УДК 519.681

Н.С. Езова, Д.Б. Кузнецов

Пермский национальный исследовательский
политехнический университет

ВЕРИФИКАЦИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Классическим способом доказательства правильности программ является использование системы Дейкстры. Эта система изначально предназначена для однопоточной последовательной программы. Рассматривается возможность применения системы Дейкстры к верификации параллельных программ.

Концептуальную основу программирования последовательных вычислений составляет понятие алгоритма, реализуемого по шагам строго последовательно во времени. В противоположность этому параллельная программа порождает совокупность одновременно протекающих процессов обработки информации, полностью независимых или связанных между собой статическими или динамическими пространственно-временными или причинно-следственными отношениями.

Параллельное программирование, суть которого заключается в использовании преимуществ многоядерных или многопроцессорных компьютеров – разделении процесса на несколько выполняющихся «параллельно» потоков, сложнее реализовывать, чем последовательное, поэтому его необходимо подвергать строгому анализу и проверке с помощью различных методов верификации.

Классические методы верификации Р. Флойда [4], Ч. Хоара [5] и Э. Дейкстры [3] рассчитаны на императивные однопоточные программы.

Тем не менее, эти проверенные временем методы можно использовать в параллельном программировании, подвергнув некоторым модификациям.

Книга Ю.Г. Карпова [2] в данном случае подходит не в полной мере, потому что там рассматривается метод формальной верификации m*od*el check*in*g, для применения которого система представляется

в виде переходов с конечным числом состояний. Здесь же не рассматривается множество состояний, переходы и разметка в качестве основы верификации [2].

Для верификации программы необходимо задаться критерием ее правильности – спецификацией – средством для точного описания того, что должно быть совершено в результате выполнения программы.

Спецификация по Дейкстре [1, 3] задается с помощью двух предикатов: предусловие $\{Q\}$ и постусловие $\{R\}$. Таким образом, сама спецификация также является предикатом $\{Q\}\ S\ \{R\}$, который истинен: если выполнение программы S началось в состоянии, удовлетворяющем Q, то оно завершится через конечное время в состоянии, удовлетворяющем R.

Слабейшим предусловием является условие, характеризующее множество всех начальных состояний, при которых выполнение обязательно приведёт к правильному завершению, причём система останется в конечном состоянии, удовлетворяющем заданному постусловию. Оно обозначается $wp(S,R)$ [3].

При доказательстве правильности цикла, кроме пред- и постусловия, используются инвариант и ограничивающая функция.

Инвариант P – логическое выражение, описывающее зависимости переменных цикла. Оно должно быть истинно после каждого прохода тела цикла и перед началом выполнения цикла.

Аннотированный цикл должен выглядеть следующим образом:

do

$B_1 \rightarrow S_1$

$\square B_2 \rightarrow S_2$

…

$\square B_n \rightarrow S_n$

od

где B – служит охраной входа ($\rightarrow$) – условие; S – выполняется только при истинности B; $B \rightarrow S$ – охраняемая команда; $\square$ – равнозначно *elseif* – команде выбора, разделитель вариантов; *do…od* – оператор из набора охраняемых команд. Если нет ни одной охраны с истинным значением, происходит правильное завершение программы, но при этом работе нельзя завершиться, пока хотя бы одна охрана является истинной.

По теореме о цикле [1, 3] для проверки существует ряд условий:

1. P истинно перед выполнением цикла $\{Q\}$ – начальная инициализация переменной $\{P\}$.

2. P является инвариантом цикла $\{P\&B_i\}S_i\{P\}$. (1)

3. Выполнение P и невыполнение BB должны дать R: $P\&\neg BB \Rightarrow R$.

4. Если цикл еще не закончен, то ограничивающая функция положительна: $P \,\&\, BB \Rightarrow t > 0$.

5. Каждый шаг цикла ведет к концу цикла $\{P \,\&\, B_i\}\ t_1{:=}t; S_1\ \{t{<}t_1\}$.

Для применения метода Дейкстры происходит выделение в параллельной программе слоёв за счёт разбиения на последовательность операторов каждого её процесса. При этом первым слоем будут считаться все те операторы, которым требуются только входные параметры и константы. После того как эти операторы окажутся вычисленными, на следующем шаге для вычисления выбираются все те операторы, которые имеют на своем входе вычисленные данные; таким образом формируется следующий слой [6]. Порядок выполнения данных слоёв не важен, поскольку взаимодействие процессов происходит только внутри них, то есть слои являются коммуникационно-замкнутыми. Программа называется безопасной, если все её слои коммуникационно-замкнутые. В этом случае вместо верификации всей параллельной программы можно проводить её послойную верификацию – последовательное доказательство вход-выходных соотношений для каждого слоя. Иными словами, каждый слой представляет собой последовательную программу [7], к которой применим метод Дейкстры, что было неоднократно доказано.

