Математические вопросы кибернетики и вычислительной техники 17, 1997, 46-51.

О полной системе примеров для обнаружения конфликтов взаимодействующих программ

А. Ю. Шукурян

Институт проблем информатики и автоматизации

НАН РА и ЕрГУ

Одним из важных направлений в анализе алгоритмов и программ является проверка их правильности на тестовых примерах [1,2,4,5]. В настоящей работе на основе подхода к построению полной системы тестовых примеров для программ, впервые предложенного Я.М.Барздинем [1], рассматривается задача, связанная с тестированием взаимодействующих программ. Необходимость изучения этого класса программ обосновывается тем, что к ошибкам, возникающим при распределенной обработке относятся так называемые ошибки ситуаций или конфликтов. Они проявляются тогда, когда процесс А ждет окончания процесса В, в то время, когда процесс В ждет окончания процесса А. К ним относятся также ошибки семафора в операционных системах [5]. С аналогичной ситуацией можно встретиться при сетевой организации информационного обеспечения банковской деятельности, моделировании взаимодействующих физических процессов, проектировании сетевых протоколов [3]. Полученный результат касается установлению разрешимости проблемы построения полной системы тестовых примеров, обнаруживающих конфликты для подкласса взаимодействующих программ.

Определения и обозначения

В основном используется терминология работ [1,2]. *Абстрактная машина* Барздиня имеет конечное множество X входных лент $X_1, X_2, \ldots X_n$, конечное множество Y выходных лент $Y_1, Y_2, \ldots Y_m$, и конечное множество внутренних ячеек (рис.1). Ленты разделены на ячейки, в каждой из которых может быть записано

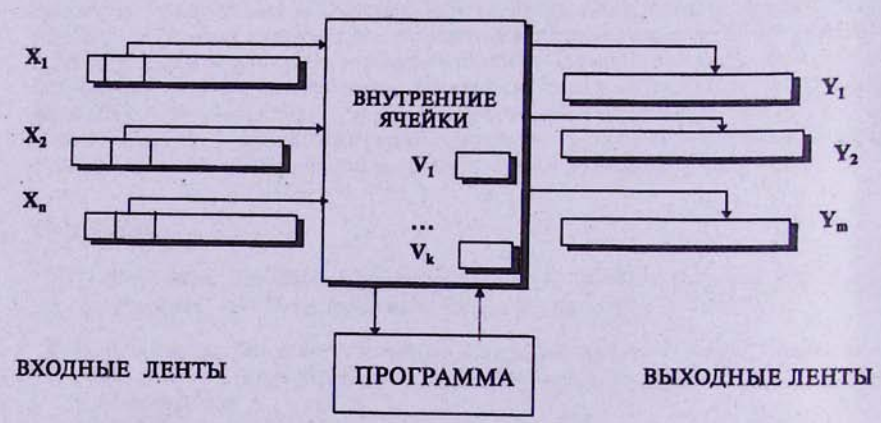

Рис.1 Машина Барздиня

произвольное целое число. Каждая входная (выходная) лента снабжена одной считывающей (записывающей) головкой. Машина имеет систему команд, последовательность которых образует программу.

Опишем систему команд абстрактной машины:

а) $v_j := X_i$. Содержимое обозреваемой считывающей головкой входной ленты X_i записывается во внутреннюю ячейку v_j . Команда является условной, т.е. имеет два выхода " + " и " - ". Выход " + " соответствует случаю, когда в обозреваемой считывающей головкой входной ленты X_i записано целое число. В этом случае после передачи числа в ячейку v_j считывающая головка ленты X_i перемещается на одну ячейку вправо. Выход " - " соответствует случаю, когда обозреваемая ячейка X_i пуста: содержимое внутренней ячейки v_j не меняется, сдвига головки на ленте не происходит.

б) $Y_i := v_j$. Целое число, содержащееся во внутренней ячейке v_j записывается в обозреваемую записывающей головкой ячейку выходной ленты Y_i. Головка перемещается на одну ячейку вправо. Команда безусловная, т.е. имеет один выход.

в) $v_i := v_j$. Содержимое v_j переписывается в v_i .

г) $v_i := c$. Во внутреннюю ячейку v_i записывается константа c .

д) Команды сравнения. Каждая команда имеет вид $v \sigma w$, где v, w - внутренние ячейки или целые числа, $\sigma \in \{\geq, \leq\}$. Команда имеет два выхода. По выходу " + " управление передается в случае выполнения проверяемого условия. В противном случае управление передается по выходу " - " команды.

е) НОП. Холостая команда, никаких действий не производится, управление передается по единственному выходу команды.

ж) СТОП. Машина останавливается, команда выходов не имеет.

Обозначим через $\mathfrak{R}$ систему команд а)- ж).

