УДК 004.42 : 519.85

ТЕХНОЛОГИЯ ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ. ЯЗЫК PGRAPH

© 2018 А.Н. Коварцев, В.В. Жидченко, Д.А. Попова-Коварцева, А.Н. Даниленко

Самарский национальный исследовательский университет имени академика С.П. Королёва

Статья поступила в редакцию 12.12.2018

Рассматриваются основные принципы разработки моделей параллельных алгоритмов на основе визуального стиля программирования. Визуальное моделирование параллельных вычислительных процессов обладает достоинством наглядного представления информации и гораздо лучше соответствует природе человеческого восприятия, чем методы традиционного, текстового программирования. Предложенная технология и созданный на её основе язык визуального представления параллельных алгоритмов имеют большое практическое значение, поскольку позволяют упростить и ускорить разработку алгоритмов параллельных вычислений.
Ключевые слова: визуальное программирование, технология программирования, параллельные вычисления, синхронизация вычислительных процессов.

ВВЕДЕНИЕ

Современное программное обеспечение (ПО) характеризуется сложностью выполняемых функций и значительностью объемов исполняемых кодов. При этом к программам предъявляются высокие требования к качеству и надежности ее кодов. Использование графических методов программирования позволяет повысить производительность труда программиста, сократить время разработки параллельных вычислительных процессов, а также повысить их надежность.

Визуальные средства разработки алгоритмов используют для создания программ графические элементы (диаграммы и схемы). Технология графосимволического программирования (ГСП), созданная на кафедре программных систем Самарского национально-исследовательского университета, является одним из способов наглядного представления алгоритмов программы в виде графа управления и в большей степени соответствует образному способу мышления человека.

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ ТЕХНОЛОГИИ ГСП

Технология ГСП предоставляет возможности проектирования и создания алгоритмов программ с использованием графической нотации. Данная технология разрабатывалась для автоматизации процессов проектирования, кодирования и тестирования программного обеспечения.

В основу рассматриваемой технологии легла традиционная модель объекта с дискретными состояниями, когда можно выделить конечное число состояний для любого объекта программирования. Тогда вычислительный процесс можно описать переходами объекта из одного состояния в другое.

1.1. Моделирование последовательных алгоритмов

В технологии ГСП понятие алгоритма связывается с совокупностью вычислимых функций (модулей, операций). Тогда программу алгоритма A можно интерпретировать как некоторую результирующую вычислимую функцию:

$$f : in(D) \to out(D),$$

где $in(D)$ – множество входных данных функции f, $out(D)$ – множество выходных данных функции f.

В [2] изложена концепция описания алгоритма для вычислительной модели, названной машиной Колмогорова. В машине Колмогорова на каждом шаге работы алгоритма реализуется переработка текущего состояния структур данных D в новое состояние D^* с помощью некоторой локальной функции $D^*=f_k(D)$. Процесс переработки $D^0=D$ в $D^1=f_1(D^0)$, D^1 в $D^2=f_2(D^1)=f_1(f_2(D))$ и т.д. продолжается до тех пор, пока не появится сигнал о получении решения.

В ГСП тоже реализуется преобразование структур данных с помощью вычислимых функций, но по другим правилам. Граф состояний G - ориентированный помеченный граф, вершины этого графа – состояния, а дуги определяют переходы системы из состояния в состояние [1].

Формально состояние – это достижение объ-

Коварцев Александр Николаевич, доктор технических наук, профессор, заведующий кафедрой программных систем. E-mail: kovr_ssau@mail.ru
Жидченко Виктор Викторович, кандидат технических наук, доцент кафедры программных систем. E-mail: vzhidchenko@yandex.ru
Попова-Коварцева Дарья Александровна, кандидат технических наук, доцент кафедры программных систем. E-mail: dakkovr@mail.ru
Даниленко Александра Николаевна, кандидат технических наук, доцент кафедры программных систем. E-mail: danilenko.al@gmail.com

ектом *O* (моделью алгоритма) определенной конкретизации структур данных, требующее выполнения соответствующих действий над данными предметной области с помощью некоторой вычислимой функции A_k. Поэтому каждая вершина графа помечается локальной вычислимой функцией A_k. При этом следует помнить, что состояние графа S_j – это некое понятие («концепт»), связанное с абстрагированием объекта *O*, а вычислимая функция A_k, «приписанная» состоянию, – функциональное действие, которое необходимо выполнить при переходе объекта в это состояние.

