УДК 519 716.35

**ПАРАЛЛЕЛЬНЫЕ ПРОЦЕССЫ И ИХ СЛОЖНОСТЬ\***

**© 2019 г. В.П. Кутепов, В.Н. Фальк**

Федеральное государственное бюджетное образовательное учреждения высшего образования «Национальный исследовательский университет «МЭИ»,   
111250, Москва, Красноказарменная ул., д. 14. МЭИ, Кафедра Прикладной математики.

В статье предложен метод оценки сложности параллельных процессов по критериям среднего времени их абсолютно параллельного выполнения и необходимых для этого ресурсов (количеству узлов выполняющей процесс системы). Эти критерии рассматриваются для достаточно общего языка параллельных процессов и по их значениям можно судить насколько реальные их значения, полученные при выполнении процесса на конкретной системе, отличаются от предельно возможных.

**Введение.** Понятие процесса (далее мы рассматриваем дискретные процессы) уточняется как наблюдаемое частично-упорядоченное следование во времени его актов (неделимых действий), подчиненных определенным причинно-следственным отношениям. Применительно к системам и физическим явлениям протекающие в них процессы обычно определяются через понятие состояния и правила изменения состояния. Для описания и исследования различных классов процессов создаются специальные модели и языки. Однако, чтобы получить полную информацию об интересующем нас классе процессов, мы вынуждены рассматривать триаду: организацию и алгоритм выполнения процессами работ, язык, на котором эти процессы с теми или иными ограничениями могут быть описаны, и возможные варианты практической реализации процессов на предполагаемых для использования системах. При этом время выполнения процессов и используемые для этого ресурсы системы – основные критерии сложности процессов и их способности выполнять соответствующие работы и алгоритмы. Проблема достижения определенного времени выполнения процесса предполагает выполнение распараллеливания на трех указанных стадиях, как необходимого условия для достижения требуемого времени выполнения процесса или конкретного множества процессов при заданных ограничениях ресурсной среды.

Именно параллелизм с его различными формами [1,2] существенно усложняет работу по созданию соответствующих языков для их описания и эффективной реализации процессов на системах [3-5] по сравнению, например, с последовательным программированием.

Понятие акта и актора процесса, средства группирования актов процесса, формы динамического порождения процессов при их выполнении – главные конституенты, определяющие особенности моделей и языков процессов [6]. В то же время на самом нажнем уровне выполнение процесса можно представить как изменяемое во времени множество одновременно выполняемых и взаимодействующих между собой актов процесса.

Эта абстракция процесса реализована в средах параллельного программирования PVM [7], MPI [8]. В средах параллельного программирования ERLANG [9], MULTITHREADING [10], FPTL [5], HOPE [11], Haskell [12] имеется набор примитивов, предназначенных для группирования процессов, их описания и определений рекурсивно порождаемых процессов.

Процессы часто ассоциируются с функционированием систем и в серии статей [13] эта взаимосвязь обсуждается с разных сторон. В частности, дано формальное определение состояния процесса, введено естественное ограничение на правила изменения состояний, которые могут происходить только на основе информации о поведении процесса в прошлом [14], рассмотрена обобщенная модель динамического порождения процессов и реконфигурирования системы на основе теории самовоспроизведения [15]. Эти основополагающие результаты не утратили своей актуальности при исследовании процессов и систем.

Различные варианты формализации дискретных параллельных процессов получили свою в определенном смысле завершенную форму в работах [16-17]. В отличие от моделей последовательных процессов, при построении которых используются бинарные операции последовательной композиции, операции выбора по условию продолжения процесса и рекурсивные определения процессов или определения их с помощью циклов (частный случай правосторонней рекурсии) в моделях процессов [16-17] добавлена операция параллельной композиции. Последняя операция трактуется как произвольное чередование актов процессов при их выполнении, а операция выбора интерпретируется как операция случайного выбора одного из соединяемых ею процессов.

В [6] мы предложили версию языка параллельных процессов, который позволяет описывать более широкий класс параллельных процессов, которые, в частности, порождаются при выполнении сложных рекурсивных параллельных программ [5,18]. Этот язык позволяет более точно отображать в процессах временной фактор, возможность выполнения актов процесса с упреждением [19] и др.

Сети Петри – еще одна популярная модель описания дискретных процессов и систем [20], которая интересна двумя факторами: наглядным графическим представлением описываемого процесса в виде сети и трактовкой акта процесса как элемента со многими входами, принимающего инициализирующие акт сигналы от других актов, и многими выходами, по которым акт после своего завершения воздействует на другие акты. Можно показать, что двумерная графическая форма представления процессов более выразительна по сравнению с их одномерным текстовым представлением, позволяя часто избежать повторное вхождение одного и того же по смыслу акта в описании процесса [6].

Основная цель настоящей статьи – предложить конструктивные методы оценки сложности параллельных процессов по критериям времени их параллельного выполнения и необходимых для этого ресурсов системы. Это позволит разработчику процессов и систем оценивать границу между предельными и реально достижимыми значениями этих критериев.

В разделе 1 статьи описан вариант языка параллельных процессов из [6] и определены процессная семантика рекурсивно определенных процессов и множество траекторий выполнения процесса, по которым далее определяется сложность процесса.

В разделе 2 кратко рассмотрена проблема однозначной идентификации динамически порождаемых процессов, которая важна при реализации процессов на системах.

В разделе 3 описаны методы оценки среднего времени абсолютно параллельного выполнения процессов и требуемые для этой цели ресурсы (количество узлов системы).

В заключение указаны ждущие решения проблемы.

**1. Язык параллельных процессов.** Рассматриваемый язык параллельных процессов основан на языке функционального параллельного программирования FPTL (Functional Parallel Typified Language) [5] и модели параллельных процессов, определённой в [6]. Модели параллельного выполнения программ на этих языках являются достаточно общими и известные модели параллельных процессов [16,17] представляют их частный случай.

Последнее означает, что любой процесс, описываемый на этих языках, имеет эквивалентное представление на предлагаемом языке параллельных процессов [6] с сохранением всех возможностей его параллельного выполнения. Более того, многие формы параллелизма [1,2], выразимые средствами этого языка, не имеют адекватного представления на языках процессов [16,17].

**1.1. Формальное определение языка.** Пусть – не более чем счетное множество актов, где – множество константных актов, обозначаемых ; – множество пар ортогональных актов-условий, обозначаемых ; – множество переменных актов, обозначаемых ; – акт останова процесса.

Процесс на множестве актов определяется в виде системы в общем случае рекурсивных уравнений , где , а – процессный терм, представляющий композицию актов, полученную посредством следующих операций композиции: .

Терм определяется индуктивно.

1. Каждый акт – терм.
2. Если – термы, то – термы.
3. Если – термы, и – акты множества , то и – термы.
4. