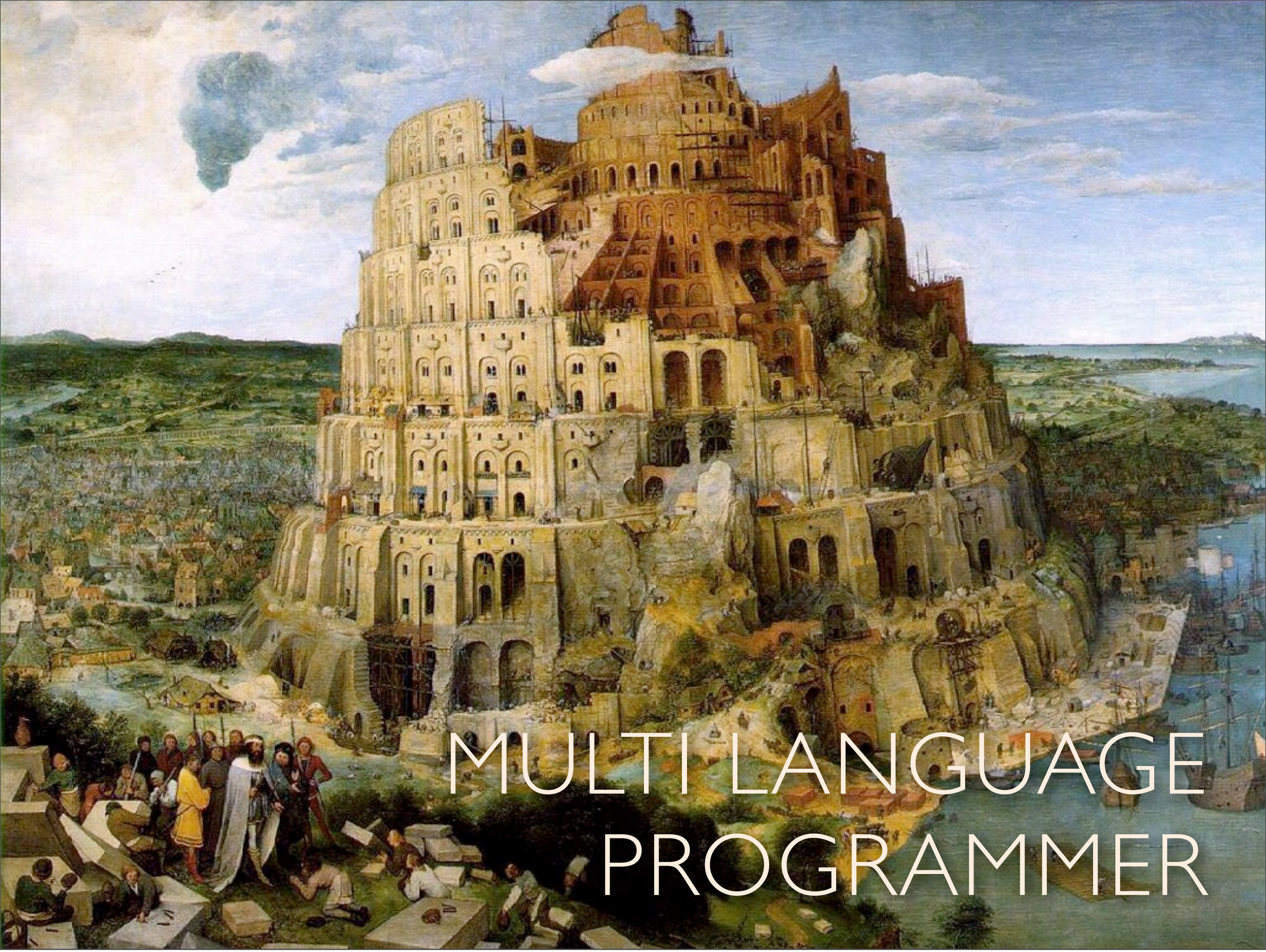


MULTI LANGUAGE PROGRAMMER

I PROBLEM

5 LANGUAGES

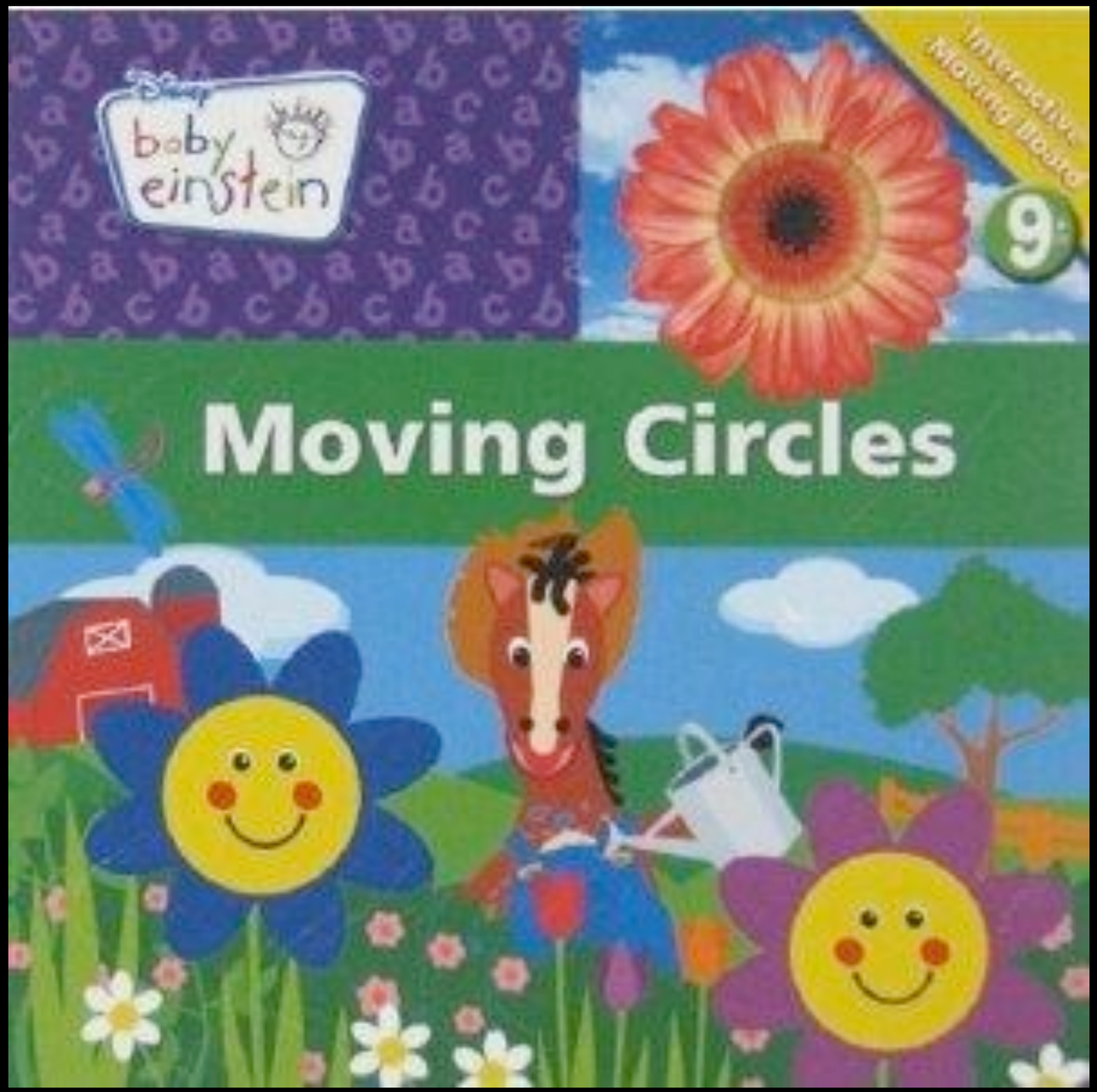
N + 1 SOLUTIONS

THE PROBLEM IS ABOUT...

