Rules of correctness proof for programs with simple logic

Vladimir Shelekhov

A.P.Ershov Institute of Informatics Systems, Novosibirsk, Russia, vshel@iis.nsk.su

1. Program logic

For a program S(z) a program logic is the predicate L(S(z)).

L(S(z)) ⇔ ∃ execution of S finishing with the same values of z

A(x: y) ≅ the statement A with the arguments x and results y, x and y are variable sets.

The logics of the basic statements:

$$L(B(x{:}\ z);\ C(z{:}\ y)) \cong \exists z.\ L(B(x{:}\ z)) \ \&\ L(C(z{:}\ y)) \quad (1)$$

$$L(B(x{:}\ y)\ ||\ C(x{:}\ z)) \cong L(B(x{:}\ y)) \ \&\ L(C(x{:}\ z)) \quad (2)$$

$$L(\textbf{if}\ (E)\ B(x{:}\ y)\ \textbf{else}\ C(x{:}\ y)) \cong (E \Rightarrow L(B(x{:}\ y))) \ \&\ (\neg E \Rightarrow L(C(x{:}\ y))) \quad , \quad (3)$$

where E is an expression that depends on x, the variable sets x, y, and z are disjoint and the set x may be empty.

The constructs with simple logic are sufficient to represent any algorithm for problems in the discrete and computational mathematics but not for reactive systems consisted of interacting processes.

The **while** loops and pointers (but not **goto**) are harmful constructs; their program logic are complicated.

2. Program correctness

A *specification* of the program S(x: y) is [P(x), Q(x, y)], P(x) is a *precondition* and Q(x, y) is a *postcondition*.

The condition of *total correctness* of the program S(x: y) relative to the specification [P(x), Q(x, y)]:

$$Corr(S(x{:}\ y), P(x), Q(x, y)) \cong P(x) \Rightarrow [\ L(S(x{:}\ y)) \Rightarrow Q(x, y)\ ] \ \&\ \exists y.\ L(S(x{:}\ y)) \quad (6)$$

Single-valued program $S(x\mathbf{:}\ y)$:

$$P(x)\ \&\ L(S(x\mathbf{:}\ y_1))\ \&\ L(S(x\mathbf{:}\ y_2)) \Rightarrow y_1 = y_2; \qquad (7)$$

Total specification $[P(x), Q(x, y)]$:

$$T(P(x), Q(x, y)) \cong P(x) \Rightarrow \exists y.\ Q(x, y); \qquad (8)$$

Single-valued specification:

$$SV(P(x), Q(x, y)) \cong P(x)\ \&\ Q(x, y_1)\ \&\ Q(x, y_2) \Rightarrow y_1 = y_2\,. \qquad (9)$$

3. Theorem of identity between specification and program

T1: $\dfrac{T(P(x), Q(x, y));\ \ P(x)\ \&\ Q(x, y) \Rightarrow L(S(x\mathbf{:}\ y))}{Corr(S(x\mathbf{:}\ y), P(x), Q(x, y))}$

4. Rules of correctness proof for statements

Rules for general case with specifications for substatements:

RP: $\dfrac{Corr(B(x\mathbf{:}\ y), P_B(x), Q_B(x, y));\ \ Corr(C(x\mathbf{:}\ z), P_C(x), Q_C(x, z));\ \ P(x) \vdash P_B(x)\ \&\ P_C(x);\ \ Q_B(x, y)\ \&\ Q_C(x, z) \vdash Q(x, y, z)}{Corr(B(x\mathbf{:}\ y)\ ||\ C(x\mathbf{:}\ z), P(x), Q(x, y, z))}$

RS: $\dfrac{Corr(B(x\mathbf{:}\ z), P_B(x), Q_B(x, z));\ \ Corr(C(z\mathbf{:}\ y), P_C(z), Q_C(z, y));\ \ P(x) \vdash P_B(x)\ \&\ \forall z.\ Q_B(x, z) \Rightarrow P_C(z);\ \ P(x)\ \&\ \exists z.\ Q_B(x, z)\ \&\ Q_C(z, y) \vdash Q(x, y)}{Corr(B(x\mathbf{:}\ z);\ \ C(z\mathbf{:}\ y), P(x), Q(x, y))}$