Для реализации событийного управления на графе состояний *G* вводится множество логических функций (предикатов) $P=\{P_1, P_2,..., P_l\}$. Здесь $P_i(D) \in \{0,1\}$ в зависимости от значений данных *D*.

Дугам графа *G* поставим в соответствие функции-предикаты. Событие, реализующее переход $S_i \rightarrow S_j$ на графе состояний *G*, инициируется, если модель объекта *O* на текущем шаге работы алгоритма находится в состоянии S_i и соответствующий предикат $P_{ij}(D)$ (помечающий данный переход) истинен.

Ориентированный помеченный граф наглядно представляет пути развития вычислительного процесса. Фактически граф *G* является строгим описанием графа управления программы.

Модель алгоритма в технологии графосимволического алгоритма можно описать следующим образом:

$$M=<D,F,P,G>,$$

где *D* – данные некоторой предметной области, *F* – множество вычислимых функций, *P* – множество предикатов, действующих над структурами данных, *G* – граф состояний объекта *O* [1].

С помощью данной четверки *M* можно описать любой алгоритм программы. При этом алгоритм программы представляется в визуальной форме, в виде графа управления. Кроме того, происходит декомпозиционное расслоение основных компонент описания алгоритма. Так структура алгоритма представляется графом управления *G*, логические выражения, определяющие управления, собраны в едином хранилище - множестве предикатов *P*.

Технология ГСП допускает построение иерархически вложенных алгоритмических моделей, уровень вложенности граф-моделей не ограничен.

Рисунок 1 содержит описание модели алгоритма решения задачи о ханойских башнях. Текст программы на языке C++ в соответствии с графическими нотациями, представленными на рисунке, генерируется автоматически [3].

Предложенный стиль визуального программирования языка *PGRAPH* имеет много общего с другими языками визуального программирования. С одной стороны, графические нотации языка *PGRAPH* часто воспринимаются как разновидность автоматного программирования, например, *Switch*-технологии Шалыто А.А. [4]. С другой стороны, язык *PGRAPH* имеет схожие черты с языками, использующими блок-схемы: ДРАКОН [5]. В определенном смысле *PGRAPH* можно считать алгоритмическим языком моделирования, реализующим процесс проектирования ПО, и тогда он похож на язык *UML* [6] за тем исключением, что на языке *PGRAPH* после завершения описания модели алгоритма всегда автоматически порождается код программы.

Используемый в ГСП математический формализм - граф управления (ГУ) - служит хорошей основой для решения ряда задач повышения качества разрабатываемых моделей алгоритмов: структурного тестирования графа управления, структурной оптимизации ГУ, построения схем маршрутов вычислительных процессов, классификации используемых данных по признаку вход/выход и т.д. Но удивительной особенностью данной технологии является возможность разработки модели алгоритма без использования его

