УДК 004.415.52

Методы доказательства корректности программ с хорошей логикой

В.И. Шелехов

Институт Систем Информатики СО РАН, г. Новосибирск,
e-mail: vshel@iis.nsk.su

Логика программы – предикат, истинный на значениях переменных тогда и только тогда, когда некоторое исполнение программы завершается с этими значениями. Для программ с хорошей логикой (без циклов типа **while** и указателей) определяется формула тотальной корректности. Теорема тождества спецификации и программы использует более простую формулу корректности, применимую для однозначных спецификаций. Разработана система правил доказательства корректности для различных видов операторов программы. Генерация и доказательство формул корректности проводится проще, чем для верификации по методу Хоара. Правила доказательства корректности применимы также для программного синтеза.

***Ключевые слова**: тотальная корректность программы, дедуктивная верификация, программный синтез, формальная семантика языка программирования*

Rules of correctness proof for programs with simple logic

Vladimir Shelekhov

A.P.Ershov Institute of Informatics Systems, Novosibirsk, Russia

e-mail: vshel@iis.nsk.su

The notion of a program logic is introduced to denote a predicate which is true for some variable values if and only if some program execution finishes with that values. For a program with restrictions typical for functional languages, total correctness formula which uses the program logic is defined. A theorem of identity between specification and program introduces simple correctness formula applicable for a single-valued specification. The rules of program correctness proof have been developed for proving the statements of various kinds. Generating and proving the correctness formulas are different than in the classic Hoare verification. The rules of program correctness proof are also applicable for program synthesis.

***Keywords**: total program correctness, deductive verification, program synthesis, formal semantics of a programming language*

Введение

Всякая программа обладает логикой. *Логика программы* – предикат (логическое утверждение), зависящий от значений переменных программы в момент ее завершения. Предикат истинен на фиксированных значениях переменных тогда и только тогда, когда существует исполнение программы, завершающиеся с этими значениями.

Наше рассмотрение ограничено классом программ для задач дискретной и вычислительной математики. Эти программы реализуют функции, отображающие значения аргументов в значения результатов. Предполагается, что взаимодействие с внешним окружением программы отсутствует, т.е. в программе нет операций ввода-вывода, а перед началом исполнения программы значения аргументов находятся в памяти программы.

Предлагаемое понятие логики программы неприменимо для реактивных систем и программ системного программирования, поскольку эффект исполнения программы нельзя представить в виде функции – эффект определяется только в виде процесса. Логика таких программ может быть представлена набором предикатов в разных точках программы в рамках алгебры процессов Р. Милнера [12] или Т. Хоара [13].

При анализе программы для определения причины ее ошибочного исполнения программист пытается изъять из кода программы необходимую ему часть логики программы. Логика обычно извлекается в виде содержательных утверждений, а не логических формул. Квалифицированный программист обязан уметь «читать» программу, т.е. определять ее свойства по коду программы. Однако в действительности извлечение свойств (логики) программы оказывается нетривиальным. Типичным способом исследования свойств программы является ее исполнение в отладочном режиме.

Извлечение логики из программы проводится также в целях дедуктивной верификации программы. Логические формулы автоматически извлекаются из программы на основе формальной (денотационной [11] или аксиоматической [2]) семантики языка программирования. Логику программы, извлекаемую статическим анализом из кода программы в процессе реинжиниринга программы [16], принято называть бизнес-логикой.

Для иллюстрации понятия логики программы рассмотрим простейшую программу умножения натуральных чисел a и b через операцию сложения. *Логика решения задачи* определяется очевидными свойствами целых чисел:

$$a * b = b + (a - 1) * b, \qquad 0 * b = 0 . \tag{1}$$

С помощью данных формул можно реализовать вычисление умножения $a * b$ в стиле логического программирования многократной заменой правой части формулы на левую. Однако формулы (1) не могут использоваться в качестве императивной программы. Для получения программы их следует переписать в виде следующей рекурсивной функции:

```
nat mult(nat a, b)                                              (2)
{ if (a = 0) return 0 else return b + mult(a – 1, b) }
```

Логика программы (2) в точности определяется логикой решения (1).

Поскольку программа (2) неэффективна, обычно строится следующая программа в виде цикла:

```
nat c = 0; while (a != 0) { c = c + b; a = a – 1 }              (3)
```

Программа (3) также построена на основе логики решения (1), однако ее логика уже иная. Логика тела цикла есть предикат $c' = c + b \,\&\, a' = a - 1$, а логика цикла **while** определяется формулой: $c' = c + a * b$. Для модифицируемых переменных в приведенных формулах переменная со штрихом обозначает итоговое значение переменной, а без штриха – исходное.

Более адекватной логикой цикла является *инвариант цикла*, истинный перед началом очередного исполнения тела цикла. Инвариантом цикла **while** является предикат $c = (a - a\tilde{}) * b$, где a – исходное значение переменной a, а $a\tilde{}$ – текущее. Инвариант необходим для дедуктивной верификации цикла в логике Хоара [2]. В технологии

программирования на базе инвариантов (Invariant Based Programming) [18] для цикла сначала строится инвариант, а затем цикл программируется на базе инварианта.

При «чтении» кода (3) программист определяет результат вычисления цикла в виде суммы c' = b + b + ... + b, где слагаемое b встречается a раз, что эквивалентно c' = a ∗ b. Отметим, что логика (1), применяемая при построении программы (3), отлична от используемой при ее анализе. Впрочем, программист редко занимается извлечением логики цикла. Чтобы убедится в правильности цикла, часто бывает достаточно проверить его исполнение для пары случаев, например, при a = 0 и a = 2.

Итак, логика программы является трансформацией *логики решения задачи* – набора математических свойств, определяющих алгоритм решения задачи, а программа является реализацией логики решения задачи. Требования эффективности вынуждают программиста оптимизировать программу в процессе ее построения. Оптимизации программы трансформируют ее логику, обычно в сторону усложнения.

Использование циклов типа **while** и указателей переменных приводит к существенному усложнению (искривлению) логики программы. Эти конструкции являются «вредными» в программировании; по сравнению с ними оператор перехода[1] является совершенно безобидным. Для любой, даже простой, императивной программы ее логика оказывается существенно искривленной. Искривление логики программы серьезно ухудшает ее понимание. Как следствие, противоречие между надежностью и эффективностью неразрешимо для традиционного стиля императивного программирования.

Любая оптимизация императивной программы, совершаемая программистом в процессе ее построения, может быть воспроизведена в предикатном программировании [20, 21] без искривления логики программы. Например, оптимизация программы (2) с целью получения эффективной программы (3) реализуется следующим образом. Рекурсивный вызов функции mult в программе (2) не является хвостовым, что затрудняет автоматическое преобразование рекурсивной функции в цикл **while** при трансляции программы. Для преобразования рекурсии к хвостовому виду применяется метод *обобщения исходной задачи* mult вычисления произведения a ∗ b. Рассматривается более общая задача mult1 для вычисления выражения c + a ∗ b с похожей логикой решения:

$$c + a * b = c + b + (a - 1) * b, \qquad 0 * b = 0 . \tag{4}$$

Решение задачи mult сводится к mult1 при c = 0 следующим образом:

```
nat mult(nat a, b) { return mult1(0, a, b) }                    (5)
nat mult1(nat a, b, c)
{ if (a = 0) return c else return mult1(a – 1, b, c + b) }
```

Данная программа декларативна, хотя и не является компактной. Однако она имеет простую логику, совпадающую с логикой решения (4), и поэтому более понятна для программиста, чем программа (3). Программа (3) может быть получена как результат оптимизирующей трансляции программы (5): хвостовая рекурсия в функции mult1 заменяется циклом, после чего тело mult1 подставляется на место вызова mult1(0, a, b).

Технология построения программ с правильной логикой для класса задач дискретной и вычислительной математики исследуется в работах [20, 21].

В настоящей работе представлены методы дедуктивной верификации программ с правильной логикой, совпадающей с логикой решения задачи. Понятие логики программы

[1] Структурное программирование Э. Дейкстры исходит из признания «вредным» оператора перехода.

формально представлено в разделе 2. Понятие тотальной корректности программы сформулировано в виде формулы (11) в разделе 3. Для программ с однозначной спецификацией определена более простая формула тотальной корректности в разделе 5. Система правил доказательства корректности для разных видов операторов разработана в разделе 6. Использование правил доказательства корректности для дедуктивной верификации программы mult1 иллюстрируется в разделе 7. В заключении суммируются основные положения работы, описывается опыт дедуктивной верификации небольших программ применением разработанной системы правил и определяются возможности применения разработанных методов для императивных программ.

