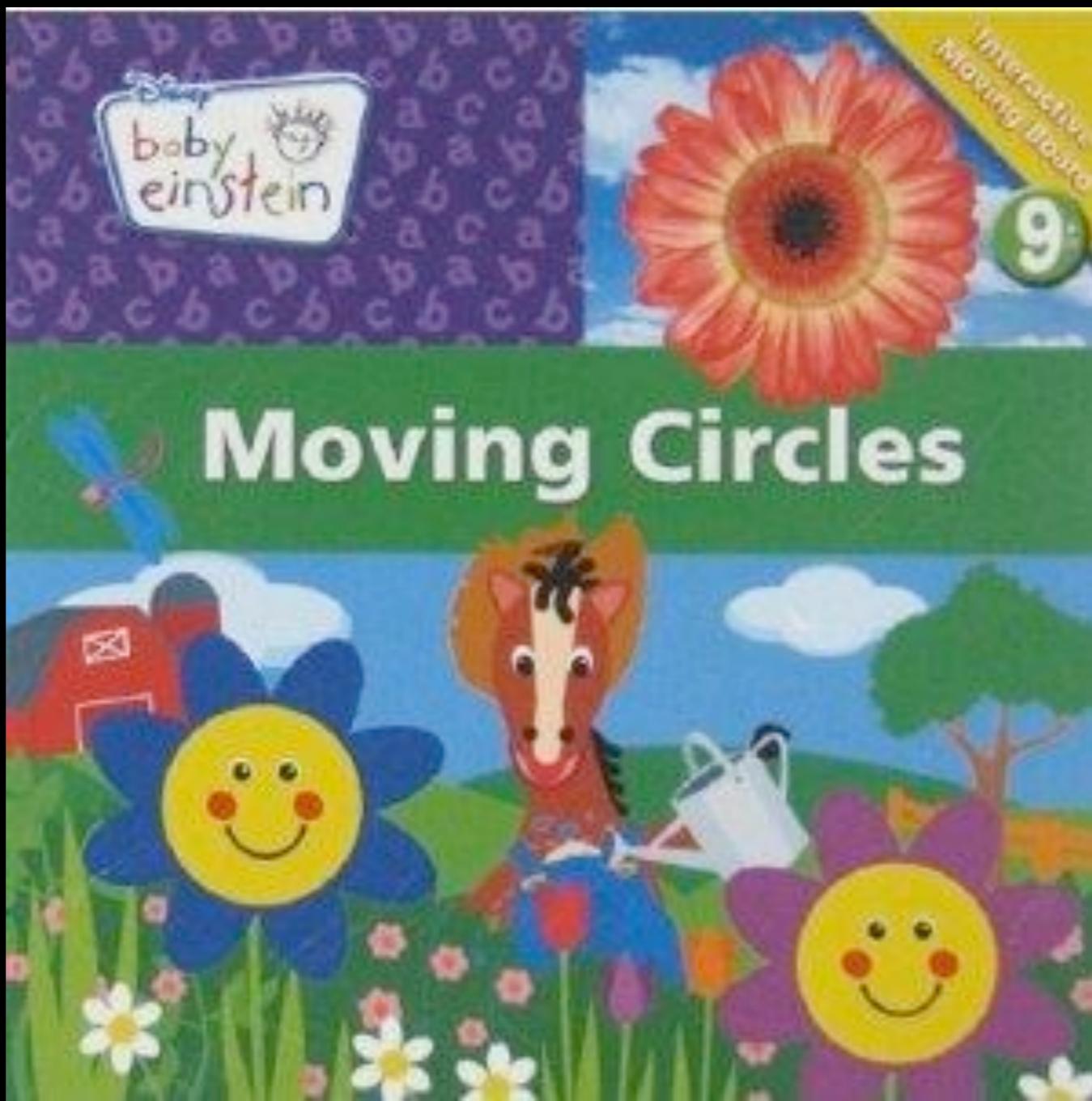








RICHARD SHILLIN
L·A·N·D·A·R·T



$$\textcircled{A} + \textcircled{B} = \textcircled{A}\textcircled{B}$$

$$C < A \\ B > A$$
$$\textcircled{A}\textcircled{B} + \textcircled{C} = \textcircled{A+C}\textcircled{B}$$

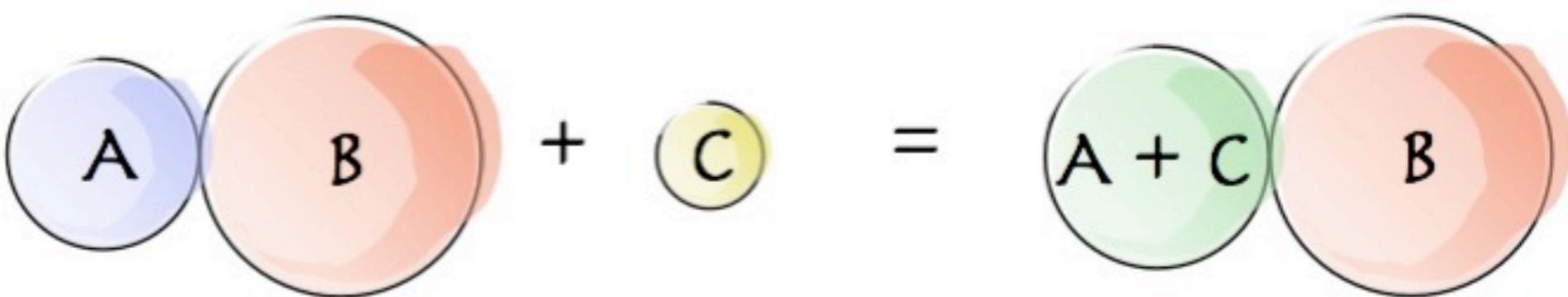
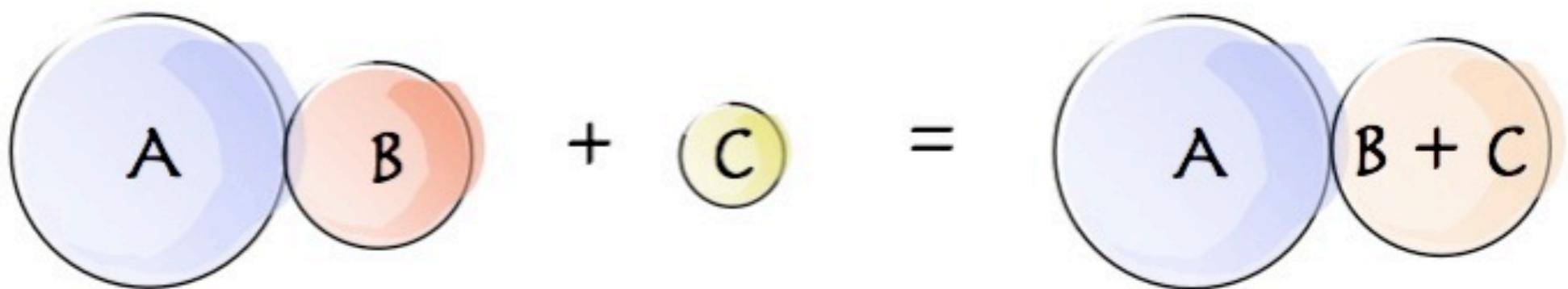
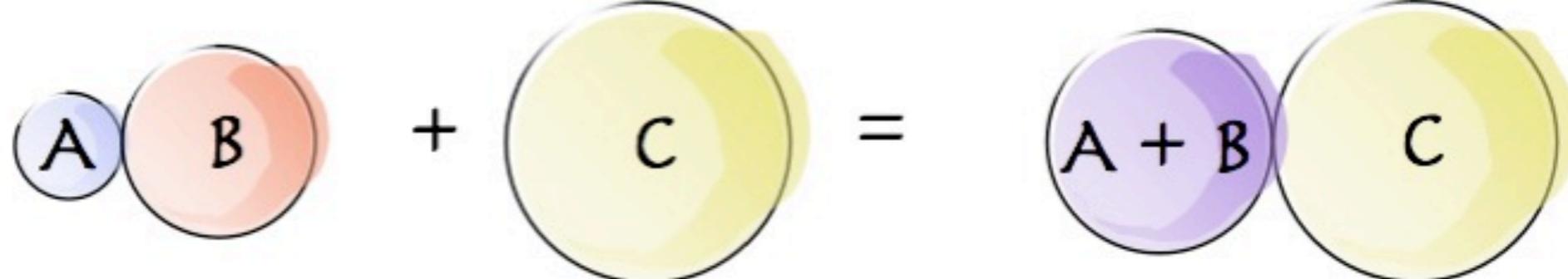
$$C < B \\ A > B$$
$$\textcircled{A}\textcircled{B} + \textcircled{C} = \textcircled{A}\textcircled{B+C}$$

$$C > A \\ C > B$$
$$\textcircled{A}\textcircled{B} + \textcircled{C} = \textcircled{A+B}\textcircled{C}$$

$$\textcircled{A} - \textcircled{B} = \textcircled{|B-A|}$$

$$B > A \\ C < A \\ C < B$$
$$\textcircled{A}\textcircled{B} - \textcircled{C} = \textcircled{A}\textcircled{B-C}$$

$$C > A \\ C > B$$
$$\textcircled{A}\textcircled{B} - \textcircled{C} = \textcircled{|C-|B-A|}$$

