Applications of Logic

Aritra Hazra

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur,
Paschim Medinipur, West Bengal, India - 721302.

Email: *aritrah@cse.iitkgp.ac.in*

Autumn 2020

Program Behavior Analysis: An Application of Logic

Program – Variety of Implementations

"Computing Quotient and Remainder while dividing one integer with another"
(Two Different Prog.)

```
readInt x;  readInt y;
q = 0;  r = x;
loop until y < r, do:
  r = r - y; q = q + 1;
output q;  output r;
```

```
readInt x;  readInt y;
q = 0;  t = 0;
loop until x>=(t+y), do:
  t = t + y; q = q + 1;
r = x - t;
output q;  output r;
```

Assertions (Specifications)

Input Assertion: $\mathtt{I}$: $[y \neq 0]$ (disallow non-zero divisor)

Predicate: $[\neg \mathrm{eq}(y, 0)]$ (Pre-Condition)

Output Assertion: $\mathtt{O}$: $[(x = q * y + r) \wedge (r < y)]$ (verify correct program behavior)

Predicate: $[\mathrm{eq}(x, \mathtt{sum}(\mathtt{mult}(q, y), r)) \wedge \mathrm{gt}(y, r)]$ (Post-Condition)

Program Requirement

$$\forall x \; \forall y \; \exists q \; \exists r \; \left([\, \neg \mathrm{eq}(y, 0) \,] \overset{\mathtt{Program}}{\rightsquigarrow} [\, \mathrm{eq}(x, \mathtt{sum}(\mathtt{mult}(q, y), r)) \wedge \mathrm{gt}(y, r) \,] \right)$$

Simple Programs: *Assignments and Operations*

Program (Swapping) and Input/Output Assertions

```
0. readInt x; readInt y;     // Input Assertion, I: True
1. x = x - y;
2. y = x + y;
3. x = y - x;
4. output x; output y;       // Output Assertion, O: (x'=y) & (y'=x)
```

Program Requirement: $\forall \mathtt{x}\ \forall \mathtt{y}\ \exists \mathtt{x}'\ \exists \mathtt{y}'\ \big(\ [\mathtt{True}] \rightsquigarrow [(\mathtt{x}' = \mathtt{y}) \wedge (\mathtt{y}' = \mathtt{x})]\ \big)$

Program Execution

Steps	Normal Execution		Symbolic Execution	
0.	inputs: $\mathtt{x} = 3$,	$\mathtt{y} = 1$	inputs: $\mathtt{x} = \alpha$,	$\mathtt{y} = \beta$
1.	$\mathtt{x} = 3 - 1 = 2$,	$\mathtt{y} = 1$	$\mathtt{x} = \alpha - \beta$,	$\mathtt{y} = \beta$
2.	$\mathtt{x} = 2$,	$\mathtt{y} = 2 + 1 = 3$	$\mathtt{x} = \alpha - \beta$,	$\mathtt{y} = \alpha - \beta + \beta = \alpha$
3.	$\mathtt{x} = 3 - 2 = 1$,	$\mathtt{y} = 3$	$\mathtt{x} = \alpha - \alpha + \beta = \beta$,	$\mathtt{y} = \alpha$
4.	$\mathtt{x} = 1$,	$\mathtt{y} = 3$	$\mathtt{x} = \beta$,	$\mathtt{y} = \alpha$

Formal Program Analysis

Assume that, the initial value of x and y are α and β, respectively.

$\mathtt{VC(I - O)}$: $[\mathtt{True}] \rightsquigarrow [(((\alpha - \beta) + \beta) - (\alpha - \beta) = \beta) \wedge ((\alpha - \beta) + \beta = \alpha)]$

Simple Programs: *Conditional Branching*

Program (Conditional-Swapping) and Input/Output Assertions

```
0. readInt x; readInt y;  // I: True
1. if x > y, do:          // (Branching-Condition) C: (x>y)
2.     t = x; x = y; y = t;
3. output x; output y;  // O: [(x<=y)&((x'=x)&(y'=y))]V[(x'=y)&(y'=x)]
```

Program Requirement

$$\forall x\ \forall y\ \exists x'\ \exists y'\ \left(\ [\texttt{True}] \rightsquigarrow [((x \leq y) \wedge ((x' = x) \wedge (y' = y))) \vee ((x' = y) \wedge (y' = x))]\ \right)$$

Formal Program Analysis

Assume that, the initial values of x and y are α and β, respectively.