1. Обзор смежных работ

В классической работе по дедуктивной верификации программ [1] Р. Флойд оперирует понятием интерпретации программы в виде множества предикатов для входов и выходов операторов программы. Для программы на языке блок-схем в [1] каждая дуга блок-схемы снабжается предикатом. Формальная семантика языка описывается условиями корректности для всех видов операторов. Параметры условий корректности для оператора – предикаты на входе и выходе оператора. В соответствии с семантикой языка предикаты интерпретации должны соответствовать условиям корректности. Для сравнения, предикаты логики программы должны быть согласованы с формальной операционной семантикой. Таким образом, понятие интерпретации является близким к понятию логики программы, введенному в данной работе.

Дедуктивная верификация реальных программных систем в основном базируется на логике Хоара [2] (известной также как логика Флойда – Хоара), определяющей набор правил вывода (аксиом) для троек Хоара. Последовательное применение правил позволяет получить формулу – условие частичной корректности императивной программы. В нашем подходе правила доказательства тотальной корректности операторов являются конкретизациями формулы тотальной корректности (11) и используют логики подоператоров. Последовательное применение правил генерирует набор коротких формул, в целом определяющих тотальную корректность программы. Вместо инвариантов циклов здесь достаточно построить функцию меры, строго убывающую на аргументах рекурсивных вызовов. Применение правил для случая однозначных спецификаций позволяет получить более простые формулы корректности с меньшим числом кванторов. Итак, генерация формул корректности во многом проще, чем для верификации по логике Хоара.

Отметим, что в ряде работ после 2003 г., а затем по инерции (см., например [14]), термин «логика программы» (program logic) начинает использоваться как синоним логики Хоара и ее расширений, что не вполне адекватно для обозначения системы правил вывода.

Отказ от использования в программе циклов и указателей фактически приводит к функциональному программированию. Здесь проповедуется декларативный стиль программирования и не приветствуется оптимизация программы программистом на исходном языке. Для функциональных языков не удается добиться приемлемой эффективности даже с помощью изощренной оптимизации. Определенный прогресс в эффективности достигается разработкой сверхмощных интерпретаторов и генераторов кода для языка SequenceL [19], определяющего предельно декларативный стиль

программирования. В частности, в программе языка SequenceL опущены итераторы по компонентам структур; они подразумеваются неявно.

В предикатном программировании [20, 21], базирующемся на правильной логике программ, применяется противоположный подход: эффективность программы предопределяется ее оптимизацией на исходном языке предикатного программирования. Например, для получения программы (3) строится программа (5). Разумеется, существующие методы трансляции функциональных программ способны автоматически превратить нехвостовую рекурсию программы (2) в хвостовую и получить в итоге программу (3). Однако после преобразования рекурсии к хвостовому виду далее обычно следует серия нетривиальных улучшений программы (см., например, [7]), которую совершенно нереально провести автоматически.

Формальная семантика языка программирования признана элегантным и мощным формализмом, однако не для реально используемого императивного языка программирования. Для него денотационная семантика оказывается сложной и громоздкой из-за сложности самого языка, а аксиоматическая семантика (логика Хоара) может быть построена лишь для некоторого простого подмножества языка. Эта оценка, данная Джоном Бэкусом 30 лет назад в своей тьюринговской лекции [17], остается справедливой в настоящее время для современных императивных языков. Таким образом, императивные языки – это языки сложной и кривой логики, которую весьма трудно извлечь из программы.

2. Логика программы

Допустим, программа представлена оператором $S(x, y)$ на некотором языке императивного или функционального программирования, где x и y – наборы переменных, обозначающие аргументы и результаты программы. Все остальные переменные, которыми оперирует программа, являются локальными. Предполагается, что взаимодействие с внешним окружением программы отсутствует, т.е. в программе нет операций ввода-вывода, а перед началом исполнения программы аргументы x находятся в памяти программы. Таким образом, программа $S(x, y)$ реализует функцию, отображающую значения аргументов x в значения результатов y.

Логика программы – предикат (логическое утверждение) $L(S(x, y))$, зависящий от значений переменных x и y в момент завершения программы. Предикат истинен на фиксированных значениях переменных x и y тогда и только тогда, когда существует исполнение программы, завершающиеся с этими значениями.

Логика $L(K)$ может быть определена для произвольной конструкции K – оператора или выражения. Логика выражения истинна, если исполнение выражения нормально завершается. Например, логика выражения a / b есть $b \neq 0$ в предположении, что семантика не ограничивает представление сверхмалых и сверхбольших чисел; однако для чисел фиксированной разрядности логика $L(a / b)$ определяет более сложные ограничения на значения a и b.

Рассмотрим построение логики для операторов различных видов.

Пусть K есть оператор присваивания $a := E$, где E – выражение. Если выражение E не зависит от a, то $L(a := E)$ есть предикат $a = E$. Если выражение E зависит от a, то $L(a := E) \equiv a' = E$, где a' обозначает итоговое значение переменной a после присваивания.

Пусть K есть оператор B; C, определяющий последовательное исполнение операторов B и C. Построим логику L(B; C) через логики L(B) и L(C). Оказывается, необходимо отдельно рассматривать случай, когда вычисление оператора C от B не зависит. Для фиксации связей между операторами произвольный оператор A будем изображать в виде A(x: y), где наборы переменных x и y обозначают аргументы и результаты оператора, соответственно.

Вместо оператора B; C далее будем рассматривать *оператор суперпозиции* B(x: z); C(z: y) и *параллельный оператор* B(x: y) || C(x: z). Третьим рассматривается условный оператор **if (**E) B(x: y) **else** C(x: y), где логическое выражение E может зависеть от x. В предположении, что наборы x, y и z – не пересекаются, а набор x может быть пустым, определим логики указанных операторов:

$$L(B(x: z); C(z: y)) \cong \exists z.\ L(B(x: z)) \,\&\, L(C(z: y)) \quad (6)$$

$$L(B(x: y) \,||\, C(x: z)) \cong L(B(x: y)) \,\&\, L(C(x: z)) \quad (7)$$

$$L(\textbf{if (}E) B(x: y) \textbf{ else } C(x: y)) \cong (E \Rightarrow L(B(x: y))) \,\&\, (\neg E \Rightarrow L(C(x: y))) \quad (8)$$

Несмотря на очевидность этих определений для использования их в дедуктивной верификации программ необходимо формально доказать их корректность. Для этого сначала необходимо построить формальную операционную семантику для языка программирования, содержащего указанные операторы. Далее следует формализовать определение логики программы. Доказательство правильности определений (6) – (8) заключается в доказательстве их согласованности с формальной операционной семантикой, что требуется в соответствии с определением логики программы.

Рассмотрим оператор цикла **while** E **do** B(x: x) **end**, где x – набор переменных, E – логическое выражение, зависящее от x. Построим логику оператора цикла через логику L(B(x: x′)), где x′ обозначает набор переменных, не пересекающийся с набором x. Логика цикла типа **while** нетривиальна. Ее можно представить бесконечной формулой:

$$\neg E(x) \vee E(x) \,\&\, L(B(x: x_1)) \,\&\, \neg E(x_1) \vee E(x) \,\&\, L(B(x: x_1)) \,\&\, E(x_1) \,\&\, L(B(x_1: x_2)) \,\&\, \neg E(x_2) \vee \ldots \vee E(x) \,\&\, L(B(x: x_1)) \,\&\, E(x_1) \,\&\, L(B(x_1: x_2)) \,\&\, E(x_2) \,\&\, \ldots \,\&\, E(x_{k-1}) \,\&\, L(B(x_{k-1}: x_k)) \,\&\, \neg E(x_k) \vee \ldots$$

Точное определение логики цикла реализуется с помощью аппарата наименьшей неподвижной точки.

Логика может быть также построена для рекурсивно определяемых процедур и для операций со всеми типами данных, кроме указателей. Таким образом, логика программы естественным образом конструируется для любого чистого языка функционального программирования. Перечисленных выше конструкций с простой логикой вполне достаточно для представления любого алгоритма в классе задач дискретной и вычислительной математики, но не для реактивных систем с параллельными взаимодействующими процессами.

