Introduction to Software Verification

Orna Grumberg

Lectures Material
winter 2017-18

Lecture 2

Part 1 of the course

Program Correctness

- Non-automated
- Verifies program with possibly infinite number of states
- Refers to the programs as input-output transformation

Ingredients for Formal Verification

1. Specification language
   - With formal semantics
2. Programming language
   - with formal semantics
3. Proof rules
   - For proving "Program P has the property φ"

More notations

- $\perp$ - bottom : the undefined value
- **val(π)** denotes the final state of computation π (if exists)
  - val(π) = σ_k if $\pi = (\sigma_1, \ldots, \sigma_k)$
  - val(π) = $\perp$ if $\pi = (\sigma_1, \sigma_2, \ldots)$
    - π is an infinite computation
- $\sigma \models \mathbf{q(\bar{x})}$ if $q(\bar{x})$ is true when free variables in q are replaced with matching values in σ

Partial Correctness

- For every computation π and every state σ_0 :

$$(\sigma_0 \models q_1(\bar{x}) \text{ and } val(\pi(P, \sigma_0)) \neq \perp) \Rightarrow$$
$$val(\pi(P, \sigma_0)) \models q_2(\bar{x})$$

- Notation: $\{q_1\}P\{q_2\}$

Total Correctness

- For every computation π and every state σ_0 :

$$\sigma_0 \models q_1(\bar{x}) \Rightarrow$$
$$\text{val}(\pi(P, \sigma_0)) \models q_2(\bar{x})$$

- Notation: $\mathbf{<q_1>P<q_2>}$

Logical Variables in Specifications

Add fresh variables which are

- not part of the program and therefore
- their value does not change during the execution of the program

Convention:

Use logical variable X to preserve the value of variable x

Assertions q_1 ,q_2 are defined over $\bar{x}$ that includes program variables as well as logical variables

Ingredients for Formal Verification

1. Specification language
   - With formal semantics

2. Programming language
   - with formal semantics

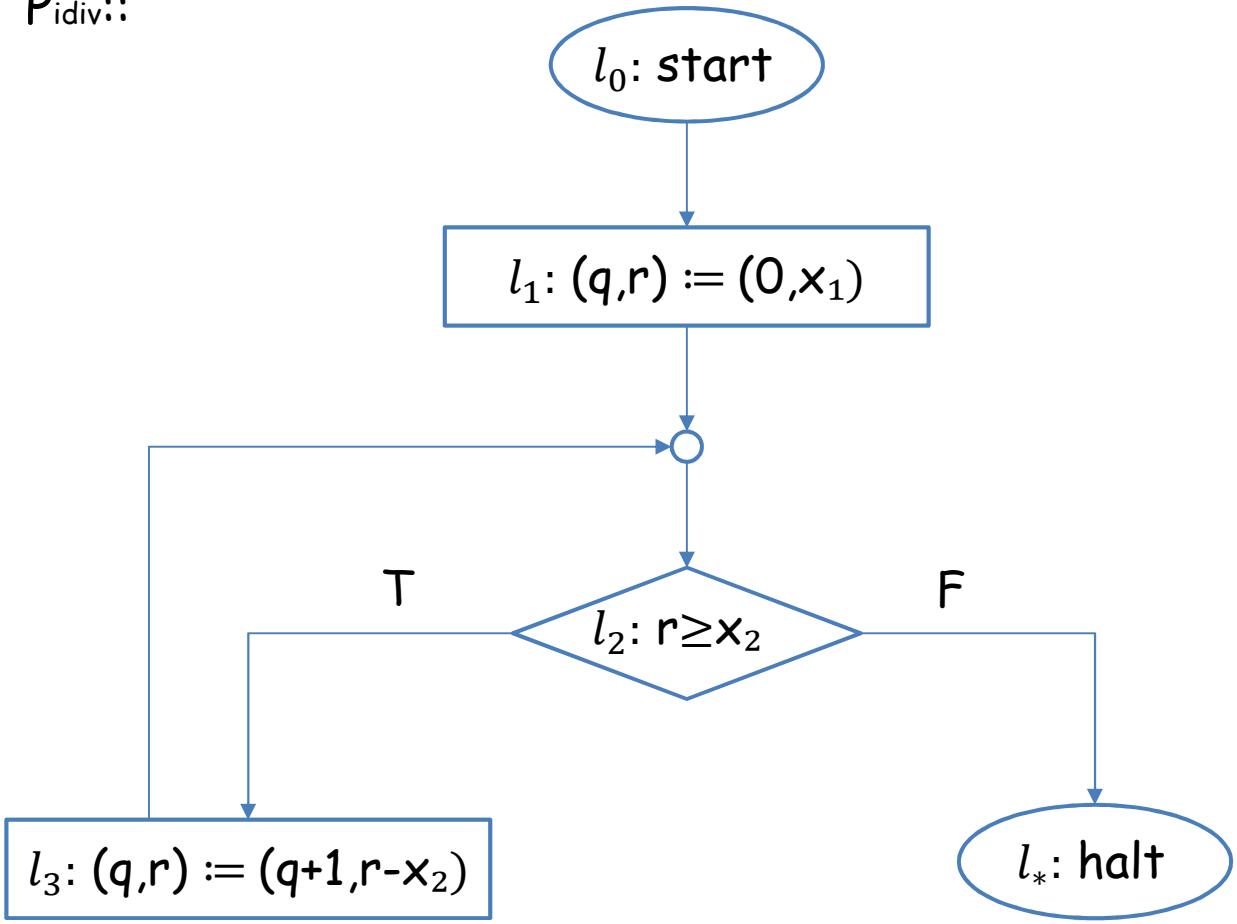
3. Proof rules
   - For proving "Program P has the property φ"