$A < B$  $C < A < B$  $C < B < A$  $C > A, C > B$ 

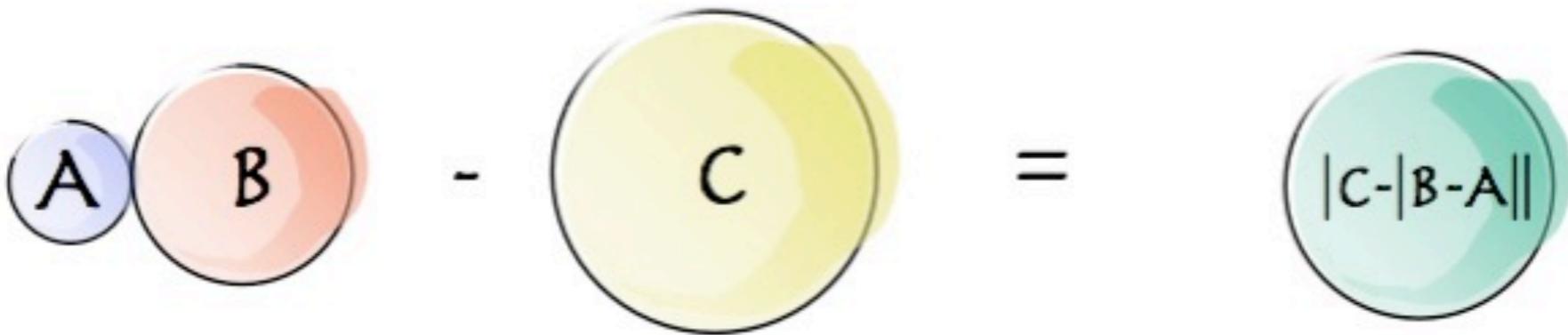
$$A < B$$



$$C < A < B$$



$$C > A, C > B$$



THE LANGUAGES



Prolog



Ruby

A Programmer's Best Friend

Haskell λ

A Purely Functional Language



Java™



Prolog



Ruby

A Programmer's Best Friend

Haskell λ

A Purely Functional Language



Java™

$$A < B \quad A + B = AB$$

The diagram illustrates the concept of function composition. It shows two overlapping circles labeled 'A' and 'B'. Circle 'A' is light blue and circle 'B' is light orange. They overlap, and their union is shaded in a darker shade of each color. This visual representation corresponds to the mathematical expression $A < B$ followed by the addition operator $+$, which is used here to denote function composition. The result of the composition is shown as a single circle containing both 'A' and 'B', with the combined shading.

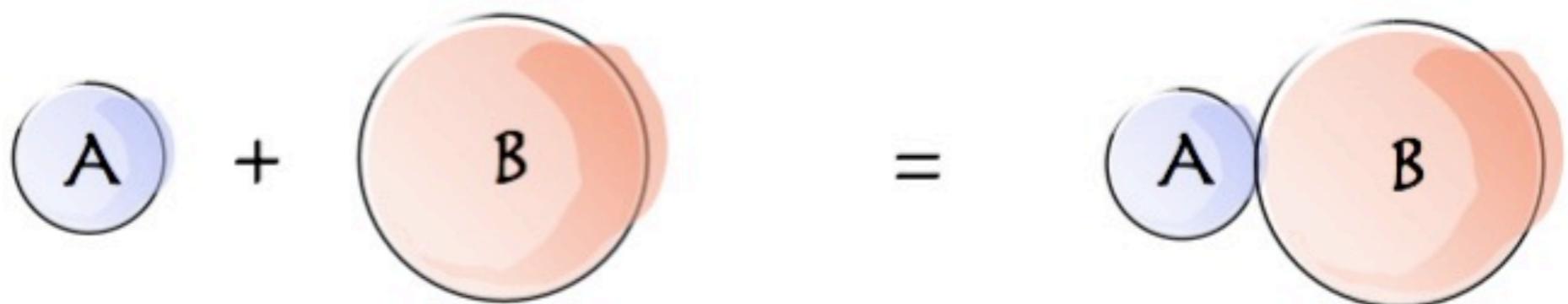
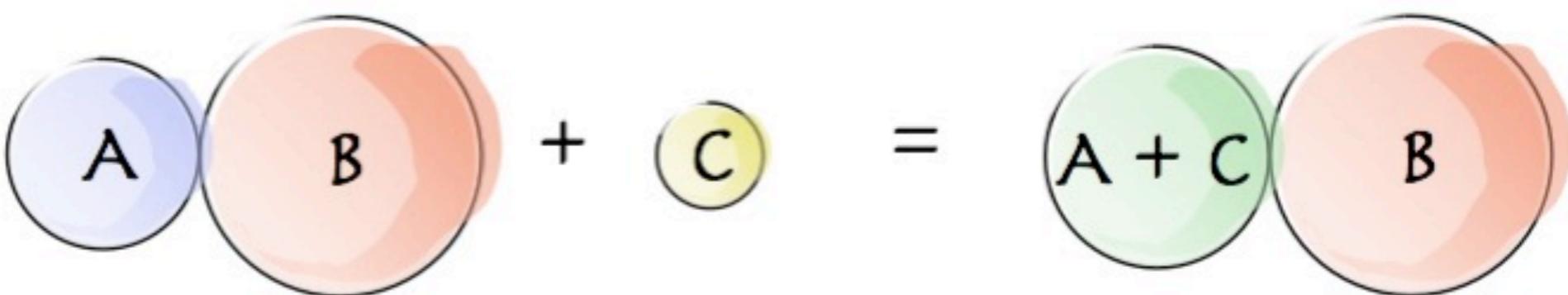
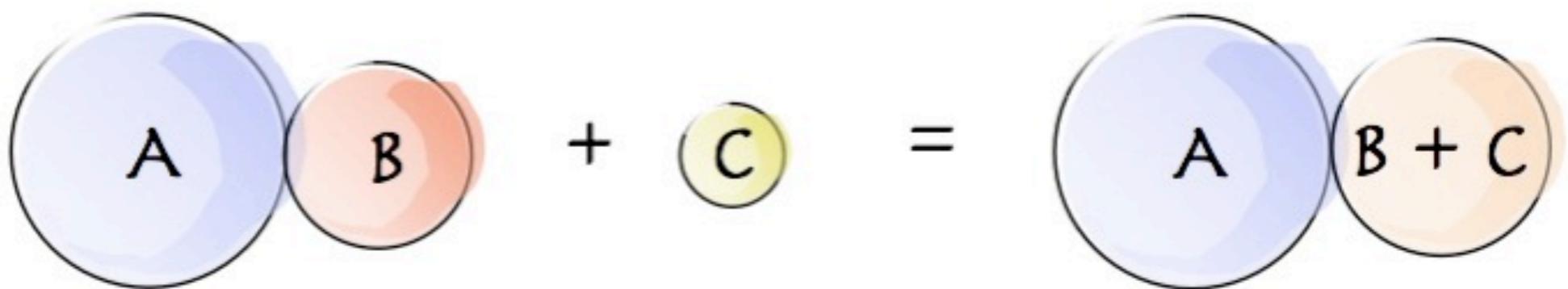
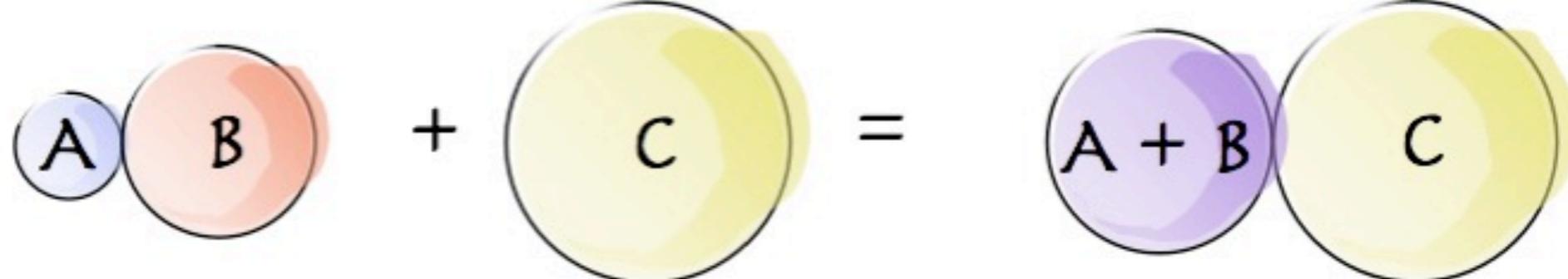
PROGRAMMING LANGUAGES

- JAVA
- SCALA
- RUBY
- PROLOG
- HASKELL



JAVA

SHOW ME THE CODE...

