МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1981 год Выпуск 89

УДК 681.324:681.142.4

ПРОВЕРКА ФУНКЦИОНИРОВАНИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ МНОГОМАШИННОГО КОМПЛЕКСА

Ю.И. Колосова

Развитие современных технологий программирования вызвано, в первую очередь, возросшими требованиями к надежности и эффективности создаваемых сложных программных комплексов [1,2]. Одно из важных мест в технологических разработках занимают программные средства, обеспечивающие контроль функционирования создаваемых комплексов и их отдельных составляющих модулей. Основное назначение средств контроля заключается в своевременном выявлении фактов отказов или сбоев, в накоплении, обобщении и анализе характеристик работоспособности таких комплексов [3].

В настоящей работе предлагается метод проверки отдельных функций операционных систем многомашинных комплексов. Метод заключается в создании программного блока, который реализует Модель проектируемой Операционной системы (МОСТ), точнее ее определенной части. Блок МОСТ помещается в существующий ОТЛАДЧИК программ в качестве составляющего модуля и проверяется на конкретных исходных данных в процессе функционирования ОТЛАДЧИКА.

Описываемый здесь блок МОСТ ориентирован на проверку той части операционной системы (ОС), которая обеспечивает режим параллельной обработки в смысле вычислительного комплекса МИНИМАКС [4]. Этот режим используется для решения задач, представленных параллельными программами и занимающих все ресурсы подсистемы комплекса. Блок включает в себя два основных типа подпрограмм, записываемых на МНЕМОКОДЕ мини-ЭВМ "М-6000". Подпрограммы одного типа непосредственно выполняют функции проектируемой части ОС и после их проверки могут быть включены в состав новой ОС без каких-либо изменений. Подпрограммы другого типа только имитируют некоторые

функции ОС и позволяют исследовать логику их работы. Исходными (проверочными) данными блока МОСТ являются параллельные программы для комплекса МИНИМАКС.

I. Вводные замечания

Под параллельной программой, выполняемой на комплексе МИНИМАКС [4], понимается организованная совокупность последовательных программ (ветвей), выполняющихся одновременно на отдельных машинах и взаимодействующих между собой в соответствии с требованиями алгоритма.

Различаются взаимодействия двух типов: групповые и индивидуальные. Организация таких взаимодействий производится под управлением операционной системы.

Первый тип взаимодействий осуществляется между всеми ветвями параллельной программы и задается оператором, присутствующим в каждой ветви. Групповое взаимодействие осуществляется как только все ветви в процессе своего выполнения выйдут на оператор, задающий такое взаимодействие.

Второй тип взаимодействий осуществляется между двумя ветвями (инициатором и инициируемой) и задается оператором, присутствующим только в ветви-инициаторе. Индивидуальное взаимодействие осуществляется как только ветвь-инициатор получит сведения о наступлении события, связываемого с истинным значением определенной логической функции над полем заданных семафоров. При невыполнении условия синхронизации ветвь-инициатор ставится в очередь ждущих процессов и переводится в состояние динамического останова либо выполнения фоновой программы. При выполнении условия синхронизации осуществляется индивидуальное взаимодействие, преобразуются семафорные переменные в соответствии с дисциплиной их обслуживания, просматривается очередь ждущих процессов. Для всех составляющих очереди, при выполнении условия синхронизации, осуществляется соответствующее индивидуальное взаимодействие.

2. Организация блока МОСТ

В любой момент выполнения каждая из ветвей параллельной программы находится в определенном состоянии k. Например: ветвь вышла на оператор группового или индивидуального взаимодействия, выполняет фоновую программу, выполнила некоторое заданное число команд (заданный интервал), обрабатывает переполнение и т.п.

Состояние ветви j ($j = 1,2,\ldots, L$; L – число ветвей параллельной программы), будем характеризовать значением вектора $Y_j = (y_{j1}, y_{j2}, \ldots, y_{jn})$, где n – число всевозможных состояний, в которых могут находиться ветви. Каждая составляющая y_{jk} ($k = 1, 2, \ldots, n$) вектора Y_j является двоичной переменной, значение которой равно 1, если ветвь вошла в состояние k .