"Programming language"

- Program are described as flowcharts
- The "programming language" is called PLF

Flowchart: Example

P_{idiv}::



Flowcharts: Syntax

Commands

1. *Start – program entry point*

2. Assignment – $\bar{x}:=\bar{e}$, where $\bar{x}$ is a list of pairwise distinct program variables, and $\bar{e}$ is a list (of the same length as $\bar{x}$) of expressions over program variables

   E.g (x,y):=(y-1,x·y)

3. Test – a Boolean expression $B(\bar{x})$ over the program variables

4. Halt – an exit point of the program

Each command has a distinct label $l \in L_P$, where L_P is a finite set of labels for program P

Flowcharts: Syntax (cont.)

A program $P \in PLF$ is a finite directed graph with labeled commands as nodes, satisfying the following restrictions:

- A **start** node is unique, has no predecessors and only one successor
- A **halt** node has no successors
- An **assignment** node has exactly one successor

Flowcharts: Syntax (cont.)

- A **test** node has two successors, with two labeled edges, one with **T** and the other with **F**
- Every node resides on a directed path from a start node to a halt node
- Different nodes have different labels, where
  - l_0 labels the start node
  - l_* labels the halt node

Paths in flowcharts

- A **path** in a flowchart program P is a sequence of consecutive nodes in the graph
- A path is **full** if it begins with a start node, and ends with a halt node
- A path is **maximal** if it is full or infinite

Operational semantics of flowcharts

- σ - program state
  - assignment for all program variables
- Configuration of a program is a pair $C=\langle l,\sigma\rangle$ such that
  - $l \in L_P$ is a flowchart label (value of the program counter) and
  - σ is a state

Operational semantics of flowcharts

- A configuration is **halting** if $l = l_*$
- A configuration is **initial** if $l = l_0$, denoted C_0

Relation $\rightarrow$ Over Configurations

$\langle l,\sigma\rangle \rightarrow \langle l',\sigma'\rangle$ iff the node labeled l' is a successor of the node labeled l, and one of the following holds:

- l is the start node and $\sigma' = \sigma$
- l is a node with test $B(\bar{x})$, $\sigma \models B(\bar{x})$, the edge from l to l' is labeled with T, and $\sigma' = \sigma$
- l is a node with test $B(\bar{x})$, $\sigma \models \neg B(\bar{x})$, the edge from l to l' is labeled with F, and $\sigma' = \sigma$

Relation $\rightarrow$ Over Configurations

- l is an assignment $\bar{x} := \bar{e}$ and $\sigma' = \sigma[x \leftarrow \sigma(e)]$