Рассмотрим применение метода Дейкстры для верификации параллельной программы, вычисляющей значение $A^n + B^n$ через сложение и умножение. Следует учесть, что данный пример можно реализовать и при помощи последовательного алгоритма.

Первый этап построения программы – составление спецификации.

$$\begin{aligned} &\{Q : n > 0 \,\&\, A \geq 0 \,\&\, B \geq 0\}, \\ &\{R : x = A^n + B^n\}. \end{aligned} \tag{2}$$

Сначала построим последовательную программу.

По постановке задачи, очевидно, что в программе будет цикл. Для этого спецификация должна быть дополнена инвариантом P и ограничивающей функцией t:

$$\{P : x = a + b \,\&\, a = A^i \,\&\, b = B^i \,\&\, 0 < i \le n\},$$
$$\{t : n - i\}. \tag{3}$$

Построенная, исходя из спецификации, программа приведена на листинге 1:

$x, i, a, b := A + B, 1, A, B;$

do

$\underbrace{i \ne n}_{B} -> \underbrace{a, b := a * A, b * B; \; x, i := a + b, i + 1;}_{S}$

od

Листинг 1

Выполним один из элементов верификации – проверку цикла по формуле (1) Дейкстры – проверка инварианта.

$$\{P \,\&\, B\} S \{P\} \tag{4}$$

$P \,\&\, B => wp(S, P)$

$x = a + b \,\&\, a = A^i \,\&\, b = B^i \,\&\, 0 < i \le n \,\&\, i \ne n =>$

$=> wp("a, b := a * A, b * B; x, i := a + b, i + 1", x = a + b \,\&\, a =$

$= A^i \,\&\, b = B^i \,\&\, 0 < i \le n)$

$x = a + b \,\&\, a = A^i \,\&\, b = B^i \,\&\, 0 < i \le n \,\&\, i \ne n => a * A =$

$= A^{i+1} \,\&\, b * B = B^{i+1} \,\&\, 0 < i + 1 \le n$

Импликация истинна, поскольку:

$a = A^i => a * A = A^{i+1}$

$b = B^i => b * B = B^{i+1}$

$i \le n \,\&\, i \ne n => i + 1 \le n$

$0 < i => 0 < i + 1$

Преобразуем последовательную программу в параллельную таким образом, чтобы она порождала три процесса:

1. Процесс вычисления A^i.

2. Процесс вычисления B^i.

3. Процесс вычисления $A^i + B^i$.

$a, i_a := A, 1;$	$i := 1;$	$b, i_b := B, 1;$
$!a;$	$?a; ?b;$	$!b;$
	$x := a + b;$	
do	do	do
$i_a \neq n -> a, i_a := a * A, i_a + 1;$	$i \neq n -> ?a; ?b;$	$i_b \neq n -> b, i_b := b * B, i_b + 1;$
$!a;$	$x, i := a + b, i + 1;$	$!b$
od	od	od

Листинг 2

Здесь:

– $!a$ передача в канал a значения переменной a,

– $!b$ передача в канал b значения переменной b,

– $?a$ получение из канала a значения переменной a,

– $?b$ получение из канала b значения переменной b.

Для анализа параллельной программы построим сеть Петри для порождаемых ее процессов (рисунок).

