Conditional Equations

Using only equations between terms is often too restrictive. For instance, suppose we are interested in the specification of Sets of Items. We introduce operations/signatures:

EMPTY: $\rightarrow$ Set
IN: Item $\times$ Set $\rightarrow$ Boolean || test membership
ADD: Item $\times$ Set $\rightarrow$ Set || add item to set

For these operations, the semantics are expressed as “equations”

IN(i, EMPTY) = False
IN(i, ADD(j,s)) = **if** i=j
then True
else IN(i, s)

But this latter “equation” is not an equation between two terms, it is another kind of construct. It cannot be immediately explained by the equivalence relation on terms, and we lose the equational equivalence semantics.

The pure term interpretation can be relatively easily restored, but there is a price to be paid. We introduce another function, one that is not an actual component of the TOI. Such a function is referred to as a **hidden function**, and it is assumed to be inaccessible to client code. For the Set ADT, it would be a function with the signature

IF: Boolean $\times$ Set $\times$ Set $\rightarrow$ Boolean

with the (equational) semantics

IF(True, b1, b2) = b1
IF(False, b1, b2) = b2

We will write such conditional expressions using the **if**–**then**–**else** format rather than using an 'IF' functional. However, the formal interpretation is understood through this hidden function device. This is the first (but not last) instance where a compromise in the perfect reflection of the abstract concept is permitted to achieve expressiveness — hidden functions are not a conceptually required component of the systems whose specifications we wish to express, but they are a practical necessity.