$\texttt{VC}(\texttt{I} - \texttt{C}[\texttt{T}] - \texttt{O})$: $\texttt{I} \wedge \texttt{C} \rightsquigarrow \texttt{O} \quad \equiv \quad [\text{True} \wedge (\alpha > \beta)] \rightsquigarrow$
$[((\alpha \leq \beta) \wedge ((\beta = \alpha) \wedge (\alpha = \beta))) \vee ((\beta = \beta) \wedge (\alpha = \alpha))]$

$\texttt{VC}(\texttt{I} - \texttt{C}[\texttt{F}] - \texttt{O})$: $\texttt{I} \wedge \neg\texttt{C} \rightsquigarrow \texttt{O} \quad \equiv \quad [\text{True} \wedge \neg(\alpha > \beta)] \rightsquigarrow$
$[((\alpha \leq \beta) \wedge ((\alpha = \alpha) \wedge (\beta = \beta))) \vee ((\alpha = \beta) \wedge (\beta = \alpha))]$

Simple Programs: *Looping / Iterations*

Program (Factorial) and Input/Output Assertions

```
0. readInt n;                          // I: (n>=0)
1. i = 0;   f = 1;
2. loop until i < n, do:               // (Loop-Invariant) L: (f=i!)&(i<=n)
3.      i = i + 1;   f = f * i;        // (Loop-Condition) C: (i<n)
4. output f;                           // O: (f=n!)
```

Program Requirement

$\forall \mathtt{n}\ \exists \mathtt{f}\ \big((\mathtt{n} \geq 0) \rightsquigarrow (\mathtt{f} = \mathtt{n}!) \big)$

Formal Program Analysis

Assume that, the initial value of `n` is γ; the current values of `i` and `f` are α and β (resp.).

$$\mathtt{VC(I-L)}:\ \mathtt{I} \rightsquigarrow \mathtt{L} \quad \equiv \quad [(\gamma \geq 0)] \rightsquigarrow [(1 = 0!) \wedge (0 \leq \gamma)]$$

$$\mathtt{VC(L-C[T]-L)}:\ \mathtt{L} \wedge \mathtt{C} \rightsquigarrow \mathtt{L} \quad \equiv \quad [((\beta = \alpha!) \wedge (\alpha \leq \gamma)) \wedge (\alpha < \gamma)] \rightsquigarrow [(\beta = (\alpha+1)\beta = (\alpha+1)!) \wedge ((\alpha+1) \leq \gamma)]$$

$$\mathtt{VC(L-C[F]-O)}:\ \mathtt{L} \wedge \neg\mathtt{C} \rightsquigarrow \mathtt{O} \quad \equiv \quad [((\beta = \alpha!) \wedge (\alpha \leq \gamma)) \wedge \neg(\alpha < \gamma)] \rightsquigarrow [\beta = \gamma!]$$

Simple Programs: *Arrays and Indexing*

Program (Minimum-Element-Location) and Input/Output Assertions

```
0. readInt n;  readArray A[1..n];  // I: (n>=1)
1. i = 1;  p = 1;
2. loop until i < n, do:  // L: (1<=i<=n)&(1<=p<=i)&(A[p]<=A[1],..,A[i])
3.     i = i + 1;                  // C1: (i<n)
4.     if A[i] < A[p], then p = i; // C2: (A[i]<A[p])
5. output p;                       // O: (1<=p<=n)&(A[p]<=A[1],..,A[n])
```

Program Requirement:

$$\forall \mathtt{n}\ \forall \mathtt{A}_1\ \cdots\ \forall \mathtt{A}_n\ \exists \mathtt{p}\ \exists \mathtt{A}_p \Big((\mathtt{n} \geq 1) \rightsquigarrow ((1 \leq \mathtt{p} \leq \mathtt{n}) \wedge ((\mathtt{A}_p \leq \mathtt{A}_1) \wedge \cdots \wedge (\mathtt{A}_p \leq \mathtt{A}_n))) \Big)$$

Formal Program Analysis

Assume that, the initial value of $\mathtt{n}$ is δ and the values of each $\mathtt{A}_k$ is α_k $(1 \leq \mathtt{k} \leq \mathtt{n})$; the current values of $\mathtt{i}$ and $\mathtt{p}$ are β and γ (resp.).