Explanation:

Let $\bar{x} = (x_1, \ldots, x_n)$, $\bar{e} = (e_1, \ldots, e_n)$ then

- $\sigma'(y) = \sigma(y)$ for $y \neq x_i$
- $\sigma'(x_i) = \sigma(e_i)$

Example: $(x,y) := (x+1, y \cdot x)$

$\sigma(x)=2$, $\sigma(y)=5$, $\sigma(z)=7$

$\sigma'(x)=3$, $\sigma'(y)=10$, $\sigma'(z)=7$

Relation $\rightarrow$ Over Configurations

- $\xrightarrow{*}$ marks the transitive closure of $\rightarrow$
- Computation from configuration C is denoted $\pi(C)$
- $\pi(C) = C_0, C_1, \ldots$ is a maximal sequence of configurations such that $C=C_0$ and for every $i \geqslant 0$, $C_i \rightarrow C_{i+1}$
- If $\pi(C)$ is finite, it ends with a halting configuration

Definition of val(π) - revisited

- val($\pi(C)$) =

  σ' if $C \xrightarrow{*} (l_*, \sigma')$

  $\perp$ if $\pi(C)$ is infinite

- The **meaning** or **semantics** of a program P is:

  M[P](σ) = val(π(P,σ))

Ingredients for Formal Verification

1. Specification language
   - With formal semantics

2. Programming language
   - with formal semantics

3. Proof rules
   - For proving "Program P has the property φ"

In order to define the proof method we need the following definitions

- Let τ be a finite path in P, we associate with τ two semantic characteristics

Semantic characteristics

- *A* **reachability condition** is a condition on states at the beginning of τ that guarantees that control will traverse τ.

- It is denoted by $\boldsymbol{R_\tau(\overline{x})}$

Semantic characteristics

- A **state transformation** is a sequence of expressions over $\bar{x}$, describing for each x_i, its value at the end of τ
- It is denoted by $T_\tau(\bar{x})$

Computing $R_\tau(\bar{x})$ and $T_\tau(\bar{x})$

Let $\tau = l_{i_0}, \ldots, l_{i_k}$ be a finite path in P.

Starting from m=k down to m=0:

Initially

$T_\tau^k(\bar{x}) = (\bar{x})$ $\qquad$ $R_\tau^k(\bar{x}) = true$

Computing $R_\tau(\bar{x})$ and $T_\tau(\bar{x})$

Given $R_\tau^{m+1}(\bar{x})$ and $T_\tau^{m+1}(\bar{x})$

1. For **start** in l_{i_m}

   $T_\tau^m(\bar{x}) = T_\tau^{m+1}(\bar{x})$ $\quad$ $R_\tau^m(\bar{x}) = R_\tau^{m+1}(\bar{x})$

   Type equation here.

2. For $\bar{y} := \bar{e}$ in l_{i_m}

   $$T_\tau^m(\bar{x}) = T_\tau^{m+1}(\bar{x})[\bar{y} \leftarrow \bar{e}]$$
   $$B_\tau^m(\bar{x}) = B_\tau^{m+1}(\bar{x})[\bar{y} \leftarrow \bar{e}]$$

Computing $R_\tau(\bar{x})$ and $T_\tau(\bar{x})$

Given $R_\tau^{m+1}(\bar{x})$ and $T_\tau^{m+1}(\bar{x})$

3. For $\boldsymbol{B(\bar{x})}$ in l_{i_m}

$$T_\tau^m(\bar{x}) = T_\tau^{m+1}(\bar{x})$$

- If $l_{i_{m+1}}$ is the T-son of l_{i_m} $R_\tau^m(\bar{x}) = R_\tau^{m+1}(\bar{x}) \wedge \boldsymbol{B(\bar{x})}$
- If $l_{i_{m+1}}$ is the F-son of l_{i_m} $R_\tau^m(\bar{x}) = R_\tau^{m+1}(\bar{x}) \wedge \neg \boldsymbol{B(\bar{x})}$