RC: $\dfrac{Corr(B(x\mathbf{:}\ y), P_B(x), Q_B(x, y));\ \ Corr(C(x\mathbf{:}\ y), P_C(x), Q_C(x, y));\ \ P(x)\ \&\ E \vdash P_B(x);\ \ P(x)\ \&\ \neg E \vdash P_C(x);\ \ P(x)\ \&\ E\ \&\ Q_B(x, y) \vdash Q(x, y);\ \ P(x)\ \&\ \neg E\ \&\ Q_C(x, y) \vdash Q(x, y)}{Corr(\mathbf{if}\ (E)\ B(x\mathbf{:}\ y)\ \mathbf{else}\ C(x\mathbf{:}\ y), P(x), Q(x, y))}$

Rules for single-valued specification with specifications for substatements:

LP:
$$\frac{\begin{array}{l}T(P(x), Q(x, y, z));\ \ Corr(B(x: y), P_B(x), Q_B(x, y));\ \ SV(P_B(x), Q_B(x, y));\\ Corr(C(x: z), P_C(x), Q_C(x, z));\ \ SV(P_C(x), Q_C(x, z));\\ P(x) \vdash P_B(x) \,\&\, P_C(x);\ \ P(x) \,\&\, Q(x, y, z) \vdash Q_B(x, y) \,\&\, Q_C(x, z)\end{array}}{Corr(B(x: y) \,||\, C(x: z), P(x), Q(x, y, z))}$$

LS:
$$\frac{\begin{array}{l}T(P(x), Q(x, y));\ \ Corr(B(x: z), P_B(x), Q_B(x, z));\\ Corr(C(z: y), P_C(z), Q_C(z, y));\ \ SV(P_C(x), Q_C(z, y));\\ P(x) \vdash P_B(x);\ \ P(x) \,\&\, Q(x, y) \,\&\, Q_B(x, z) \vdash P_C(x) \,\&\, Q_C(z, y)\end{array}}{Corr(B(x: z);\ \ C(z: y), P(x), Q(x, y))}$$

LC:
$$\frac{\begin{array}{l}T(P(x), Q(x, y));\ \ Corr(B(x: y), P_B(x), Q_B(x, y));\ \ SV(P_B(x), Q_B(x, y));\\ Corr(C(x: y), P_C(x), Q_C(x, y));\ \ SV(P_C(x), Q_C(x, y));\\ P(x) \,\&\, Q(x, y) \,\&\, E \vdash P_B(x) \,\&\, Q_B(x, y);\\ P(x) \,\&\, Q(x, y) \,\&\, \neg E \vdash P_C(x) \,\&\, Q_C(x, y)\end{array}}{Corr(\textbf{if (}E) B(x\textbf{:} y) \textbf{ else } C(x\textbf{:} y), P(x), Q(x, y))}$$

Rules for general case without specifications for substatements:

QP:
$$\frac{Corr(B(x: y), P(x), Q(x, y));\ \ Corr(C(x: z), P(x), R(x, z))}{Corr(B(x: y) \,||\, C(x: z), P(x), Q(x, y) \,\&\, R(x, z)}$$

QC:
$$\frac{Corr(B(x: y), P(x) \,\&\, E, Q(x, y));\ \ Corr(C(x: y), P(x) \,\&\, \neg E, Q(x, y))}{Corr(\textbf{if (}E) B(x\textbf{:} y) \textbf{ else } C(x\textbf{:} y), P(x), Q(x, y))}$$

QS:
$$\frac{\begin{array}{l}P(x) \vdash \exists z.\ L(B(x: z));\ \ L(B(x: z)) \vdash \exists y.\ L(C(z: y));\\ \exists z.\ L(B(x: z)) \,\&\, L(C(z: y)) \vdash Q(x, y)\end{array}}{Corr(B(x: z);\ \ C(z: y), P(x), Q(x, y))}$$

Decomposition rules for single-valued specification:

$$FB:\ \frac{\mathrm{Corr}(A(x\colon y), P(x), Q(x, y));\ \ \mathrm{SV}(P(x), Q(x, y));\ \ R(x, y) \vdash P(x) \,\&\, Q(x, y)}{R(x, y) \vdash L(A(x\colon y))}$$