$A < B$  $C < A < B$  $C < B < A$  $C > A, C > B$ 

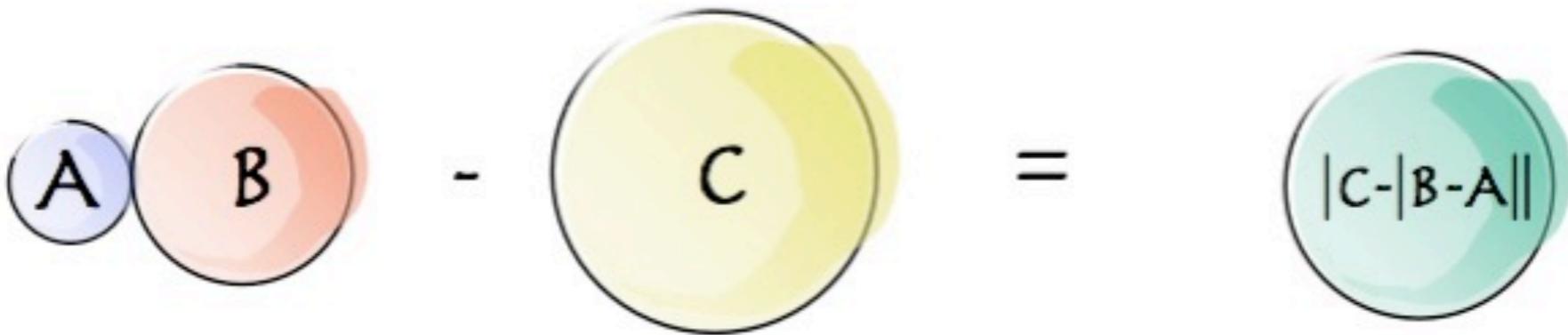
$$A < B$$



$$C < A < B$$



$$C > A, C > B$$



	Java	Scala	Ruby	Prolog	Haskell
Style	Imperative				
Type	Static (declare)				
Strong	Strong				
OO / F	Object oriented				
Class	Manually				

Scala
RESTAURANT

UPSTAIRS



SCALA

- TWITTER: RUBY -> SCALA
- AT LEAST A BRIDGE, MAYBE MORE (B.A. TATE)

SCALA OBJECTS

A comprehensive step-by-step guide

Scalability Functional Programming + Objects

Programming in

Scala

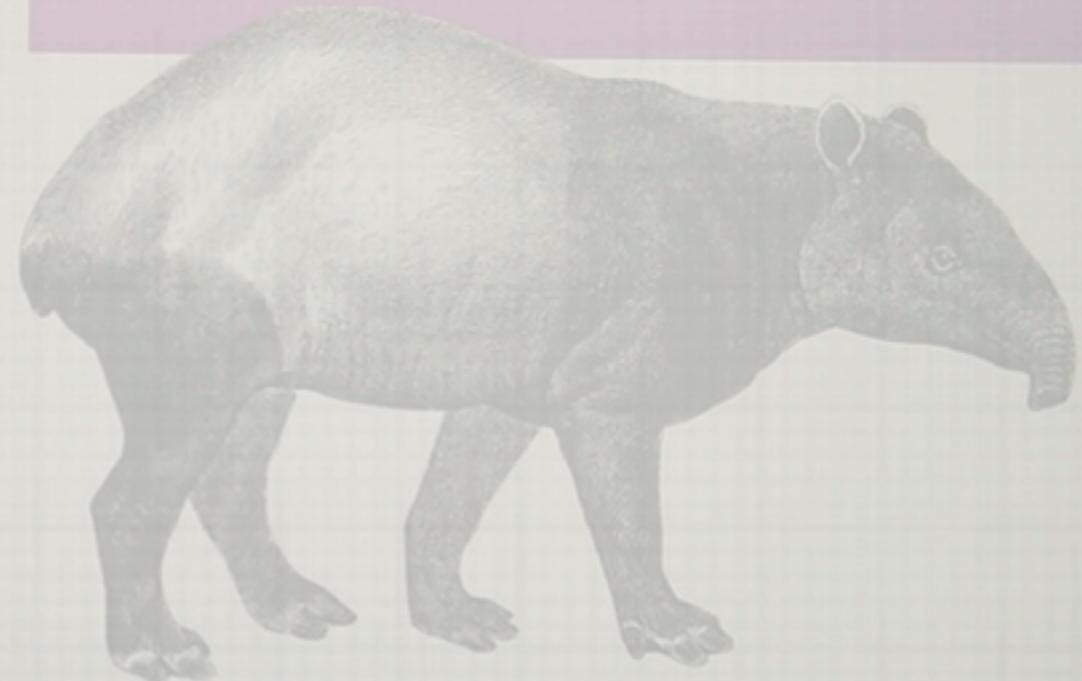
- EVERYTHING IS AN OBJECT
- SMALL EXCEPTIONS
- TYPE CHECKING
- SCALA COMPILE TIME
- (RUBY RUNTIME)

artima

Martin Odersky
Lex Spoon
Bill Venners

Programming

Scala



O'REILLY®

Dean Wampler & Alex Payne

A large, diverse crowd of people is shown from behind, with their hands raised in the air, suggesting they are at a concert or festival. The scene is slightly hazy, and a person wearing a red ladybug hat is visible in the top left corner.

IMMUTABLE SCALA

- PUBLIC BY DEFAULT
- VAR VS. VAL
- SCALA HAS TUPELS ("JOHN", 38)

METHODS

- CONSTRUCTOR = CODE AFTER CLASS DEF
- ALL METHOD DEFINITIONS:
 - PARAMETER NAMES
 - PARAMETER TYPES



COMPANION OBJECTS

- CLASS = INSTANCE
- OBJECT = SINGLETON
- THEY CAN SHARE THE SAME NAME!





THE FAILURE OF STATE by Uncle Bob

THE 8TH LIGHT = INFRARED = INVISIBLE
THE MOST POWERFUL LIGHT
LEAD WITH THE LITTLE FINGER
STATE
YOU'RE DOING IT WRONG!

WE'VE BEEN
WHERE IS OUR CRAFT HEADED?

WE WRITE MOST OF OUR PROGRAMS IN
A STATEFUL WAY. THE CODE IS
TIME DEPENDANT!

SICP!
IS THE NEXT Book
You SHOULD READ

ENTER
THE MODEL OF
ASSIGNMENT

TIME IS NOW
A FACTOR!

FUNCTIONAL PROGRAMMING

CIRCA 1957
REQUIRED MEMORY THAT WE DIDN'T HAVE

$$y = f(x)$$

REGARDLESS
OF TIME

assert Equals ($f(x)$, $f(x)$)?

WE SHOULD
CHANGE TO FUNCTIONAL
PROGRAMMING

THE TAUTOLOGY PRINCIPLE

MEMORY IS
CHEAP
WE HAVE LOTS!

WE ARE TRIWONAIRES!

PROCESSOR
SPEED
GROWTH
IS FLAT



FUNCTIONAL PROGRAMMING

THE FAILURE

by Uncle Bob

THE 8TH LIGHT = INFRARED = INVISIBLE

THE MOST
POWERFUL LIGHT
LEAD WITH THE
LITTLE FINGER

BUILT BY FUNCTIONS

YOU'RE DOING IT WRONG!

RETURNS ALWAYS

WHERE IS OUR CRAFT HEADED?

SAME INPUT - SAME OUTPUT

WE WRITE PROGRAMS IN
A STATEFUL WAY. THE CODE IS
TIME DEPENDANT!

