Дедуктивная верификация предикатных программ[1]

Чушкин М.С.
Новосибирский Государственный Университет

Описывается реализация генератора условий тотальной корректности предикатных программ в системе предикатного программирования. Разработана система правил доказательства корректности для различных видов операторов предикатных программ.

Введение

Тестирование – не единственный способ обнаружения ошибок. Комплекс различных методов обнаружения ошибок и контроля качества программ определяется как верификация программного обеспечения. Верификация включает в себя множество методов, таких как статический анализ, экспертиза и формальные методы. Одним из формальных методов является дедуктивная верификация. В отличие от тестирования с помощью дедуктивной верификации можно обнаружить все ошибки. Однако это весьма сложный и трудоемкий метод. Применение дедуктивной верификации оправдано лишь в приложениях с высокой ценой ошибки: в авиакосмической отрасли, энергетике, медицине и др.

Дедуктивная верификация реализует проверку правильности программы относительно ее спецификации, записанной на формальном языке спецификаций. Условия корректности программы генерируются автоматически по формулам логики и спецификации программы путем применения системы логических правил. Условие корректности программы обычно имеет вид $A \Rightarrow B$, где B – утверждение, определяемое спецификацией программы, а формула A, истинная для программы, извлекается из нее с помощью формальной (операционной, денотационной, …) семантики языка программирования. Доказательство условий корректности проводится с помощью некоторой системы автоматического доказательства. Ее применение, в отличие от доказательства «вручную», гарантирует правильность программы относительно спецификации.

В данной работе описывается автоматическая генерация условий корректности для программ на языке предикатного программирования P [3].

Язык предикатного программирования P находится на границе меду функциональными и логическими языками. Язык P определяет класс программ, не взаимодействующих с внешним окружением. Эти программы реализуют функции, отображающие значения входных переменных в значения результатов. Исполнение программ должно всегда завершаться, поскольку бесконечное исполнение бессмысленно. Этот класс, по меньшей мере, включает программы для задач дискретной и вычислительной математики. Спецификация предикатных программ реализуется с помощью предусловия и постусловия.

Исчисление предикатов

Ниже будет приведено несколько основных определений, заимствованных из теории исчисление предикатов. Для более прозрачной связи с данной работой, определения были слегка упрощены. Более подробно и полно этот материал описан в работе [12].

[1] Работа выполнялась в рамках гранта РФФИ № 12-01-00686.

Правилом вывода называют формальную запись вида:

$$\frac{\Phi_0, \Phi_1, \ldots, \Phi_n}{\Gamma}$$

где Φ_i и Γ – это формулы, причем Φ_i – это посылки, а Γ – это заключение.

Определим индуктивно понятие *дерева формул*:

1) Всякая формула является деревом.

2) Если $D_0, D_1, \ldots, D_n$ – деревья и S – формула, то

$$\frac{D_0, D_1, \ldots, D_n}{S}$$

– также дерево. Формула S является корнем дерева.

Дерево формул D называется *деревом вывода* формулы C, если его переходы – это применение правил вывода, а корнем в D является C.

Истинность формул, являющихся листьями дерева, и истинность правил влечет истинность корневой формулы C.

Логика оператора

Программа на языке P представляет собой совокупность определений предикатов. Определение предиката имеет вид:

$$A(x: y)\ \textbf{pre}\ P(x)\ \{\ S(x: y);\ \}\ \textbf{post}\ Q(x, y),$$

где A – это имя определяемого предиката, S - оператор, а x и y – это непересекающиеся наборы переменных, аргументы и результаты соответственно.

Спецификация предиката A(x: y) задается двумя логическими формулами: предусловием P(x), ограничивающим область определения функции, реализуемой программой, и постусловием Q(x, y), связывающим значения аргументов и результатов.

Логика оператора S(x: y) – логическое утверждение L(S(x: y)), зависящее от значений переменных оператора. Логика оператора является его точным эквивалентом [6].

Определим логики для базисных операторов. В предположении, что выражение E, зависящее от x, не содержит переменную a, логика оператора присваивания a := E есть:

$$L(a := E(x)) \cong R(x)\ \&\ a = E(x)$$

где R(x) – слабейший предикат, при истинности которого определено выражение E. Например, $L(c := a / b) = b \neq 0\ \&\ c = a / b$.