vrijdag 25 november 11


ice
watch
ice
watch
ice
watch
ice
watch
ice
watch
ice
watch
ice
watch
ice
watch


RICHARD SHILLIN
L·A·N·D·A·R·T


Disney
baby einstein
Interactive Moving Books
9
Moving Circles

$A + B = AB$

$C < A$
$B > A$

$AB + C = (A+C)\,B$

$C < B$
$A > B$

$A\,(B+C) = AB + AC$

$C > A$
$C > B$

$AB + C = (A+B)\,C$

$A - B = |B-A|$

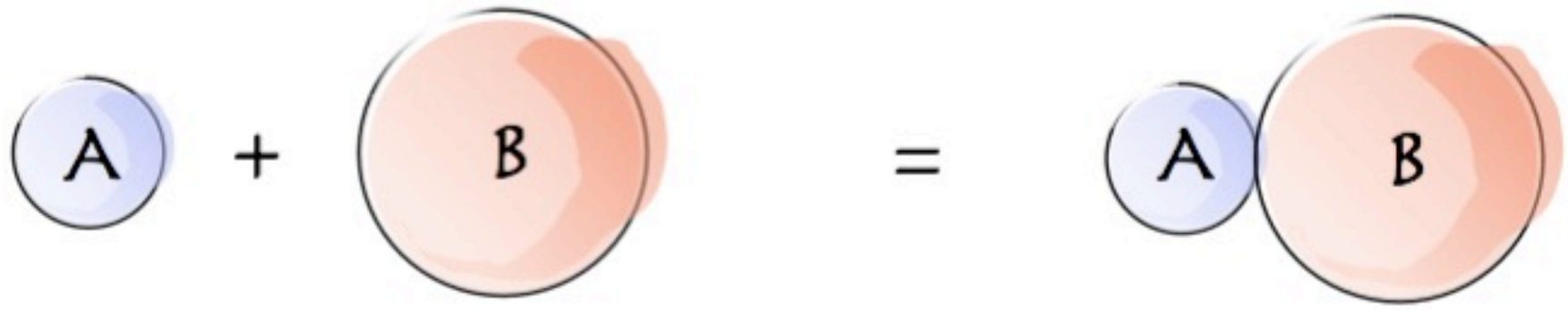
$B > A$
$C < A$
$C < B$

$AB - C = A\,(B-C)$

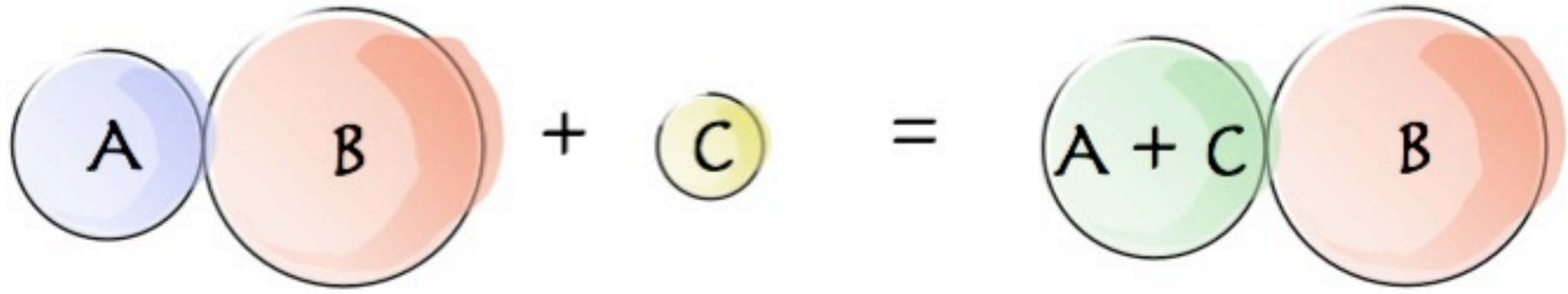
$C > A$
$C > B$

$AB - C = |C-|B-A||$

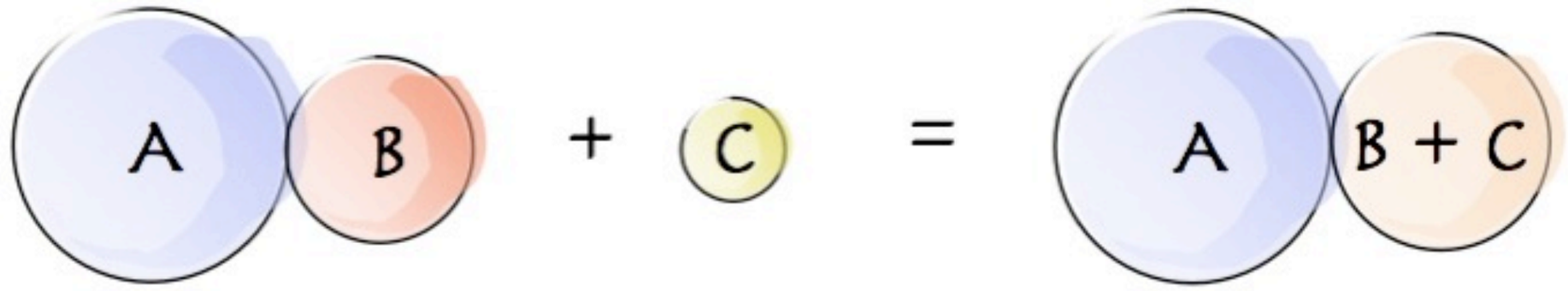
A < B



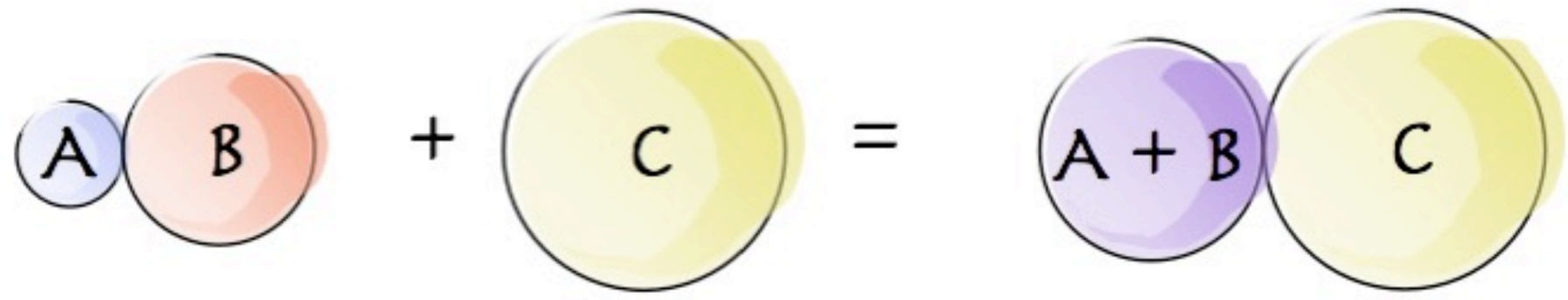
C < A < B



C < B < A



C > A, C > B



$A < B$

$C < A < B$

$C > A,\ C > B$

THE LANGUAGES


Scala


Ruby
A Programmer's Best Friend


Prolog


Haskell
A Purely Functional Language


Java

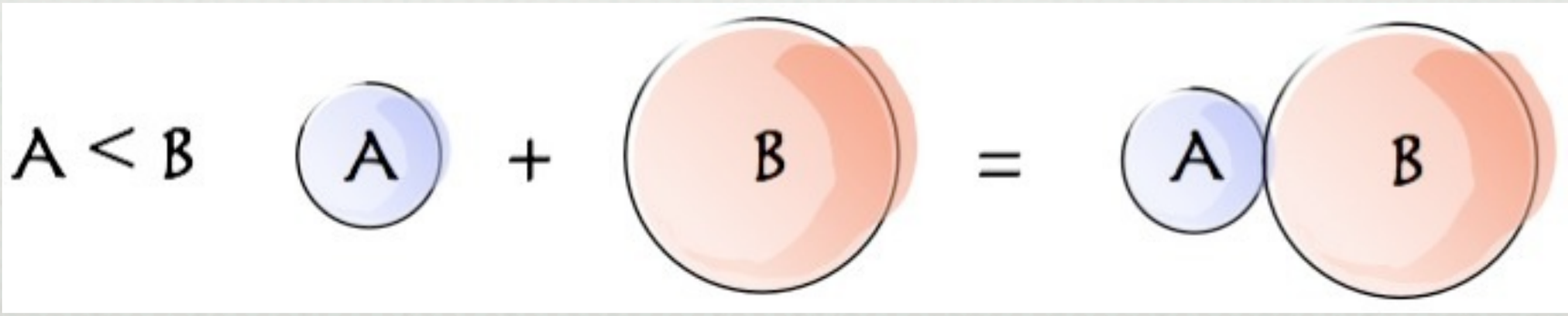

