Verifica & Convalida

PBT's Patterns

Alberto Momigliano

2015-2016

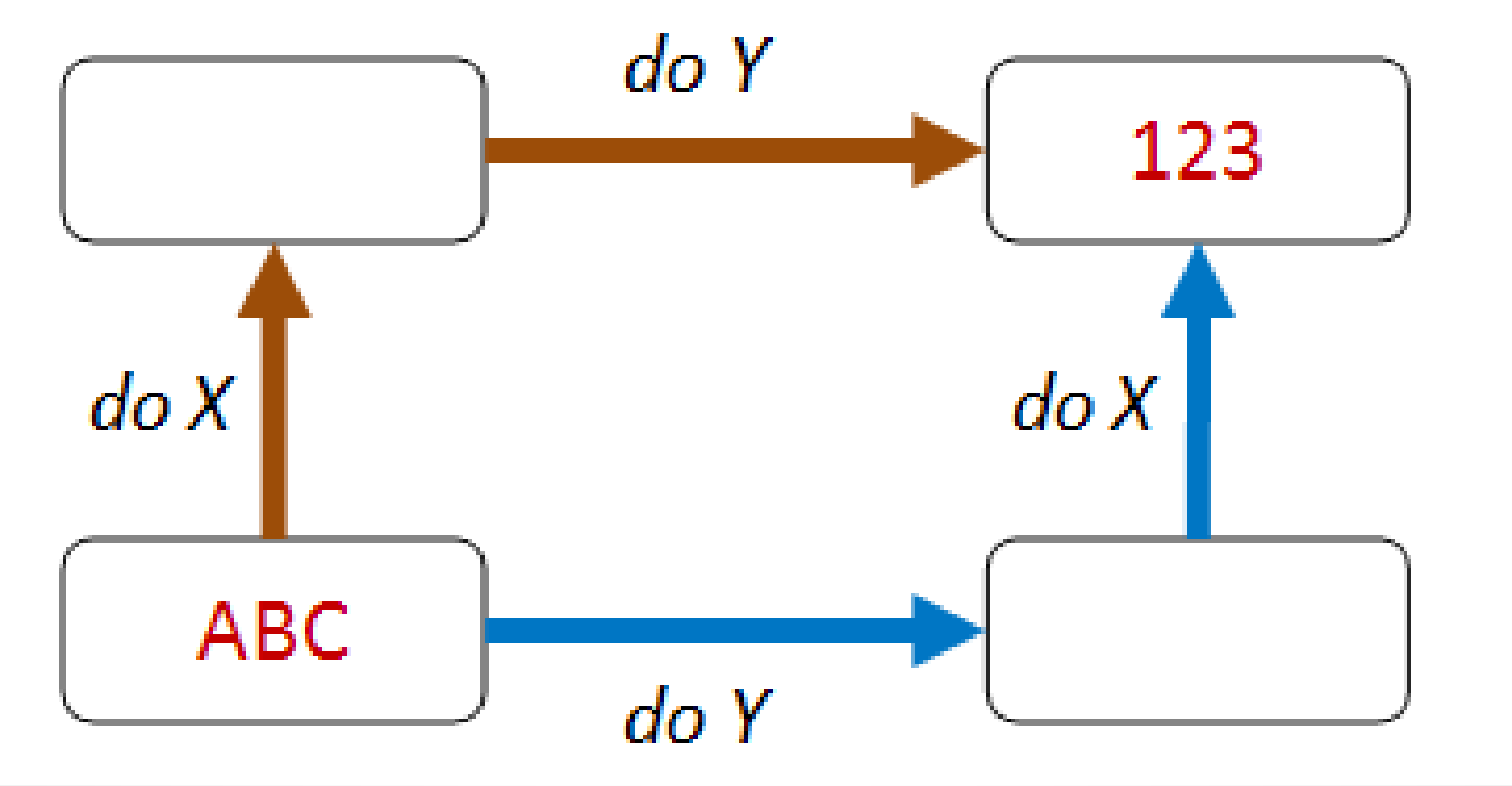
Properties: where do they come from?

- It's easy to come up with properties when we are dealing with **algebraic** notions, viz *field* properties of addition or functions over abstract data types
- Seems harder for everyday programming
- However, some **patterns** emerge from algebraic considerations that may be used even in business applications:
  - Serializations
  - State machines

Patterns

- Different paths, same destination
- There and back again
- Some things never change
- The more things change, the more they stay the same
- …