NO CHANGING STATE

MEMORY IS

HIGHER-ORDER FUNCTIONS

WE HAVE LOTS!

WE ARE TRIWONAIRES!

SICP!

IS THE NEXT Book
YOU SHOULD READ

ENTER
THE MODEL OF
ASSIGNMENT

TIME

IS NOW

A FACTOR!

WE'VE BEEN

DEPENDING ON STATE

FUNCTIONAL PROGRAMMING

CIRCA 1950s
REQUIRED MEMORY THAT
WE DIDN'T HAVE

$$y = f(x)$$

REGARDLESS
OF TIME

assert Equals (f(x), f(x))?

THE TAUTOLOGY PRINCIPLE

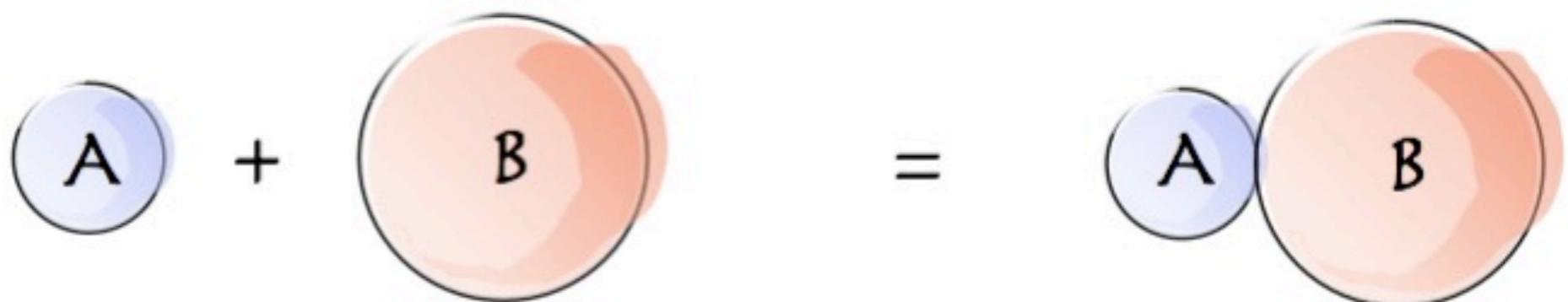
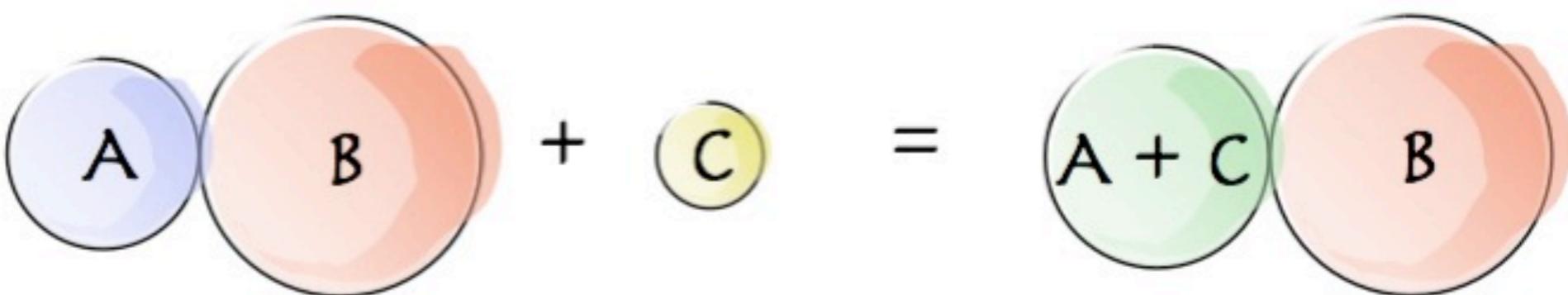
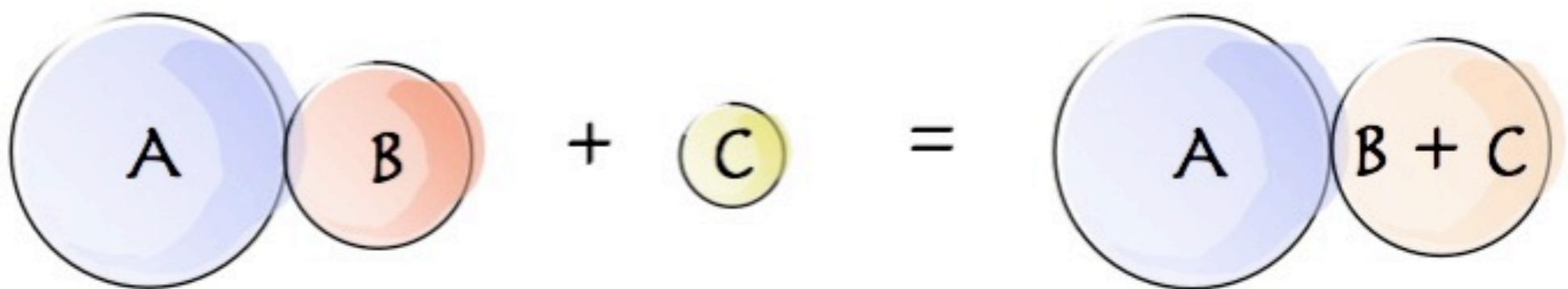
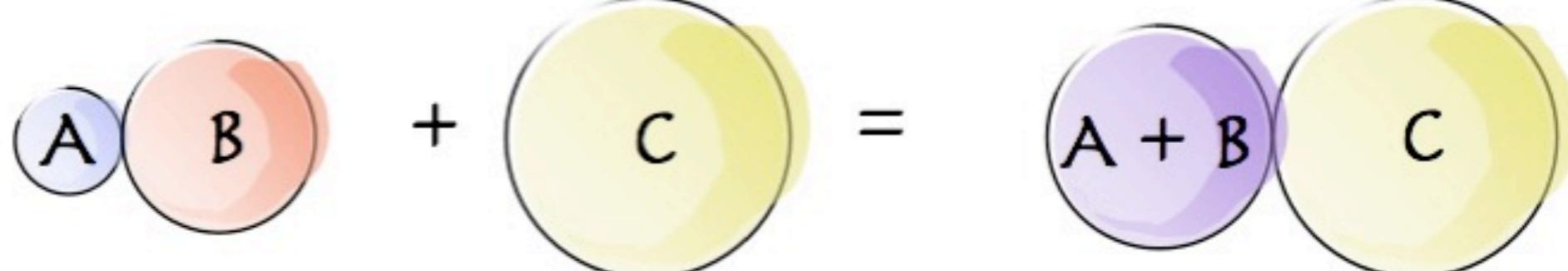
WE SHOULD

MOVE TO FUNCTIONAL
PROGRAMMING

PROCESSOR
SPEED
GROWTH
IS FLAT

SCALA

SHOW ME THE CODE...

$A < B$  $C < A < B$  $C < B < A$  $C > A, C > B$ 

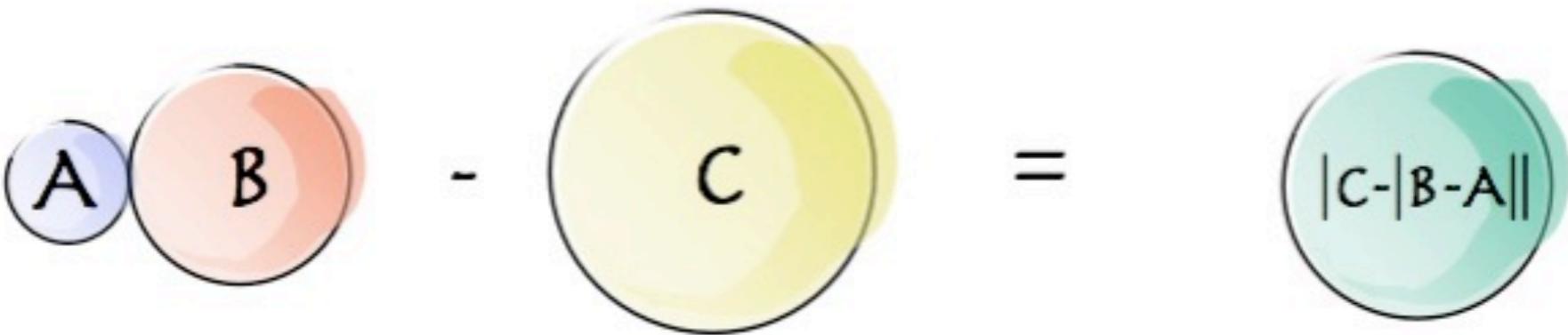
$$A < B$$



$$C < A < B$$



$$C > A, C > B$$



LINES OF CODE

- JAVA CODE: 155
- SCALA CODE: 72
- JAVA TEST CODE: 104
- SCALA TEST CODE: 61

	Java	Scala	Ruby	Prolog	Haskell
Style	Imperative	Imperative			
Type	Static (type declare)	Static (type inference)			
Strong	Strong	Strong			
OO / F	Object oriented	Object oriented / Functional			
Class	Manually	Class, Object			