3. Корректность программы

Представление о правильности программы принято формулировать в виде понятия *корректности* программы, означающего, что программа должна соответствовать спецификации. Это общее требование о соответствии программы ее спецификации формулируется в виде набора *условий корректности* программы.

Допустим, программа представлена оператором S(x: y), где x и y – непересекающиеся наборы переменных. Для программы определена логика L(S(x: y)). *Спецификация программы* S(x: y) задается двумя предикатами: *предусловием* P(x),

ограничивающим область определения функции, реализуемой программой, и *постусловием* Q(x, y), связывающим значения аргументов и результатов. Спецификация изображается следующим образом: [P(x), Q(x, y)]. Программа S(x: y) со спецификацией записывается в виде известной тройки Хоара {P(x)} S(x: y) {Q(x, y)}, см. [2].

Пусть точка 1 определяет начало исполнения программы, а точка 2 – конец исполнения. Корректность программы определяется следующими двумя условиями:

- предусловие P(x) должно быть истинным в точке 1;
- постусловие Q(x, y) должно быть истинным в точке 2.

Второе условие корректности должно быть истинным в случае завершения исполнение программы. Однако допустимо ли, чтобы исполнение не завершалась? Нет. Бесконечное исполнение бессмысленно, поскольку в соответствии с исходным предположением программа не взаимодействует с окружением. Поэтому необходимо третье условие корректности:

- исполнение программы всегда завершается.

Истинность предусловия P(x) невозможно проверить статически для программы S(x: y). Возможна лишь динамическая проверка предусловия перед началом исполнения программы. Однако статическая проверка предусловия может быть проведена для вызова программы S из другой программы.

Предусловие P(x), истинное в точке 1, будет также истинным и в точке 2, поскольку программа S(x: y) не меняет значений набора переменных x. Поэтому истинность второго и третьего условий корректности, ассоциированных с точкой 2, достаточно доказать при условии, что P(x) истинно.

Предикат L(S(x: y)) становится второй посылкой при доказательстве второго условия корректности, поскольку в точке 2 истинно L(S(x: y)). В соответствии с определением логики программы условие завершения исполнения определяется предикатом $\exists y.\ L(S(x: y))$. В итоге второе и третье условия корректности принимают следующий вид:

$$P(x)\ \&\ L(S(x: y)) \Rightarrow Q(x, y) \qquad (9)$$

$$P(x) \Rightarrow \exists y.\ L(S(x: y)) \qquad (10)$$

Формула (9) называется условием *частичной корректности*, а (10) – условием *завершения* программы. Их конъюнкция определяет условие *тотальной* (или *полной*) корректности программы:

$$Corr(S(x: y), P(x), Q(x, y)) \cong P(x) \Rightarrow [\ L(S(x: y)) \Rightarrow Q(x, y)\ ]\ \&\ \exists y.\ L(S(x: y)) \qquad (11)$$

Далее термин «корректность» будем использовать в смысле тотальной корректности.

4. Свойства тотальности и однозначности

Известные понятия тотальности и однозначности функции $f: x \to y$, где x и y – непересекающиеся наборы переменных, естественным образом переносятся на предикат H(x, y), рассматриваемый как функция из x в y. Предикат H(x, y) является *однозначным* в области X для набора переменных x, если истинно:

$$\forall x \in X\ \forall y_1, y_2.\ H(x, y_1)\ \&\ H(x, y_2) \Rightarrow y_1 = y_2\ .$$

Предикат H(x, y) является *тотальным* в области X для набора переменных x, если:

$$\forall x \in X\ \exists y.\ H(x, y) \quad .$$

Далее потребуется следующая лемма.

Лемма 1. Допустим, предикат $D(x, y)$ является однозначным в области X, а предикат $Z(x, y)$ – тотальный в области X. Пусть истинна формула $\forall x{\in}X.\ Z(x, y) \Rightarrow D(x, y)$. Тогда истинна следующая формула: $\forall x{\in}X.\ D(x, y) \Rightarrow Z(x, y)$. Как следствие, предикаты D и Z оказываются тождественными в области X.

Доказательство. Пусть истинно $D(x, y)$ для некоторого $x{\in}X$. Докажем истинность $Z(x, y)$. Поскольку предикат Z – тотальный, то существует некоторый y', для которого истинно $Z(x, y')$. Из формулы $Z(x, y) \Rightarrow D(x, y)$ получаем истинность $D(x, y')$. Поскольку истинны $D(x, y)$ и $D(x, y')$, то из однозначности D следует $y = y'$. Тогда истинно $Z(x, y)$. □

Однозначность программы $S(x{:}\ y)$, тотальность и однозначность спецификации $[P(x), Q(x, y)]$ определяются, соответственно, формулами:

$$P(x)\ \&\ L(S(x{:}\ y_1))\ \&\ L(S(x{:}\ y_2)) \Rightarrow y_1 = y_2; \qquad (12)$$

$$T(P(x), Q(x, y)) \cong\ P(x) \Rightarrow \exists y.\ Q(x, y); \qquad (13)$$

$$SV(P(x), Q(x, y)) \cong\ P(x)\ \&\ Q(x, y_1)\ \&\ Q(x, y_2) \Rightarrow y_1 = y_2. \qquad (14)$$

Отметим, что свойство тотальности программы есть в точности условие корректности (10).

Лемма 2. Если программа $S(x{:}\ y)$ корректна относительно спецификации $[P(x), Q(x, y)]$, то спецификация тотальна.

Доказательство. Требуется доказать (13). Пусть истинно $P(x)$. Докажем истинность $\exists y.\ Q(x, y)$. Из формулы тотальной корректности (11) и $P(x)$ следует истинность формул $\exists y.\ L(S(x{:}\ y))$ и $L(S(x{:}\ y)) \Rightarrow Q(x, y)$. Допустим, истинность формулы $\exists y.\ L(S(x{:}\ y))$ реализуется для некоторого $y = y_0$. Из формулы $L(S(x{:}\ y)) \Rightarrow Q(x, y)$ получаем истинность $Q(x, y_0)$, а значит и $\exists y.\ Q(x, y)$. □

Наше рассмотрение ограничено понятием тотальной корректности. Исключается ситуация, когда исполнение программы может недетерминировано завершаться или не завершаться. Такая ситуация возможна, например, для оператора суперпозиции $B(x{:}\ z);\ C(z{:}\ y)$ при нарушении свойства однозначности (12) для оператора $B(x{:}\ z)$. Пусть для некоторого x существуют разные наборы z_1 и z_2 ($z_1 \neq z_2$) такие, что истинны $L(B(x{:}\ z_1))$ и $L(B(x{:}\ z_2))$. Это означает, что исполнение оператора $B(x{:}\ z)$ для набора x может недетерминировано завершаться разными наборами значений z_1 и z_2. Предположим, что $L(C(z_1{:}\ y))$ истинно для некоторого y, а $L(C(z_2{:}\ y))$ ложно для любых y. Иначе говоря, предполагается, что исполнение оператора $C(z{:}\ y)$ завершается для z_1 и не завершается (не определено) для z_2. Как следствие, оператор суперпозиции недетерминировано завершается либо не завершается. Для программ с таким поведением трактовка тотальной корректности оказывается неадекватной. Здесь принято использовать понятие *общей корректности* (general correctness) [15].

5. Теорема тождества спецификации и программы

Формула $L(S(x{:}\ y)) \Rightarrow Q(x, y)$, являющаяся главной частью формулы тотальной корректности (11), определяет вывод спецификации из программы, точнее, ее логики. Существуют подходы (в частности, программного синтеза), в которых, наоборот, программа выводится из спецификации. Наша цель – определить условия, при которых

доказательство корректности программы можно повернуть в другую сторону. Эти условия представлены Леммой 1.

Теорема 1 тождества спецификации и программы. Рассмотрим программу S(x: y) со спецификацией [P(x), Q(x, y)]. Допустим, оператор S(x: y) является однозначным, а спецификация [P(x), Q(x, y)] является тотальной. Предположим, логика программы L(S(x: y)) выводима из спецификации, т. е.

$$P(x) \,\&\, Q(x, y) \Rightarrow L(S(x: y)) \,. \qquad (15)$$

Тогда программа S(x: y) является корректной относительно спецификации.