Scala


Ruby
A Programmer's Best Friend


Prolog


Haskell
A Purely Functional Language
λ


Java


A < B
A
+
B
=
A
B

PROGRAMMING LANGUAGES

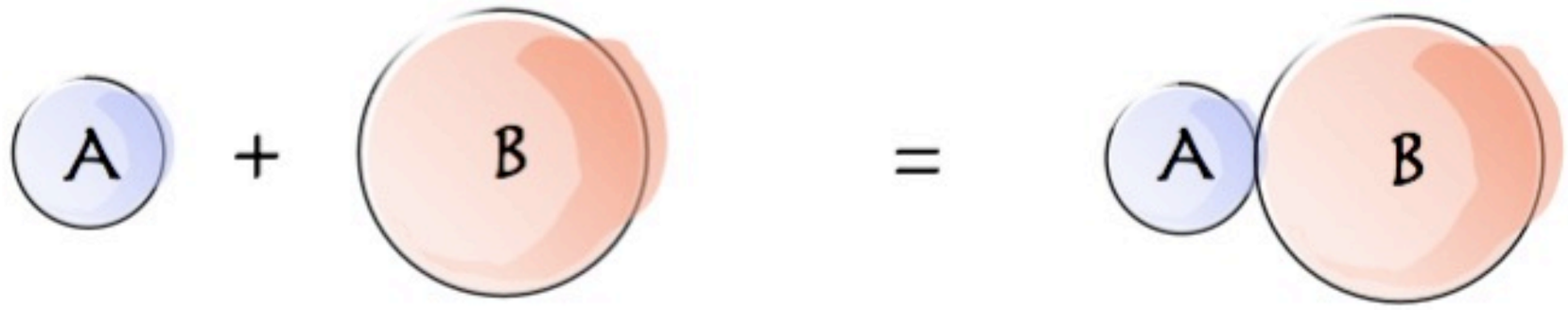
- [ ] JAVA
- [ ] SCALA
- [ ] RUBY
- [ ] PROLOG
- [ ] HASKELL


Java™

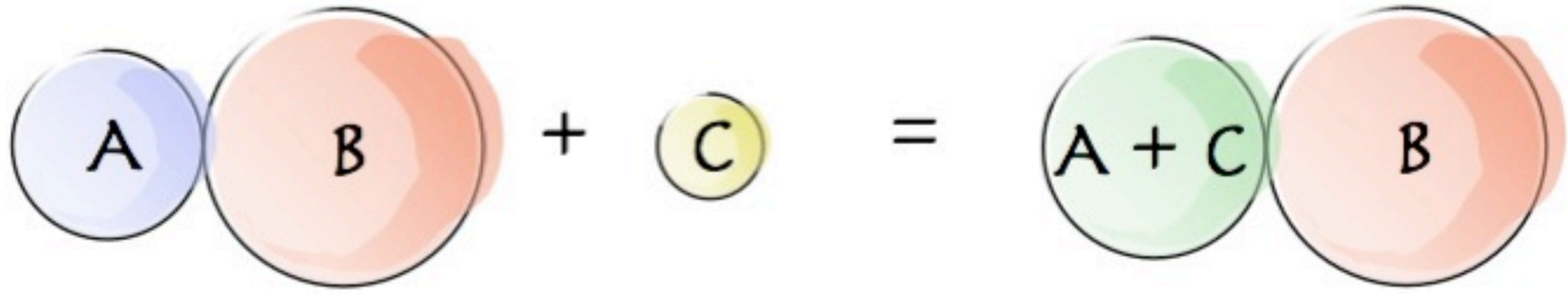
JAVA

- [ ] SHOW ME THE CODE...

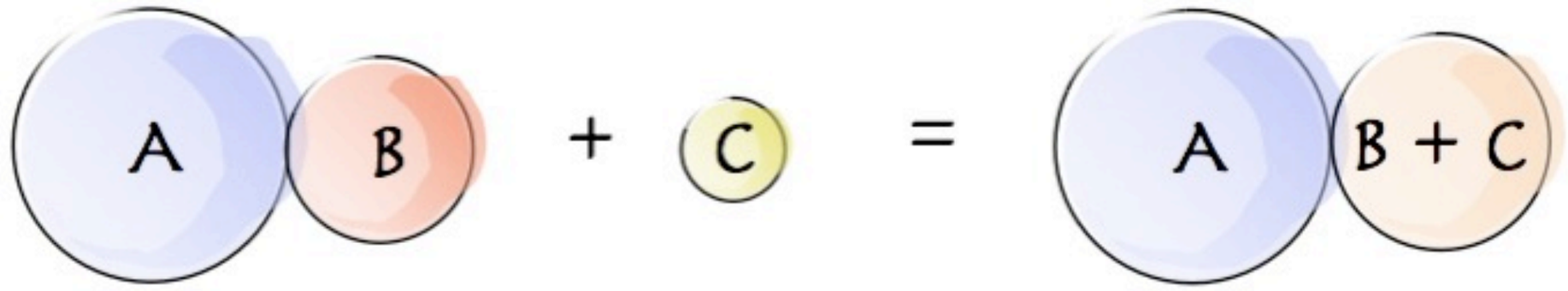
A < B



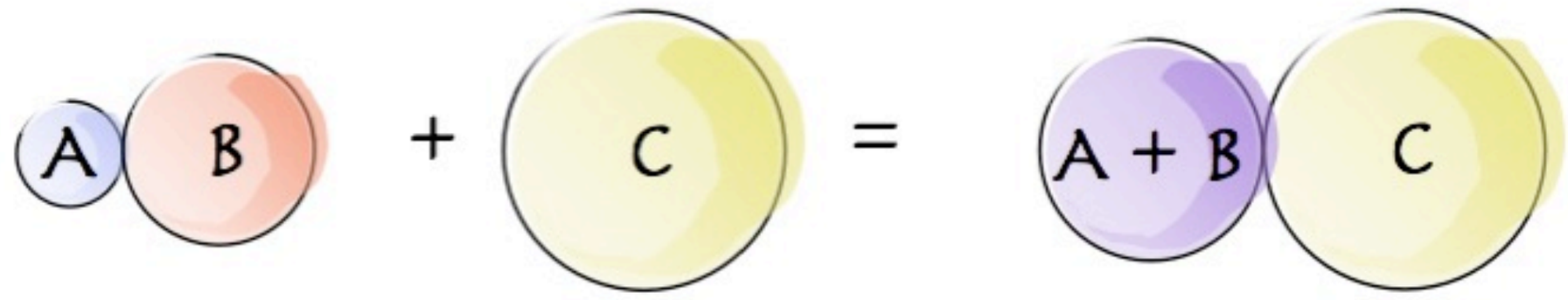
C < A < B



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**				


UPSTAIRS
Scala
RESTAURANT

SCALA

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

SCALA OBJECTS

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

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!

Software CRAFTSMANSHIP north america

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!

WHERE IS OUR CRAFT HEADED?