RUBY

- INTERPRETED, NOT (ALWAYS) COMPILED
- CLEAN SYNTAX
- `order.calculate_tax unless order.nil?`

RUBY TYPES

- STRONGLY TYPED
- DYNAMIC TYPING
- RUNTIME FAILURE
- IF IT WALKS LIKE A DUCK AND QUACKS LIKE A DUCK:
IT'S A DUCK
- "WITH FREEDOM AND POWER
COME RESPONSIBILITY"

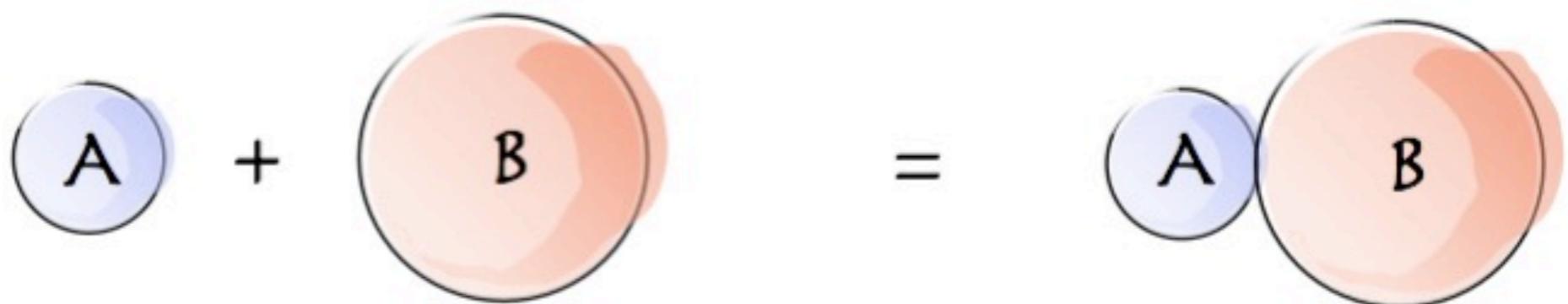
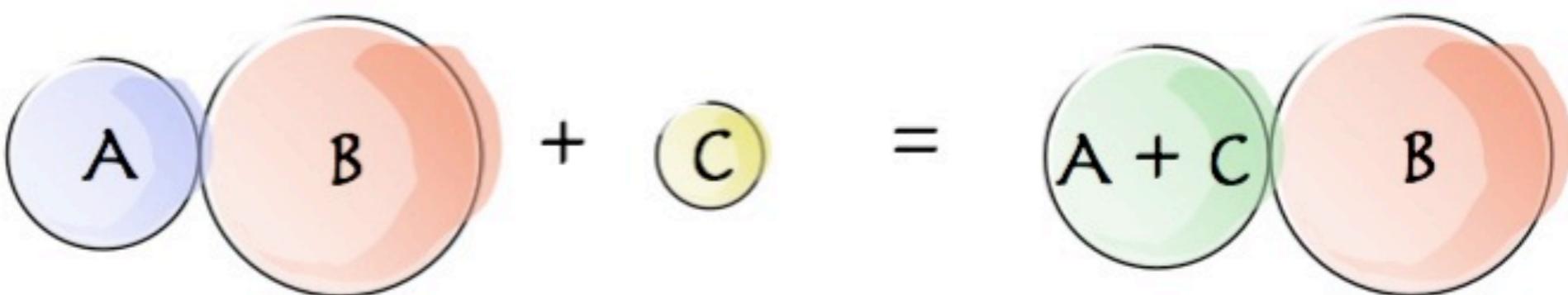
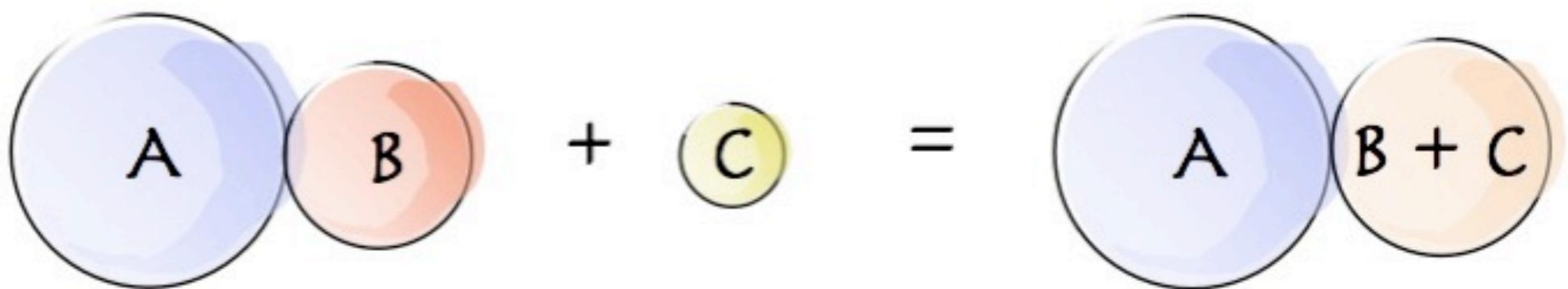
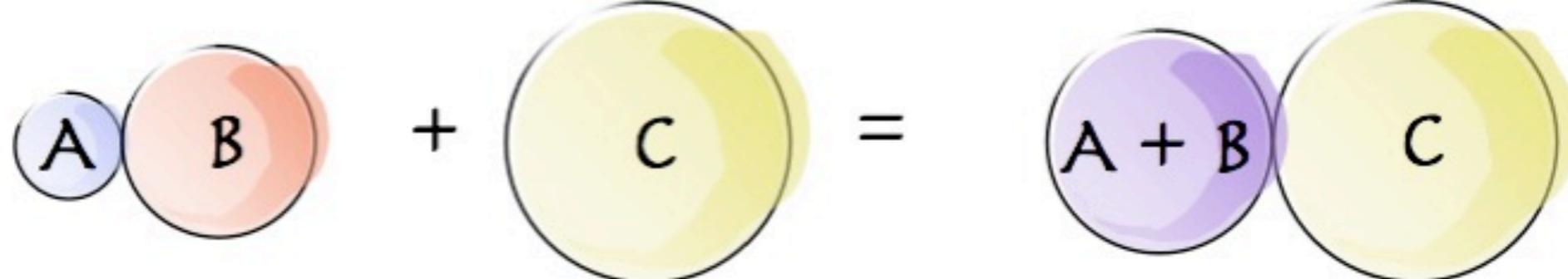
```
a = ['100', 100.0]
while i < 2
  puts a[i].to_i
  i = i + 1
end
100
100
```

TESTING RUBY

- TESTING WITH CUCUMBER
- FEATURE (SCENARIO'S)
- STEPS
 - GIVEN, WHEN, THEN
- DOMAIN

RUBY

SHOW ME THE CODE...