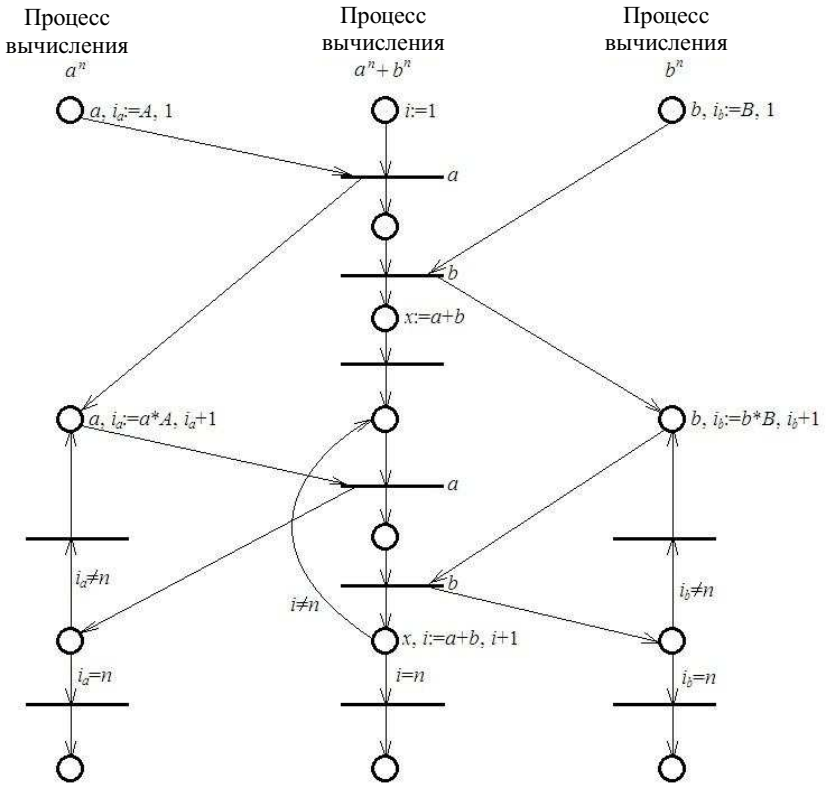

Рис. Сеть Петри

На рисунке:

– места маркируются операторами присваивания,

– переходы названиями каналов,

– немаркированные места/переходы использованы для соответствия правилам построения сетей Перти,

– дуги промаркированы только в случаях недетерминизма в выборе перехода.

Заданный для последовательной программы (листинг 1) инвариант P должен выполняться в параллельной программе (листинг 2) после места на рисунке, маркированного как «$x, i := a + b, i + 1$».

Тело цикла последовательной программы в параллельной программе состоит из двух коммуникационно-замкнутых слоев: первому слою цикла соответствуют места (см. рисунок), маркированные как «$a, i_a := a * A, i_a + 1$» и «$b, i_b := b * B, i_b + 1$», второму слою – «$x, i := a + b, i + 1$».

Таким образом, спецификацию для выполнения команд, входящих в коммуникационно-замкнутые слои цикла, по аналогии со второй проверкой цикла Дейкстры можно записать в следующем виде:

$\{i_a \neq n \,\&\, i_b \neq n \,\&\, i \neq n \,\&\, P\}$ <команды 1-го слоя>;<команды 2-го слоя>$\{P\}$.

В инвариант, по сравнению с использованным в последовательной версии, необходимо добавить соотношение между счетчиками циклов $i = i_a = i_b$.

Команды первого слоя, как выполняющиеся одновременно и независимо, можно подставить в формулу в любом порядке. Важно, что команды второго слоя должны строго следовать за командами первого.

После этих преобразований можно воспользоваться обычной формулой Дейкстры для доказательства правильности спецификации:

$i_a \neq n \,\&\, i_b \neq n \,\&\, i \neq n \,\&\, x = a + b \,\&\, a = A^i \,\&\, b = B^i \,\&\, i = i_a = i_b \,\&\, 0 < i \leq n \Rightarrow$

$\Rightarrow wp("a, i_a := a * A, i_a + 1; b, i_b := b * B, i_b + 1; x, i := a + b, i + 1",$

$i = i_a = i_b \,\&\, 0 < i \leq n \,\&\, x = a + b \,\&\, a = A^i \,\&\, b = B^i)$.

Доказывается истинность импликации аналогично варианту с последовательной программой (4).

В примере рассмотрена самая важная и сложная проверка цикла – на сохранение инварианта. Остальные проверки строятся аналогично.

Библиографический список

1. Грис Д. Наука программирования. – М.: Мир, 1984. – 416 с.

2. Карпов Ю.Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010. – 560 с.

3. Дейкстра Э. Дисциплина программирования. – М.: Мир, 1978. – 276 с.

4. Floyd R.W. Assigning Meaning to Programs // Proceedings of Symposium on Applied Mathematics. – 1967. – Vol. 19. (J.T. Schwartz (Ed.), A.M.S.). – P. 19–32.

5. Hoare C.A.R. An axiomatic basis for computer programming // Communs ACM. – 1969. – Vol. 12, N 1. – P. 576–580.

6. Удалова Ю.В., Легалов А.И., Сиротинина Н.Ю. Методы отладки и верификации функционально-потоковых параллельных программ. – Красноярск: Изд-во Сибир. федерал. ун-та, 2011. – 224 с.

7. Малышкин В.Э., Корнеев В.Д. Параллельное программирование мультикомпьютеров. – Новосибирск: Изд-во Новосиб. гос. техн. ун-та, 2006. – 439 с.

Получено 05.09.2012