Доказательство. Для доказательства корректности программы S(x: y) достаточно доказать истинность формулы (11), т. е.

$$P(x) \Rightarrow [\, L(S(x: y)) \Rightarrow Q(x, y) \,] \,\&\, \exists y.\, L(S(x: y)) \,.$$

Допустим, предусловие P(x) истинно.

Докажем истинность ∃y. L(S(x: y)). Поскольку спецификация предиката S тотальна, то истинна формула ∃y. Q(x, y). Пусть эта формула истинна для некоторого y′. Тогда из (15) истинна L(S(x: y′)) и, следовательно, ∃y. L(S(x: y)).

Докажем истинность формулы L(S(x: y)) ⇒ Q(x, y). Поскольку истинны P(x) и формула (15), то истинна формула Q(x, y) ⇒ L(S(x: y)). В соответствии с Леммой 1 истинна формула L(S(x: y)) ⇒ Q(x, y), поскольку постусловие Q(x, y) тотально, а L(S(x: y)) – однозначно. □

Лемма 3. В условиях теоремы 1 истинна следующая формула:

$$P(x) \Rightarrow (L(S(x: y)) \equiv Q(x, y)) \,.$$

Доказательство. Допустим, предусловие P(x) истинно. Истинность формулы L(S(x: y)) ⇒ Q(x, y) установлена при доказательстве Теоремы 1. Обратная импликация следует из (15). □

Следствие Леммы 3: в условиях Теоремы 1 спецификация [P(x), Q(x, y)] является однозначной.

Лемма 4. Допустим, программа S(x: y) является корректной, а спецификация [P(x), Q(x, y)] – однозначной. Тогда истинна формула (15).

Доказательство. Допустим, истинны P(x) и Q(x, y). Докажем истинность L(S(x: y)). Из формулы тотальной корректности (11) следует истинность формул L(S(x: y)) ⇒ Q(x, y) и ∃y. L(S(x: y)). Пусть для некоторого y′ истинно L(S(x: y′)). Тогда истинно Q(x, y′), а из однозначности Q следует y = y′. Следовательно, истинно L(S(x: y)). □

Итак, для доказательства корректности программы S(x: y) достаточно доказать формулу (15). Формула (15) существенно проще (11) и поэтому является более предпочтительной. Однако она применима лишь для однозначных спецификаций. Для неоднозначных спецификаций формула (15) недоказуема.

Для применения формулы (15) достаточно обеспечить однозначность программы и тотальность спецификации. Однозначность программы будет гарантирована при условии, что все используемые базисные операции программы являются однозначными. Это свойство программы необходимо формально доказать при разработке формальной операционной семантики языка программирования.

Переформулируем Теорему 1 в виде правила доказательства корректности программы:

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

Здесь и далее в правилах на базе формулы (15) условие однозначности программы опускается, хотя и подразумевается.

6. Система правил доказательства корректности операторов

Используя формулу (11) или (15) можно автоматически построить формулу корректности для программы $S(x: y)$ при условии, что для языка программирования построена логика программы. Итоговая формула корректности будет длинной и сложной даже для коротких программ; она будет длиннее программы $S(x: y)$. Специализация формул (11) и (15) для разных видов операторов позволяет декомпозировать длинную формулу корректности к нескольким более коротким и простым формулам.

В данном разделе представлена специализация формул (11) и (15) в виде правил доказательства корректности для оператора суперпозиции $B(x: z); C(z: y)$, параллельного оператора $B(x: y) \,||\, C(x: z)$ и условного оператора **if (**E**)** $B(x: y)$ **else** $C(x: y)$.

6.1. Правила для общего случая

Рассмотрим случай, когда для операторов B и C имеются спецификации и эти операторы корректны по отношению к своим спецификациям. Ниже представлены правила на базе формулы (11). Они позволяют свести доказательство корректности оператора к доказательству корректности составляющих операторов B и C.

6.1.1 Правило корректности для параллельного оператора

Допустим, операторы $B(x: y)$ и $C(x: z)$ корректны относительно своих спецификаций $[P_B(x), Q_B(x, y)]$ и $[P_C(x), Q_C(x, z)]$. Параллельный оператор $B(x: y) \,||\, C(x: z)$ имеет спецификацию $[P(x), Q(x, y, z)]$. Представим правило для доказательства корректности параллельного оператора.

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

Доказательство истинности правила *RP*. Необходимо доказать формулу корректности (11) для параллельного оператора. Пусть предусловие $P(x)$ истинно. Тогда в соответствии с формулой (11) следует доказать тотальность $L(B(x: y) \,||\, C(x: z))$ и выводимость постусловия $Q(x, y, z)$ из $L(B(x: y) \,||\, C(x: z))$. В соответствии с определением (7) формула $L(B(x: y) \,||\, C(x: z))$ эквивалентна $L(B(x: y))\ \&\ L(C(x: z))$.

Из истинности предусловия $P(x)$ и третьей посылки правила *RP* следует истинность $P_B(x)$ и $P_C(x)$. Далее, из корректности операторов $B(x: y)$ и $C(x: z)$ следует истинность формул $\exists y.\ L(B(x: y))$ и $\exists z.\ L(C(x: z))$. Их конъюнкция определяет тотальность $L(B(x: y) \,||\, C(x: z))$.

Докажем выводимость постусловия $Q(x, y, z)$ из $L(B(x: y))\ \&\ L(C(x: z))$. Допустим, истинна формула $L(B(x: y))\ \&\ L(C(x: z))$. Из истинности $P_B(x)$ и $P_C(x)$ и корректности

операторов $B(x{:}\ y)$ и $C(x{:}\ z)$ следует истинность формул $L(B(x{:}\ y)) \Rightarrow Q_B(x, y)$ и $L(C(x{:}\ z)) \Rightarrow Q_C(x, z)$. Как следствие, будут истинны $Q_B(x, y)$ и $Q_C(x, z)$. Наконец, из последней посылки правила *RP* следует истинность постусловия $Q(x, y, z)$. □

6.1.2. Правило корректности для оператора суперпозиции

Допустим, операторы $B(x{:}\ z)$ и $C(z{:}\ y)$ корректны относительно своих спецификаций $[P_B(x), Q_B(x, z)]$ и $[P_C(z), Q_C(z, y)]$. Оператор суперпозиции $B(x{:}\ z); C(z{:}\ y)$ имеет спецификацию $[P(x), Q(x, y)]$. Представим правило для доказательства корректности оператора суперпозиции.

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

Доказательство истинности правила *RS*. Необходимо доказать формулу корректности (11) для оператора суперпозиции. Пусть предусловие $P(x)$ истинно. Тогда в соответствии с формулой (11) следует доказать тотальность $L(B(x{:}\ z); C(z{:}\ y))$ и выводимость постусловия $Q(x, y)$ из $L(B(x{:}\ z); C(z{:}\ y))$. В соответствии с определением (11) формула $L(B(x{:}\ z); C(z{:}\ y))$ эквивалентна $\exists z.\, L(B(x{:}\ z)) \,\&\, L(C(z{:}\ y))$.

Из истинности предусловия $P(x)$ и третьей посылки правила *RS* следует истинность формул $P_B(x)$ и $\forall z\, (Q_B(x, z) \Rightarrow P_C(z))$. Из истинности $P_B(x)$ и корректности оператора $B(x{:}\ z)$ следует истинность формул $\exists z.\, L(B(x{:}\ z))$ и $L(B(x{:}\ z)) \Rightarrow Q_B(x, z)$. Допустим, для некоторого z_0 формула $L(B(x{:}\ z_0))$ истинна. Как следствие, истинно $Q_B(x, z_0)$. Далее, из истинности $\forall z\, (Q_B(x, z) \Rightarrow P_C(z))$ следует истинность $P_C(z_0)$. Ввиду корректности оператора $C(z{:}\ y)$ истинна формула $\exists y\, L(C(z_0{:}\ y))$. Далее, истинна конъюнкция $L(B(x{:}\ z_0)) \,\&\, \exists y\, L(C(z_0{:}\ y))$, и затем – формула $\exists y.\, \exists z.\, L(B(x{:}\ z)) \,\&\, L(C(z{:}\ y))$, т. е. доказана тотальность $L(B(x{:}\ z); C(z{:}\ y))$.