WE'VE BEEN DEPENDING ON STATE

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

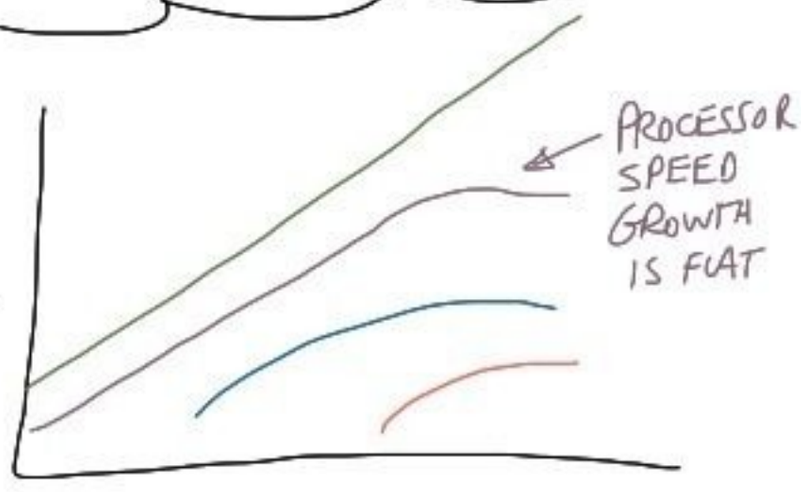
assertEquals(f(x), f(x))?

THE TAUTOLOGY PRINCIPLE

MEMORY IS CHEAP WE HAVE LOTS!

WE ARE TRILLIONAIRES!

WE SHOULD CHANGE TO FUNCTIONAL PROGRAMMING



Software CRAFTSMANSHIP north america

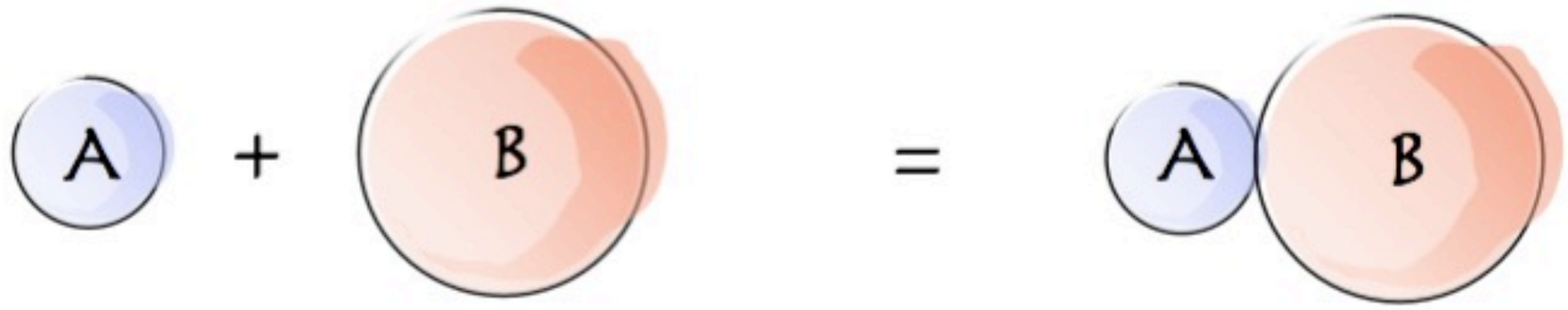
FUNCTIONAL PROGRAMMING

- ☐ BUILT BY FUNCTIONS
- ☐ RETURNS ALWAYS
- ☐ SAME INPUT - SAME OUTPUT
- ☐ NO CHANGING STATE
- ☐ HIGHER-ORDER FUNCTIONS

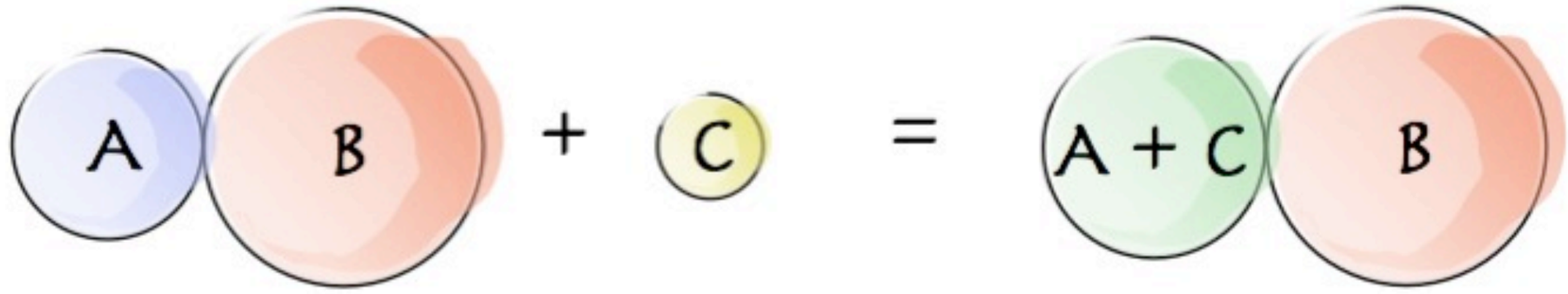
SCALA

- [ ] SHOW ME THE CODE...

