УДК 004.2

Паулин О.Н., Поляков Ю.С., Шульгин В.А.

ПРЕДСТАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ СЕТЯМИ ПЕТРИ

Одесский национальный политехнический университет

Одесса, просп. Шевченко, 1, 65044

Paulin O.N., Poljakov J.S., Shulgin V.A.

PRESENTATION COMPUTING PROCESSES BY PETRI NETS

Odessa National Polytechnic University

Odessa, ave. Shevchenko, 1, 65044

Аннотация. В статье рассматривается новый подход к анализу программ. Он обеспечивает представление вычислительных процессов (ВП) сетями Петри (СП), который является более мощным инструментом чем конечный автомат. При этом в ВП выделяются базовые операции, называемые макрооперациями (МО), для которых строятся фрагменты сети Петри. Из фрагментов собирается вся сеть Петри для данного ВП. Предложенный подход иллюстрируется примером ВП сортировки методом простых вставок. Полученная СП моделируется на эмуляторе СП, который также разработан авторами.

Ключевые слова: вычислительный процесс, сеть Петри, макрооперация, фрагмент, сортировка простыми включениями, эмулятор сети Петри, моделирование.

Abstract. In this paper a new approach to the analysis of programs is considered. It provides a representation of computational processes (CP) Petri nets (PN), which is a more powerful tool than a finite-state automaton. In the CP are extracted basic operations, called macrooperation (MO), for which are built fragments of Petri nets. The Petri net is construced with the fragments for current

CP. The proposed approach by an example CP by straight insertions sorting is illustrated. The obtained SP is modeled on the emulator SP, which is also developed by the authors.

Keywords: computational process, Petri net, macro operation, fragment, straight insertions sorting, emulator, simulation.

Введение. Процесс получения качественного программного продукта включает в себя следующую цепочку этапов его разработки: формулирование идеи решения задачи, развёртывание её в словесное описание вычислительного процесса (процедуры) решения, доказательство того, что процедура является алгоритмом, построение схемы алгоритма, реализация вычислительных процессов на компьютере, т.е. написание программ на языке высокого уровня, её отладку на контрольном примере и тестирование [1, 2].

Ошибки в процедуре уже не могут быть исправлены программно, поэтому важно их обнаружить на более раннем этапе. Оптимизация также должна быть проведена как можно раньше – уже на этапе анализа схемы алгоритма. И только после этого можно переходить к написанию программы и её анализу.

В настоящее время термин «вычисление» приобрёл весьма широкий смысл, который более точно отражается термином «обработка данных». В этот термин входят: транспонирование матриц, сортировка и поиск определённой информации в базах данных, наилучшее расположение в пространстве некоторых объектов, получение оптимальной схемы соединения выводов микросхем на плате, представление на экране дисплея заданного разреза некоторой конструкции и др., а также разнообразные формирования и преобразования образов (световых, цветовых, аудио) в мультимедиа.

Сложность программ непрерывно растёт и качественный анализ программ порой занимает в разы большее время, чем её написание. Поэтому весьма актуально снижение (относительного) времени анализа программы.

Многообразие вычислительных процессов (ВП) при традиционном подходе приводит к излишним затратам времени на программирование каждого

отдельного ВП и на анализ программ, в то время как в вычислительных процессах имеется много общего в виде одинаковых подпроцессов.

В последнее время популярным стал автоматный подход к программированию, который позволяет более тщательно проводить анализ программ. По Википедии: «Автоматное программирование – это парадигма программирования, при использовании которой программа или её фрагмент осмысливается как модель какого-либо формального автомата».

Отметим, что термин «автоматный» трактуется слишком широко: сюда включают не только конечные автоматы, но и машину Тьюринга, сеть Петри, нейронные сети и клеточные автоматы. На наш взгляд, такое широкое толкование термина не является оправданным, ибо, например, конечный автомат (КА) [3] и машина Тьюринга (МТ) [1] качественно отличаются по вычислительной мощности.

В области автоматного программирования можно выделить 2 наиболее развитых подхода: SWITCH-технология [4] и КА-технология [5] разработки программного обеспечения. Основное отличие данных технологий состоит в реализации логики автоматных программ: в 1-й она реализуется переводом автоматного описания в программные инструкции языка программирования, во 2-й реализуется прямое исполнение автоматов путём интерпретации его исходного табличного представления.

Недостатком автоматного подхода является малая вычислительная мощность, выражаемая автоматным языком. Представляется более адекватным использование сетей Петри [6], которые обладают существенно большей выразительностью своего языка, занимая промежуточное положение между КА и МТ. К тому же автомат не отражает более сложного понятия, чем «операция», а именно понятия «событие», а также взаимную зависимость «событие – условие», что характерно для СП; это сужает сферу применения КА.