$A < B$  $C < A < B$  $C < B < A$  $C > A, C > B$ 

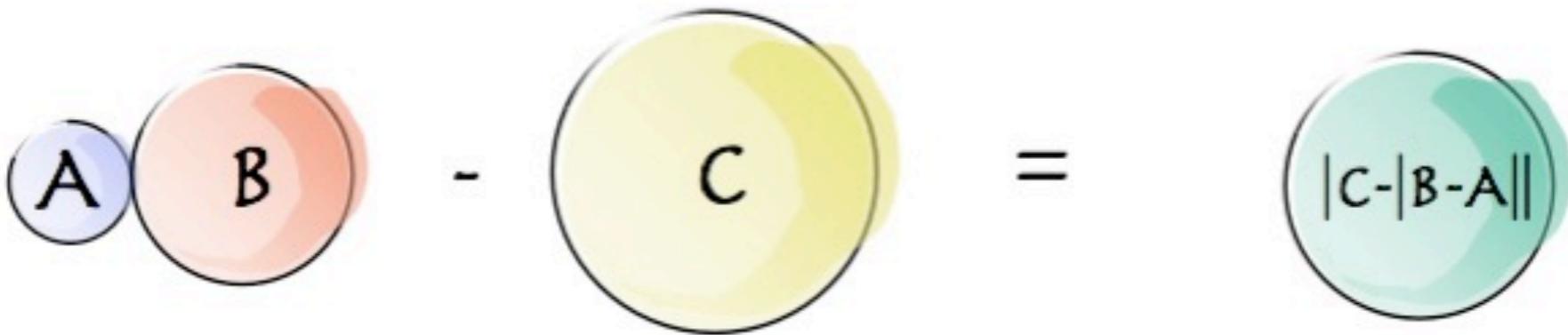
$$A < B$$



$$C < A < B$$



$$C > A, C > B$$



	Java	Scala	Ruby	Prolog	Haskell
Style	Imperative	Imperative	Imperative		
Type	Static (declare)	Static (type inference)	Dynamic		
Strong	Strong	Strong	Strong		
OO / F	Object oriented	Object oriented / Functional	Object oriented		
Class	Manually	Class, object, case object	Open can change		

Prolog

R A I D | PROLOG ZÜRICH

Willkommen

- LOGIC PROGRAMMING LANGUAGE
- DECLARATIVE LANGUAGE
- SAY WHAT, NOT HOW
- JUST DESCRIBE THE PROBLEM

PROLOG

- FACTS
- RULES
- QUERY

PROLOG

PROLOG

□ IT'S ALL ABOUT ... = UNIFICATION

PROLOG

- IT'S ALL ABOUT ... = UNIFICATION
- $(A, 2, C) = (1, B, 3)$

PROLOG

- IT'S ALL ABOUT ... = UNIFICATION
- $(A, 2, C) = (1, B, 3)$
- $A = 1$
 $B = 2$
 $C = 3$

PROLOG

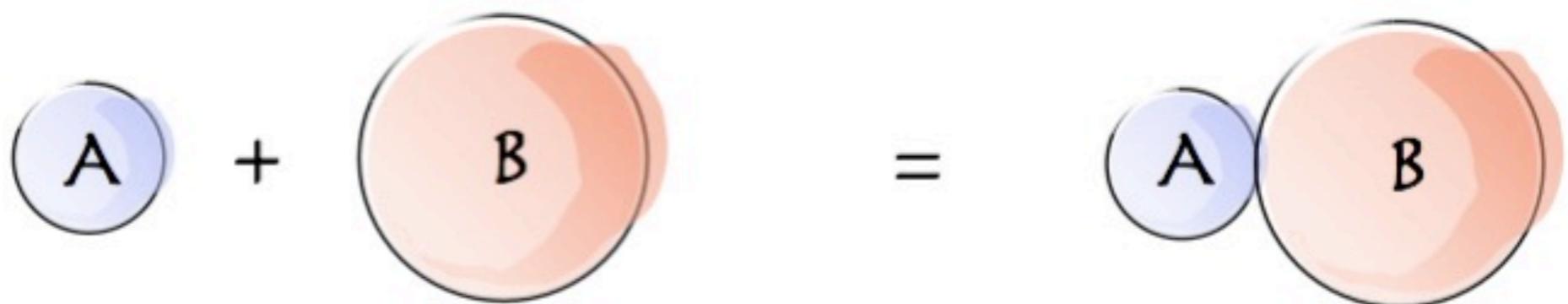
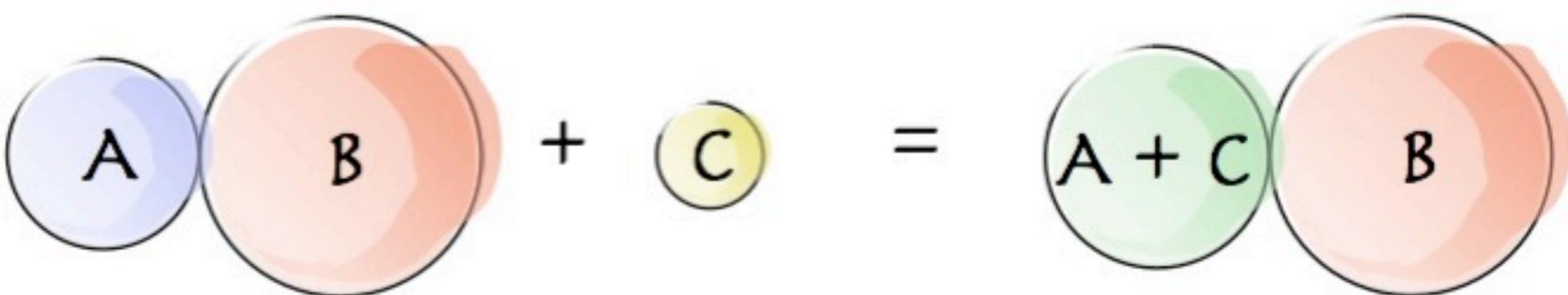
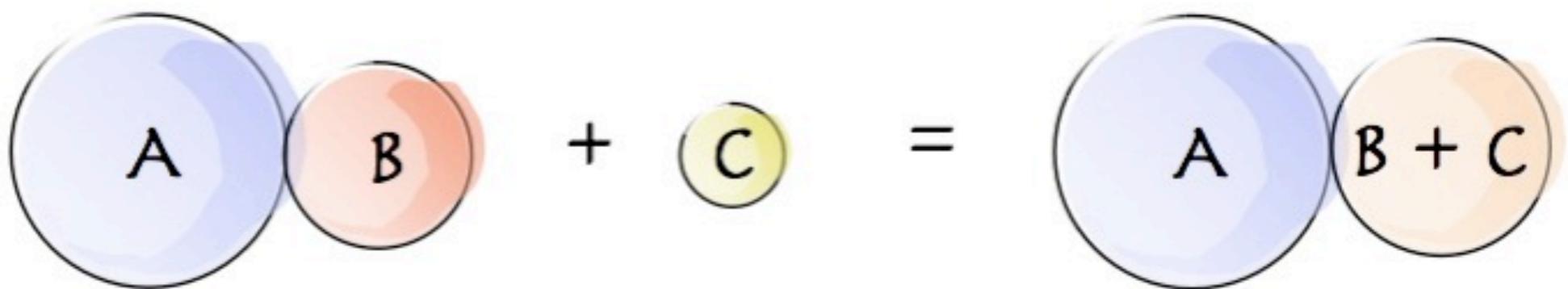
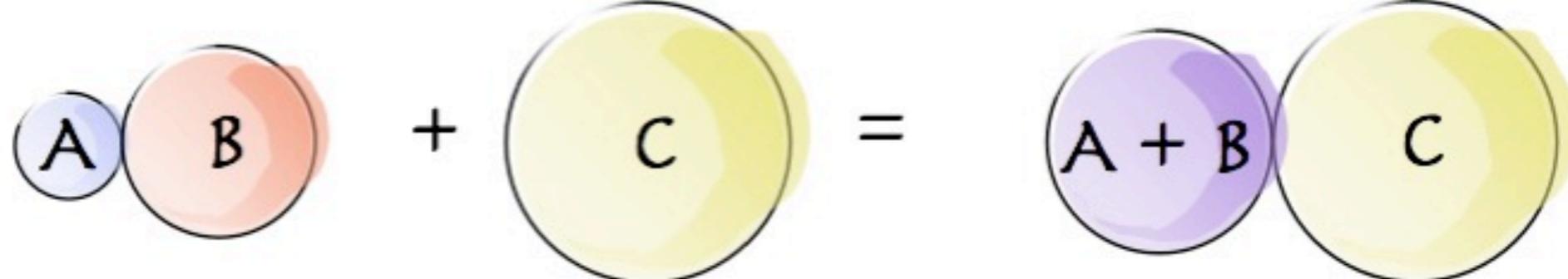
- IT'S ALL ABOUT ... = UNIFICATION
- $(A, 2, C) = (1, B, 3)$
 - $A = 1$
 - $B = 2$
 - $C = 3$
- $x = x + 1$

PROLOG

- IT'S ALL ABOUT ... = UNIFICATION
- $(A, 2, C) = (1, B, 3)$
 - $A = 1$
 - $B = 2$
 - $C = 3$
- $x = x + 1$
- NO.

PROLOG

SHOW ME THE CODE...