$$\mathtt{VC(I - L)}:\ \mathtt{I} \rightsquigarrow \mathtt{L} \quad \equiv \quad [(\delta \geq 1)] \rightsquigarrow [(1 \leq 1 \leq \delta) \wedge (1 \leq 1 \leq 1) \wedge (\alpha_1 \leq \alpha_1)]$$

$$\mathtt{VC(L - C1[T] - C2[T] - L)}:\ \mathtt{L} \wedge \mathtt{C1} \wedge \mathtt{C2} \rightsquigarrow \mathtt{L} \quad \equiv$$
$$[((1 \leq \beta \leq \delta) \wedge (1 \leq \gamma \leq \beta) \wedge (\alpha_\gamma \leq \alpha_1, \cdots, \alpha_\beta)) \wedge (\beta < \delta) \wedge (\alpha_{\beta+1} < \alpha_\gamma)]$$
$$\rightsquigarrow [(1 \leq \beta + 1 \leq \delta) \wedge (1 \leq \beta + 1 \leq \beta + 1) \wedge (\alpha_{\beta+1} \leq \alpha_1, \cdots, \alpha_{\beta+1})]$$

Simple Programs: *Arrays and Indexing*

Program (Minimum-Element-Location) and Input/Output Assertions

```
0. readInt n;  readArray A[1..n];  // I: (n>=1)
1. i = 1;   p = 1;
2. loop until i < n, do:  // L: (1<=i<=n)&(1<=p<=i)&(A[p]<=A[1],..,A[i])
3.      i = i + 1;                 // C1: (i<n)
4.      if A[i] < A[p], then p = i; // C2: (A[i]<A[p])
5. output p;                       // O: (1<=p<=n)&(A[p]<=A[1],..,A[n])
```

Program Requirement:

$$\forall n\ \forall A_1\ \cdots\ \forall A_n\ \exists p\ \exists A_p \Big((n \geq 1) \rightsquigarrow ((1 \leq p \leq n) \wedge ((A_p \leq A_1) \wedge \cdots \wedge (A_p \leq A_n))) \Big)$$

Formal Program Analysis

$\mathtt{VC(L - C1[T] - C2[F] - L)}$: $\mathtt{L} \wedge \mathtt{C1} \wedge \neg\mathtt{C2} \rightsquigarrow \mathtt{L} \quad \equiv$

$$[((1 \leq \beta \leq \delta) \wedge (1 \leq \gamma \leq \beta) \wedge (\alpha_\gamma \leq \alpha_1, \cdots, \alpha_\beta)) \wedge (\beta < \delta) \wedge \neg(\alpha_{\beta+1} < \alpha_\gamma)]$$
$$\rightsquigarrow [(1 \leq \beta + 1 \leq \delta) \wedge (1 \leq \gamma \leq \beta + 1) \wedge (\alpha_\gamma \leq \alpha_1, \cdots, \alpha_{\beta+1})]$$

$\mathtt{VC(L - C1[F] - O)}$: $\mathtt{L} \wedge \neg\mathtt{C1} \rightsquigarrow \mathtt{O} \quad \equiv$

$$[((1 \leq \beta \leq \delta) \wedge (1 \leq \gamma \leq \beta) \wedge (\alpha_\gamma \leq \alpha_1, \cdots, \alpha_\beta)) \wedge \neg(\beta < \delta)]$$
$$\rightsquigarrow [(1 \leq \gamma \leq \delta) \wedge (\alpha_\gamma \leq \alpha_1, \cdots, \alpha_\delta)]$$

Mathematical Logic: A Summary

- **Propositional Logic:**
  - Logical formula constructed with Boolean propositions and connectors
  - Interpretations over finite combinations of truth values of propositions (using truth-table or deduction rules)
  - Notions of validity and satisfiability
- **First-Order Predicate Logic:**
  - Addition of quantifiers, predicates and functions to propositional logic
  - Interpretations using universal generalization and specification rules
  - Complete expressibility power of computable and logical functionalities
- **Limitations and Extensions:**
  - Higher-order logic allowing quantification of predicates and functions
  - Temporal logic to express relationship among different time worlds
  - Unsolvable and Undecidable computation cannot be modeled
- **Applications:**
  - Problem solving, goal finding and cause-effect analysis
  - Program behavior analysis and correctness checking
  - **Many more ...** *Tell me if you find anything or apply logic anywhere!*

Thank You!