Докажем выводимость постусловия $Q(x, y)$ из $L(B(x{:}\ z); C(z{:}\ y))$. Пусть $L(B(x{:}\ z); C(z{:}\ y))$ истинно, т. е. истинна формула $\exists z.\, L(B(x{:}\ z)) \,\&\, L(C(z{:}\ y))$. Пусть формула истинна для некоторого z_1. Ввиду корректности оператора $B(x{:}\ z)$ истинна формула $L(B(x{:}\ z_1)) \Rightarrow Q_B(x, z_1)$ и далее – $Q_B(x, z_1)$. Истинность $Q_B(x, z_1)$ и $\forall z\, (Q_B(x, z) \Rightarrow P_C(z))$ влечет истинность $P_C(z_1)$. Ввиду корректности оператора $C(z{:}\ y)$ истинно $L(C(z_1, y)) \Rightarrow Q_C(z_1, y)$. Поскольку $L(C(z_1, y))$ истинно, то истинно $Q_C(z_1, y)$. В итоге, истинна правая часть последней посылки правила *RS*, а значит – и левая, т. е. истинно постусловие $Q(x, y)$. □

6.1.3. Правило корректности для условного оператора

Допустим, операторы $B(x{:}\ y)$ и $C(x{:}\ y)$ корректны относительно своих спецификаций $[P_B(x), Q_B(x, y)]$ и $[P_C(x), Q_C(x, y)]$. Условный оператор **if (**E) $B(x{:}\ y)$ **else** $C(x{:}\ y)$ имеет спецификацию $[P(x), Q(x, y)]$. Представим правило для доказательства корректности условного оператора.

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

Доказательство истинности правила *RC*. Необходимо доказать формулу корректности (11) для условного оператора. В нее дважды входит подформула $L(\textbf{if }(E)\ B(x{:}\ y)\ \textbf{else}\ C(x{:}\ y))$, которая согласно определению (8) эквивалентна:

$$(E \Rightarrow L(B(x{:}\ y))) \ \&\ (\neg E \Rightarrow L(C(x{:}\ y)))\ . \qquad (16)$$

Пусть предусловие $P(x)$ истинно. В соответствии с формулой (16) следует доказать тотальность формулы (16) и выводимость из нее постусловия $Q(x, y)$.

Допустим, что условие E истинно. Из истинности предусловия $P(x)$ и третьей посылки правила *RC* следует истинность $P_B(x)$. Ввиду корректности оператора $B(x{:}\ y)$ истинна формула $\exists y.\ L(B(x{:}\ y))$. Далее будет истинной формула $\exists y.\ (E \Rightarrow L(B(x{:}\ y)))$. Из истинности E следует истинность формулы $\neg E \Rightarrow L(C(x{:}\ y))$ и, следовательно, формулы $\exists y.\ [(E \Rightarrow L(B(x{:}\ y))) \ \&\ (\neg E \Rightarrow L(C(x{:}\ y)))]$. Это доказывает тотальность формулы (16) в случае истинности E. Тотальность (16) в случае ложности E доказывается аналогичным образом.

Докажем выводимость постусловия $Q(x, y)$ из формулы (16). Допустим, истинна формула (16). Пусть E истинно. Тогда истинно $L(B(x{:}\ y))$. Из третьей посылки правила *RC* следует истинность $P_B(x)$. Ввиду корректности оператора $B(x{:}\ y)$ истинна формула $L(B(x{:}\ y)) \Rightarrow Q_B(x, y)$, а значит – и $Q_B(x, y)$. В итоге, истинна правая часть пятой посылки правила *RC*, и, следовательно, истинна левая часть посылки, т. е. истинно постусловие $Q(x, y)$. Доказательство истинности постусловия $Q(x, y)$ для случая, когда E ложно, проводится аналогично с использованием четвертой и шестой посылок правила *RC*. □

6.2. Правила для однозначной спецификации

Допустим, подоператоры B и C имеют спецификации и эти операторы корректны по отношению к своим спецификациям. Предлагаемая ниже система правил доказательства корректности операторов базируется на Теореме 1, в соответствии с которой для тотальной спецификации и при условии однозначности используемых операторов требуется доказать истинность формулы (15). Правила применимы только для однозначной спецификации.

6.2.1 Правило корректности для параллельного оператора

Допустим, операторы $B(x{:}\ y)$ и $C(x{:}\ z)$ корректны относительно своих спецификаций $[P_B(x), Q_B(x, y)]$ и $[P_C(x), Q_C(x, z)]$. Параллельный оператор $B(x{:}\ y)\ ||\ C(x{:}\ z)$ имеет спецификацию $[P(x), Q(x, y, z)]$. Представим правило для доказательства корректности параллельного оператора.

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

Доказательство истинности правила *LP*. Поскольку спецификация [P(x), Q(x, y, z)] тотальна, в соответствии с Теоремой 1 для доказательства правила достаточно доказать истинность формулы:

$$P(x)\ \&\ Q(x, y, z) \Rightarrow L(B(x{:}\ y)\ ||\ C(x{:}\ z))\ .$$

В соответствии с определением (7) формула $L(B(x{:}\ y)\ ||\ C(x{:}\ z))$ эквивалентна $L(B(x{:}\ y)\ \&\ L(C(x{:}\ z))$.

Пусть истинны P(x) и Q(x, y, z). Докажем истинность $L(B(x{:}\ y)$ и $L(C(x{:}\ z))$. Из истинности предусловия P(x) и посылки $P(x) \vdash P_B(x)\ \&\ P_C(x)$ следует истинность $P_B(x)$ и $P_C(x)$. Из последней посылки правила *LP* становятся истинными $Q_B(x, y)$ и $Q_C(x, z)$. Для предикатов B и C выполняются условия Леммы 4. Поэтому истинны формулы:

$$P_B(x)\ \&\ Q_B(x, y) \Rightarrow L(B(x{:}\ y);$$
$$P_C(x)\ \&\ Q_C(x, z) \Rightarrow L(C(x{:}\ z)).$$

Поскольку посылки этих формул истинны, то истинны $L(B(x{:}\ y)$ и $L(C(x{:}\ z))$. □

6.2.2 Правило корректности для оператора суперпозиции

Допустим, операторы $B(x{:}\ z)$ и $C(z{:}\ y)$ корректны относительно своих спецификаций $[P_B(x), Q_B(x, z)]$ и $[P_C(z), Q_C(z, y)]$. Оператор суперпозиции $B(x{:}\ z);\ C(z{:}\ y)$ имеет спецификацию [P(x), Q(x, y)]. Представим правило для доказательства корректности оператора суперпозиции.

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

Доказательство истинности правила *LS*. В соответствии с Теоремой 1 достаточно доказать истинность формулы:

$$P(x)\ \&\ Q(x, y) \Rightarrow L(B(x{:}\ z);\ C(z{:}\ y))\ .$$

В соответствии с определением (11) формула $L(B(x{:}\ z);\ C(z{:}\ y))$ эквивалентна $\exists z.(L(B(x{:}\ z))\ \&\ L(C(z{:}\ y))$.

Пусть истинны P(x) и Q(x, y). Докажем истинность $\exists z.(L(B(x{:}\ z))\ \&\ L(C(z{:}\ y))$. Из истинности предусловия P(x) и посылки $P(x) \vdash P_B(x)$ следует истинность $P_B(x)$. Из корректности оператора B следует истинность формул $\exists z.\ L(B(x{:}\ z))$ и $L(B(x{:}\ z)) \Rightarrow Q_B(x, z)$. Допустим для некоторого z_0 истинно $L(B(x{:}\ z_0))$. Тогда истинно $Q_B(x, z_0)$. В соответствии с последней посылкой правила *LS* истинна формула $P_C(z_0)\ \&\ Q_C(z_0, y)$. В соответствии с Леммой 4 истинна формула

$$P_C(z)\ \&\ Q_C(z, y) \Rightarrow L(C(z{:}\ y))\ .$$

Тогда истинна $L(C(z_0{:}\ y))$. В итоге, будет истинна формула $\exists z.(L(B(x{:}\ z))\ \&\ L(C(z{:}\ y))$. □

6.2.3. Правило корректности для условного оператора

Допустим, операторы $B(x{:}\ y)$ и $C(x{:}\ y)$ корректны относительно своих спецификаций $[P_B(x), Q_B(x, y)]$ и $[P_C(x), Q_C(x, y)]$. Условный оператор **if (**E) $B(x{:}\ y)$ **else** $C(x{:}\ y)$ имеет спецификацию [P(x), Q(x, y)]. Представим правило для доказательства корректности условного оператора.