С другой стороны, при автоматном подходе в КА преобразуются программные конструкции. Это также является недостатком, ибо

целесообразно связать с автоматом не программные конструкции, а вычислительные конструкции. Это тем более относится к сетям Петри.

Целью работы является сокращение времени на анализ программы, реализующей конкретный ВП, за счёт более ранней увязки вычислительных конструкций с более мощным средством их описания – с сетью Петри.

Представление вычислительного процесса. Для сокращения времени анализа программ нами предлагается новый подход, который заключается в представлении обобщённых вычислительных конструкций фрагментами сетей Петри; эти конструкции мы назвали *макрооперациями* (МО).

Определение. Макрооперацией называется логически законченный фрагмент вычислительного процесса.

МО включает в себя простейшие стандартные операции присваивания, перестановки элементов массива, сдвига элементов последовательности, сравнение строк (столбцов) таблицы и т.п., а также структуры управления (СУ) элементарными операциями: композиция, альтернатива, итерация. Отметим, что СУ присутствуют на любом уровне иерархии ВП.

Предлагаемый нами подход основан на **утверждении**: ВП может быть сконструирован из МО и структур управления (СУ), причём структур управления всего 3 (композиция, альтернатива, итерация), а количество МО относительно невелико.

Для получения МО проводится декомпозиция ВП и выделяются базовые вычислительные конструкции. Идея выделения базовых фрагментов в арифметических выражениях при решении нелинейных уравнений была рассмотрена в [7].

Пример вычислительного процесса. В качестве примера ВП рассмотрим процессы сортировки [8]. В простейшем варианте эти сортировки осуществляются следующими методами: простыми вставками, простым выбором, простым обменом.

Рассмотрим схему реализации сортировки методом вставки «изнутри» процесса (рис. 2), а затем уточним его параметры в начале и конце процесса.

Сортировку будем проводить по возрастанию. Этот метод заключается в следующем. Элементы массива условно разделяются на *готовую* подпоследовательность $a_1,\ldots,a_{i-1}$ и *входную* подпоследовательность $a_i,\ldots,a_n$ (рис. 1). Затем используется идея: очередной элемент из входной подпоследовательности (сразу за границей) вставляется на подходящее место в готовой подпоследовательности.

Рассмотрим подробно текущее состояние процесса сортировки и далее уточним его состояния в начале и конце этого процесса. Для удобства описания процесса введём понятие «граница», отделяющую отсортированную подпоследовательность от входной (т.е. оставшейся неотсортированной части исходной последовательности).

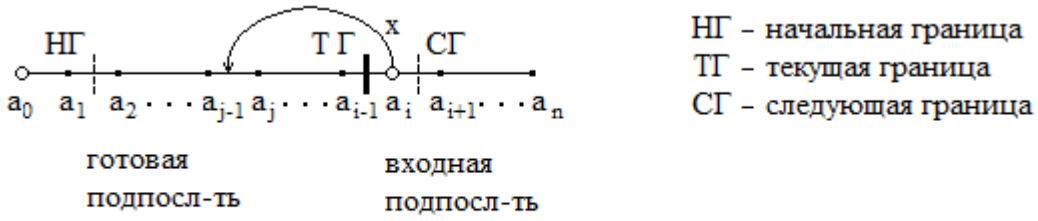

Рисунок 1 – Модель задачи сортировки простыми включениями

Процесс сортировки протекает следующим образом.

1. Обозначим x текущий вставляемый элемент, в качестве которого берём первый сразу после границы элемент a_i ($x:=a_i$), и вставляем его на подходящее место в готовой подпоследовательности; это место определяется соотношением: $a_{j-1}\le x<a_j$.

2. Сдвигаем элементы $a_j..a_{i-1}$ на одну позицию вправо, начиная с правого конца: $a_i:=a_{i-1}$; $a_{i-1}:=a_{i-2}$; … $a_{j+1}:=a_j$.

3. Вставляем текущий элемент x в готовую подпоследовательность: $a_j:=x$.

4. Перемещаем границу вправо: $i:=i+1$ (это номер элемента сразу после границы). Переходим на п. 1.

Отметим, что циклы поиска подходящего места и сдвига элементов можно совместить. Тогда каждая итерация будет состоять из двух последовательных операций: сравнение текущего элемента с очередным элементом готовой

подпоследовательности и сдвиг (при необходимости) очередного элемента вправо. Такой процесс в [1] назван *просеиванием*.