$A < B$  $C < A < B$  $C < B < A$  $C > A, C > B$ 

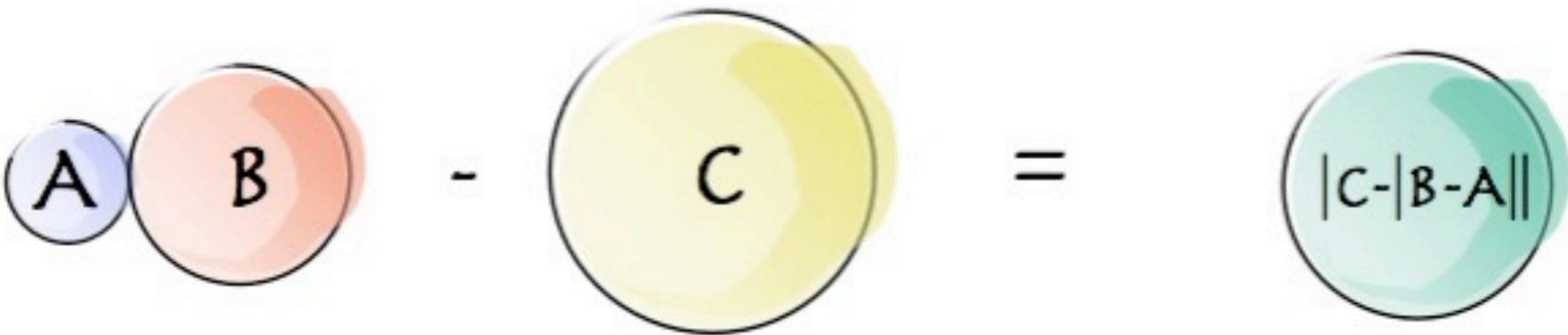
$$A < B$$



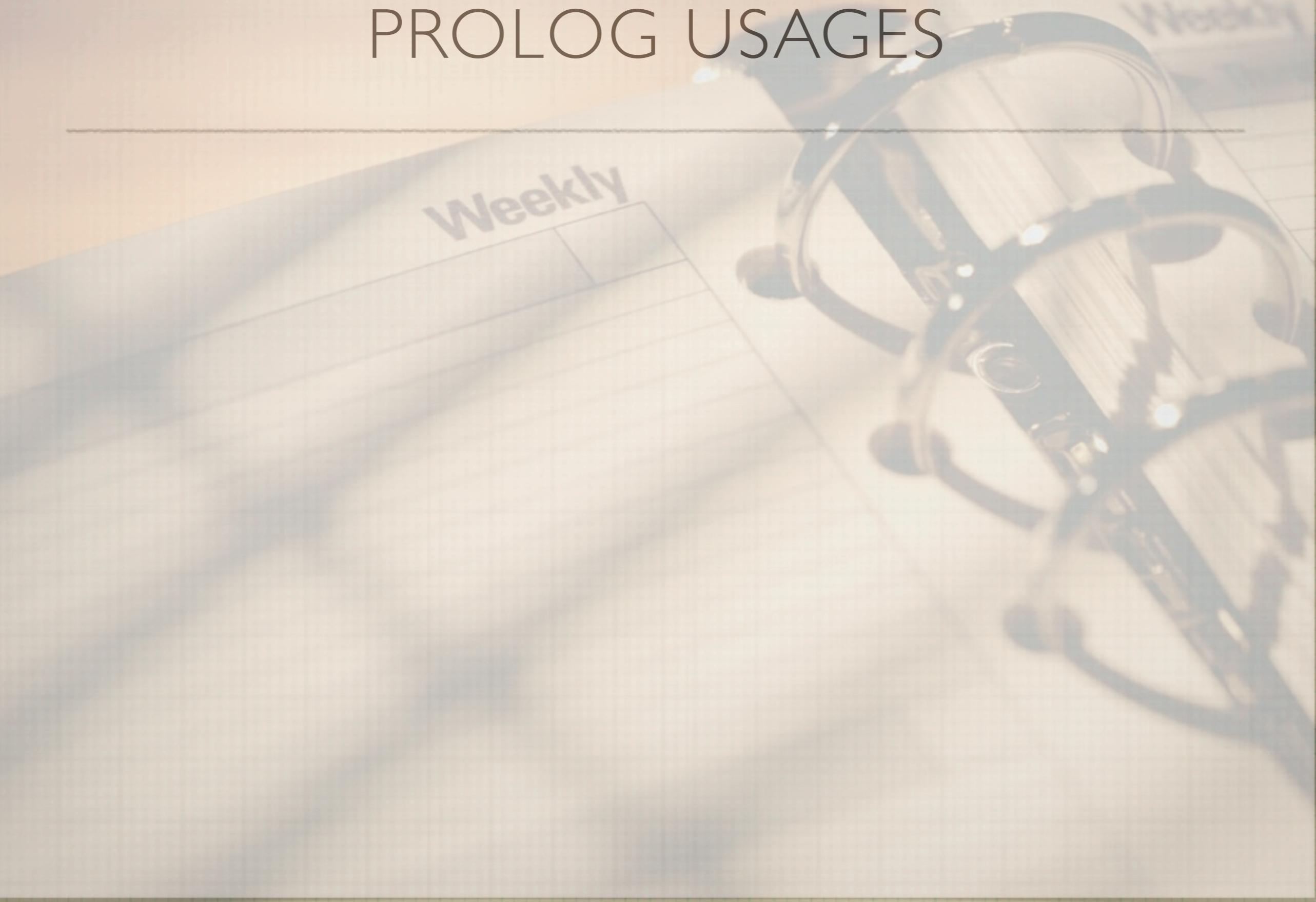
$$C < A < B$$



$$C > A, C > B$$



PROLOG USAGES



PROLOG USAGES

SCHEDULES

PROLOG USAGES

- SCHEDULES
- ARTIFICIAL INTELLIGENCE

PROLOG USAGES

- SCHEDULES
- ARTIFICIAL INTELLIGENCE
- GAMES
- 4-COLOUR MAP
- SUDOKU
- 8-QUEENS PUZZLE

	Java	Scala	Ruby	Prolog	Haskell
Style	Imperative	Imperative	Imperative	Declarative	
Type	Static (declare)	Static (type inference)	Dynamic	Unification	
Strong	Strong	Strong	Strong		
OO / F	Object oriented	Object oriented / Functional	Object oriented	Logic	
Class	Manually	Class, object, case object	Open can change		