Построим логику L(B; C) оператора B; C, определяющего последовательное исполнение операторов B и 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. Предполагается, что наборы x, y и z – не пересекаются, а набор x может быть пустым. Определим логики указанных операторов:

$$L(B(x: z);\ C(z: y)) \cong \exists z\ L(B(x: z))\ \&\ L(C(z: y))$$
$$L(B(x: y)\ ||\ C(x: z)) \cong L(B(x: y))\ \&\ L(C(x: z))$$
$$L(\textbf{if}\ (E)\ B(x: y)\ \textbf{else}\ C(x: y)) \cong (E \Rightarrow L(B(x: y)))\ \&\ (\neg E \Rightarrow L(C(x: y)))$$

Логика программы должна быть *согласована* с формальной операционной семантикой языка P. Для произвольного оператора B(x: y) и фиксированных значений переменных наборов x и y его логика L(B(x: y)) истинна тогда и только тогда, когда любое исполнение оператора B(x: y) на наборе значений x завершается, причем результатами исполнения являются значения набора y. Отметим, что условием завершения оператора B(x: y) является формула ∃y L(B(x: y)).

Корректность программы

Допустим, программе в целом соответствует главный предикат A(x: y) со спецификацией [P(x), Q(x, y)], оператором которого является S(x: y). Тогда, корректность программы определяется следующими условиями: во-первых, предусловие P(x) должно быть истинным до исполнения оператора S(x: y), во-вторых, постусловие Q(x, y) должно быть истинным после исполнения оператора S(x: y) и, в-третьих, исполнение оператора S(x: y) всегда завершается. Данные условия корректности формулируются в виде следующих утверждений [6]:

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

Первое утверждение называется условием частичной корректности, а второе – условием завершения программы. Их конъюнкция определяет условие *тотальной (или полной) корректности* оператора S(x: y) относительно спецификации [P(x), Q(x, y)]:

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

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

Корректность рекурсивно определяемого предиката

Используется следующая схема доказательства по индукции для некоторого произвольного утверждения W(z):

$$\forall t \in X\, [\, (\forall u \in X\ m(u) < m(t) \Rightarrow W(u)\ ) \Rightarrow W(t)\,] \Rightarrow \forall z \in X\ W(z)$$

Функция m, называемая мерой, отображает X во множество натуральных чисел со стандартным отношением порядка <.

Допустим, имеется определение рекурсивного предиката A:

$$A(x: y)\ \textbf{pre}\ P(x)\ \{\ K(x: y);\ \}\ \textbf{post}\ Q(x, y)$$

Внутри оператора K(x: y) имеется рекурсивный вызов предиката A.

В соответствии со схемой индукции корректность рекурсивного предиката–может быть определена следующей формулой:

$$\forall t\ (\forall u\ m(u) < m(t) \Rightarrow Corr(A(u: y), P(x), Q(x, y))) \Rightarrow Corr(K(x: y), P(x), Q(x, y))$$

Введем формулу, которая будет обозначать индуктивное предположение для набора аргументов t:

$$Induct(t, A) \cong \forall u\ m(u) < m(t) \Rightarrow Corr(A(u : y), P(x), Q(x: y))$$

Таким образом, корректность рекурсивного предиката определяется формулой:

$$Corr(t, A, K(x: y), P(x), Q(x, y)) \cong Induct(t, A) \Rightarrow Corr(K(x: y), P(x), Q(x, y))$$

Следуя подходу Флойда, изменим предусловия у вызова предиката, дополнив их проверками монотонности по мере:

$$P^*(x) \cong m(x) < m(t)\ \&\ P(x)$$

Здесь t – формальные параметры рекурсивного определения предиката, а x – фактические параметры рекурсивного вызова.

В итоге, индукционная схема доказательства корректности рекурсивного определения предиката A представляется правилом:

$$\textbf{R0:}\ \frac{Corr(t, A, K(x: y), P(x), Q(x, y))}{Corr(A, P(x), Q(x, y))}$$

Далее условимся, что при отсутствии рекурсивного вызова внутри оператора K запись Corr(t, A, K(x: y), P(x), Q(x, y)) будет эквивалентна Corr(K(x: y), P(x), Q(x, y)). В случае нерекурсивного вызова запись P*(x) эквивалентна P(x).

Система правил вывода условий корректности

Используя формулу тотальной корректности можно автоматически построить формулу корректности оператора 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 формулы тотальной корректности для разных видов операторов позволяет декомпозировать длинную формулу корректности к нескольким более коротким и простым формулам.

В данном разделе определяется система правил вывода условий корректности для различных операторов. Однако будут опущены доказательства правил, приведенные в работах [5], [6] и [7].