Особенность процесса сортировки: может оказаться, что текущий элемент надо размещать на первой позиции. Однако при этом не образуется пара элементов, между которыми должен разместиться текущий элемент. Такую пару создают искусственно – вводят так называемый «барьер» в виде элемента a_0. Можно ввести его одноразово на весь процесс с большой отрицательной величиной; можно, исходя из соотношения для сравнения (см. п. 1), сделать его значение переменным, равным текущему элементу: $a_0=x$. Последний вариант предпочтительней; в этом случае нужно расширить диапазон индексов в описании массива A до $0, ..., n$.

Анализ *крайних случаев*. В начале процесса считаем элемент a_1 отсортированным, а элемент a_2 – текущим, т.е. принимаем $i_{нач}=2$. Процесс заканчивается вставкой элемента a_n на подходящее место, т.е. $i_{кон}=n$. Таким образом, граница перемещается в пределах от 2 до n.

Это предварительное рассмотрение позволяет сформулировать *словесное описание* алгоритма. Отталкиваясь от него, уже достаточно просто составить программу на выбранном языке программирования.

Словесное описание алгоритма простых включений

0. В готовую подпоследовательность записывается a_1, во входную – $a_2,\ldots, a_n$.

1. i :=2.

2. Переносим элемент $x=a_i$ из входной подпоследовательности в готовую так, чтобы готовая подпоследовательность осталась отсортированной. Для этого:

а) расширяем готовую подпоследовательность слева с помощью барьера $a_0 := x$.

б) параметр цикла поиска подходящего места принимает значение $j:=i-1;$

в) пока $x<a_j$, выполняется сдвиг элемента a_j вправо ($a_{j+1}:=a_j$) и уменьшение j на единицу ($j:=j-1$);

г) $a_{j+1}:=x$.

3. $i:=i+1$.

4. Если $i \leq n$, то переходим на п. 2, иначе сортировка закончена.

Процесс сортировки простыми включениями показан в табл. 1 на примере восьми случайно взятых чисел (вертикальной чертой – границей – отделяются элементы готовой подпоследовательности от входной).

Таблица 1.

Процесс сортировки простыми вставками

Начальные ключи	44	55	12	42	94	18	06	67
i=2	44	55	12	42	94	18	06	67
i=3	12	44	55	42	94	18	06	67
i=4	12	42	44	55	94	18	06	67
i=5	12	42	44	55	94	18	06	67
i=6	12	18	42	44	55	94	06	67
i=7	06	12	18	42	44	55	94	67
i=8	06	12	18	42	44	55	67	94

Анализ ВП сортировок (в том числе более сложных: Шелла, пирамидальной, быстрой [8], а также параллельной – см. [9]) показывает, что ядром ВП является операции сравнения элементов последовательности чисел и перестановки их при необходимости; поскольку они идут в паре, то имеет смысл их объединить в одну макрооперацию «сравнить – переставить».

Выбор элементов для сравнения и способ передвижения границы организуют конкретный ВП сортировки.

В сортировке методом вставки можно выделить 3 МО: 1) «сравнить-переставить»; 2) «счётный цикл»; 3) «цикл с условием».

Для рассматриваемой сортировки построена сеть Петри (рис. 2) с учётом указанных МО. Здесь сплошными линиями показаны обычные связи сети, а пунктирными линиями – ингибиторные (запрещающие) связи.

Рассмотрим функционирование сети. Инициализация ВП осуществляется введением фишки в позицию P0. Дальнейшее перемещение фишки происходит в соответствии со сценарием, заданным алгоритмом. При установлении

начальной границы (i:=2) срабатывает переход t0. В позиции P1 происходит проверка условия $i \le n$; если оно выполняется, возможно срабатывание перехода t1 при возникновении событий: переобозначение переменной (x:=a_i), введение барьера (a_0:=x) и запуск цикла по j (j:=i-1) поиска подходящего места для вставки очередного элемента. Фишка перемещается в P2.

Позиция P2 реализует условие (пока $x<a_j$), что означает разрешение поиска, а событие t2 – это шаг поиска с помощью сравнения текущего элемента с парой элементов ($a_{j-1} \le x < a_j$) отсортированной подпоследовательности, причём правый элемент пары сдвигается вправо. При срабатывании t2 фишка возвращается в P2; так будет происходить, пока не будет найдено подходящее место. В этом случае связи P2-t2 и P2-t3 изменят свою суть на противоположные: ингибиторная станет обычной, а обычная – ингибиторной. Событие t3 означает помещение элемента x на подходящее место (a_{j+1}:=x) и увеличение на 1 значения счётчика номера i элемента, расположенного сразу за границей. Фишка перемещается из P2 в P1.