Совокупность состояний всех ветвей параллельной программы в некоторый момент выполнения образует **событие выполнения ветвей** в этот момент. Такое событие будем характеризовать значением вектора $X = (x_1, x_2, \ldots, x_n)$, каждая составляющая которого x_i ($i = 1,2,\ldots,n$) есть дизъюнкция соответствующих составляющих y_{ji} векторов Y_j ($j = 1,2,\ldots, L$), т.е.

$$x_i = \bigvee_{j=1}^{L} y_{ji}, \quad i = 1,2,\ldots,n .$$

Таким образом, вектор $X = (x_1, x_2, \ldots, x_n)$ представляет собой n-разрядный двоичный символ и определен на конечном множестве из 2^n различных элементов.

Блок МОСТ можно рассматривать как конечный автомат, определяемый упорядоченной последовательностью $A = (X, R, D, \gamma, r_0, r_\beta)$, где X – конечное множество (алфавит) символов или конечное множество событий на входе в блок МОСТ; R – конечное множество состояний блока, $r_0, r_\beta \in R$; D – конечное множество действий блока, осуществляющих управляющие воздействия на каждую из ветвей параллельной программы в соответствии с событием; γ – отношение на множестве $R \times X \times D \times R$, заданное в виде набора правил:

$$r_i: \begin{array}{lll} X_{i1} & D_{i1} & r_{i1}, \\ \cdot & \cdot & \cdot \\ X_{im} & D_{im} & r_{im}, \\ \cdot & \cdot & \cdot \\ X_{ib} & D_{ib} & r_{ib}. \end{array}$$

Здесь $x_{i\delta} \in X$, $D_{i\delta} \in D$, $r_{i\delta} \in R$ ($\delta = 1,2,\ldots,b$) .

В состоянии автомата r_i на его входе разрешены события $X_{i1}, X_{i2}, \ldots, X_{im}, \ldots, X_{ib}$. При наступлении события X_{im} выполняется подмножество действий D_{im} и автомат переходит в состояние r_{im} . При появлении на входе символа, отличного от предусмотренных в правиле r_i , происходит выдача соответствующей информации.

На практике для задания символов на входе автомата целесообразнее использовать вместо множества X совокупность двух множеств $d \cdot X$ и S, между которыми определено взаимно-однозначное соответствие. Здесь S - конечное множество событий (символов) на входе автомата, допустимых на определенном шаге исследования ОС ($S \subset X$); d - константа (фильтр), обеспечивающая эквивалентность символа, поступающего на вход автомата, элементам множества S.

Представление блока МОСТ в виде конечного автомата делает прозрачной его реализацию, позволяя использовать для описания алгоритма его работы таблицу переходов [3] и интерпретирующую программу.

3. Функционирование блока МОСТ

Сущность описываемого метода заключается в погружении блока МОСТ в среду ОТЛАДЧИКА программ и использовании им в процессе своей работы сервиса и возможностей последнего.

В основные функции блока МОСТ входят:

- анализ событий при выполнении ветвей;
- осуществление управляющих воздействий на каждую из ветвей в зависимости от конкретного события;
- работа с семафорными переменными и с очередью ждущих процессов.

Работа блока по анализу события в основном совпадает с аналогичной работой предполагаемой операционной системы. Входной алфавит блока определяется как множество событий, последовательно распознаваемых на входе. Распознавание слов соответствует функции синтаксиса некоторой автоматной грамматики. В то же время алгоритм получения значений векторов Y_j ($j = 1,2,\ldots,L$), т.е. сбор информации о состояниях ветвей перед входом в блок МОСТ, различен. При работе блока в среде ОС (т.е. после включения блока в работающую ОС) значения составляющих вектора Y_j он получает посредством программного сбора соответствующей информации через модуль межмашинной связи. При работе блока в среде ОТЛАДЧИКА, интерпретирующего выполнение каждой команды ветви, значения указанных составляющих сообщаются блоку модулями ОТЛАДЧИКА, фиксирующими вход в конкретное состояние.