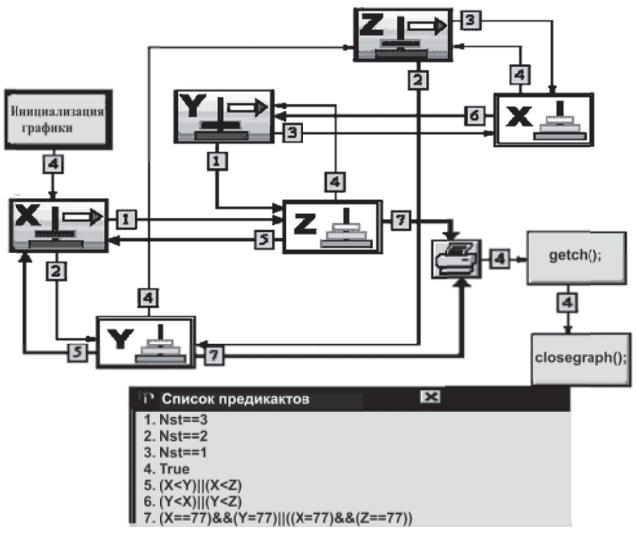

Рис. 1. Пример последовательной программы

готового облика. В технологии ГСП для разработки алгоритма достаточно иметь лишь идею решения задачи, а алгоритм строится шаг за шагом в процессе прорисовки графа управления.

1.2. Моделирование параллельных алгоритмов

Параллельная программа содержит несколько вычислительных процессов, поэтому для описания модели параллельного алгоритма в технологии ГСП необходимо ввести новые элементы. Следует обозначить начало и конец параллельных процессов, а также определить условия их запуска и завершения [7]. Для этих целей вводится понятие дуг различного типа.

Рассматриваемая технология допускает для описания параллельного алгоритма использование следующих типов дуг: последовательная, параллельная (обозначение начала параллельного процесса) и терминирующая (обозначение завершения параллельного процесса).

Для формального описания типов дуг Ψ_{ij} используется функция, принимающая три значения $T(\Psi_{ij}) \in \{1,2,3\}$ (1 – последовательная дуга, 2– параллельная дуга, 3 – терминирующая).

Каждый отдельный вычислительный процесс начинается параллельной дугой и заканчивается терминирующей. В технологии ГСП для описания такого процесса вводится понятие параллельной ветви b, являющейся подграфом графа G. Параллельную ветвь алгоритма можно формально определить следующим образом:

$$b = <A_b, \Psi_b, R_b>,$$

где A_b – множество вершин ветви, Ψ_b – множество дуг управления ветви, R_b – отношение над множествами вершин и дуг ветви, определяющее способ их связи.

Дуга, которая обозначает вход в параллельный процесс называется параллельной дугой и помечается «кружком». Дуга, обозначающая выход из параллельной ветви, называется терминирующей и помечается перечеркиванием. Допускается вложенность параллельных ветвей. Ветви, породившиеся в одной вершине некоторой ветви, должны терминироваться также в одной вершине этой же ветви. Можно создать неограниченное количество параллельных ветвей. Число параллельных ветвей программы фиксируется при создании алгоритма. Модель параллельного алгоритма можно представить как совокупность всех параллельных ветвей:

$$G=\cup\beta_k,$$

где β_k, – параллельные ветви графа *G*.

Функционирование модели начинается с запуска единственной ветви, называемой мастер-ветвью. На рисунке 2 представлен пример параллельной программы, предназначенной для решения системы обыкновенных дифференциальных уравнений (ОДУ) больших размерностей методом Рунге-Кутты. Данный модуль входит в более общую программу расчета движения космической тросовой системы (КТС) [8].

Применение КТС открывает новые возможности в использовании космического пространства: создание искусственной тяжести, транспортные операции в космосе, возвращение полезных грузов с орбиты, запуск малых спутников с базового космического аппарата (КА), использование геомагнитного поля Земли для орбитальных маневров – вот далеко не полный перечень возможностей КТС [9].

Несмотря на свою внешнюю простоту, задача высокоточного расчета движения КТС имеет свои сложности. Трос может иметь длину 30 километров и более. Для достижения приемлемой точности расчетов необходимо разбивать трос на значительное количество участков (100000 и более). На одном процессоре (для последовательного алгоритма) потребуется слишком много машинного времени. Однако исходную задачу можно разбить на p частных задач, меньших размерностей, и решать их на высокопроизводительной кластерной вычислительной системе, что достигается за счет разбиения всего троса на p участков.