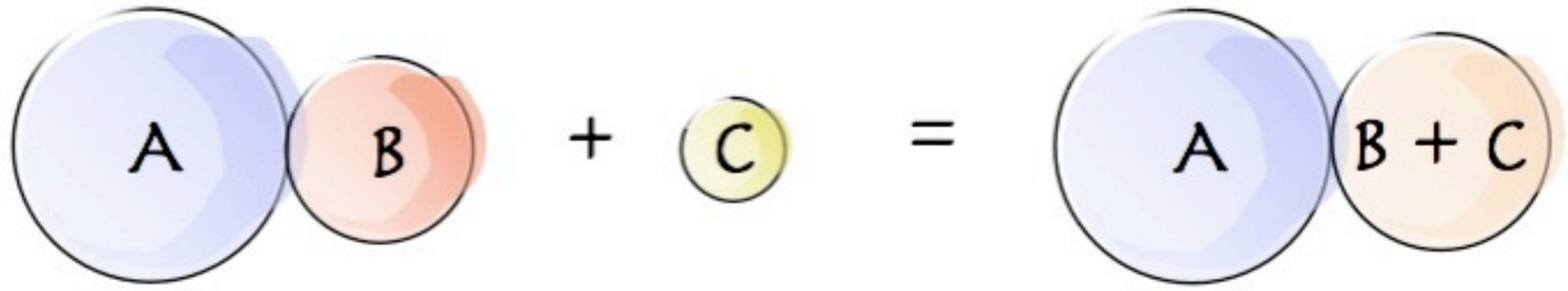
A < B



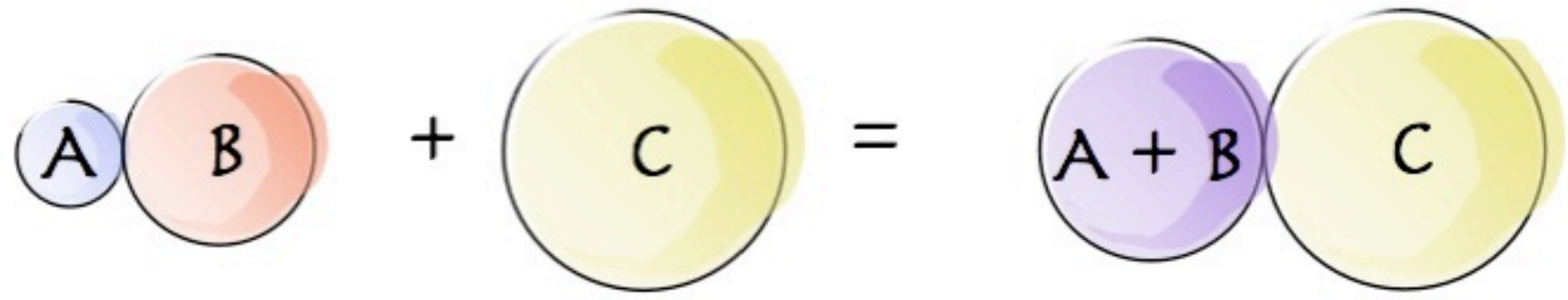
C < A < B



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
```

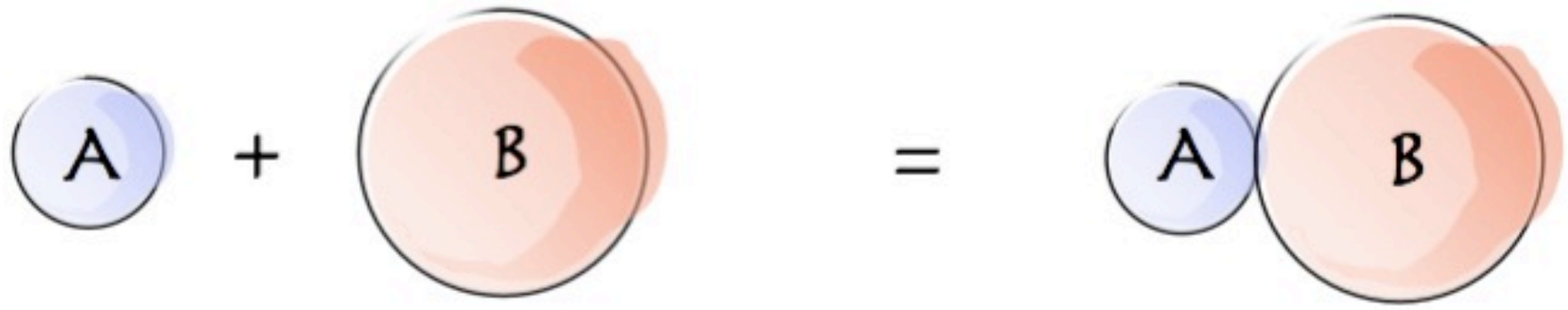
TESTING RUBY

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

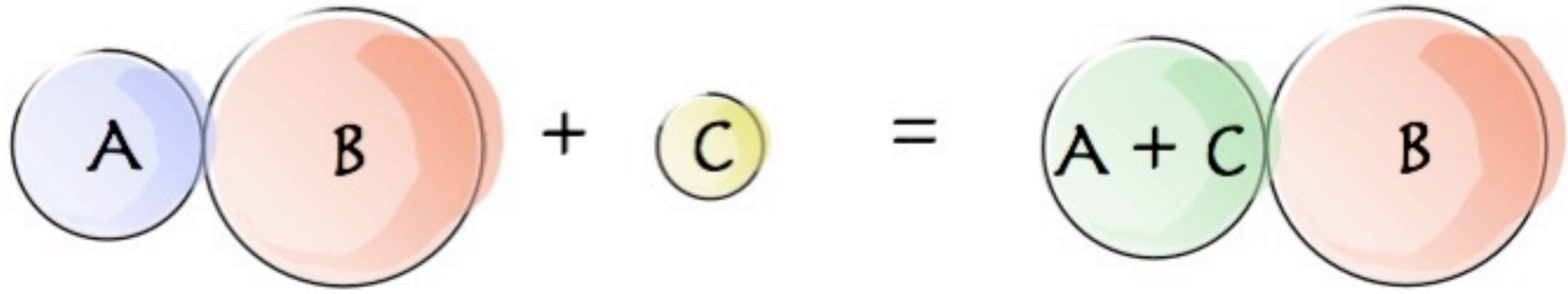
RUBY

- [ ] SHOW ME THE CODE...

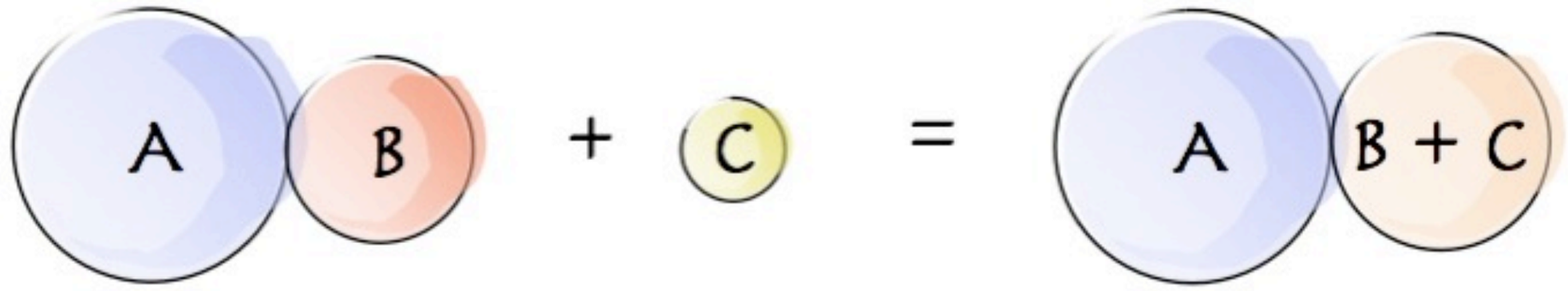
A < B



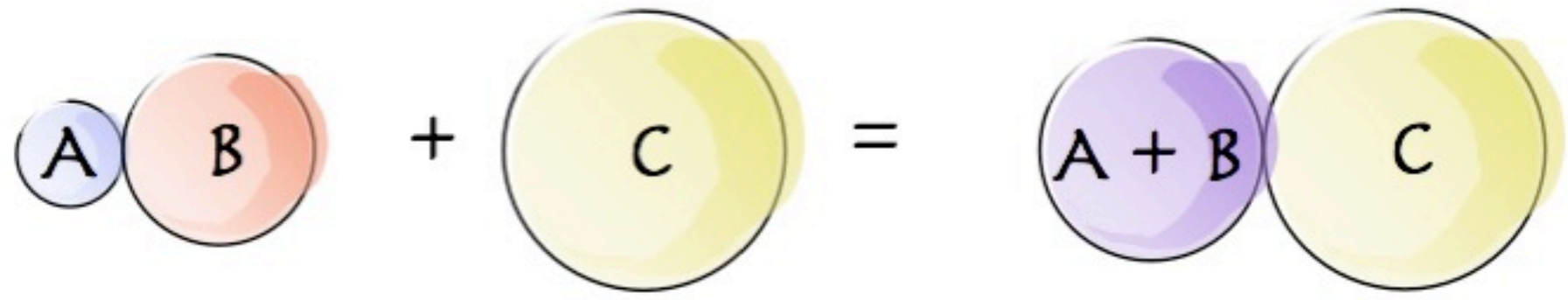
C < A < B