Работа блока по осуществлению управляющих воздействий (множество D) имитирует аналогичную работу операционной системы и соответствует функции семантики автоматной грамматики распознающего типа.

Выполнение параллельной программы под управлением ОТЛАДЧИКА осуществляется поэтапно. На каждом этапе выполняется очередной фрагмент этой программы, заключающийся в реализации очередных фрагментов ее ветвей. Выполнение очередного фрагмента каждой ветви j ($j = 1,2,\ldots,L$) на любом из этапов завершается входом ветви в какое-либо из состояний i ($i = 1,2,\ldots,n$), что характеризуется установлением в "1" значения соответствующей составляющей y_{ji} вектора Y_j . Детерминированность данного выполнения обуславливается функциями ОТЛАДЧИКА. Далее формируется вектор X и управление получает блок МОСТ, на вход которого поступает символ, совпадающий с результатом логического произведения фильтра d на вектор X и равный некоторому символу $s_i \in S$ ($i = 1,2,\ldots, q$; $q \leq n$). При несовпадении полученного символа ни с одним s_i выдается соответствующая информация. Использование фильтра обеспечивает пошаговое исследование процесса функционирования операционной системы многомашинного комплекса. Причем, подобно [5], в значительной степени обеспечивается проверяемость правильности выполнения шага, безотносительно шагов, которые еще не выполнялись.

Завершение работы блока МОСТ характеризуется либо выдачей информации о некоторой конфликтной ситуации, либо передачей управления ОТЛАДЧИКУ для организации очередного этапа выполнения ветвей.

4. Пример использования блока МОСТ

Пусть на определенном шаге исследования ОС требуется проверить ее функционирование при входах ветвей параллельной программы в одно из следующих состояний:

- завершение интервала (состояние ЗИ), т.е. выполнение ветвью заданного числа команд;
- достижение оператора группового взаимодействия (состояние ДГВ);
- достижение оператора индивидуального взаимодействия (состояние ДИВ).

Задается фильтр, выделяющий только эти состояния в векторе X, остальные состояния ветвей игнорируются.

Состояние ЗИ вводится в целях отладки. ИНТЕРВАЛ - это число, указывающее количество команд ветви, выполняемых за один этап. Интервал может принимать постоянное (стандартное) значение, установленное ОТЛАДЧИКОМ (например, 100 команд), либо задаваться поль-

зователем и принимать любые целые положительные значения. При единичном значении интервала ветвь входит в состояние ЗИ после выполнения одной команды. При значении интервала, превышающем число выполняемых команд ветви, последняя никогда не достигает состояния ЗИ. Предполагается, что ветвь переходит в это состояние не только после выполнения заданного числа команд, но и сразу же после выполнения очередного оператора системных взаимодействий. При этом этап выполнения ветви считается законченным. Два других состояния обслуживаются ОТЛАДЧИКОМ в соответствии с работой операционной системы по организации их выполнения.

При входе в состояние ДГВ ветвь не выполняется дальше до тех пор, пока все ветви не выйдут на аналогичный оператор и не будет установлена корректность предполагаемого взаимодействия или же пока не обнаружится конфликтная ситуация.

При входе в состояние ДИВ управление получает модуль ОТЛАДЧИКА, организующий работу с семафорными переменными. В его функции входят проверка условия синхронизации и выполнение преобразований над семафорами. При невыполнении условия синхронизации данная ветвь, в соответствии с дисциплиной обслуживания в операционной системе, ставится в очередь. ОТЛАДЧИК оставляет ее в состоянии ДИВ и переходит к выполнению очередной ветви текущего этапа. При выполнении условия синхронизации ОТЛАДЧИК осуществляет имитацию выполнения данного оператора, преобразует семафоры в соответствии со значениями параметров этого оператора, устанавливает данную ветвь в состояние ЗИ. Далее ОТЛАДЧИК осуществляет просмотр очереди и организует имитацию выполнения оператора индивидуального взаимодействия тех ветвей, для которых выполнено условие синхронизации. Такие ветви также устанавливаются в состояние ЗИ. По завершению просмотра всей очереди ОТЛАДЧИК переходит к выполнению очередной ветви текущего этапа.

