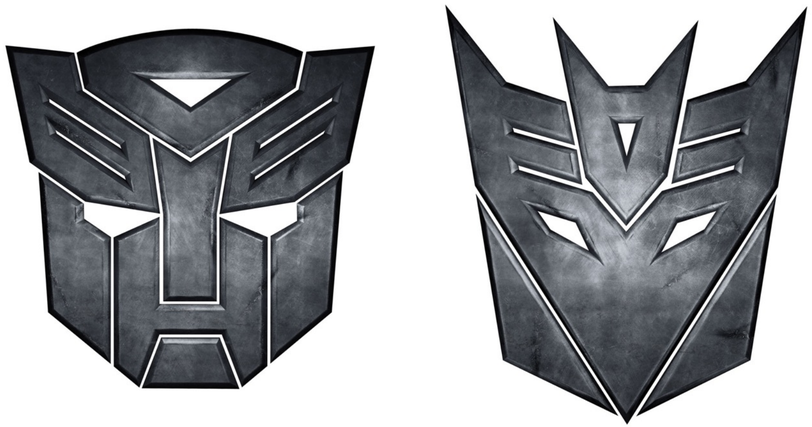
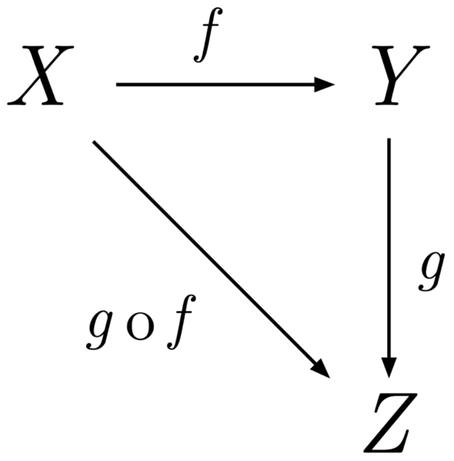
The case for Case Classes: More than meets the eye

Yoel Spotts
Experience, LLC



Goals

- Really understand case classes
- Insight into Scala compiler
- Influence of math in Scala
  - aka: you can run, but you can’t hide
- Everything in Scala: more than meets the eye



Case Classes: The Known

- Class and companion object
- Apply factory method
- Immutable class
- toString, hashCode, equals
- copy constructor
- Unapply destructor

Case Classes: The Unknown

- Extends Product
  - Case classes are cartesian products
  - Aka Tuples
- Extends AbstractFunctionN
  - Underscores view as tuples
- Curried form
- ADT
  - Very brief introduction

Algebraic Data Types

“In computer programming, particularly functional programming and type theory, an algebraic data type is a kind of composite type, i.e. a type formed by combining other types. Two common classes of algebraic types are product types—i.e. tuples and records—and sum types, also called tagged or disjoint unions or variant types.” (Wikipedia)

Sum Types

“In computer science, a tagged union, also called a variant, variant record, discriminated union, disjoint union, or sum type, is a data structure used to hold a value that could take on several different, but fixed, types. Only one of the types can be in use at any one time, and a tag field explicitly indicates which one is in use. It can be thought of as a type that has several "cases," each of which should be handled correctly when that type is manipulated. Like ordinary unions, tagged unions can save storage by overlapping storage areas for each type, since only one is in use at a time.” (Wikipedia)

The End

Thanks to http://www.alessandrolacava.com/blog/scala-case-classes-in-depth/