Semantic characteristics: example

Let the initial state be: $\sigma_0(x) = 2,\ \sigma_0(y) = 3.$

If $T_\tau(x, y) = (y + 1, x)$

then

$$\sigma_k(x) = \sigma_0(y + 1) = 4,\ \sigma_k(y) = \sigma_0(x) = 2$$

If $R_\tau(\bar{x})$ = x> y

Then, since 2>3 is false, the path τ will not be traversed from the initial state

Semantic characteristics

Lemma:

For a finite path $\tau = l_{i_0}, \ldots, l_{i_k}$ in P and a state σ_0,

- τ is traversed from σ_0 iff $\sigma_0 \vDash R_\tau(\bar{x})$
- If the path ends at l_{i_k} with state σ_k then $\sigma_k = \sigma_0[\bar{x} \leftarrow T_\tau(\bar{x})]$

Floyd Proof Rule for Partial Correctness

To prove $\{q_1\}P\{q_2\}$:

1. Choose a set of cut points such that:
   i. start and halt are cut points
   ii. every cycle in the graph of P contains at least one cut point

2. For every cut point l find an inductive assertion $I_l(\bar{x})$, such that $I_{l_0}(\bar{x}) = q_1(\bar{x})$, $I_{l_*}(\bar{x}) = q_2(\bar{x})$

A **basic path** (l, l') is a path from l to l' such that l, l' are cut points, and those are the only cut points in the path

Floyd Proof Rule for Partial Correctness (cont.)

3. For every basic path $\alpha = (l, l')$ prove:
   $\forall \bar{x}[I_l(\bar{x}) \wedge R_\alpha(\bar{x}) \rightarrow I_{l'}(T_\alpha(\bar{x}))]$

If we successfully applied the proof rule for some invariants we will write

$\vdash_F \{q_1\}P\{q_2\}$

Floyd Proof Rule for Partial Correctness (cont.)

Remark:

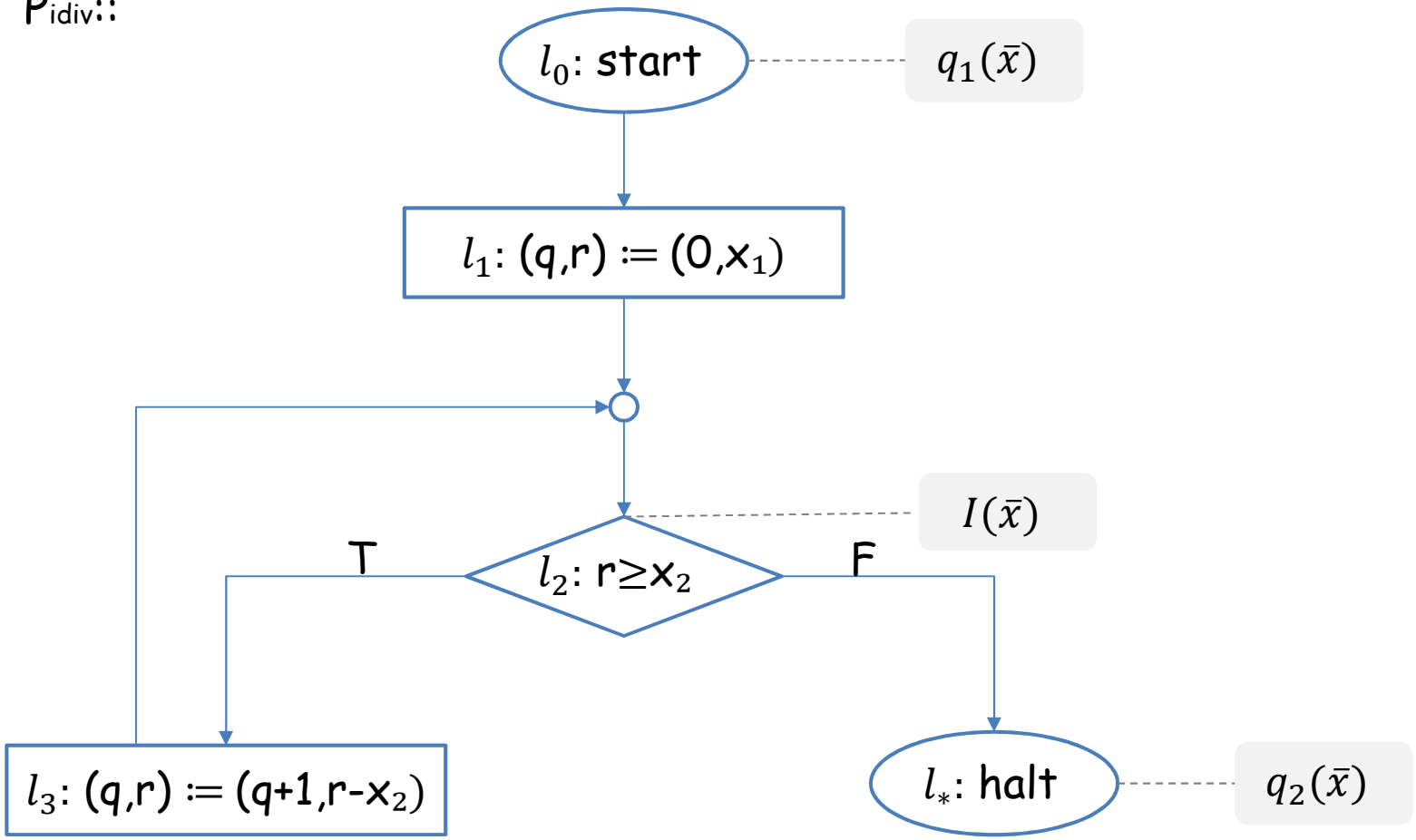
We have reduced a proof of $\{q_1\}P\{q_2\}$ to a proof in first order logic.