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

Доказательство истинности правила *LC*. В соответствии с Теоремой 1 достаточно доказать истинность формулы:

$$P(x) \,\&\, Q(x, y) \Rightarrow LS(\textbf{if (}E) B \textbf{ else } C)(x, y) .$$

Согласно определению (8) формула $L(\textbf{if (}E) B(x\textbf{:} y) \textbf{ else } C(x\textbf{:} y))$ эквивалентна:

$$(E \Rightarrow L(B(x\textbf{:} y))) \,\&\, (\neg E \Rightarrow L(C(x\textbf{:} y))) .$$

Пусть истинны $P(x)$ и $Q(x, y)$. Докажем истинность формулы $E \Rightarrow L(B(x\textbf{:} y))$. Пусть истинно E. Докажем истинность $L(B(x\textbf{:} y))$. Можно применить правило LC1, Поскольку истинны $P(x)$, $Q(x, y)$ и E в правой части предпоследней посылки правила *LC*, то истинна левая часть, т.е. формула $P_B(x) \,\&\, Q_B(x, y)$. В соответствии с Леммой 4 истинна формула

$$P_B(x) \,\&\, Q_B(x, y) \Rightarrow L(B(x\textbf{:} y)) .$$

Поскольку истинна посылка, то истинно $L(B(x\textbf{:} y))$. Следовательно, доказана истинность формулы $E \Rightarrow L(B(x\textbf{:} y))$. Истинность формулы $\neg E \Rightarrow L(C(x\textbf{:} y))$ доказывается аналогично. □

В приведенных правилах посылка вида $SV(...)$ для операторов B и C может быть опущена, если корректность этих операторов доказывается по основе формулы (15). Это вытекает из Следствия Леммы 3.

6.3. Система правил декомпозиции доказательства корректности для произвольной спецификации

Спецификации подоператоров как правило отсутствуют. Ниже представлены правила на базе формулы (11). Они позволяют свести доказательство корректности оператора к доказательству корректности составляющих операторов B и C.

6.3.1. Правило корректности для параллельного оператора

Предположим, что для параллельного оператора $L(A(x\textbf{:} y) \,||\, B(x\textbf{:} z))$ постусловие представимо в виде конъюнкции предикатов $Q(x, y) \,\&\, R(x, z)$.

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

Доказательство истинности правила *QP*. Допустим, истинно предусловие $P(x)$. Необходимо доказать тотальность формулы $L(A(x\textbf{:} y) \,||\, B(x\textbf{:} z))$ и выводимость из этой формулы постусловия $Q(x, y) \,\&\, R(x, z)$. В соответствии с определением (7) формула $L(B(x\textbf{:} y) \,||\, C(x\textbf{:} z))$ эквивалентна $L(B(x\textbf{:} y) \,\&\, L(C(x\textbf{:} z))$. Из истинности $P(x)$ и корректности операторов $B(x\textbf{:} y)$ и $C(x\textbf{:} z)$ следует истинность следующих формул: $L(B(x\textbf{:} y)) \Rightarrow Q(x, y)$, $\exists y.\ L(B(x\textbf{:} y))$, $L(C(x\textbf{:} z)) \Rightarrow R(x, z)$ и $\exists z.\ L(C(x\textbf{:} z))$. Конъюнкция $\exists y.\ L(B(x\textbf{:} y))$ и $\exists z.\ L(C(x\textbf{:} z))$ дает тотальность формулы $L(A(x\textbf{:} y) \,||\, B(x\textbf{:} z))$. Выводимость $Q(x, y) \,\&\, R(x, z)$ также очевидна. □

6.3.2. Правило корректности для условного оператора

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

Доказательство истинности правила *QC*. Допустим, истинно предусловие P(x). Необходимо доказать тотальность формулы L(**if (**E) B(x**:** y) **else** C(x**:** y)) и выводимость из нее постусловия Q(x, y). В соответствии с определением (8) формула L(**if (**E) B(x**:** y) **else** C(x**:** y)) эквивалентна:

$$(E \Rightarrow L(B(x{:}\ y))) \ \&\ (\neg E \Rightarrow L(C(x{:}\ y)))\ .$$

Допустим, что условие E истинно. Из истинности P(x) & E и корректности оператора B(x**:** y) по первой посылке правила *QC* следует истинность формулы ∃y. L(B(x**:** y)). Далее будет истинной формула ∃y. (E ⇒ L(B(x**:** y))). Из истинности E следует истинность формулы ¬E ⇒ L(C(x**:** y)) и, следовательно, формулы

$$\exists y.\ [(E \Rightarrow L(B(x{:}\ y))) \ \&\ (\neg E \Rightarrow L(C(x{:}\ y)))]\ .$$

Это доказывает тотальность L(**if (**E) B(x**:** y) **else** C(x**:** y)) для истинного E. Тотальность в случае ложного E доказывается аналогичным образом.

Пусть формула L(**if (**E) B(x**:** y) **else** C(x**:** y)) истинна. Докажем истинность Q(x, y). Допустим, что условие E истинно. Из истинности P(x) & E и корректности оператора B(x**:** y) следует истинность формулы L(B(x, y)) ⇒ Q(x, y). Из истинности (E ⇒ L(B(x**:** y)) следует истинность L(B(x**:** y) и далее – Q(x, y). Истинность Q(x, y) при ложном E доказывается аналогично. □

6.3.3. Правило корректности для оператора суперпозиции

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

Доказательство истинности правила *QS*. Необходимо доказать формулу корректности (11) для оператора суперпозиции. Пусть предусловие P(x) истинно. Тогда в соответствии с формулой (11) следует доказать тотальность L(B(x**:** z); C(z**:** y)) и выводимость постусловия Q(x, y) из L(B(x**:** z); C(z**:** y)). В соответствии с определением (11) формула L(B(x**:** z); C(z**:** y)) эквивалентна ∃z. L(B(x**:** z)) & L(C(z**:** y)).

Из истинности предусловия P(x) и первой посылки правила *QS* следует истинность формулы ∃z. L(B(x**:** z)). Допустим, для некоторого z_0 формула L(B(x**:** z_0)) истинна. Из второй посылки правила *QS* следует истинность формулы ∃y. L(C(z**:** y)). Далее, истинна L(B(x**:** z_0)) & ∃y L(C(z_0**:** y)), и затем – формула ∃y. ∃z. L(B(x**:** z)) & L(C(z**:** y)), т. е. доказана тотальность L(B(x**:** z); C(z**:** y)).

Докажем выводимость постусловия Q(x, y) из L(B(x**:** z); C(z**:** y)). Пусть L(B(x**:** z); C(z**:** y)) истинно, т. е. истинна формула ∃z. L(B(x**:** z)) & L(C(z**:** y)). Истинность Q(x, y) следует из третьей посылки правила *QS*. □

6.3.4. Правило корректности для оператора суперпозиции с первым корректным оператором

$$QS1:\ \frac{\begin{array}{l}\mathrm{Corr}(B(x{:}\ z),\ P_B(x),\ Q_B(x,\ z));\\ P(x)\ \vdash P_B(x)\ \&\ \forall z.\ (Q_B(x,\ z) \Rightarrow \exists y.\ L(C(z{:}\ y)));\\ P(x)\ \&\ \exists z.\ Q_B(x,\ z)\ \&\ L(C(z{:}\ y))\ \vdash Q(x,\ y)\end{array}}{\mathrm{Corr}(B(x{:}\ z);\ C(z{:}\ y),\ P(x),\ Q(x,\ y))}$$

Доказательство истинности правила *QS1*. Необходимо доказать формулу корректности (11) для оператора суперпозиции. Пусть предусловие $P(x)$ истинно. Тогда в соответствии с формулой (11) следует доказать тотальность $L(B(x{:}\ z);\ C(z{:}\ y))$ и выводимость постусловия $Q(x, y)$ из $L(B(x{:}\ z);\ C(z{:}\ y))$. В соответствии с определением (11) формула $L(B(x{:}\ z);\ C(z{:}\ y))$ эквивалентна $\exists z.\ L(B(x{:}\ z))\ \&\ L(C(z{:}\ y))$.