В качестве аргументов предикатов в предлагаемых ниже правилах используются переменные. Нетрудно показать, что в позициях аргументов можно использовать выражения, при условии, что предусловия выражений выводимы из предусловий предикатов.

Допустим, операторы B(x: z) и C(z: y) корректны относительно своих спецификаций $[P_B, Q_B]$ и $[P_C, Q_C]$. Тогда истины следующие правила:

$$\textbf{RP:}\ \frac{\begin{array}{l} Corr(t, A, B(x: y), P_B(x), Q_B(x, y));\quad Corr(t, A, C(x: y), P_C(x), Q_C(x, z)); \\ P(x) \rightarrow P^*_B(x)\ \&\ P^*_C(x);\quad Q_B(x, y)\ \&\ Q_C(x, z) \rightarrow Q(x, y, z) \end{array}}{Corr(t, A, B(x: y) \parallel C(x: z), P(x), Q(x, y, z))}$$

RS: $$\frac{\begin{array}{l}\mathrm{Corr}(t, A, B(x: z), P_B(x), Q_B(x, z)); \quad \mathrm{Corr}(t, A, C(z: y), P_C(z), Q_C(z, y)); \\ P(x) \to (P^*_B(x) \,\&\, \forall z\, Q_B(x, z) \to P^*_C(z)); \quad P(x) \,\&\, \exists z\, Q_B(x, z) \,\&\, Q_C(z, y) \to Q(x, y)\end{array}}{\mathrm{Corr}(t, A, B(x, z); C(z, y), P(x), Q(x, y))}$$

RC: $$\frac{\begin{array}{l}\mathrm{Corr}(t, A, B(x: y), P_B(x), Q_B(x, y)); \quad \mathrm{Corr}(t, A, C(x: y), P_C(x), Q_C(x, y)); \\ P(x) \,\&\, E \to P^*_B(x); \quad P(x) \,\&\, \neg E \to P^*_C(x); \\ P(x) \,\&\, E \,\&\, Q_B(x, y) \to Q(x, y); \quad P(x) \,\&\, \neg E \,\&\, Q_C(x, y) \to Q(x, y)\end{array}}{\mathrm{Corr}(t, A, \textbf{if}\ (E)\ B(x: y)\ \textbf{else}\ C(x, y), P(x), Q(x, y))}$$

Допустим, необходимо доказать тотальную корректность оператора вызова C(B(x): y), где C – вызываемый предикат, а B(x) – набор аргументов в виде выражений, не содержащих других вызовов. Для этого необходимо воспользоваться следующим правилом:

RB: $$\frac{\begin{array}{l}\mathrm{Corr}(t, A, B(x: z), P_B(x), Q_B(x, z)); \quad \mathrm{Corr}(t, A, C(z: y), P_C(z), Q_C(z, y)); \\ P(x) \to P^*_B(x) \,\&\, P^*_C(z); \quad P(x) \,\&\, Q_C(z, y) \to Q(x, y); \quad \mathrm{SV}(P_B(x), Q_B(x, z));\end{array}}{\mathrm{Corr}(t, A, C(B(x): y), P(x), Q(x, y))}$$

В случае, если вызов рекурсивен (C совпадает с A), то доказательства корректности предиката C проводить не нужно. Здесь SV(P(x), Q(x, y)) – это утверждение об однозначности спецификации:

$$\mathrm{SV}(P(x), Q(x, y)) \quad \cong \quad P(x) \,\&\, Q(x, y1) \,\&\, Q(x, y2) \Rightarrow y1 = y2$$

Однако, не всегда известны спецификации для подоператоров. В таком случае, следует применять другие правила:

QP: $$\frac{\mathrm{Corr}(t, A, B(x: y), P(x), Q_B(x, y)); \quad \mathrm{Corr}(t, A, C(x: z), P(x), Q_C(x, z));}{\mathrm{Corr}(t, A, B(x: y) \,||\, C(x: z), P(x), Q_B(x, y) \,\&\, Q_C(x, z))}$$

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

QSB: $$\frac{\begin{array}{l}\mathrm{Corr}(t, A, B(x: z), P_B(x), Q_B(x, z, t)); \quad P(x) \to P^*_B(x); \\ \mathrm{Corr}(t, A, C(x, z: y), P(x) \,\&\, Q_B(x, z, t), Q(x, t, y))\end{array}}{\mathrm{Corr}(t, A, B(x: z, t); C(x, z: y), P(x), Q(x, t, y))}$$