We will assume that every assertion is provable iff it is true (in $\mathbb{Z}$)

- We have an oracle of all true FOL statements

Proof: Example

P_{idiv}::



Proof: Example

We need to show:

$$q_1(\bar{x}) \wedge R_{l_0 l_2}(\bar{x}) \rightarrow I\left(T_{l_0 l_2}(\bar{x})\right)$$
$$I(\bar{x}) \wedge R_{l_2 l_2}(\bar{x}) \rightarrow I\left(T_{l_2 l_2}(\bar{x})\right)$$
$$I(\bar{x}) \wedge R_{l_2 l_*}(\bar{x}) \rightarrow q_2\left(T_{l_2 l_*}(\bar{x})\right)$$

We choose

$$\boldsymbol{I}(x_1, x_2, q, r, X_1, X_2) =$$
$$(x_1 = X_1) \wedge (x_2 = X_2) \wedge (X_1 = qX_2 + r) \wedge r \geq 0$$

We assume all variables are in $\mathbb{Z}$

Proof: Example

Where,

$q_1(x_1, x_2, q, r, X_1, X_2) =$

$$(x_1 = X_1) \wedge (x_2 = X_2) \wedge x_1 \geq 0 \wedge x_2 > 0$$

$q_2(x_1, x_2, q, r, X_1, X_2) =$

$$(X_1 = qX_2 + r) \wedge 0 \leq r < X_2$$

We assume all variables are in $\mathbb{Z}$

Proof: Example

For example for the basic path $(l_2, l_2) = l_2, l_3, l_2$ we have:

$$R_{l_2 l_2}(\bar{x}) = \boldsymbol{r \geq x_2}$$

$$T_{l_2 l_2}(x_1, x_2, q, r, X_1, X_2) = (x_1, x_2, \boldsymbol{q + 1}, \boldsymbol{r - x_2}, X_1, X_2)$$

So we need to prove:

$$(x_1 = X_1 \wedge x_2 = X_2 \wedge X_1 = qX_2 + r \wedge r \geq 0) \wedge \boldsymbol{r \geq x_2}$$

$$\rightarrow \begin{pmatrix} x_1 = X_1 \wedge x_2 = X_2 \wedge \\ X_1 = (\boldsymbol{q + 1})X_2 + (\boldsymbol{r - x_2}) \wedge (\boldsymbol{r - x_2}) \geq 0 \end{pmatrix}$$

Floyd Proof Rule for Partial Correctness

Soundness of Floyd proof system (F):

If $\vdash_F \{ q_1 \} P \{ q_2 \}$

then $\models \{ q_1 \} P \{ q_2 \}$