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

RAID | 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

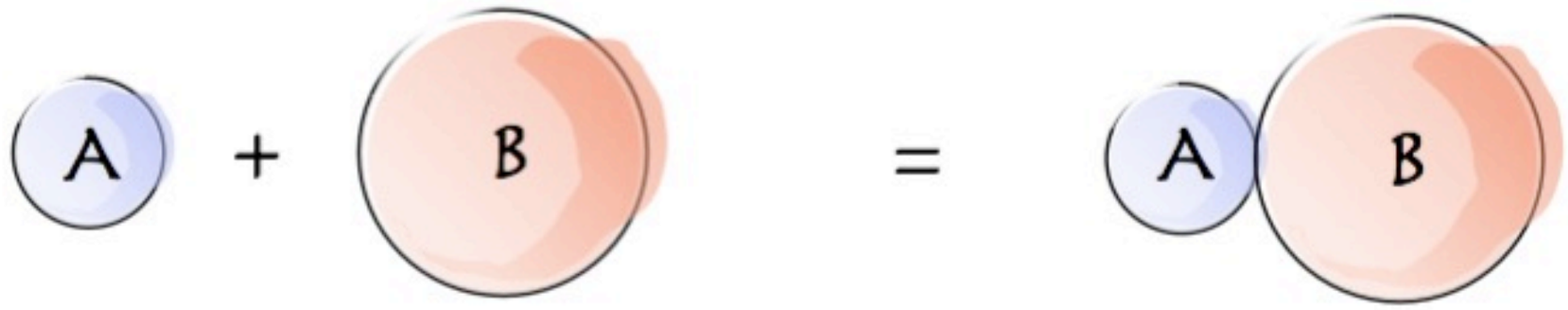
PROLOG

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

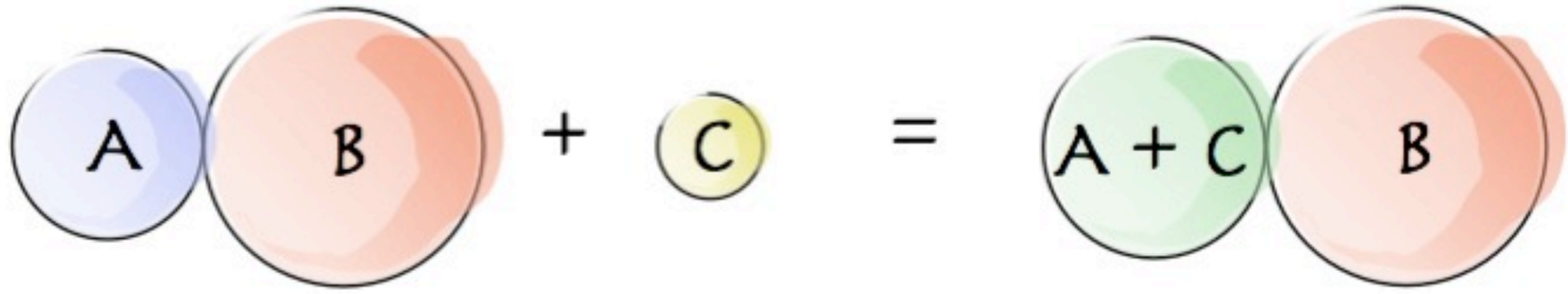
PROLOG

- [ ] SHOW ME THE CODE...

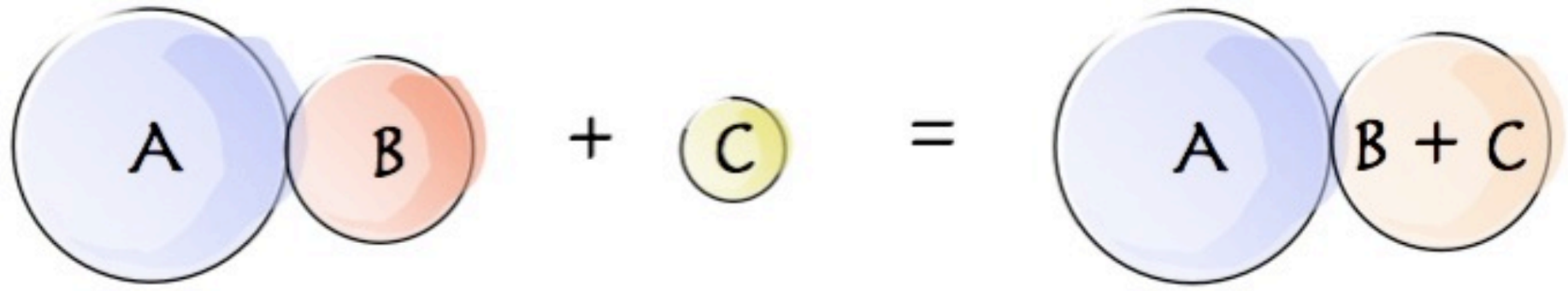
A < B



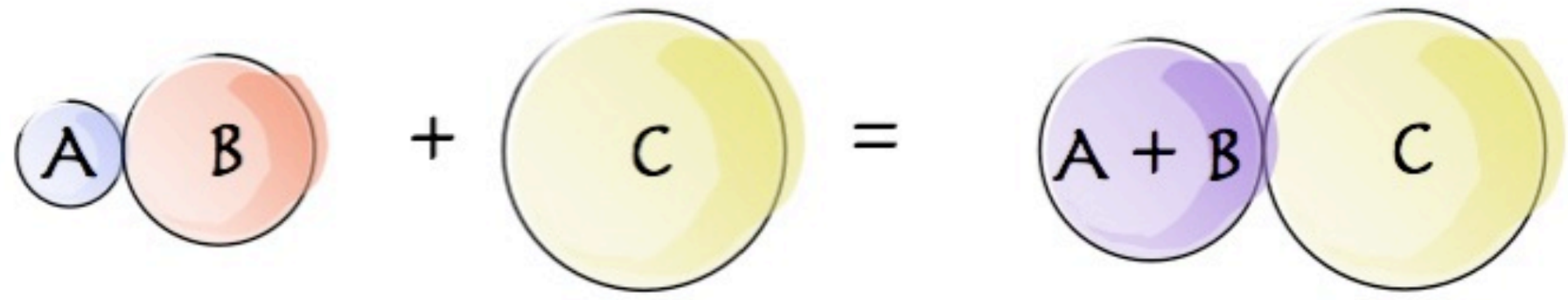
C < A < B



C < B < A



C > A, C > B



$A < B$