The rule *FB* is used, when one needs to prove the formula $R(x, y) \Rightarrow L(A(x\colon y))$ and the statement $A(x\colon y)$ is correct relative to the specification $[P(x), Q(x, y)]$. If the statement $A(x\colon y)$ is a recursive call of the procedure A, then the rule *FB* must include the additional premise $m(x) < m(z)$, where z denotes the arguments of the procedure A and m is a natural *measure* function defined on arguments.

$$FP:\ \frac{R(x, y, z) \vdash L(B(x\colon y));\ \ R(x, y, z) \vdash L(C(x\colon z))}{R(x, y, z) \vdash L(B(x\colon y) \,||\, C(x\colon z))}$$

$$FC:\ \frac{R(x, y) \,\&\, E \vdash L(B(x\colon y));\ \ R(x, y) \,\&\, \neg E \vdash L(C(x\colon y))}{R(x, y) \vdash L(\mathbf{if}\ (E)\ B(x\colon y)\ \mathbf{else}\ C(x\colon y))}$$

$$FS:\ \frac{R(x, y) \vdash \exists z.\ L(B(x\colon z));\ \ R(x, y) \,\&\, L(B(x\colon z)) \vdash L(C(z\colon y))}{R(x, y) \vdash L(B(x\colon z);\ \ C(z\colon y))}$$

Decomposition rules for ∃-part:

$$FE:\ \frac{R(x) \vdash \exists z.\ L(B(x\colon z));\ \ R(x) \,\&\, L(B(x\colon z)) \vdash \exists y.\ L(C(z\colon y))}{R(x) \vdash \exists y.\ L(B(x\colon z);\ \ C(z\colon y))}$$

$$FL:\ \frac{R(x) \,\&\, L(B(x\colon z)) \,\&\, L(C(z\colon y)) \vdash H(x, y)}{R(x) \,\&\, L(B(x\colon z);\ \ C(z\colon y)) \vdash H(x, y)}$$

4. Example of generating correctness formulas

The problem mult(a, b, d**:** c) with the specification
[a ≥ 0 & b ≥ 0 & d ≥ 0, c = a ∗ b + d] and the following program:

proc mult(**nat** a, b, d**: nat** c) (11)
{ **if** (a = 0) c := d **else** mult(a – 1, b, d + b**:** c) } **measure** a;

First, we introduce the definitions:

formula P_mult(**nat** a, b, d) = a ≥ 0 & b ≥ 0 & d ≥ 0;
formula Q_mult(**nat** a, b, d, c) = c = a ∗ b + d;
function m(**nat** a**: nat**) = a;

An application of the rule *T1* (Theorem 1) to program (11) produces the totality lemma (the first premise):

lemma P_mult(a, b, d) ⇒ ∃c. Q_mult(a, b, d, c)

and the following formula (the second premise) for proving:

P_mult(a, b, d) & Q_mult(a, b, d, c) ⇒ (12)
L(**if** (a = 0) c := d **else** mult(a – 1, b, d + b**:** c));

An application of the rule *FC* to formula (12) produces the lemma (the first premise):

lemma P_mult(a, b, d) & Q_mult(a, b, d, c) & a = 0 ⇒ c = d .

and the following formula (the second premise) for proving:

P_mult(a, b, d) & Q_mult(a, b, d, c) & ¬ a = 0 ⇒
L(mult(a – 1, b, d + b**:** c)) . (13)

For the recursive call of the procedure mult in (13), we apply the rule ***FB3*** which is similar to ***FB*** and includes an additional premise to ensure termination of recursion. It produces the following lemma:

lemma P_mult(a, b, d) & Q_mult(a, b, d, c) & ¬ a = 0 ⇒
m(a - 1) < m(a) & P_mult(a – 1, b, d + b) &
Q_mult(a – 1, b, d + b, c) .

One can compare the described verification of program (11) with the classic Hoare verification of the following imperative program obtained from program (11) by transforming the tail-recursion to the **while** statement:

c := d; **while** a != 0 **do** a := a – 1; c := c + b **end**