В то же время, особенности решаемой задачи не позволяют использовать классические схемы организации параллельных вычислений, часто используемые численные решения дифференциальных уравнений [10]. Дело в том, что для определения сил натяжения для каждой точки троса требуется знать положение соседних с ней точек, что не позволяет организовать независимое решение частных систем ОДУ на каждом процессоре. Каждая частная задача должна получать и передавать информацию о положении точек, смежных с соседним участком троса (каждый участок обсчитывается на собственном процессоре).

В методе Рунге-Кутты изменение фазовых координат для уравнения $\frac{dX}{dt} = F(X,t)$ рассчитывается по формуле

$$X^{(r+1)} = X^{(r)} + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4). \quad (1)$$

Здесь X – матрица фазовых координат для каждой точки КТС; $F(X,t)$ – правые части ОДУ движения тросовой системы; K_1, K_2, K_3, K_4 – матрицы коэффициентов формулы (1). Использование метода Рунге-Кутты применительно для КТС не позволяет производить вычисления коэффициентов K_i независимо друг от друга. И, как следствие, возникает такая странная 4-х фазная (пульсирующая) модель организации параллельных вычислений, представленная на рис. 2. В каждой фазе независимо друг от друга вычисляются коэффициенты K_1, K_2, K_3, K_4. При этом при переходе от фазы к фазе передаются каждому

процессору положения точек, смежных с каждым участком троса. Последняя фаза вычисляет новое положение всех точек троса.

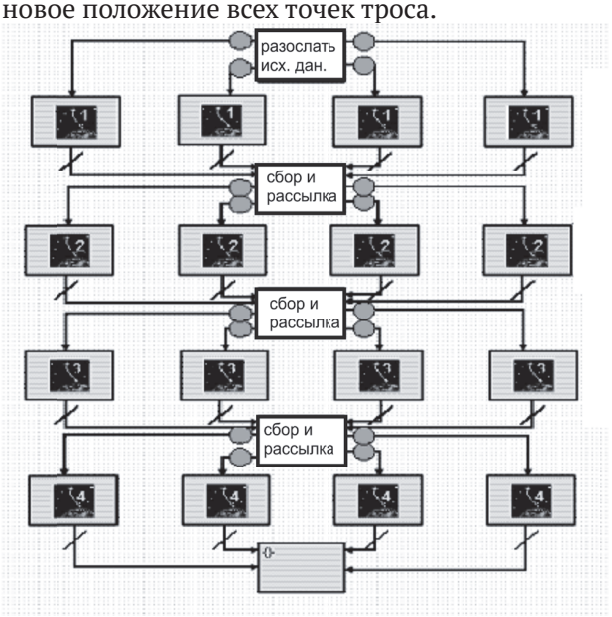

Рис. 2. Пример параллельной программы

2. МОДЕЛЬ СИНХРОНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ

При выполнении параллельных вычислений требуется синхронизация работы модулей программы, выполняемых разными процессорами. Задача синхронизации – это задача, в рамках которой требуется согласовать выполнение нескольких процессов. Данная задача решается с использованием различных механизмов [11]. В технологии ГСП применяется комбинированный способ, использующий одновременно механизмы передачи сообщений и принципы мониторной синхронизации.

Определим почтовый ящик L_{post} как список сообщений, с помощью которых вершины модели информируют друг друга о своем состоянии: $L_{post} = [\mu_{i_0,j_0},...,\mu_{i_m,j_n}]$, где $\mu_{i,j}$ – сообщение, посылаемое от вершины A_i вершине A_j. Возможность передачи сообщения от одной вершины граф-модели к другой вершине изображается графически дугой синхронизации Ψ_{ij}, проведенной из вершины-источника сообщения в вершину-получателя сообщения.