Программой в системе команд $\mathfrak{R}$ называется ориентированный граф, каждая вершина которого отмечена какой-либо командой из $\mathfrak{R}$. Каждая вершина, кроме одной, выделенной в качестве начальной, имеет хотя бы одного предшественника. Вершины, отмеченные условными командами, имеют двух преемников, вершины, отмеченные безусловными командами, имеют одного преемника. Вершины, отмеченные командой СТОП, преемников не имеют. Преемники вершины, отмеченной условной командой, называются " + " и " - " премниками соответственно. Дуга, ведущая из условной вершины к ε преемнику обозначается ε ($\varepsilon \in \{+, -\}$). Для удобства изложения будем полагать, что все вершины перенумерованы и преемник безусловной команды является ее " + " преемником. Будем отождествлять вершины программы с отмеченными командами.

Пусть зафиксирована машина Барздиня.

Конфигурацией программы называют систему

$$C = (s, u_0, \ldots, u_m) \qquad (1)$$

где s - вершина программы, u_0 - n - мерный вектор целых чисел, составленный из содержимых внутренних ячеек, u_i - последовательность целых чисел, записанных в ячейках входной ленты, справа от ячейки, обозреваемой считывающей головкой. Если эта ячейка пуста, то $w_i = \varnothing$.

Конфигурация $(S', u_0', \ldots, u_m')$ называется 1- достижимой из конфигурации $(S, u_0, \ldots, u_m)$, если выполняется одно из следующих условий:

1. S отмечена одной из команд б),с), S' является преемником S и $u_i' = u_i$,(i=0,...,m)
2. S отмечена командой а) ($v_j := X_i$), S' является "-" преемником S, $u_i' = u_i$, (i= 0,...,m), $u_i = \varnothing$
3. S отмечена командой а), ($v_i := X_j$) ,S' является " + " преемником S, $u_j = a_1, \ldots, a_k$, $v_i' = a_1, u_t' = u_t$, t= 1, 2,..., j-1, j+1,..., m, $u_j' = a_2, \ldots, a_k$; $v_t = v_t'$, t= 1, 2, ..., j-1, j+1, ...,n

4. S отмечена командой в) ($v_i := v_j$), S′ является преемником S, $u_t' = u_t$ для t=1, 2, ..., m, $v_t' = v_t$ для $t \neq i$, $v_i' = v_j$

5. S отмечена командой г), S′ является преемником S, $u_t' = u_t$ для t=1, 2, ..., m, $v_t' = v_t$ для $t \neq i$, $v_i' = c$.

6. S отмечена командой сравнения, S′ является " + " (" - ") преемником S, $u_t' = u_t$ для t=1,2,...,m и подстановка значений в соответствующие ячейки из u_0 обращают условие в И (Л).

Если в конфигурации (1) S = СТОП, то С называется *заключительной*.

Вычислением по программе Р *над примером* $(u_0^0, u_1^0, \ldots, u_m^0)$ называется последовательность конфигураций $C_0 = (s_0, u_1^0, \ldots, u_m^0), C_1, \ldots$, где S_0 - начальная конфигурация и C_{i+1} 1 - достижима из C_i. Если при некотором n конфигурация C_n заключительная, то вычисление называется *терминальным,* а пример $(u_0^0, u_1^0, \ldots, u_m^0)$ - *допустимым* для программы Р.

Путем в программе называется последовательность $S_1\varepsilon_1 \ldots S_k\varepsilon_k \ldots$, где вершина S_{i+1} является ε_i - преемником вершины S_i, $i = 1,2,\ldots$, $\varepsilon \in \{+, -\}$. Путь, соответствующий вычислению (последовательность кофигураций) над некоторым примером, называется также путем, *активизируемым* в программе этим примером. Если пример является допустимым, то активизируемый им путь называется *реализуемым.*

Конечное множество допустимых примеров называется *полной системой примеров* (ПСП) для программы Р, если каждая реализуемая дуга программы Р принадлежит пути, акивизируемому некоторым примером из Σ. Говорят, что проблема построения ПСП для программ из некоторого класса Э разрешима, если существует алгоритм, который по любой программе $P \in Э$ строит ПСП для Р.

Имеет место

Теорема 1 [1]. *Проблема построения ПСП для программ в системе команд* $\mathfrak{R}$ *разрешима.*

Приведем некоторые основные моменты, связанные с этим результатом. Состоянием программы после выполнения пути α, начинающегося с начальной вершины, называется некоторое множество соотношений F_α, обладающее свойством: для любых путей α и β, завершающихся дугами, ведущими в одну и ту же вершину программы, из идентичности соотношений F_α и F_β следует, что множество реализуемых продолжений путей α и β совпадают. Очевидно, что такое определение предполагает существование алгоритма, который по пути α строит состояние F_α программы после выполнения пути α.