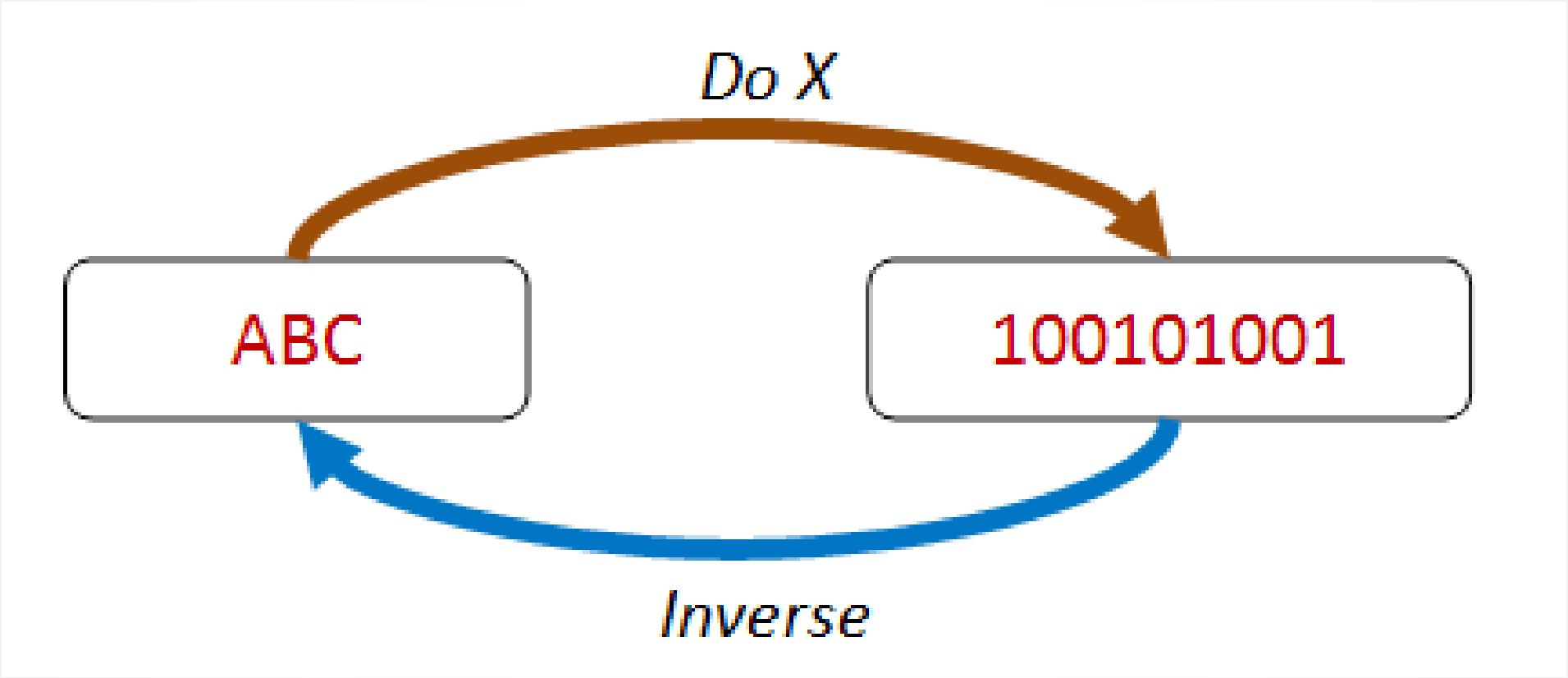
"Different paths, same destination"



"Different paths, same destination"

- Basic **commutation** property:
  - $F \circ G = G \circ F$
- Some simple examples later on
- More in general, there could be different ways to achieve the same goal:
  - think about initial and final states in a state machine representing say a web application for some payment system. You may want to check that whether you chose to pay by credit card or PayPal you're gonna be charged the same and the transaction succeeds.

"There and back again"



"There and back again"

- Combining an operation with its **inverse**, ending up with the same value you started with.
- $F \circ F^{-1} = id$
  - Serialisation
  - Add/subtract
- Other patterns that are not strictly inverse
  - insert/contains,
  - create/exists
  - write/read

"Some things never change"

"Some things never should"

- Based on an invariant that is preserved after some transformation:
  - Length of a collection after a `map`
    - `List.length xs =`

      `(List.map f xs |> List.length)`
  - the contents of a collection after a `sort`
    - `set xs = set (List.sort xs)`
  - PLT results such as type preservation

“The more things change, the more they stay the same"

- These kinds of properties are based on "**idempotence**" -- that is, doing an operation twice is the same as doing it once:
  - F ° F = F
  - Can be generalized to n-fold application
- `List.sort (List.sort xs) = List.sort xs`
- Think: database updates and message processing
- HTTP verbs: GET, PUT, and DELETE

More extensive examples?

- Not many. One with comment is this Roman numeral kata See also this for red & black trees
- There are significant applications in the **Erlang** community, but are mostly proprietary
- A project: PBT the crap out of some F# code posted around, such as
  - F# Snippets
  - Ninety-Nine F# Problems