Таким образом, на входе в блок МОСТ возможны следующие состояния выполнения ветвей параллельной программы:

1) недопустимое - при выбранном фильтре;

2) все ветви вошли в состояние ЗИ;

3) все ветви вошли в состояние ДГВ;

4) часть ветвей вошла в состояние ЗИ, часть - в состояние ДГВ;

5) все ветви вошли в состояние ДИВ;

6) часть ветвей вошла в состояние ДИВ, часть - в состояние ЗИ;

7) часть ветвей вошла в состояние ДИВ, часть - в состояние ДГВ;

8) часть ветвей вошла в состояние ДИВ, часть - в состояние ДГВ, часть - в состояние ЗИ.

В первом случае выдается сообщение о недопустимости появившегося символа на входе в блок МОСТ. Во втором - ОТЛАДЧИК переходит к выполнению всех ветвей на очередном этапе. В третьем - ОТЛАДЧИК имитирует выполнение заданного оператора группового взаимодействия и переходит к выполнению всех ветвей на очередном этапе. В пятом и седьмом случаях выдается сообщение о зависании: все ветви в очереди либо часть ветвей в очереди, а часть ждет выхода остальных на оператор группового взаимодействия. В четвертом, шестом и восьмом случаях организуется список ветвей, вошедших в состояние ЗИ и ОТЛАДЧИК осуществляет очередной этап выполнения только для ветвей из списка.

На данном шаге исследования имеется возможность полностью проверить блоки операционной системы многомашинного комплекса, осуществляющие анализ и преобразование семафорных переменных, обработку символов, соответствующих состояниям выполнения ветвей параллельной программы, работу с очередью ждущих процессов. Кроме того, на данном шаге, благодаря имитации выполнения управляющих воздействий операционной системы, можно создать набор отлаженных тестов, которые в дальнейшем используются при отладке самой операционной системы.

На очередном шаге исследования к уже известным состояниям (ЗИ,ДГВ,ДИВ) можно добавить состояние входа ветви в режим выполнения фоновой программы, что обеспечивается использованием соответствующего фильтра. В этом случае на входе в блок МОСТ возможны 2^4 состояний выполнения ветвей.

5. Заключение

Использование предлагаемого метода при конструировании операционной системы позволяет:

- проводить одновременно и независимо от создания других частей операционной системы построение и анализ блоков работы с семафорными переменными, обработки символов, идентифицирующих состояния выполнения ветвей, работы с очередью;

- обеспечивать быструю и достаточно полную автономную отладку таких блоков и сохранить их целостность вплоть до включения в операционную систему;

- проводить, используя систему фильтров, многократное, целенаправленное исследование возможностей управляющих воздействий операционной системы при всевозможных состояниях выполнения ветвей параллельной программы;

- создавать систему отлаженных тестов с целью их дальнейшего использования непосредственно при отладке операционной системы.

Погружение программного блока МОСТ в среду ОТЛАДЧИКА позволяет:

- оперативно вносить изменения в логику работы отдельных модулей блока без их перетрансляции;

- исследовать, изменяя значения интервала, функции операционной системы по выполнению взаимодействия ветвей.

Литература

1. ГОЛОВКИН Б.А. Надежное программное обеспечение. Обзор.- Зарубежная радиоэлектроника, 1978, № 12, с. 3-61.

2. МАЙЕРС Г. Надежность программного обеспечения. - М.: Мир, 1980, - 353 с.

3. ЛИПАЕВ В.В. Надежность программного обеспечения АСУ. - М.: Энергоиздат, 1981, - 241 с.

4. КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И., КРЫЛОВ Э.Г., КОРНЕЕВ В.М., МИРЕНКОВ Н.Н. Программное обеспечение системы МИНИМАКС. - Новосибирск, 1979. - 43 с. (Препринт/Институт математики СО АН СССР, ОВС-09).

5. JACKON M.A. Constructive methods of program design. - Lecture Notes in Computer Science, 1976, v.44, p.236-262.

Поступила в ред.-изд.отд.
9 октября 1981 года