Из истинности предусловия $P(x)$ и второй посылки правила *QS1* следует истинность формул $P_B(x)$ и $\forall z\ (Q_B(x, z) \Rightarrow \exists y.\ L(C(z{:}\ y)))$. Из истинности $P_B(x)$ и корректности оператора $B(x{:}\ z)$ следует истинность формул $\exists z.\ L(B(x{:}\ z))$ и $L(B(x{:}\ z)) \Rightarrow Q_B(x, z)$. Допустим, для некоторого z_0 формула $L(B(x{:}\ z_0))$ истинна. Как следствие, истинно $Q_B(x, z_0)$. Далее, из истинности $\forall z\ (Q_B(x, z) \Rightarrow \exists y.\ L(C(z{:}\ y)))$ следует истинность $\exists y.\ L(C(z_0{:}\ y)))$. Далее, истинна $L(B(x{:}\ z_0))\ \&\ \exists y\ L(C(z_0{:}\ y))$, и затем – формула $\exists y.\ \exists z.\ L(B(x{:}\ z))\ \&\ L(C(z{:}\ y))$, т. е. доказана тотальность $L(B(x{:}\ z);\ C(z{:}\ y))$.

Докажем выводимость постусловия $Q(x, y)$ из $L(B(x{:}\ z);\ C(z{:}\ y))$. Пусть $L(B(x{:}\ z);\ C(z{:}\ y))$ истинно, т. е. истинна формула $\exists z.\ L(B(x{:}\ z))\ \&\ L(C(z{:}\ y))$. Из истинности $P_B(x)$ и корректности оператора $B(x{:}\ z)$ следует истинность формулы и $L(B(x{:}\ z)) \Rightarrow Q_B(x, z)$. Как следствие, истинна формула $\exists z.\ Q_B(x, z)\ \&\ L(C(z{:}\ y))$. Из истинности правой части третьей посылки правила *QS1* следует истинность левой части, т.е. $Q(x, y)$. □

6.4. Система правил декомпозиции доказательства корректности для однозначной спецификации

Теорема 1 сводит доказательство корректности оператора к формуле (15): $P(x)\ \&\ Q(x, y) \Rightarrow L(S(x{:}\ y))$. Декомпозиция доказательства формулы (15) реализуется для вхождения $L(S(x{:}\ y))$. Таким образом, имеется задача доказательства формулы вида $R(x, y) \Rightarrow L(S(x{:}\ y))$, где $R(x, y)$ – произвольная посылка. Решением задачи являются правила доказательства формулы для различных видов операторов в позиции оператора $S(x{:}\ y)$.

6.4.1. Правило корректности для параллельного оператора

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

Доказательство истинности правила *FP*. В соответствии с определением (7) формула $L(B(x{:}\ y)\ ||\ C(x{:}\ z))$ эквивалентна $L(B(x{:}\ y))\ \&\ L(C(x{:}\ z))$. Поэтому достаточно доказать истинность двух формул:

$$R(x, y, z) \Rightarrow L(B(x{:}\ y))$$
$$R(x, y, z) \Rightarrow L(C(x{:}\ z))$$

Эти формулы представлены посылками правила *FP*. □

6.4.2. Правило корректности для условного оператора

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

Доказательство истинности правила *FC*. В соответствии с определением (8) формула $L(\textbf{if (}E)\ B(x\textbf{:}\ y)\ \textbf{else}\ C(x\textbf{:}\ y))$ эквивалентна

$$(E \Rightarrow L(B(x\textbf{:}\ y))) \ \&\ (\neg E \Rightarrow L(C(x\textbf{:}\ y)))$$

Таким образом, требуется доказать истинность:

$$R(x, y) \Rightarrow (E \Rightarrow L(B(x\textbf{:}\ y))) \ \&\ (\neg E \Rightarrow L(C(x\textbf{:}\ y)))$$

Последняя формула эквивалентна конъюнкции формул:

$$R(x, y) \Rightarrow (E \Rightarrow L(B(x\textbf{:}\ y)))$$
$$R(x, y) \Rightarrow (\neg E \Rightarrow L(C(x\textbf{:}\ y)))$$

А эти формулы представлены посылками правила *FC*. □

6.4.3. Правило корректности для оператора суперпозиции

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

Доказательство истинности правила *FS*. В соответствии с определением (6) формула $L(B(x\textbf{:}\ z); C(z\textbf{:}\ y))$ эквивалентна $\exists z.\ L(B(x\textbf{:}\ z))\ \&\ L(C(z\textbf{:}\ y))$. Пусть истинно $R(x, y)$. Докажем истинность $\exists z.\ L(B(x\textbf{:}\ z))\ \&\ L(C(z\textbf{:}\ y))$. Из первой посылки правила *FS* истинна формула $\exists z.\ L(B(x\textbf{:}\ z))$. Допустим для некоторого z_0 истинно $L(B(x\textbf{:}\ z_0))$. Из второй посылки истинна формула $L(C(z_0\textbf{:}\ y))$. В итоге, будет истинна формула $\exists z.\ L(B(x\textbf{:}\ z))\ \&\ L(C(z\textbf{:}\ y))$. □

6.4.4. Правило для нерекурсивного вызова предиката

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

Доказательство истинности правила *FB*. Пусть истинно $R(x, y)$. Докажем истинность $L(A(x\textbf{:}\ y))$. Из третьей посылки правила *FB* истинна формула $P(x)\ \&\ Q(x, y)$. В соответствии с Леммой 4 истинна формула $P(x)\ \&\ Q(x, y) \Rightarrow L(A(x\textbf{:}\ y))$ и, следовательно, $L(A(x\textbf{:}\ y))$. □

Правило *FB* используется для доказательства формулы $R(x, y) \Rightarrow L(A(x: y))$ при условии, что оператор $A(x\textbf{:}\ y)$ корректен относительно спецификации $[P(x), Q(x, y)]$. Если оператор $A(x\textbf{:}\ y)$ является рекурсивным вызовом процедуры A, правило *FB* (точнее его модификация *FB3*) должно включать дополнительную посылку $m(x) < m(z)$, где z обозначает аргументы процедуры A, а m – функция меры, определенная на аргументах. Заметим, что посылка $SV(P(x), Q(x, y))$ может быть опущена, если корректность вызова доказывается на базе формулы (15).

6.4.5. Правила для оператора суперпозиции в позиции квантора существования и в левой части

Приведенные выше правила достаточно просты. Их можно применять многократно для декомпозиции вхождений L(S(x: y)) при доказательстве формул вида: R(x, y) ⇒ L(S(x: y)). Таких формул большинство среди посылок в приведенных выше правилах. Однако правило *FS* использует посылки двух других видов: R(x, y) ⇒ ∃y. L(S(x: y)) и R(x, y) & L(S(x: y)) ⇒ H(x, y), где R(x, y) и H(x, y) – произвольные предикаты. Необходимо разработать правила для декомпозиции вхождений L(S(x: y)) в этих новых видах формул. Ограничимся правилами для оператора суперпозиции. Правила для параллельного и условного операторов значительно проще.

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

Доказательство истинности правила *FE*. В соответствии с определением (6) формула ∃y. L(B(x: z); C(z: y)) эквивалентна ∃y. ∃z.(L(B(x: z)) & L(C(z: y))). Пусть истинно R(x). Из первой посылки правила *FE* истинна ∃z. L(B(x: z)). Допустим, для некоторого z_0 истинна L(B(x: z_0)). Из второй посылки правила *FE* истинна формула ∃y. L(C(z_0: y)). В итоге будет истинна ∃y. ∃z.(L(B(x: z)) & L(C(z: y))). □

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

Доказательство истинности правила *FL*. Пусть истинно R(x) & L(B(x: z); C(z: y)). Докажем истинность H(x, y). В соответствии с определением (6) формула L(B(x: z); C(z: y)) эквивалентна ∃z. L(B(x: z)) & L(C(z: y)). Допустим, формула истинна при некотором z_0. Тогда истинны L(B(x: z_0)) и L(C(z_0: y)). Применение посылки правила *FL* доказывает истинность H(x, y). □

7. Пример применения правил доказательства корректности

Дадим иллюстрацию применения некоторых из приведенных правил для доказательства корректности программы умножения mult1, представленной во Введении. Программа mult1(a, b, d: c) имеет спецификацию

[a ≥ 0 & b ≥ 0 & d ≥ 0, c = a ∗ b + d] .

