Rigorous Methods for Software Engineering (F21RS-F20RS) Program Verification Part 2: Theorem Proving

Andrew Ireland
Department of Computer Science
School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh

Overview

- Logical arguments and proofs.
- Constructing proofs for VCs.

Logic and Validity of Arguments

- Logic is concerned with identifying valid arguments.
- Argument = Hypotheses + Conclusion
- Hypotheses are said to support the conclusion, *e.g.*
  $$\text{“} \underbrace{\ldots\ldots}_{\textbf{hypotheses}} \textbf{ therefore } \underbrace{\ldots\ldots}_{\textbf{conclusion}} \text{”}$$
- Both hypotheses and conclusion are denoted by **statements** which have an associated truth value, *i.e.* *true* or *false*.
- An **argument is valid** if the conclusion is true whenever all the hypotheses are true.
- A **theorem** is the name given to a valid argument.

Arguments and Verification Conditions

- When we specify the partial correctness of a program we are expressing a logical argument in terms of assertions and code.
- When we generate the set of verification conditions for a program specification we are expressing the logical argument purely in terms of logical formulae, *i.e.*

$$\underbrace{Hypothesis_1, \ldots Hypothesis_n}_{\text{Givens}} \rightarrow \underbrace{Conclusion}_{\text{Goal}}$$

- When determining the validity of a verification condition (argument) we will refer to the hypotheses as the **givens** and the conclusion as our **goal**.

Validity via Proof

- A **formal proof** is a sequence of statements each of which corresponds to i) a previously proved **theorem** or ii) an instance of an **axiom** or iii) follows from earlier statements by a **proof rule**:

$$A < B \leftrightarrow \neg(B \leq A)$$

theorem

$$A = A$$

reflexivity axiom

$$\frac{A = B \quad P(A)}{P(B)}$$

substitution proof rule

- Theorems will be implicitly universally quantified.
- Proof rules and axioms are **templates** which represent general purpose **chunks** of valid arguments.

Strategies for Proof Construction

- **Forwards proof:** apply proof rules to axioms and known theorems to derive new theorems until theorem-hood is established for the conjecture (VC).
- **Backwards proof:** start from the conjecture (VC) and apply proof rules backwards until you reach the level of axioms or known theorems.

"The grand thing is to be able to reason backwards."

Sir Arthur Conan Doyle, *A Study in Scarlet*

Backward Proof Construction

- **Tautologies:** goals that are always *true*, *e.g.* $P \vee \neg P$, can be replaced by *true*.
- **Axioms:** a goal that matches an axiom can be replaced by *true*, *e.g* the goal $N + 1 = N + 1$ can be replaced by *true* because it matches the reflexivity axiom.
- **Hypotheses:** a goal G (or subterm of G) can be replaced by *true* if it matches a given hypothesis.
- **Rewriting:** a subterm L of a goal G can be replaced by R if we know that L and R are equal (equivalent) or $R \to L$. Such knowledge comes from definitions, properties and givens.

Quotient-Remainder Specification

$$\{true\}$$

```
R:= X;
Q:= 0;
```

$$\{R = X \wedge Q = 0\}$$

```
while Y<=R loop
```

$$\{X = R + (Y * Q)\}$$

```
    R:= R - Y;
    Q:= Q + 1;
end loop;
```

$$\{X = R + (Y * Q) \wedge R < Y\}$$

Quotient-Remainder VCs

$$
\begin{aligned}
\mathit{true} &\rightarrow (X = X \wedge 0 = 0) \\
(R = X \wedge Q = 0) &\rightarrow (X = R + (Y * Q)) \\
(X = R + (Y * Q)) \wedge \neg(Y \leq R) &\rightarrow (X = R + (Y * Q) \wedge R < Y) \\
(X = R + (Y * Q)) \wedge Y \leq R &\rightarrow (X = (R - Y) + (Y * (Q + 1)))
\end{aligned}
$$