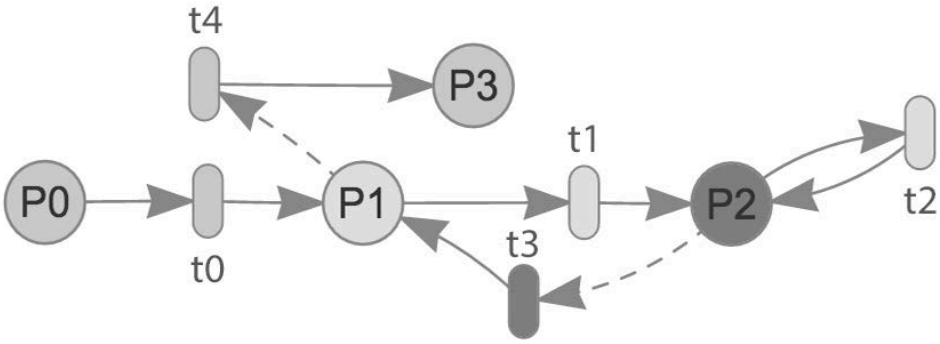

Рисунок 2 – Сеть Петри для ВП сортировки простыми вставками

Далее повторяется процесс поиска подходящего места для нового элемента неотсортированной части массива. Процесс закончится, когда для элемента a_n найдётся подходящее место. В этом случае связи P1- t1 и P1- t3 поменяют свою суть. Событие t4 означает окончание процесса сортировки, а перемещение фишки при срабатывании t4 в позицию P3 об этом сигнализирует.

Для моделирования построенной сети Петри нами проведен обзор эмуляторов сети Петри, имеющихся в доступных источниках, в том числе на

сайте **StudioPetri.** Однако в них отсутствуют возможности записи отчета работы сети и её анализа. Поэтому возникла необходимость повышения функциональности данного эмулятора и строить собственный эмулятор.

Заключение. В последнее время получило широкое распространение автоматное программирование, при котором программные конструкции представляются в виде конечных автоматов (КА). Наибольшую известность приобрели 2 подхода: SWITCH-технология и КА-технология. Однако автомат обладает относительно слабой описательной мощью и использует узкое понятие «операция». В связи с этим в работе предложен более мощный аппарат – сеть Петри, которая представляет собой двудольный граф: здесь имеется два вида вершин (позиции, которые формализуют условия, и переходы, которые формализуют события) и, соответственно, два вида связей между условием и событием. Именно двудольность графа определяет б***о***льшую вычислительную мощь сети Петри по сравнению с конечным автоматом.

Кроме того, в работе предложено представлять в виде сети Петри не программу, а именно ВП на этапе алгоритмизации, для чего в ВП выделяются макрооперации (МО), являющиеся логически законченными компонентами процессов.

Отметим, что разные ВП могут иметь аналогичные МО, следовательно, небольшое количество МО можно использовать для описания разнообразных ВП. Так, во всех сортировках общей является МО «Сравнить и переставить»; особенность той или иной сортировки выражается в способе организации ВП.

Рассмотрен пример ВП сортировки методом простых вставок. Для данной сортировки построена сеть Петри, в которой используются базовая МО «Сравнить и переставить», МО «цикл с условием» и МО «счётный цикл». Проведено моделирование этой сети на разработанном эмуляторе сети Петри, которое показало, что построенная сеть жива.

Литература

1. Паулин О.Н. Основы теории алгоритмов: Учебное пособие. – Одесса: Автограф, 2003. – 188 с.

2. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов. – М.: Мир, 1981. – 368 с.

3. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1974. – 216 с.

4. Шалыто А.А. SWITCH-технология. Апгоритмизация и программирование задач логического управления. – СПб: Наука, 1998. – 628 с.

5. Любченко В.С. К проблеме создания модели параллельных вычислений / В.С. Любченко / Труды Третьей международной конференции «Параллельные вычисления и задачи управления – PACO'2006»

6. Котов В.Е. Сети Петри. – М.: Наука,1984. – 160 с.

7. Паулин О.Н., Усова Т.И. Алгоритм распараллеливания решения нелинейных уравнений на основе информационного графа /О.Н. Паулин, Т.И. Усова/ МНТК «Искусственный интеллект. Интеллектуальные системы. ИИ-2010». – Донецьк: ІПШІ, «Наука і освіта», 2010. – С. 163-165.

8. Кнут Д. Искусство программирования. Т.3 – Сортировка и поиск. – М.: Изд. дом «Вильямс», 2000. – 832 с.

9. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Учебное пособие. – Н. Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. – 184 с.