Вершина A_k посылает сообщения другим вершинам, записывая сообщения $L_{A_kCom} = [\mu_{k,j_0},...,\mu_{k,j_n}]$ в список L_{post}. Наличие сообщения $\mu_{i,j}$ в списке L_{post} говорит о том, что оператор вершины A_i завершил работу и хочет сообщить об этом вершине A_j.

Каждой вершине ставится в соответствие семафорный предикат $R_{A_j} = r(b_{i_0,j},...,b_{i_m,j})$, представляющий собой правильно построенную формулу относительно булевых переменных $b_{i_k,j}$, связанных логическими связками типа &,∨. Булевы переменные определяются следующим образом:

$$b_{k,j} = \begin{cases} 1, & если\ \mu_{k,j} \in L_{post}, \\ 0, & если\ \mu_{k,j} \notin L_{post}. \end{cases}$$

Семафорный предикат разрешает или запрещает запуск соответствующей вершины, в которую входит дуга синхронизации. Если $R_{A_j} = 1$, то запуск оператора вершины A_j разрешается, иначе вычисления приостанавливаются до того момента, когда R_{A_j} станет равным 1.

Проиллюстрируем применение рассмотренных методов на реальном примере. Рассмотрим модель асинхронного потенциального *RS*-триггера, построенного на элементах И-НЕ (рис. 3) [7].

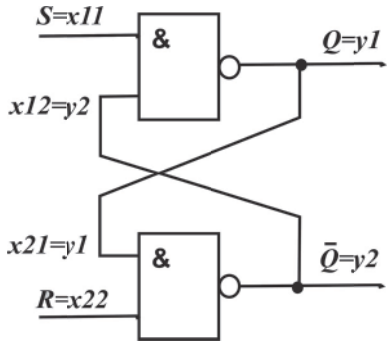

Рис. 3. Принципиальная схема RS-триггера на элементах И-НЕ

Логика работы *RS*-триггера во времени может быть описана следующей системой функций:

$$\begin{cases} y1(t+1) = \overline{x11(t)\ \&\ y2(t)}, \\ y2(t+1) = \overline{x22(t)\ \&\ y1(t)}. \end{cases} \quad (2)$$

Поскольку триггеры в основном применяются в цифровых устройствах, они, как правило, работают в дискретном времени, и, как следствие, значения их выходных сигналов анализируются только в определенные моменты времени.

Пусть *S=x11(t), R=x22(t), x12(t)=y2(t), x21(t)=y1(t), Q=y1(t+1), Q̄=y2(t+1)*. На рисунке 4 представлена параллельная модель алгоритма *RS*-триггера. Параллельные ветви данной модели совместно используют переменные *y1* и *y2* (вершина 3.0 использует переменную y2 для чтения, а вершина 4.1 - для записи). Дуги синхронизации, на рисунке 4 - пунктирные «стрелки», позволяют разрешить конфликт совместного использования данных.

ЗАКЛЮЧЕНИЕ

Визуальное программирование является удобным способом написания параллельных программ. В статье рассмотрены основные аспекты создания параллельных программ в рамках технологии ГСП. В теоретическом плане для организации параллельных вычислений