Definitions & Properties

$$
\begin{aligned}
A * 0 &= 0 && (1)\\
A * (B + 1) &= A + (A * B) && (2)\\
A + 0 &= A && (3)\\
A < B &\leftrightarrow \neg(B \leq A) && (4)\\
A + (B + C) &= (A + B) + C && (5)\\
A - B &= A + (-B) && (6)\\
(-A) + A &= 0 && (7)
\end{aligned}
$$

Proof of VC1

Givens:

Goal: $(X = X \wedge 0 = 0)$

by reflexivity

$\mathit{true} \wedge \mathit{true}$

by tautology

true

Proof of VC2

Givens: $R = X$

$Q = 0$

Goal: $X = R + (Y * \underline{Q})$

by given $Q = 0$

$X = \underline{R} + (Y * 0)$

by given $R = X$

$X = X + \underline{(Y * 0)}$

by (1) left-to-right

$X = \underline{X + 0}$

by (3) left-to-right

$X = X$

by reflexivity

true

Proof of VC3

Givens:

$$X = R + (Y * Q)$$
$$\neg(Y \leq R)$$

Goal:

$$\underline{X = R + (Y * Q)} \wedge R < Y$$

by given $X = R + (Y * Q)$

$$true \wedge \underline{R < Y}$$

by (4) left-to-right

$$true \wedge \underline{\neg(Y \leq R)}$$

by given $\neg(Y \leq R)$

$$true \wedge true$$

by tautology

$$true$$

Proof of VC4

Givens:

$$X = R + (Y * Q)$$
$$Y \leq R$$

Goal:

$$X = (R - Y) + (\underline{Y * (Q + 1)})$$

by (2) left-to-right

$$X = \underline{(R - Y) + (Y + (Y * Q))}$$

by (5) left-to-right

$$X = ((\underline{R - Y}) + Y) + (Y * Q)$$

by (6) left-to-right

$$X = ((R + (-Y)) + Y) + (Y * Q)$$

Proof of VC4 [more]

Givens:

$$X = R + (Y * Q)$$
$$Y \leq R$$

Goal:

$$\ldots$$

$$X = ((\underline{R + (-Y)) + Y}) + (Y * Q)$$

by (5) right-to-left

$$X = (R + ((\underline{-Y) + Y})) + (Y * Q)$$

by (7) left-to-right

$$X = (\underline{R + 0}) + (Y * Q)$$

by (3) left-to-right

$$\underline{X = R + (Y * Q)}$$

by given $X = R + (Y * Q)$

$$\mathit{true}$$

A Conditional Program Specification

Prove the following:

$$\{true\} \ \textbf{if} \ \texttt{even(N)} \ \ \textbf{then} \ \texttt{N:= N+1} \ \textbf{end if} \ \ \{odd(N)\}$$

given:

$$odd(X) \leftrightarrow \neg(even(X)) \tag{8}$$
$$even(X) \leftrightarrow odd(X+1) \tag{9}$$

VC Generation

(1) $\{true\}$ **if** `even(N)` **then** `N:= N+1` **end if** $\{odd(N)\}$

Apply if-then generation to (1) giving:

(2) $\{true \wedge even(N)\}$ `N:= N+1` $\{odd(N)\}$

and VC1:

$$true \wedge \neg(even(N)) \rightarrow odd(N)$$

Apply assignment generation to (2) giving VC2:

$$true \wedge even(N) \rightarrow odd(N+1)$$

Proof of VC1

Givens: $\neg(even(N))$

Goal:

$$\frac{\dfrac{odd(N)}{\neg(even(N))}\ \text{by (8) left-to-right}}{true}\ \text{by given } \neg(even(N))$$

Proof of VC2

Givens: $even(N)$

Goal:

$$\dfrac{\dfrac{odd(N+1)}{even(N)}\ \text{by (9) right-to-left}}{true}\ \text{by given } even(N)$$

Use of Conditional Rewrite Rules

Prove the following:

$\{N \geq 2\}$
if `even(N)` **then** `N:= N-2` **else** `N:= N-1` **end if**
$\{even(N)\}$

given:

$$
\begin{aligned}
(X = 0) \rightarrow (even(X) &\leftrightarrow true) && (10)\\
(X = 1) \rightarrow (even(X) &\leftrightarrow false) && (11)\\
(X > 1) \rightarrow (even(X) &\leftrightarrow even(X-2)) && (12)\\
(X > 0) \rightarrow (odd(X) &\leftrightarrow even(X-1)) && (13)\\
odd(X) &\leftrightarrow \neg(even(X)) && (14)\\
(X \geq 2) &\leftrightarrow (X > 1) && (15)\\
(X \geq 2) &\rightarrow (X > 0) && (16)
\end{aligned}
$$

VC Generation

- Then branch gives VC1:

$$((N \geq 2) \land even(N)) \rightarrow even(N-2)$$

- Else branch gives VC2:

$$((N \geq 2) \land \neg(even(N))) \rightarrow even(N-1)$$

Proof of VC1

Givens: $N \geq 2$
$even(N)$

Goal: $even(N - 2)$

given $N \geq 2$ infer $N > 1$ using (15)

Givens: $N \geq 2$
$even(N)$
$N > 1$

Goal: $even(N - 2)$

Proof of VC1 [more]

Givens: $N \geq 2$
$even(N)$
$N > 1$

Goal:

$$\frac{\dfrac{even(N-2)}{even(N)}\ \text{by (12)}}{true}\ \text{by given } even(N)$$

Note that applying a conditional property, *i.e.* $C \rightarrow (\ldots)$, involves proving that the condition C holds within the given context. As a consequence, we needed to infer $N > 1$ in order to apply (12).

Proof of VC2

Givens: $N \geq 2$
$\neg(even(N))$

Goal: $even(N-1)$

given $N \geq 2$ infer $N > 0$ using (16)

Givens: $N \geq 2$
$\neg(even(N))$
$N > 0$

Goal: $even(N-1)$

Proof of VC2 [more]

Givens: $N \geq 2$
$\neg(\mathit{even}(N))$
$N > 0$

Goal:

$$\frac{\frac{\frac{\mathit{even}(N-1)}{\mathit{odd}(N)}\ \text{by (13)}}{\neg(\mathit{even}(N))}\ \text{by (14)}}{\mathit{true}}\ \text{by given } \neg(\mathit{even}(N))$$

Note that $N > 0$ is required in order to apply (13).

Verification of Nested Conditionals

$\{true\}$

if not `(X = Y)` **then**
  `W:= X`
 else
  if `Y = Z` **then**
    `W:= Z`
   else
    `W:= Y`
  end if
end if

$\{W = X\}$

$$VC1: \quad \neg(X = Y) \rightarrow (X = X)$$
$$VC2: \quad \neg(\neg(X = Y)) \wedge (Y = Z) \rightarrow (Z = X)$$
$$VC3: \quad \neg(\neg(X = Y)) \wedge \neg(Y = Z) \rightarrow (Y = X)$$

Proof of VC2

Givens: $\neg(\neg(X = Y))$
$Y = Z$

Goal: $Z = X$

given $\neg(\neg(X = Y))$ infer $X = Y$ by $\neg(\neg(X)) \leftrightarrow X$

Givens: $\neg(\neg(X = Y))$
$Y = Z$
$X = Y$

Goal: $Z = X$

Proof of VC2 [more]

Givens: $\neg(\neg(X = Y))$

$Y = Z$

$X = Y$

Goal: $Z = \underline{X}$

given $X = Y$

$Z = Y$

by given $Y = Z$

true

$VC1$ and $VC3$ are left as an exercise to the reader.

Summary

Learning outcomes:

- Understand the notion of a valid logical argument.
- Understand the notion of formal proof.
- Be able to prove simple VCs using a backward style of proof based upon (conditional) rewriting.

Recommended reading:

- Dijkstra, E.W. *A Discipline of Programming*, Prentice-Hall, 1976.
- Gordon, M.J.C. *Programming Language Theory and its Implementation*, Prentice-Hall, 1988.
- Gries, D. *The Science of Programming*, Springer-Verlag, 1981.