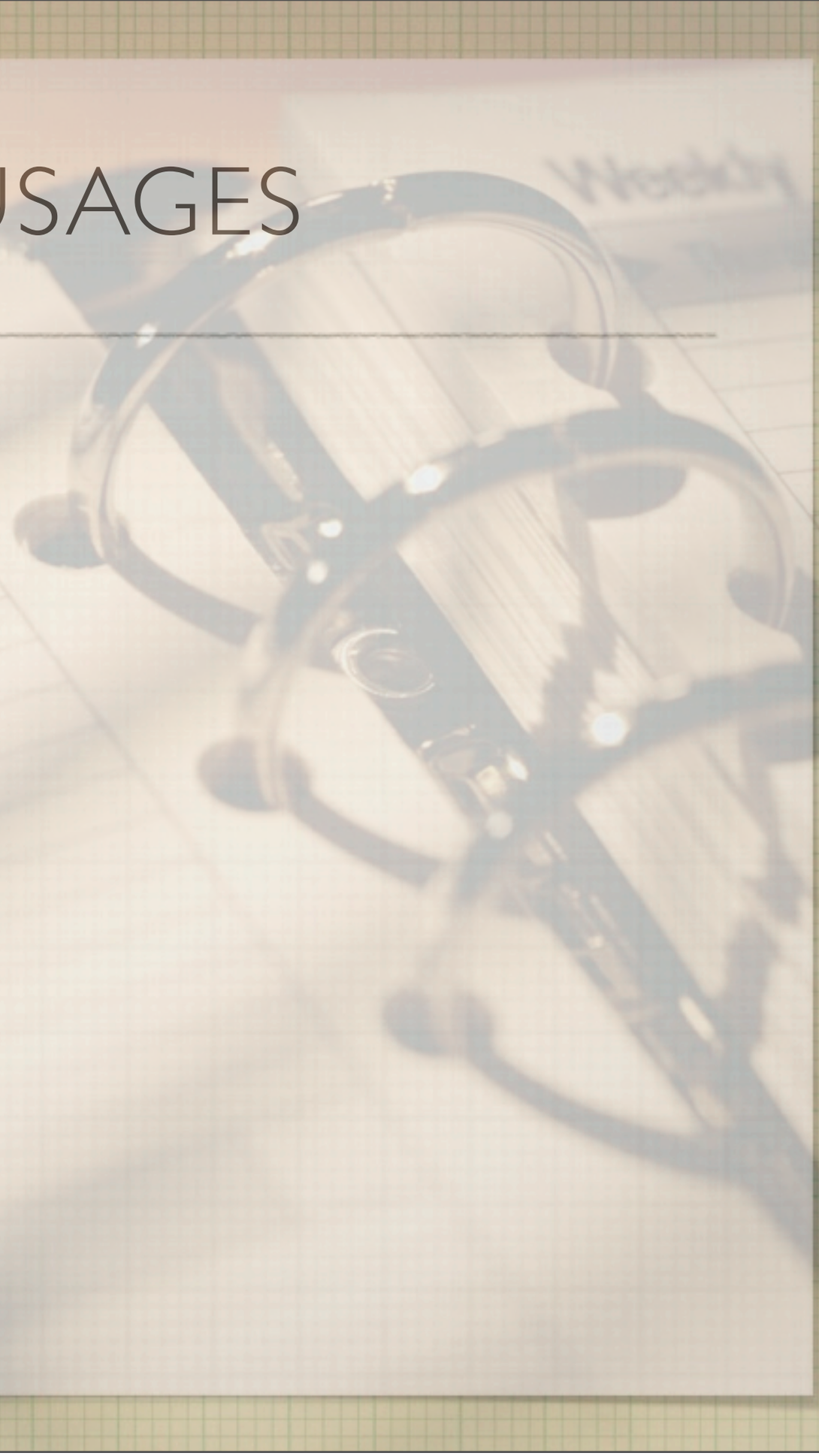
$C < A < B$

$C > A,\ C > B$

PROLOG USAGES

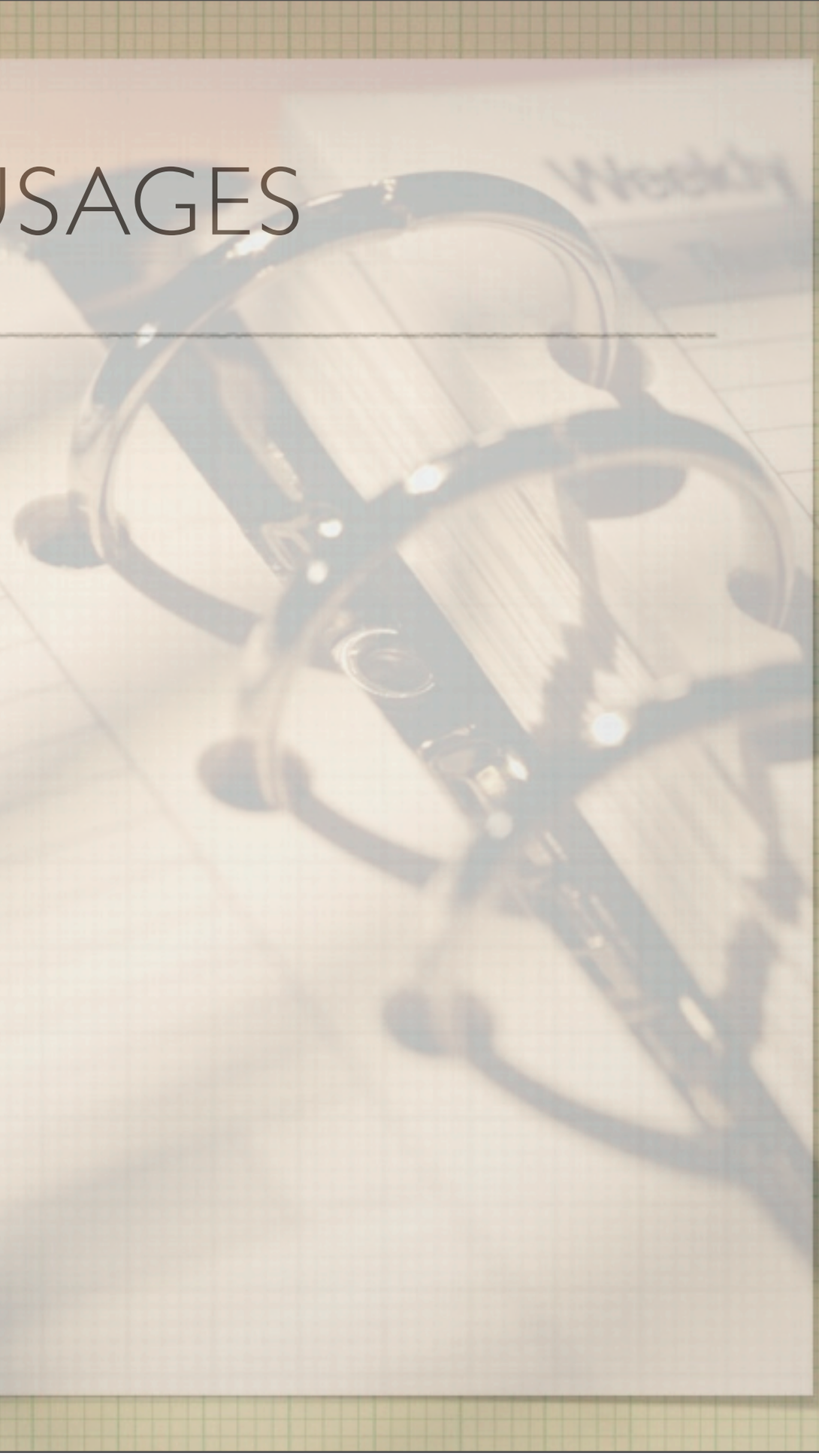
PROLOG USAGES

- [ ] 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
A Purely Functional Language

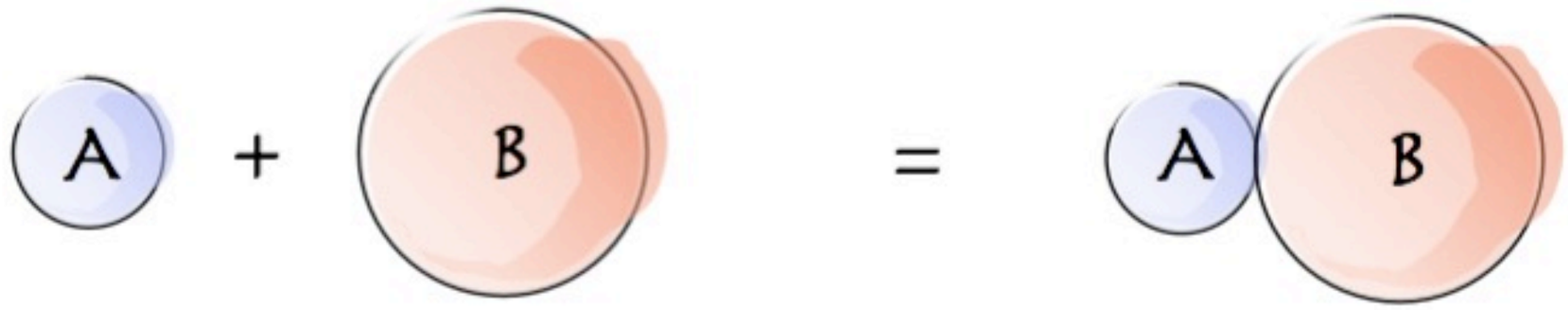
HASKELL

- [ ] PURELY FUNCTIONAL
- [ ] STRONG MATHEMATICAL FOUNDATION

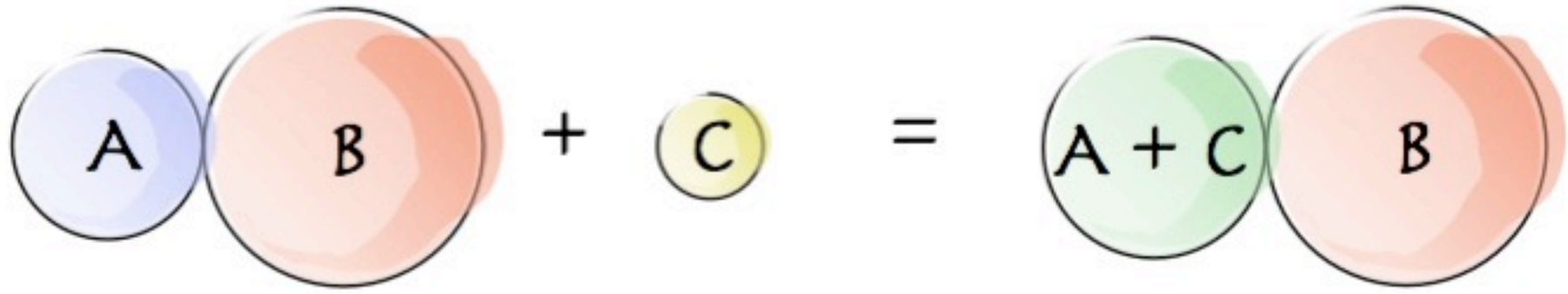
HASKELL

- [ ] SHOW ME THE CODE...

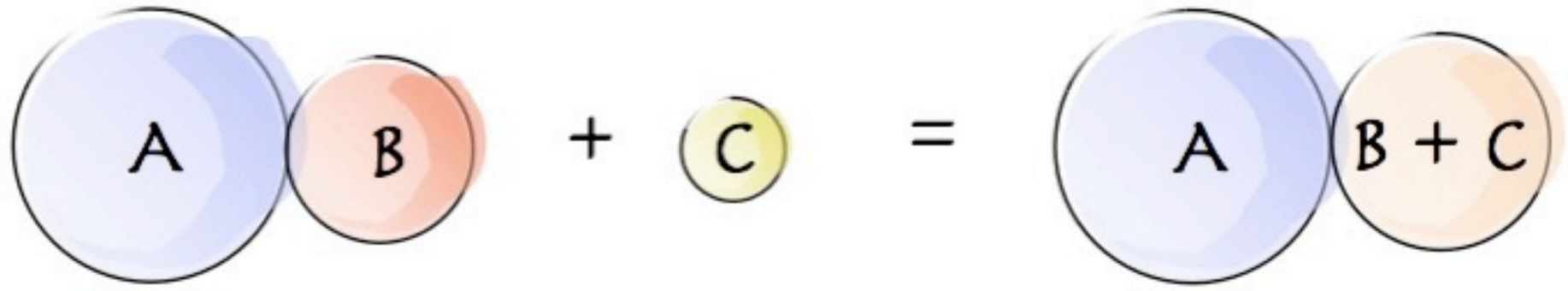