Перепишем программу (5) в новых обозначениях.

```
proc mult1(nat a, b, d: nat c)                                    ( 17)
    { if (a = 0) c = d else mult1(a – 1, b, d + b: c) }
    measure a;
```

Для удобства генерации формул корректности программы введем обозначения для формул предусловия, постусловия и функции меры:

```
formula P_mult1(nat a, b, d) = a ≥ 0 & b ≥ 0 & d ≥ 0;
formula Q_mult1(nat a, b, d, c) = c = a ∗ b + d;
function m(nat a: nat) = a;
```

Поскольку спецификация однозначна, к программе (17) применяется правило *T1* (Теорема 1). Первая посылка правила *T1* порождает лемму тотальности спецификации:

```
lemma      P_mult1(a, b, d) ⇒ ∃c. Q_mult1(a, b, d, c) .
```

На базе второй посылки правила *T1* генерируется цель для доказательства:

P_mult1(a, b, d) & Q_mult1(a, b, d, c) ⇒ (18)
L(**if** (a = 0) c = d **else** mult1(a – 1, b, d + b**:** c));

Формулу (18) следует декомпозировать для вхождения функции логики L для условного оператора. Применяется правило декомпозиции *FC*. Первая посылка правила порождает лемму:

lemma P_mult1(a, b, d) & Q_mult1(a, b, d, c) & a = 0 ⇒ c = d .

На базе второй посылки правила *FC* генерируется следующая цель для доказательства:

P_mult1(a, b, d) & Q_mult1(a, b, d, c) & ¬ a = 0 ⇒ L(mult1(a – 1, b, d + b**:** c)) . (19)

Для рекурсивного вызова процедуры mult1 в формуле (19) применяется правило *FB3* (модификация *FB*), включающее дополнительную посылку, гарантирующую завершение рекурсии. Это правило порождает лемму:

lemma P_mult1(a, b, d) & Q_mult1(a, b, d, c) & ¬ a = 0 ⇒
m(a - 1) < m(a) & P_mult1(a – 1, b, d + b) & Q_mult1(a – 1, b, d + b, c) .

Данные три леммы определяют набор формул для доказательства тотальной корректности программы (17).

Описанную верификацию программы (17) можно сравнить с классической верификацией по Хоару для программы, полученной из (17) заменой хвостовой рекурсии на цикл типа **while**:

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

Заключение

Для программы, построенной с использованием оператора суперпозиции, параллельного и условного операторов, имеющих простую логику, представлена система правил доказательства корректности программы. На базе данных трех видов операторов разработан язык предикатного программирования P [3], находящийся на границе меду функциональными и логическими языками. Для полного языка P разработана формальная операционная и логическая семантика, доказана согласованность операционной и логической семантики [4].

Множество правил доказательства корректности программы делится на две части в зависимости от того, является ли спецификация однозначной или нет. Каждая из двух частей, в свою очередь, делится на две части в зависимости от того, имеется ли спецификация для подоператоров. Из этих четырех частей правила для однозначных спецификаций на базе формулы (15) при отсутствии спецификации для подоператоров оказались наиболее простыми и удобными для верификации. Эта часть правил разработана для полного языка P. На базе этой части правил для программы на языке P разработан детальный алгоритм генерации формул корректности на язык системы автоматического доказательства PVS [5].

Алгоритм генерации формул корректности опробован примерно для 20 небольших программ. Наиболее значимыми являются программа сумматора Линь (Ling adder) [6] и эффективные программы для стандартных функций floor, isqrt, и ilog2 [7]. Сгенерированные формулы корректности для всех программ были доказаны в системе PVS. Результаты данного эксперимента следующие. Задача автоматической генерации формул корректности решена успешно. Генерируемые формулы корректности декомпозированы хорошо: они достаточно короткие и вполне понятные. Доказательство на PVS оказалось нетривиальным и трудоемким процессом. Обнаружено 15 случайных

ошибок (описок) в программе или спецификации. Лишь одна обнаруженная при верификации ошибка в программе функции floor оказалась нетривиальной и опасной.

Правила для доказательства корректности программы достаточно просты. Они годятся также для программного синтеза [6, 7]. Причем при разработке программы в стиле синтеза доказательство формул корректности проводится существенно легче.

Из-за ограничений на циклы типа **while** и указатели предлагаемый подход к дедуктивной верификации и программному синтезу напрямую применим лишь к чистым языкам функционального программирования. Тем не менее, подход может быть применен к частям императивных программ с простой операторной структурой, сходной с представленной в данной работе. Подобные смешанные модели программ становятся популярными в дедуктивной верификации. Например, дедуктивная верификация для частей программ без указателей переменных проводится более простым способом [8]. В другом подходе для упрощения верификации вместо произвольных указателей в программе допускаются лишь двухсвязные списки или структуры типа «лес деревьев» [9, 10].

Литература

1. *Floyd R. W.* Assigning meanings to programs // Proceedings Symposium in Applied Mathematics, Mathematical Aspects of Computer Science. AMS, 1967. P. 19–32.

2. *Hoare C. A. R.* An axiomatic basis for computer programming // Communications of the ACM. 1969. Vol. 12 (10). P. 576–585.

3. *Карнаухов Н.С., Першин Д.Ю., Шелехов В.И.* Язык предикатного программирования Р. Новосибирск, 2010. 42с. (Препр. / ИСИ СО РАН; N 153).

4. *Shelekhov V.* The language of calculus of computable predicates as a minimal kernel for functional languages // BULLETIN of the Novosibirsk Computing Center. Series: Computer Science. IIS Special Issue. 2009. 29(2009). P.107-117.

5. PVS Specification and Verification System. SRI International. http://pvs.csl.sri.com/

6. *Шелехов В.И.* Верификация и синтез программ сложения на базе правил корректности операторов // Computer Science in Russia CSR-2010. Workshop on Program Semantics and Verification: Theory and Applications. Казань, 2010. С. 150-156.

7. *Шелехов В.И.* Верификация и синтез эффективных программ стандартных функций floor, isqrt и ilog2 в технологии предикатного программирования // Тр. 12-й межд. конф. «Проблемы управления и моделирования в сложных системах». Самара, Самарский научный центр РАН, 2010. С. 622-630.

8. *Anureev I. S., Maryasov I. V., Nepomniaschy V. A.* C-programs Verification on Basis of Mixed Axiomatic Semantics // Modeling and analysis of informations systems. Yaroslavl, 2010. Vol.17, No. 4, P. 5-28.

9. *Cohen E., Dahlweid M., Hillebrand M., Leinenbach D., Moskal M., Santen T., Schulte W., Tobies S. VCC: A Practical System for Verifying Concurrent C //* LNCS, 5674, P. 1–22. 2009.

10. *Ball T., Hackett B., Lahiri S.K., Qadeer S., and Vanegue J.* Towards Scalable Modular Checking of User-Defined Properties // LNCS, 6217, P. 1-24. 2010.

11. *Scott D. S. and Strachey C.* Towards a mathematical semantics for computer languages // Computers and Automata. 1971. P. 19–46.

12. *Milner R. A.* Calculus of Communicating Systems // Lecture Notes in Computer Science, 1980. Vol. 92.

13. *Hoare C. A. R*. Communicating Sequential Processes. Prentice-Hall. 1985.

14. *Appel A.W.* Verified Software Toolchain // LNCS 6602, 2011. P. 1–17.

15. *Dawson J. E.* Formalising General Correctnes // Electronic Notes in Theoretical Computer Science, Vol. 91, 2004, P. 21-42.

16. Автоматизированный реинжиниринг программ / Под ред. проф. А. Н. Терехова и А. А. Терехова. СПб.: Изд-во С.-Петерб. ун-та, 2000. 332 с.

17. *Backus J*. Can programming be liberated from the von Neumann style? A. Functional Style and Its Algebra of Programs // Communications of the ACM. 1978. Vol. 21 (8). P. 613–641.

18. *Eriksson J. and Back R.-J*. Applying PVS Background Theories and Proof Strategies in Invariant Based Programming // LNCS 6447, 2010. P. 24–39.

19. *Cooke D. E., Rushton J. N*. Taking Parnas's Principles to the Next Level: Declarative Language Design // Computer, Vol. 42, no. 9. 2009. P. 56-63.

20. Методы предикатного программирования / Под ред. Шелехова В.И. Вып.1. ИСИ СО РАН. Новосибирск, 2003. 62 С.

21. Методы предикатного программирования / Под ред. Шелехова В.И. Вып.2. ИСИ СО РАН. Новосибирск, 2006. 116 С.