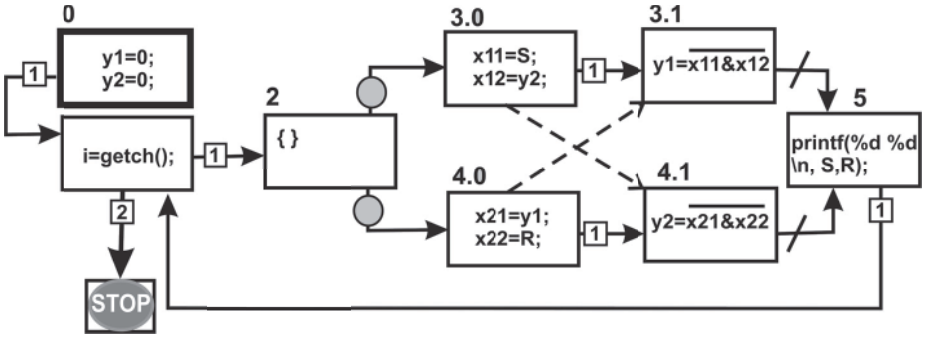

Рис. 4. Параллельная модель RS-триггера

последовательная алгоритмическая модель расширена дополнительными компонентами. Для описания параллельной программы введено понятия типа дуг и изменен алгоритм управления вычислительными процессами, для этого используются средства организации синхронизации параллельных вычислений.

Разработка и запись параллельного алгоритма в графическом виде значительно облегчает понимание такого алгоритма. Графический способ записи программы в технологии ГСП предоставляет пользователям возможности анализа структуры и производительности параллельной программы, а так же корректности применяемых механизмов синхронизации вычислений.

БЛАГОДАРНОСТИ

Работа выполнена при государственной поддержке Министерства образования и науки РФ, а также гранта РФФИ №16-41-630637.

СПИСОК ЛИТЕРАТУРЫ

1. *Коварцев А.Н.* Автоматизация разработки и тестирования программных средств. Самара: Самар. гос. аэрокосм. ун-т, 1999. 150 с.
2. *Успенский В.А., Семенов А.Л.* Теория алгоритмов: основные открытия и приложения. М.: Наука, 1987. 288 с.
3. *Коварцев А.Н.* Онтологический аспект модели вычислений графосимволического программирования// Онтология проектирования. 2012. № 3. С. 38-48.
4. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. 628 с.
5. *Пароджанов В.Д.* Язык ДРАКОН. Краткое описание. М.: 2009. 124 с.
6. *Буч Г., Рамбо Дж., Джекобсон А.* Язык UML. Руководство пользователя. СПб.: Питер, 2004. 432 с.
7. *Коварцев А.Н., Жидченко В.В.* Методы и средства визуального параллельного программирования. Автоматизация программирования. Самара: Самар. гос. аэрокосм. ун-т, 2011. 168 с.
8. *Заболотнов Ю.М., Фефелов Д.И.* Динамика движения капсулы с тросом на внеатмосферном участке спуска с орбиты // Известия Самарского научного РАН. 2006. Т. 8. № 3. С. 841–848.
9. *Белецкий В.В., Левин Е.М.* Динамика космических тросовых систем. М.: Наука, 1990. 336 с.
10. *Гергель В.П.* Теория и практика параллельных вычислений. 2-е изд. М.: Интуит, 2016. 500 с.
11. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

A TECHNIQUE FOR THE VISUAL MODELING OF PARALLEL ALGORITHMS: THE PGRAPH LANGUAGE

© 2019 A.N. Kovartsev, D.A. Popova-Kovartseva, V.V. Zhidchenko, A.N. Danilenko

Samara National Research University named after Academician S.P. Korolyov

This paper concerns basic principles for developing parallel-algorithm models based on a visual programming style. The visual modeling of parallel computational processes has the advantage of presenting information visually and so agrees with the nature of human reception much better than conventional textual-programming methods. The proposed technique and the resulting language for the visual presentation of parallel algorithms are of significant practical importance because they simplify and accelerate the development of parallel-computation algorithms.
Keywords: visual programming, programming technique, parallel computation, synchronization of computational processes.

Alexsandr Kovartsev, Doctor of Technics, Professor, Head of Software Systems Department. E-mail: kovr_ssau@mail.ru
Victor Zhidchenko, Candidate of Technics, Associate Professor of Software Systems Department. E-mail: vzhidchenko@yandex.ru
Darya Popova-Kolarceva, Candidate of Technics, Associate Professor of Software Systems Department.
E-mail: dakkovr@mail.ru
Alexandra Danilenko, Candidate of Technics, Associate Professor of Software Systems Department.
E-mail: danilenko.al@gmail.com