# An “Infamous” Example

We examine an initial algebra specification by Goguen et al (chap. 5 of Yeh book) to describe the (signed) integer ADT (with sum and product operations) given Boolean and Nat ADTs. The intuitive idea is to pair a natural number with a Boolean value that represents its sign, and then express signed arithmetic (assuming we already know unsigned arithmetic). However, it will be seen that the initial algebra of this ADT is *not* the usual algebra of the integers.

## The specification

<u>Signat signature</u>

Pre-defined types: Boolean (with '='), and Nat ({0, 1, 2, ...}) with operations as usual, including +, ∸, *, ≤ (note that ∸ is “proper subtraction”, n∸ m yields 0 when n≤m).

<u>Operation signatures:</u>

PAIR: Nat, Bool □ Signat
ABS: Signat □ Nat
SGN: Signat □ Bool
SUM: Signat, Signat □ Signat
PROD: Signat, Signat □ Signat

Semantic equations (for all s∈Signat, n∈Nat, b∈Bool)

```
1. PAIR(ABS(s), SGN(s)) = s
2. ABS(PAIR(n,b)) = n
3. SGN(PAIR(n,b)) = b
4. SUM(s1, s2) =
      if SGN(s1)=SGN(s2)
       then PAIR(ABS(s1)+ABS(s2), SGN(s1))
       else if ABS(s1)≤ABS(s2)
               then PAIR(ABS(s2)∸ABS(s1), SGN(s2))
               else PAIR(ABS(s1)∸ABS(s2), SGN(s1))
5. PROD(s1,s2) =
      PAIR(ABS(s1)*ABS(s2), SGN(s1)=SGN(s2))
```

## The flaw

**The fault to be found with this specification of the signed integers is that there are *two* "zeros" — PAIR(0,True) or +0, and PAIR(0,False) or –0. There are no equations that enable us to deduce these two different pairs are equivalent, and so in the initial algebra view they are different.**

**This might appear to be a minor oversight, but in fact having two distinct representations of zero causes numerous familiar identities to be invalidated. For instance, for all x, x+0 = x in the integers. But neither of the corresponding values in this specification has this property — note that SUM(+0,–0) = –0, and SUM(–0,+0) = +0 (use the equations in the specification on the term forms of these values to confirm this). Also for all x, x*0 = 0 in the integers, but in the specification PROD(–5,+0) = –0, and PROD(–0,–0) = +0. If it is really the system of signed natural numbers we seek to specify, the specification given does not qualify.**

## A defective repair

**In a widely circulated IBM technical report that preceded the Yeh publication, the authors "corrected" the problem in the SigNat ADT described above by proposing the single additional axiom**

**6. PAIR(0,True) = PAIR(0,False)**

**to unify these two different equivalence classes (i.e., +0 = –0). While at first glance this seems like a simple and obvious solution, it is a *huge* blunder. If we add this axiom to those we already have for SigNat, then**

**True $\underset{3}{\equiv}$ SGN(PAIR(0,True)) $\underset{6}{\equiv}$ SGN(PAIR(0,False)) $\underset{3}{\equiv}$ False!**

**Hence an inconsistency in the pre-defined type Boolean has been introduced into the specification, and the original flawed "approximate specification" has been destroyed completely rather than repaired.**

## An actual repair

**A suitable correction to the original flaw is to not add an equation, but to replace equation 3 by**

**3'. SGN(n,b) = if n=0 then True else b**

**Then the two representations of zero are equivalent:**

**PAIR(0,False)** $\underset{1}{\equiv}$

**PAIR(ABS(PAIR(0,False), SGN(PAIR(0,False))** $\underset{2}{\equiv}$

**PAIR(0, SGN(PAIR(0,False)))** $\underset{3'}{\equiv}$

**PAIR(0, True).**

**So now the two zero terms fall into the same equivalence class, and so we have a true, unique zero. But no inconsistency is introduced — it's impossible to deduce that e.g., True ≡ False (why?).**