Простым примером состояния программы в системе команд $\mathfrak{R}$ является описание состояний внутренних ячеек и входных лент, входящее в описание конфигурации. В самом деле, пусть γ - общее продолжение путей α и β. идентичность F_α и F_β означает покомпонетное равенство описаний значений внутренних ячеек и входных лент в конфигурациях программы после выполнения путей α и β. Тогда очевидна эквивалентность условий реализуемости пути γ из обоих состояний.

В случае конечности числа различных возможных состояний некоторой программы Р для определения ПСП применяется метод, основанный на построении *дерева развертки программы* Р [1, 2]. Оно строится поярусно. Корень дерева (нулевой ярус) соответствует первой выполняемой команде программы Р. Пусть S_0 - номер этой команды. Первый ярус дерева образуют команды, являющиеся преемниками команды с номером S_0, второй ярус составляют команды, являющиеся преемниками команд из первого яруса и т. д. Каждый узел дерева идентифицируется

номером соответствующей команды программы P и состоянием программы после выполнения пути программы, ведущего из корня дерева в данный узел.

Построение развертки выполняется с учетом следующих требований:

а) путь, ведущий из корня дерева в любой ее узел должен быть реализуем;

б) на пути, ведущем из корня дерева в некоторый узел, не должно быть другого узла с тем же идентификатором.

Ветви дерева развертки обрезаются по достижению некоторого узла, для которого одно из приведенных правил не выполняется. Если число состояний программы конечно, то процесс построения развертки завершается построением конечного дерева. Правило а) обеспечивает реализуемость всех путей из корня дерева в его узлы, а из определения программы следует, что всякая реализуемая дуга программы принадлежит некоторому пути из корня дерева в узел, соответствующий некоторой команде СТОП программы. Таким образом ПСП для программы P составят примеры, реализующие все те пути, которые соответствуют путям из корня развертки в узел, соответствующий команде СТОП.

Рассмотрим программу P в системе команд $\Re$. Заметим, что в команде сравнения делается сравнение внутренних ячеек на $\geq$, $\leq$ и внутенних ячеек с константами, определим свстояние программы как двоичную таблицу, столбцами которой являются внутренние переменные и константы , а строками - внутренние переменные. Клетка (v_i, v_j) в данном состоянии равна 1, если $v_i \leq v_j$ и 0 в противном случае. Аналогично (v_i, c_t) клетка равна 1, если $v_i \leq c_t$. Значение клетки равно нулю в противном случаях соответственно. Понятно, что таких состояний (таблиц) конечное множество. Легко проверить, что указанные таблицы являются состояниями программы P (они могут изменяться при чтении со входных лент и при пересылках между внутренними ячейками, однако их число конечно и не зависит от заполнений входных лент). Поэтому граф развертки будет конечным.

Тестовые примеры для обнаружения конфликтов

В настоящее время распределенная обработка информации играет большую роль в таких различных областях как информационное обеспечение банков, автоматизация физического эксперимента. Характерным для этого способа обработки является наличие взаимодействующих программ. частными примерами являются программы, реализующие протоколы передачи данных или программы клиентов, взаимодействующих через общую память серверов и т.д. Исходной для формулировки рассматриваемой задачи является возможность параллельного выполнения программы с некоторым распределением вычислительных ресурсов. При этом важной является разработка формализованных методов генерации тестовых примеров, обеспечивающих полную проверку правильности функционирования взаимодействующих программ.

Рассмотрим машину Барздиня и предположим, что даны разбиения множества входных и выходных лент

$$X = X_A \cup X_B \,,\; Y = Y_A \cup Y_B,$$
$$X_A \cap X_B = \varnothing,\; Y_A \cap Y_B = \varnothing.$$

Пусть далее множество внутренних ячеек имеет вид $V = V_A \cup V_B$, $V_A \cap V_B = V_{AB} = \varnothing$ и даны две программы P_A и P_B в системе команд $\Re$ такие, что X_A , Y_A , V_A являются лентами и внутренними ячейками, используемыми командами программмы P_A , а X_B , Y_B , V_B - используемыми командами программмы P_B . Предполагается, что программы выполняются на машине одновременно. При этом возможно, что выполнение осуществляется запуском двух таких команд S_A и S_B, которые образуют несовместимую пару. Последнее ознзчает, что имеет место одно из следующих условий:

а) S_A и S_B осуществляют присваивание одной и той же внутренней ячейке;
б) S_A (S_B) вляется командой сравнения с внутренней ячейкой, а S_B (S_A) присваивает этой же ячейке значение;
в) S_A = НОП, S_B = СТОП.

При выполнении этих условий будем говорить, что имеет место *конфликт* между взаимодействующими программами.

