Introduction to Software Verification

Orna Grumberg

Lectures Material
winter 2017-18
Lecture 3
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 \( I_{l_0}(\bar{x}) = q_1(\bar{x}) \), \( I_{l^*}(\bar{x}) = q_2(\bar{x}) \)
Floyd Proof Rule for Partial Correctness (cont.)

3. For every basic path $\alpha = (l, l')$ prove:

$$\forall \bar{x} [I_l(\bar{x}) \land R_{\alpha}(\bar{x}) \rightarrow I_{l'}(T_{\alpha}(\bar{x}))]$$

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

**Soundness** of Floyd proof system $(F)$:

If $\vdash_F \{ q_1 \} P \{ q_2 \}$
then $\models \{ q_1 \} P \{ q_2 \}$
Floyd Proof Rule for Partial Correctness

Lemma:
If $\vdash F \{q_1\} P \{q_2\}$ then for every computation $\pi$ of $P$ from $l_0$ with state $\sigma$ such that $\sigma \models q_1(\bar{x})$ if the computation reaches cut point $l'$ with state $\sigma'$ then $\sigma' \models I_{l'}(\bar{x})$

Proof:
By induction on the number of cut points traversed in $\pi$
Floyd Proof Rule for Partial Correctness

Completeness of the proof system $F$:

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

We will not prove this.
Floyd Proof Rule for Partial Correctness (cont.)

If we change the requirement

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

To
   $\forall \bar{x}[I_l(\bar{x}) \rightarrow I_{l'}(T_\alpha(\bar{x}))]$

Will the new rule be sound? Complete?
Floyd’s significance

Floyd suggested the use of

• $R$ and $T$
  – Nowadays used for automated software verification, static analysis

• Invariants
  – Evolved into assertions

• There exist extensions to procedures, arrays, parallel programs
F* Proof Rule for Proving Termination (full correctness)

We would like to prove \( <p> S <q> \)

Example:
Flowchart: Example

\[ P_{\text{div}}:: \]

\[ l_0: \text{start} \]

\[ l_1: (q,r) := (0,x_1) \]

\[ l_2: r \geq x_2 \]

\[ l_3: (q,r) := (q+1,r-x_2) \]

\[ l_*: \text{halt} \]

\[ l_*: \text{halt} \]

\[ l_0: \text{start} \]

\[ l_1: (q,r) := (0,x_1) \]

\[ l_2: r \geq x_2 \]

\[ l_3: (q,r) := (q+1,r-x_2) \]

\[ l_*: \text{halt} \]
Well Founded Sets

A set $W$ with a (possibly partial) order $<$ on $(W,<)$ is a well founded set if there is no infinitely decreasing sequences in $W$. That is, there is no sequence $w_i \in W$ such that:

$$w_0 > w_1 > w_2 > ...$$
Well Founded Sets - Examples

The partially ordered set \((2^A, \subseteq)\) for \(A=\{1,2\}\)
Well Founded Sets - Examples

• Naturals with the usual order \( \langle N, \langle \rangle \) is a well founded set
• Integers with the usual order \( \langle \) is not well founded
• Positive rational numbers with the usual order \( \langle \) is not well founded
• \( (2^A, \subseteq) \) for any finite \( A \) is well founded
• \( (2^A, \subseteq) \) for an infinite \( A \) is not well founded
• \( N \times N \) with the lexicographical order is a well founded set
F* Proof System for Proving Termination (full correctness)

To prove $<q_1> P <true>$:
1. Choose $(W,\prec)$ to be $(\mathbb{N},\prec)$ with the usual order
2. Choose a cut set as in F
3. For every cut point $l$ find a parameterized inductive assertion $I_l(\bar{x},w)$
   where $w$ is a variable over domain $W$
4. Prove (in First order logic):

- **(INIT)** \( \forall \bar{x} \left[ q_1(\bar{x}) \rightarrow \exists w \left( I_{l_0}(\bar{x}, w) \right) \right] \)

- **(DEC)** For every basic path \( \alpha = (l, l') \) prove:

\[
\forall w \forall \bar{x} \left[ I_l(\bar{x}, w) \land R_{\alpha}(\bar{x}) \rightarrow \exists w' \left( w' < w \land I_{l'}(T_{\alpha}(\bar{x}), w') \right) \right]
\]
If we successfully applied the proof rule for some invariants we will denote

\[ \vdash_{F^*} < q_1 > P < \text{true} > \]
F* Proof System for Proving Termination (full correctness)

To prove $<q_1>P<q_2>$ we need to prove in addition in First order logic:

$$\forall w \forall \bar{x} \left[ I_{l_*}(\bar{x}, w) \rightarrow q_2(\bar{x}) \right]$$
**F* Proof System for Proving Termination (full correctness)**

**Soundness** of the proof system F*:  

If \( \vdash_{F^*} < q_1 > P < q_2 > \)  

then \( \models < q_1 > P < q_2 > \)
Lemma:

If $\vdash_{F^*} <q_1> P<\text{true}>$ then for every computation $\pi$ of $P$ from $l_0$ with state $\sigma$ such that $\sigma \models q_1(\bar{x})$ if the computation reaches cut point $l'$ with state $\sigma'$ then there is $v \in W$ such that $\sigma' \models I_{l'}(\bar{x}, v)$

In addition, if the computation passes through cut points $l_0, l_1, \ldots$ with states $\sigma_0, \sigma_1, \ldots$ then there exists a sequence $v_0 > v_1 > \ldots$ such that for every $i$, $\sigma_i \models I_{l_i}(\bar{x}, v_i)$
F* Proof System for Proving Termination (full correctness)

Completeness of the proof system F*:

If $\equiv < q_1 > P < q_2 >$
then $\vdash_{F*} < q_1 > P < q_2 >$
F* Proof System for Proving Termination (full correctness)

Completeness proof sketch for the termination rule for:

\[
\text{If } \vdash < q_1 > P \langle \text{true} \rangle \\
\text{then } \vdash_{F^*} < q_1 > P \langle \text{true} \rangle
\]

Full completeness proof in [Francez, program verification]
Model Checking

Automated formal verification:

A different approach to formal verification
Formal Verification

Given

• a model of a (hardware or software) system and
• a formal specification

does the system model satisfy the specification?

Not decidable!

To enable automation, we restrict the problem to a decidable one:

• **Finite-state** reactive systems
• **Propositional** temporal logics
Properties in Propositional Temporal Logic - Examples

- **mutual exclusion:**
  \[\text{always } \neg (cs_1 \land cs_2)\]

- **non starvation:**
  \[\text{always } (\text{request} \Rightarrow \text{eventually granted})\]

- **communication protocols:**
  \[\neg \text{get-message} \text{ until } \text{send-message}\]
Finite State Systems - Examples

- Hardware designs
- Controllers (elevator, traffic-light)
- Communication protocols (when ignoring the message content)
- High level (abstracted) description of non finite state systems