Haskell

$$\text{smiley}^{-1} = \text{smiley}$$

$$\text{smiley}^2 = \text{smiley}$$

$$\text{smiley}^3 = \text{dice}$$

$$\text{Re}(\text{smiley}) = \text{smiley}$$
 No i's

$$\text{Im}(\text{smiley}) = \dots$$

$$\nabla \times (\text{smiley}) = \text{spiral smiley}$$

PURELY FUNCTIONAL

$\text{sun}(\text{smiley}) = \text{eggs}$

$$\nabla (\text{smiley}) = \text{grad smiley}$$

$$\partial (\text{smiley}) = \text{smiley}$$

$$\log (\text{smiley}) = \text{hand}$$

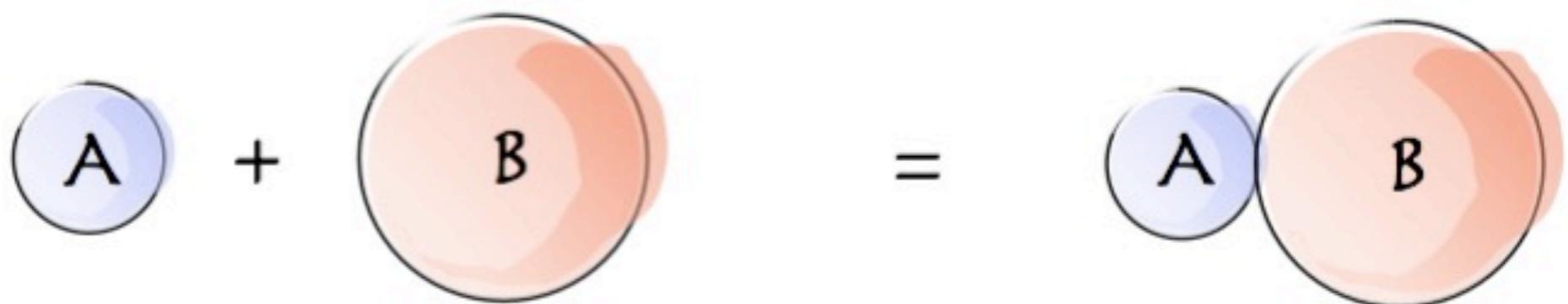
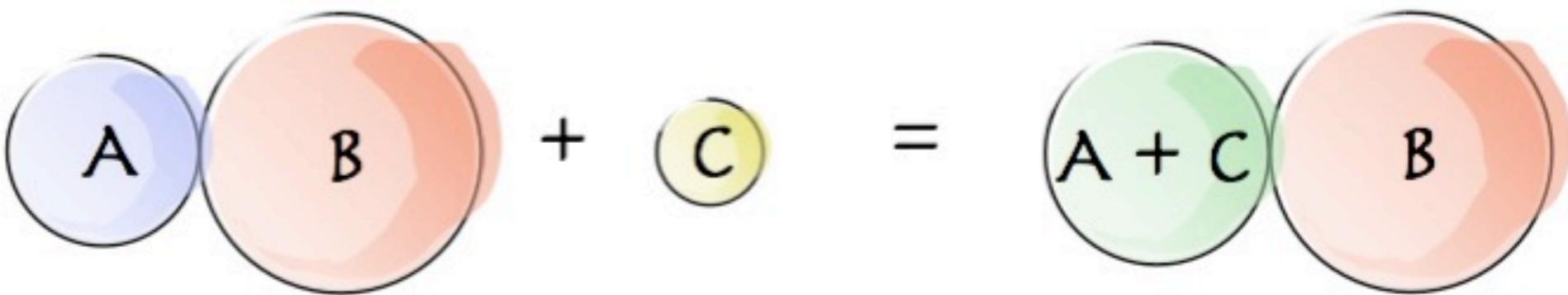
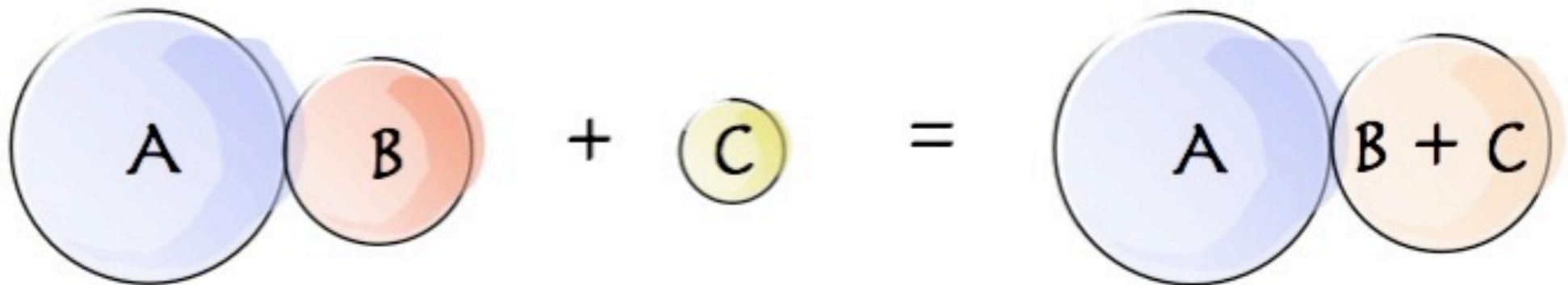
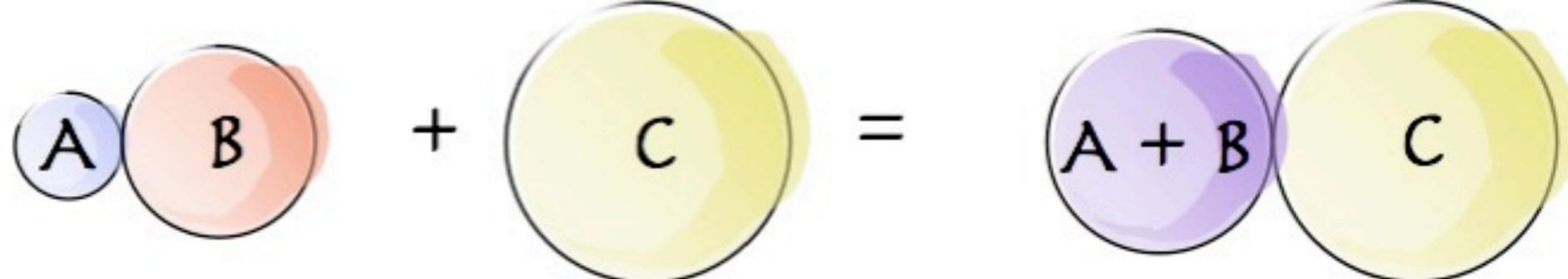
$$\sin(\text{smiley}) = \text{wavy smiley}$$



Happy Face Math by Charlie Smith

HASKELL

SHOW ME THE CODE...

$A < B$  $C < A < B$  $C < B < A$  $C > A, C > B$ 

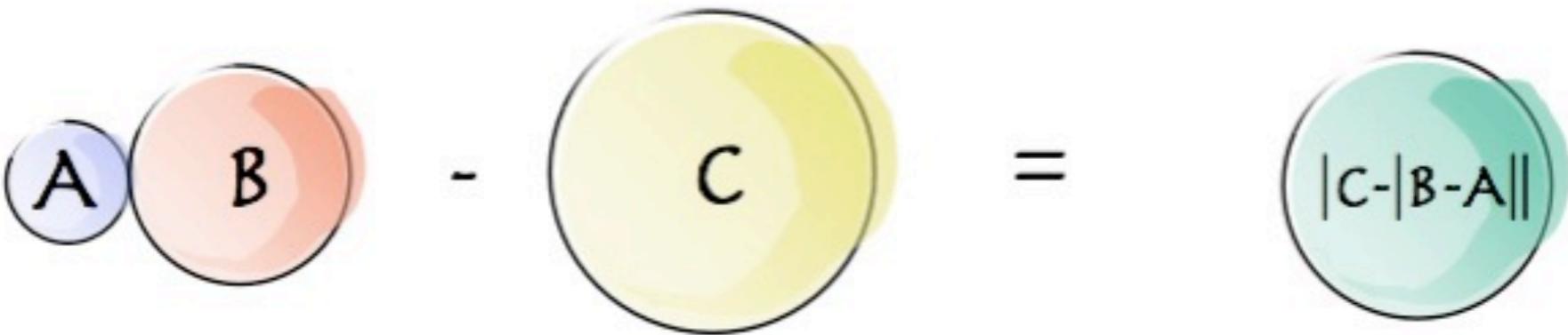
$$A < B$$



$$C < A < B$$



$$C > A, C > B$$

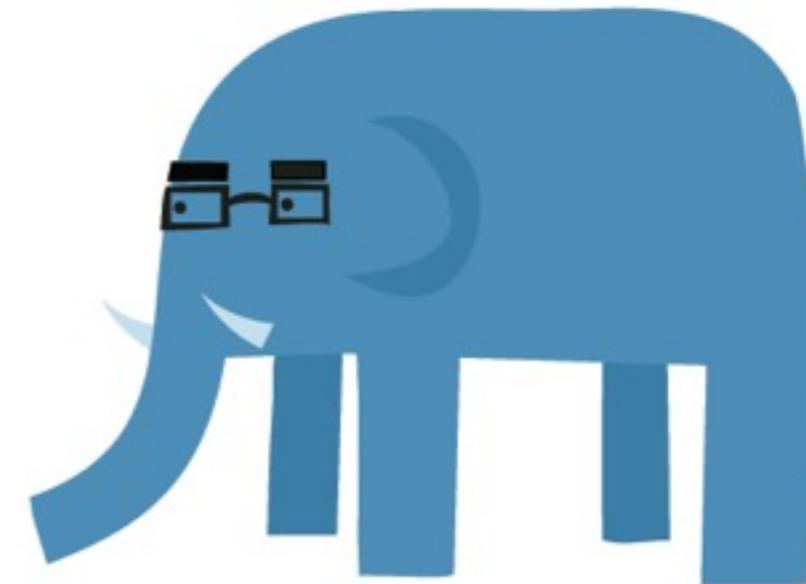


DESERVES A DEEPER
INVESTIGATION...



Learn You a Haskell for Great Good!

A Beginner's Guide



Miran Lipovača



	Java	Scala	Ruby	Prolog	Haskell
Style	Imperative	Imperative	Imperative	Declarative	Imperative
Type	Static (declare)	Static (type inference)	Dynamic	Unification	Static
Strong	Strong	Strong	Strong		Strong
OO / F	Object oriented	Object oriented / Functional	Object oriented	Logic	Functional
Class	Manually	Class, object, case object	Open can change		Monad (?)

NICE TO READ

The
Pragmatic
Programmers

Seven Languages in Seven Weeks

A Pragmatic
Guide to
Learning
Programming
Languages

Bruce A. Tate

Edited by Jacquelyn Carter

