COMP40018.3 Reasoning
Imperial College London

Boxuan Tang

Spring 2021

Contents

1 Functional Programs

1.1 Induction over Numbers

1.1.1 Mathematical Induction

Principle: For any $P \subseteq \mathbb{N}$:

$$P(0) \land \forall k : \mathbb{N}.[P(k) \to P(k+1)] \longrightarrow \forall n : \mathbb{N}.P(n)$$

Proof Schema:
Base Case:
  To Show: P(0)
Inductive Step: Take arbituary k
  Inductive Hypothesis: P(k)
  To Show: P(k+1)

1.1.2 Mathematical Induction Technique

Principle: For any $P \subseteq \mathbb{N}$ and any $m : \mathbb{Z}$

$$P(m) \land \forall k \geq m.[P(k) \to P(k+1)] \longrightarrow \forall n \geq m.P(n)$$

Proof Schema:
Base Case:
  To Show: P(m)
Inductive Step: Take arbituary k. Assume that $k \geq m$.
  Inductive Hypothesis: P(k)
  To Show: P(k+1)

1.1.3 Strong Induction

Principle: For any $P \subseteq \mathbb{N}$:

$$P(0) \land \forall k : \mathbb{N}.[\forall j \in [0..k].P(j) \to P(j+1)] \longrightarrow \forall n : \mathbb{N}.P(n)$$

Proof Schema: (for 2 base cases)
Base Case:
  To Show: P(0)
Inductive Step: Take arbituary k
  Inductive Hypothesis: $\forall j \in [0..k].P(j)$
  To Show: P(k+1)
  1st Case: k = 0
    To Show: P(1)
  2nd Case: $k \neq 0$
    (A) $k \geq 1$ because $k : \mathbb{N}$ and $k \neq 0$ by case
    (B) $k, k-1 \in [0..k]$ because $k : \mathbb{N}$ and $k \neq 0$

1.1.4 Strong Induction Technique

Principle: For any $P \subseteq \mathbb{N}$ and any $m : \mathbb{Z}$

$$P(m) \land \forall k \geq m.[\forall j \in [m..k].P(j) \to P(j+1)] \longrightarrow \forall n \geq m.P(n)$$

1.2 Structural Induction

1.2.1 Induction over Lists

Principle: For any type T, and $P \subseteq [T]$:

$$P([]) \land \forall vs : [T].\forall v : T.[P(vs) \to P(v : vs)] \longrightarrow \forall vs : [T].P(xs)$$

Proof Schema:
Base Case:
  To Show: P([])
Inductive Step: Take arbituary v':a, vs':[a]
  Inductive Hypothesis: P(vs')
  To Show: P((v':vs'))
List Lemmas:

(A) $us ++ [] = us$

(B) $[] ++ us = us$

(C) $(u : us) ++ vs = u : (us ++ vs)$

(D) $(us ++ vs) ++ ws = us ++ (vs ++ ws)$

1.2.2 Induction over arbituary data structures

- data Nat = Zero | Succ Nat

$$P(Zero) \land \forall n : Nat.[P(n) \to P(Succ\ \ n)] \longrightarrow \forall n : Nat.P(n)$$

- data Tree a = Empty | Node (Tree a) a (Tree a)

$$P(Empty) \land \forall t1, t2 : Tree\ \ T.\forall x : T.[P(t1) \land P(t2) \to P(\mathit{Node\ t1\ x\ t2})] \\ \longrightarrow \forall t : Tree\ \ T.P(t)$$

- data BExp = Tr | Fl | BNt BExp | BAnd BExp BExp

$$P(Tr) \land P(Fl) \land \forall b : BExp.[P(b) \to P(\mathit{BNt\ b})] \\ \land \forall b1, b2 : BExp.[P(b1) \land P(b2) \to P(\mathit{BAnd\ b1\ b2})] \longrightarrow \forall b : BExp.P(b)$$

- data T = C1 [Int] | C2 Int T

$$\forall is : [Int].P(\mathit{C1\ is}) \land \forall i : Int.\forall t : T.[P(t) \to P(\mathit{C2\ i\ t})] \longrightarrow \forall t : T.P(t)$$

- data Reds = BaseR | Red Greens
  data Greens = BaseG | Green Reds

$$\begin{aligned}&P(\mathit{BaseR}) \wedge \forall g : \mathit{Greens}.[Q(g) \to P(\mathit{Red}\ g)]\\ &\wedge Q(\mathit{BaseG}) \wedge \forall r : \mathit{Reds}.[P(r) \to Q(\mathit{Green}\ r)]\\ &\longrightarrow \forall r : \mathit{Reds}.P(r) \wedge \forall g : \mathit{Greens}.Q(g)\end{aligned}$$

- data Cactus = Root Tree
  data Tree = Leaf | Node Trees
  data Trees = Empty | Cons Tree Trees

$$\begin{aligned}&P(\mathit{Leaf}) \wedge \forall ts : \mathit{Trees}.[Q(ts) \to P(\mathit{Node}\ ts)] \wedge\\ &Q(\mathit{Empty}) \wedge \forall t : \mathit{Tree}.\forall ts : \mathit{Trees}.[P(t) \wedge Q(ts) \to Q(\mathit{Cons}\ t\ ts)]\\ &\longrightarrow \forall t : \mathit{Tree}.P(t) \wedge \forall ts : \mathit{Trees}.Q(ts)\end{aligned}$$

