Identity and Change

- Assignment poses the new problem of deciding whether two expressions are "the same"
- When one excludes assignments and writes: ***val x = E; val y = E***, where E is an arbitrary expression, then it is reasonable to assume that x and y are the same. That is to say that we could have also written: ***val x = E; val y = x***
- This property is called *referential transparency*
- But once we allow the assignment, the two formulations are different. For example:

  val x = new BankAccount
  val y = new BankAccount

  Question: Are x and y the same? - **No**

Operational Equivalence

- To respond to the last question, we must specify what is meant by "the same"
- The precise meaning of "being the same" is defined by the property of *operational equivalence*
- In a somewhat informal way, this property is stated as follows: Suppose we have two definitions x and y. x and y are operationally equivalent if *no possible test* can distinguish between them

Testing for Operational Equivalence

To test if x and y are the same, we must:

- Execute the definitions followed by an arbitrary sequence f of operations that involves x and y, observing the possible outcomes.

  val x = new BankAccount
  val y = new BankAccount
  f(x, y)

  val x = new BankAccount
  val y = new BankAccount
  f(x, x)
- Then execute the definitions with another sequence S' obtained by renaming all occurrences of y by x in S
- If the results are different, then the expressions x and y are certainly different
- On the other hand, if all possible pairs of sequences (S, S') produce the same result, then x and y are the same

Counterexample for Operational Equivalence

- Based on this definition, let's see if the expressions

  val x = new BankAccount
  val y = new BankAccount

  define values x and y that are the same
- Let's follow the definitions by a test sequence:

```
val x = new BankAccount
val y = new BankAccount
x deposit 30          // result: 30
y withdraw 20         // java.lang.Error: insufficient funds
```

- Now rename all occurrences of y with x in this sequence. We obtain:

  val x = new BankAccount
  val y = new BankAccount
  x deposit 30 *// result: 30*
  x withdraw 20 *// result: 10*
- The final results are different. We conclude that x and y are not the same

Establishing Operations Equivalence

- On the other hand, if we define:

  val x = new BankAccount
  val y = x

  then no sequence of operations can distinguish between x and y; x and y are the same
- To prove that two objects are the same is considerably harder than to prove that they are different because you have to show that no possible sequence of operations can distinguish between them. So, you have to argue about a possibly infinite number of possible experiments, whereas to prove that two definitions are different, a single counter example is enough

Assignment and Substitution Model

- The given examples show that our model of computation by substitution cannot be used
- Indeed, according to this model, one can always replace the name of a value by the expression that defines it. For example, in

  val x = new BankAccount ***val x = new BankAccount***
  val y = x ***val y = new BankAccount***

  the x in the definition of y could be replaced by new BankAccount
- But we have seen that this change leads to a different program
- The substitution model ceases to be valid when we add the assignment
- It is possible to adapt the substitution model by introducing a store, but this becomes considerably more complicated