Примером называется некоторое заполнение массивов входных лент и внутренних ячеек. Пример,на котором выявляется конфликт, называется примером, обнаруживающим конфликт между программами P_A и P_B.

Конечная система примеров Σ называется полным тестом, обнаруживающим конфликт взаимодействующих программ, если конфликт при одновременном выполнении программ P_A и P_B возможен тогда и только тогда, когда существует пример, такой, что одновременное выполнение программ P_A и P_B над этим примером завершается аварийно.

Задача состоит в исследовании разрешимости проблемы построения полного теста (ППТ), обнаруживающего конфликт в произвольных двух программах в системе $\mathfrak{R}$.

Расширим понятие программы для групповых команд. Рассмотрим систему команд $\mathfrak{R}$. Введем понятие групповой системы команд. Пару команд S_1 и S_2 назовем совместимой, если:
1) они не осуществляют присваивание одной и той же переменной;
2) если одна из них является командой сравнения с внутренней ячейкой, то другая не присваивает этой же ячейке некоторое значение;
3) S_1 = НОП, S_2 = НОП или S_1 = СТОП, S_2 = СТОП;
4) S_1 и S_2 являются командами сравнения.

Заметим, что пара (S_1 , S_1) если не является командой сравнения, является несовместимой. Подмножество S_1 , ... , S_n команд назовем *групповой командой*, если любая пара S_i , S_j является совместимой, либо n = 1 . Систему всевозможных групповых команд обозначим через Г($\mathfrak{R}$). Ясно, что $\mathfrak{R} \subseteq Г(\mathfrak{R})$.

Для того, чтобы говорить о программе в системе групповых команд, сделаем уточнение. Пусть γ и $\gamma \in Г(\mathfrak{R})$ содержит r ($r \neq 0$) команд сравнения. Тогда эта команда имеет 2^r выходов (в соответствии со всеми наборами значений r команд сравнения). По аналогии с вышеприведенным понятием программы в системе $\mathfrak{R}$ программой в системе Г($\mathfrak{R}$) назовем ориентированный начальная и СТОП вершины. Преемники, отмеченные командой, включающей r команд сранения, имеют 2^r преемников и дуга, ведущая из этой граф, каждая вершина которого отмечена какой-либо командой из Г($\mathfrak{R}$) . Выделены вершины к ε преемнику, обозначается ε ($\varepsilon \in 2^r$).

Лемма. *Для любой программы в системе* Г($\mathfrak{R}$) *групповых команд разрешима проблема построения полной системы примеров.*

Легко показывается,что для любой программы P в системе Г($\mathfrak{R}$) можно построить эквивалентную ей программу P′ в системе команд. Для этого достаточно выполнять команды групповой команды последовательно.

Основной результат формулируется следующим образом.

Теорема 2. *Для любой пары программм P_A и P_B в системе команд $\mathfrak{R}$ существует программа P_{AxB} в системе команд Г($\mathfrak{R}$) такая, что полный тест, обнаруживающий конфликт взаимодействующих программм P_A и P_B совпадает с полной системой примеров для P_{AxB} .*

Приведем основные шаги доказательства. Рассмотрим графы программ и построим "прямое произведение" этих графов в следующем смысле. В качестве вершин графа возьмем всевозможные пары вершин (a,b), $a \in P_A$, $b \in P_B$. Начальной вершиной объявим пару, состоящую из начальных вершин программ P_A и P_B. Каждой

вершине, образованной из пар (a,b) припишем групповую команду, состоящую из двух команд из $\Re$, приписанных вершинам a и b в программах P_A и P_B соответственно. Определим все вершины, для которых образовавшаяся групповая команда является несовместимой и отметим эти вершины командами СТОП. Все вершины, для которых соответствующая пара команд имеет вид (СТОП, СТОП) заменим командами вида (НОП, НОП) с переходами на себя. Для построенной программы P_{AxB} ПСП и составит полный тест, обнаруживающий конфликты для взаимодействующих программ P_A и P_B.

Литература

1. Я.М.Барздинь и др. Построение полной системы примеров для проверки программ.В сб." Уч.записки Латвийского ГУ", 210, Рига, 1974г.

2. А.Г.Тадевосян. О разрешимости проблемы построения полной системы примеров в одном классе программ. ДАН Армении, т. LXXII,N 3, 1981г.

3. Ю.К.Апраксин. К вопросу автоматизации проектирования протоколов вычислительных сетей на основе конечноавтоматных моделей. Автоматика и вычислительная техника, N2, 1987г.

4. Г.Мейерс. Надежность программного обеспечения. М., "Мир" ,1980г.

5. Дж.Шварц. Обзор ошибок. Средства отладки больших систем. Москва, 1977г.