A < B



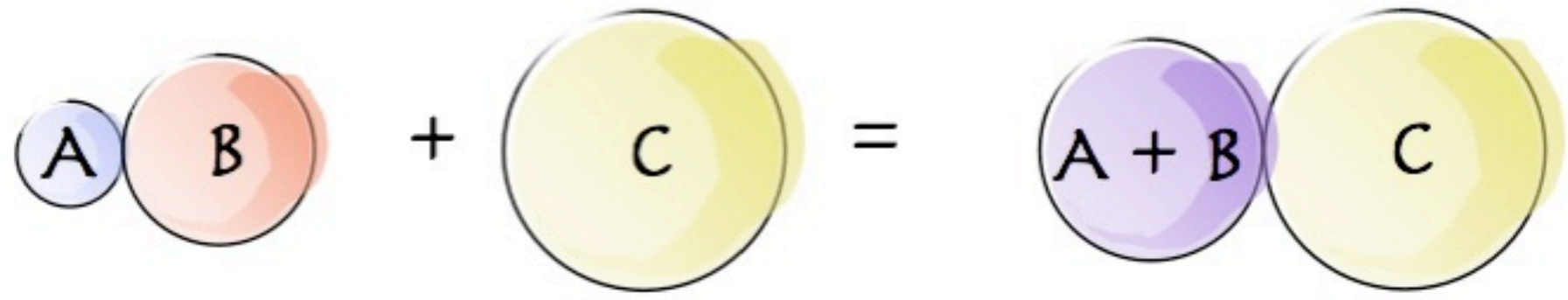
C < A < B



C < B < A



C > A, C > B

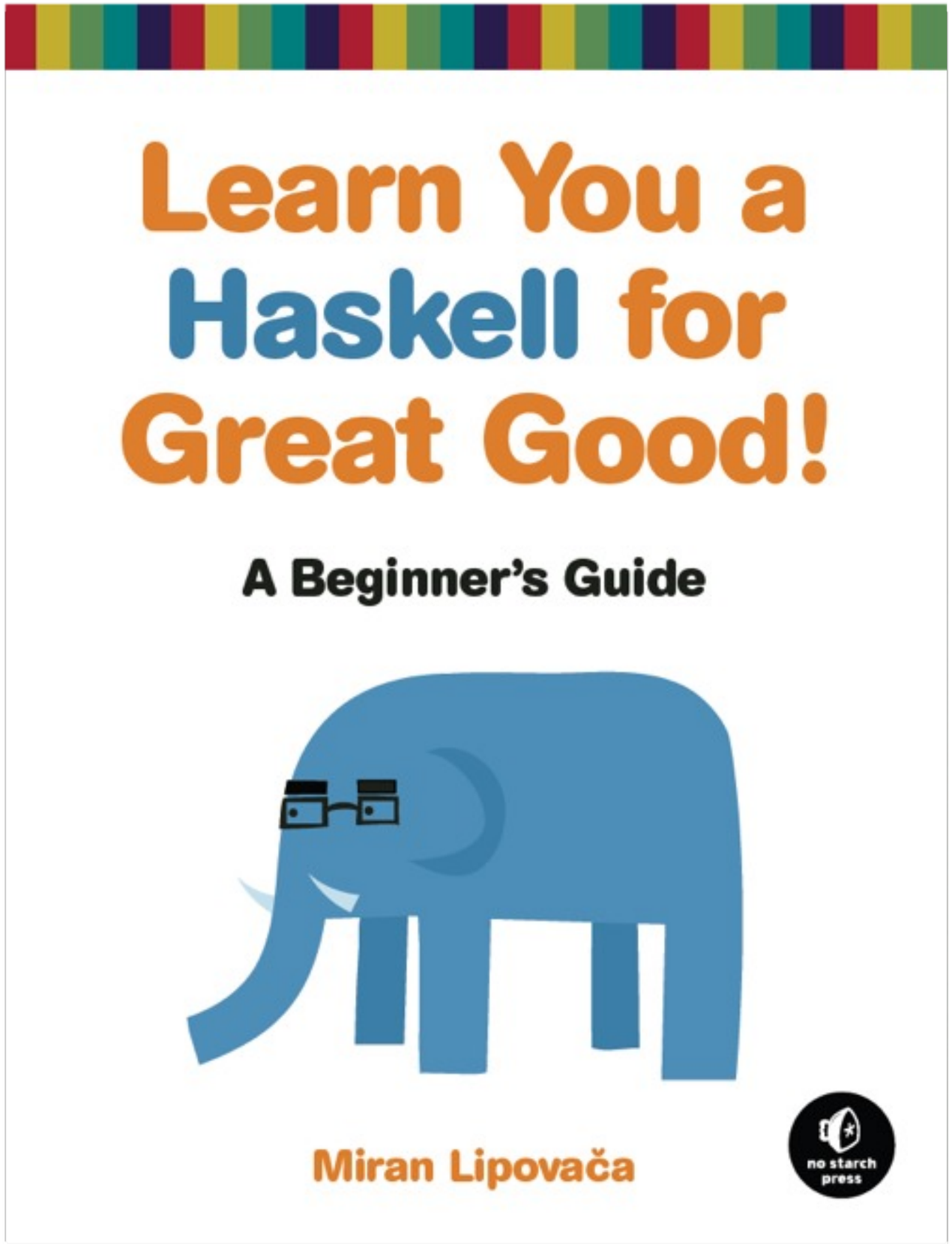


$A < B$

$C < A < B$

$C > A,\ C > B$

- [ ] DESERVES A DEEPER INVESTIGATION...



	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 (?)**

The Pragmatic Programmers

Seven Languages in Seven Weeks

A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

Edited by Jacquelyn Carter

NICE TO READ