I217E: Functional Programming

Nao Hirokawa

JAIST

Term 2-1, 2022

`http://www.jaist.ac.jp/~hirokawa/lectures/fp/`

Schedule

10/12	introduction	11/9	interpreters
10/14	algebraic data types I	11/11	compilers
10/19	algebraic data types II	11/16	termination
10/21	applications	11/18	confluence
10/26	program reasoning	11/25	verification
10/28	data structures I	11/30	review
11/2	data structures II	12/5	exam
11/4	computational models		

Evaluation

exam (60) + reports (40)

Why Is Correctness Important?

failure of Ariane 5 (C) ESA

Correctness of Functional Program

Specification

$$\mathtt{sum} : \mathbb{N} \to \mathbb{N}$$

$$\mathtt{sum}(n) = \sum_{i=0}^{n} i \quad \Big(= \frac{n(n+1)}{2}\Big)$$

Implementation

$$\mathtt{sum}(n) = \begin{cases} 0 & \text{if } n = 0 \\ \mathtt{sum}(n-1) + n & \text{otherwise} \end{cases}$$

Question

show $\mathtt{sum}(n) = \dfrac{n(n+1)}{2}$

From Loop (in C) to Recursion: Outline

```
int sum(int n) {
  int x = 0;
  while (n > 0) {
    x = x + n;
    n = n - 1;
  }
  return x;
}
```

Question

how to show correctness?

From Loop to Recursion

```
       int sum(int n) {
sum:     int x = 0;

 f1:     while (n > 0) {

 f2:         x = x + n;
 f3:         n = n - 1;
         }
 f4:     return x;
       }
```

$\mathtt{sum}(n) = \mathtt{f}_1(0, n)$

$$\mathtt{f}_1(x, n) = \begin{cases} \mathtt{f}_2(x, n) & \text{if } n > 0 \\ \mathtt{f}_4(x, n) & \text{otherwise} \end{cases}$$

$\mathtt{f}_2(x, n) = \mathtt{f}_3(x + n, n)$

$\mathtt{f}_3(x, n) = \mathtt{f}_1(x, n - 1)$

$\mathtt{f}_4(x, n) = x$

Question

Show $\mathtt{sum}(n) = \dfrac{n(n+1)}{2}$. Hint: find lemma $\mathtt{f}_1(x, n) = \boxed{?}$

Proof Scores

CafeOBJ and Proof Score Approach

Observation

proof consists of

- non-trivial part (e.g. lemma discovery)
- trivial part (just calculation)

automate trivial part!

CafeOBJ

- available at `https://cafeobj.org/`
- rewriting-based functional programming language
- aiming at software verification
- programmer is responsible for termination and confluence

Arithmetic Operations

```
mod! PNAT {
  [Nat]
  op 0 : -> Nat .
  op s : Nat -> Nat .
  op _+_ : Nat Nat -> Nat {assoc comm prec 30}.
  op _*_ : Nat Nat -> Nat {assoc comm prec 29}.
  op _=_ : Nat Nat -> Bool {comm}.

  eq 0        + Y:Nat = Y .
  eq s(X:Nat) + Y:Nat = s(X + Y) .

  eq 0        * Y:Nat = 0 .
  eq s(X:Nat) * Y:Nat = X * Y + Y .

  eq (X:Nat    = X:Nat)    = true .
  eq (0        = s(Y:Nat)) = false .
  eq (s(X:Nat) = s(Y:Nat)) = (X = Y) .
}
```


Sum

```
mod! SUM {
  pr(PNAT) .
  op sum : Nat -> Nat .
  ops f1 f2 f3 f4 : Nat Nat -> Nat .

  eq sum(N:Nat)           = f1(0, N) .
  eq f1(X:Nat, s(N:Nat)) = f2(X, s(N)) .
  eq f1(X:Nat, 0)         = f4(X, 0) .
  eq f2(X:Nat, N:Nat)     = f3(X + N, N) .
  eq f3(X:Nat, s(N:Nat)) = f1(X, N) .
  eq f4(X:Nat, N:Nat)     = X .
}

open SUM .
red sum(s(s(s(0)))) .     -- s(s(s(s(s(s(0))))))
close .
```


Proof Score for Lemma

```
mod! LEMMA {  pr(SUM) .
  op lemma : Nat Nat -> Bool .
  eq lemma(X:Nat, N:Nat) =
       (f1(X, N) + f1(X, N) = X + X + N * s(N)) .
}
--> Proof.  We perform induction on N.
--> If N = 0 then the claim is trivial.
open LEMMA .
red lemma(X:Nat, 0) .     -- true
close .
--> If N = s(m) then the claim follows from I.H.
open LEMMA .
op m : -> Nat .
eq f1(X:Nat, m) + f1(X:Nat, m) = X + X + m * s(m) .
red lemma(X:Nat, s(m)) . -- true
close .
--> QED.
```


Proof Score for Main Theorem

```
mod! THEOREM {
  pr(SUM) .
  eq f1(X:Nat, N:Nat) + f1(X, N) = X + X + N * s(N) .

  op theorem : Nat -> Bool .
  eq theorem(N:Nat) = (sum(N) + sum(N) = N * s(N)) .
}
--> Proof.
--> The theorem is immedidate from the lemma.
open THEOREM .
red theorem(N:Nat) .      -- true
close .
```