Using Z3 to solve a birthday logic puzzle

Logic Puzzle

Alex, Brook, Cody, Dusty, and Erin recently found out that all of their birthdays were on the same day, **though they are different ages**.

On their mutual birthday, I overheard that...

- Dusty said to Brook: "I'm nine years older than Erin."
- Erin said to Brook: "I'm seven years older than Alex."
- Alex said to Brook: "Your age is exactly 70% greater than mine."
- Brook said to Cody: "Erin is younger than you."
- Cody said to Dusty: "The difference between our ages is six years."
- Cody said to Alex: "I'm ten years older than you."
- Cody said to Alex: "Brook is younger than Dusty."
- Brook said to Cody: "The difference between your age and Dusty's is the same as the difference between Dusty's and Erin's."

I realized that when one of them spoke to someone **older**, everything they said was **true**, but when speaking to someone **younger**, everything they said was **false**.

How old is each person?

SMT Encoding

We can model this problem using Linear Integer Arithmetic (LIA). The age of each person is encoded as an integer (Alex, Brook, Cody, Dusty, and Erin —> a, b, c, d, e). The age constraints are either equality or inequality statements. These statements are either negated or validated depending on who made the statement and who they were speaking to.

```
; Variable declarations
; Five ages encoded as ints
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)
(declare-fun d () Int)
(declare-fun e () Int)

; absolute value function to help calculate difference in ages
(define-fun absolute ((x Int)) Int
  (ite (>= x 0) x (- x)))

; stmt function takes two ints (two people) x and y, and x's statement z. It returns z
; as t/f depending on if x is older/younger than y
(define-fun stmt ((x Int) (y Int) (z Bool)) Bool
  (ite (< x y) z (not z)) )

; Constraints
(assert (distinct a b c d e) ) ; Unique birthdays

(assert (stmt d b (= (+ e 9) d) )) ; d is 9 years older than e
(assert (stmt e b (= (+ a 7) e) )) ; a is 7 years older than e
(assert (stmt a b (= (+ (/ (* a 7) 10) a) b) )) ; b's age is 70% greater than a's age
(assert (stmt b c (< e c) )) ; e is younger than c
(assert (stmt c d (= (absolute (- d c)) 6) )) ; the difference in ages between c and d is 6 years
(assert (stmt c a (= (+ a 10) c) )) ; c is 10 years older than a
(assert (stmt c a (< b d) )) ; b is younger than d

; the difference in ages between c and d and between d and e are the same
(assert (stmt b c (= (absolute (- c d)) (absolute (- d e))) ))

; Solve
(check-sat)

; How old is each person?
(get-model)
```

SMT Output

```
sat
(model
  (define-fun z3name!1 () Int
    9)
  (define-fun b () Int
    51)
  (define-fun a () Int
    30)
  (define-fun c () Int
    55)
  (define-fun d () Int
    46)
  (define-fun e () Int
    37)
  (define-fun z3name!0 () Int
    9)
)
```

Interpreting the output

a (Alex) = 30, b (Brook) = 51, c (Cody) = 55, d (Dusty) = 46, e (Erin) = 37. The two unnamed ints are the result of the last assertion, calculating the difference in years between Cody and Dusty’s ages and Dusty and Erin’s ages, which is 9 for each.

Provided answer

Alex is 30

Brook is 51

Cody is 55

Dusty is 46

Erin is 37