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

Алгоритм генерации условий корректности

Задачей алгоритма является автоматический вывод условий корректности предикатной программы. На языке P предикатная программа представлена набором определений предикатов. Корректность подобной программы – это корректность каждого определенного предиката. Поэтому, задача сводится к выводу условий корректности для определения предиката:

$$A(x: y)\ \textbf{pre}\ P(x)\ \{\ S(x: y);\ \}\ \textbf{post}\ Q(x, y)$$

Доказательство формулы тотальной корректности предиката:

$$Corr(t, A, S(x: y), P(x), Q(x, y)) ,$$

реализуется построением дерева вывода с применением правил вывода, описанных в разделе 2. Генерируемыми условиями корректности являются листья этого дерева. Истинность формул, являющихся листьями дерева, и истинность правил влечет истинность формулы тотальной корректности.

По виду формулы определяется правило, которое необходимо применить. В процессе вывода могут встретиться два вида формул: это условие тотальной корректности оператора

$$Corr(S(x: y), P(x), Q(x, y))$$

и логическая формула

$$A_1 \& A_2 \& ... \& A_n \Rightarrow B$$

где в качестве A_i и B могут выступать как простые логические утверждения, так и логики операторов L(S(x: y)).

В том случае, если формула имеет первый вид, то применяется либо система правил для общего случая (Q), либо система правил для случая корректных подоператоров (R). Причем система правил Q более приоритетна, т.е. ее применение осуществляется в первую очередь.

Если же формулы имеет второй вид, то необходимо проверить, содержит ли она в себе логики операторов L(S(x: y)). Если не содержит, то это заключительная формула (одно из условий корректности) и она оформляется в виде леммы. Но если в ней присутствует логика, то вывод этого утверждения необходимо продолжить за счет системы правил декомпозиции L(S(x: y)) (F). Дальнейшая цель вывода – это исключение из формулы логики оператора.

Корректность алгоритма генерации условий корректности – это корректность каждого применяемого правила и корректность реализации вывода в виде программы на C++. Истинность правил вывода доказана в работах [5], [6] и [7]. В дальнейшем необходимо собрать все доказательства в одном документе. Корректность программы генератора реализуется тестированием. На данный момент работоспособность программы проверена на двух десятках тестов.

Заключение

В работе описан подход, позволяющий доказывать тотальную корректность предикатных программ. Построена система правил вывода условий корректности. Ранее правила, входящие в эту систему, существовали разрозненно, и были описаны в разных работах, а порой и под разными именами. Некоторые правила нуждались в обобщении на рекурсивный случай.

На основании данной системы правил был разработан и реализован генератор формул корректности в системе предикатного программирования. Генератор позволяет автоматически формировать условия корректности для программ с исходным кодом на языке P. В рамках генератора также было реализовано последующее упрощение полученных условий корректности, с использованием простейших законов логики и булевой алгебры. Генератор является частью системы предикатного программирования, и может быть вызван автоматически.

Дальнейшие планы. Необходимо собрать из разных документов полные доказательства правил и доказать те правила, доказательства которых отсутствуют. Разработать метод доказательства корректности предикатов со сложной рекурсией, а также для переменных предикатного типа в качестве параметров предикатов. Разработать правила для доказательства корректности гиперфункций.

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. Предиктное программирование. Учебное пособие / Под ред. Шелехова В.И. НГУ. Новосибирск, 2009. 111 С.

5. Предиктное программирование. Лекции / Под ред. Шелехова В.И. ИСИ СО РАН. Новосибирск, 2011.

6. *Шелехов В.И.* Методы доказательства корректности программ с хорошей логикой // Межд. конф. "Современные проблемы математики, информатики и биоинформатики", посвященная 100-летию со дня рождения А.А. Ляпунова. — 2011. — 17с. http://conf.nsc.ru/files/conferences/Lyap-100/fulltext/74974/75473/Shelekhov_prlogic.pdf

7. *Кулямин В.В.* Методы верификации программного обеспечения // Институт системного программирования РАН. — 2008.

8. *S. Owre, N. Shankar, J. M. Rushby, D. W. J. Stringer-Calvert.* PVS Language Reference.

9. Wikipedia. The free encyclopedia. http://en.wikipedia.org/

10. *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.

11. *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.

12. *Ершов Ю.Л., Палютин Е.А.* Математическая логика: Учебное пособие для вузов – 2-е изд., испр. и доп. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 336 с.