1.2.3 Two Approaches

1. Invent an Auxiliary Lemma
2. Strengthen the original property

1.3 General Induction

1.3.1 Inductively Defined Sets

- $S_{\mathbb{N}}$ defined over Zero and Succ through
  $\mathit{Zero} \in S_{\mathbb{N}}$
  $\forall n.[n \in S_{\mathbb{N}} \to \mathit{Succ}\ \ n \in S_{\mathbb{N}}]$

$$Q(\mathit{Zero}) \wedge \forall m \in S_{\mathbb{N}}.[Q(m) \to Q(\mathit{Succ}\ m)] \longrightarrow \forall n \in S_{\mathbb{N}}.Q(n)$$

- Tree
  $i \in \mathbb{N} \to \mathit{Leaf}\ i \in \mathit{Tree}$
  $\forall t1, t2 \in \mathit{Tree}.\forall c \in \mathit{Char}.\mathit{Node}\ \ c\ \ t1\ \ t2 \in \mathit{Tree}$

$$\begin{aligned}&\forall i \in \mathbb{N}.Q(\mathit{Leaf}\ i)\\ &\wedge \forall t1, t2 \in \mathit{Tree}.\forall c \in \mathit{Char}.[Q(t1) \wedge Q(t2) \to Q(\mathit{Node}\ \ c\ \ t1\ \ t2)]\\ &\longrightarrow \forall t \in \mathit{Tree}.Q(t)\end{aligned}$$

- $OL \subseteq \mathbb{N}^*$
  $[] \in OL$
  $\forall i \in \mathbb{N}.i : [] \in OL$
  $\forall i, j \in \mathbb{N}, js \in \mathbb{N}^*.[i \leq j \wedge j : js \in OL \to i : j : js \in OL]$

$$\begin{aligned}&Q([]) \wedge \forall i \in \mathbb{N}.Q(i : [])\\ &\wedge \forall i, j \in \mathbb{N}, js \in \mathbb{N}^*.[i \leq j \wedge j : js \in OL \wedge Q(j : js) \to Q(i : j : js)]\\ &\longrightarrow \forall ns \in OL.Q(ns)\end{aligned}$$

1.3.2 Inductively Defined Relations

- $SL \subseteq \mathbb{N} \times \mathbb{N}$
  $\forall k \in \mathbb{N}.SL(0, k+1)$
  $\forall m, n \in \mathbb{N}.[SL(m, n) \to SL(m+1, n+1)]$

$$\begin{aligned}&\forall k \in \mathbb{N}.Q(0, k+1)\\ &\land \forall m, n \in \mathbb{N}.[SL(m.n) \land Q(m, n) \to Q(m+1, n+1)]\\ &\longrightarrow \forall m, n \in \mathbb{N}.[SL(m, n) \to Q(m, n)]\end{aligned}$$

- $Even \subseteq S_{\mathbb{N}}$
  $Even(Zero)$
  $\forall n \in S_{\mathbb{N}}.[Even(n) \to Even(Succ\ (Succ\ n))]$

$$\begin{aligned}&Q(Zero)\\ &\land \forall n \in S_{\mathbb{N}}.[Even(n) \land Q(n) \to Q(Succ\ (Succ\ n))]\\ &\longrightarrow \forall n \in S_{\mathbb{N}}.[Even(n) \to Q(n)]\end{aligned}$$

- $Odd \subseteq S_{\mathbb{N}}$
  Odd(Succ Zero)
  $\forall n \in S_{\mathbb{N}}.[Odd(n) \to Odd(Succ\ (Succ\ n))]$

$$\begin{aligned}&Q(Succ\ Zero)\\ &\land \forall n \in S_{\mathbb{N}}.[Odd(n) \land Q(n) \to Q(Succ\ (Succ\ n))]\\ &\longrightarrow \forall n \in S_{\mathbb{N}}.[Odd(n) \to Q(n)]\end{aligned}$$

1.3.3 Inductively Defined Functions

- F 0 = 0
  F i = 1 + F(i - 3)

$$\begin{aligned}&Q(0, 0)\\ &\land \forall j, k : \mathbb{Z}.[j \neq 0 \land F(j-3) = k \land Q(j-3, k) \to Q(j, k+1)\\ &\longrightarrow \forall j, k : \mathbb{Z}.[F\ j\ = k \to Q(j,k)]\end{aligned}$$

- G'(i,j,cnt,acc)
  | i==cnt = acc
  | otherwise = G'(i,j,cnt+1,acc+j)

$$\begin{aligned}&\forall i, j, acc : \mathbb{N}.Q(i, j, i, acc, acc)\\ &\land \forall i, j, acc, cnt, r : \mathbb{N}.[i \neq cnt \land G'(i, j, cnt+1, acc+j) = r\\ &\quad \land Q(i, j, cnt+1, acc+j, r) \to Q(i, j, cnt, acc, r)]\\ &\longrightarrow \forall i, j, acc, cnt, r : \mathbb{N}.[G'(i, j, cnt, acc) = r \to Q(i, j, cnt, acc, r)\end{aligned}$$

- DM'(i,j,cnt,acc)
  | acc+j > i = (cnt,i-acc)
  | otherwise = DM'(i,j,cnt+1,acc+j)

  $$\begin{aligned}&\forall i, j, cnt, acc : \mathbb{N}.[acc + j > i \rightarrow Q(i, j, cnt, acc, cnt, i - acc)]\\ &\wedge \forall i, j, acc, cnt, k1, k2 : \mathbb{N}.[acc + j \leq i \wedge DM'(i, j, cnt + 1, acc + j) = (k1, k2)\\ &\quad \wedge Q(i, j, cnt + 1, acc + j, k1, k2) \rightarrow Q(i, j, cnt, acc, k1, k2)]\\ &\longrightarrow \forall i, j, acc, cnt, k1, k2 : \mathbb{N}.[DM'(i, j, cnt, acc) = (k1, k2) \rightarrow Q(i, j, cnt, acc, k1, k2)]\end{aligned}$$

- M'(i,cnt,acc)
  | i==cnt = acc
  | otherwise = M'(i,cnt+1,2*acc)

  $$\begin{aligned}&\forall i, acc : \mathbb{N}.Q(i, i, acc, acc)\\ &\forall i, cnt, acc, r : \mathbb{N}.[i \neq cnt \wedge M'(i, cnt + 1, 2 * acc) = r \wedge Q(i, cnt + 1, 2 * acc, r)\\ &\quad \rightarrow Q(i, cnt, acc, r)]\\ &\longrightarrow \forall i, cnt, acc, r : \mathbb{N}.[M'(i, cnt, acc) = r \rightarrow Q(i, cnt, acc, r)]\end{aligned}$$

2 Imperative Programs

2.1 Program Specifications

2.1.1 Hoare Logic

$$\frac{P[x \mapsto x_{old}] \wedge x = E[x \mapsto x_{old}] \longrightarrow Q}{\{P\} \quad x = E; \quad \{Q\}}$$

2.1.2 Straight Line Code

$$\frac{\{P\} \quad code1 \quad \{R\} \quad \{R\} \quad code2 \quad \{Q\}}{\{P\} \quad code1; code2 \quad \{Q\}}$$

2.2 Conditional Branches

$$\frac{\{P \wedge cond\} \quad code1 \quad \{Q\} \quad \{P \wedge \neg cond\} \quad code2 \quad \{Q\}}{\{P\} \quad if(cond)\{code1\}else\{code2\} \quad \{Q\}}$$

2.3 Method Calls

void someMethod(type x_1, ..., type x_n)
//Pre: R
//Post: S

$$\frac{P \longrightarrow R[\overline{x} \mapsto \overline{v}] \qquad P[\overline{v}[..) \mapsto \overline{v}[..)_{old}] \wedge S[\overline{x} \mapsto \overline{v}][\overline{v}[..)_{pre} \mapsto \overline{v}[..)_{old}] \longrightarrow Q}{\{P\} \quad someMethod(v_1, ..., v_n) \quad \{Q\}}$$

$$\frac{\begin{array}{l} P \longrightarrow R[\overline{x} \mapsto \overline{v}] \\ P[\overline{v}[..) \mapsto \overline{v}[..)_{old}][res_{old} \mapsto res] \wedge res = r \\ \wedge\, S[\overline{x} \mapsto \overline{v}][\overline{v}[..)_{pre} \mapsto \overline{v}[..)_{old}][res \mapsto res_{old}] \longrightarrow Q \end{array}}{\{P\} \quad res = someMethod(v_1, ..., v_n) \quad \{Q\}}$$

2.4 Iteration

1. I holds before the loop is entered
2. Given condition, the loop re-establishes I
3. Termination of loop and I establishes Q

$$\frac{P \longrightarrow I \quad \{I \wedge cond\}\ body\ \{I\} \quad I \wedge \neg cond \longrightarrow Q}{\{P\} \quad while(cond)\{body\} \quad \{Q\}}$$

$$\frac{I[\overline{mod} \mapsto \overline{mod}_{old}] \wedge cond[\overline{mod} \mapsto \overline{mod}_{old}] \wedge body\text{-}effect \longrightarrow I}{\{I \wedge cond\} \quad body \quad \{I\}}$$

4. V is bounded
5. V decreases with each iteration

$$I[\overline{mod} \mapsto \overline{mod}_{old}] \wedge cond[\overline{mod} \mapsto \overline{mod}_{old}] \wedge body\text{-}effect$$
$$\longrightarrow V \geq n \wedge V[\overline{mod} \mapsto \overline{mod}_{old}] > V$$

6. Array access are legal

$$I[\overline{mod} \mapsto \overline{mod}_{old}] \wedge cond[\overline{mod} \mapsto \overline{mod}_{old}] \longrightarrow 0 \leq x \leq a.length$$

   for any array a and access x (i_{old} or i)

7. No integer overflows — Assume perfect machine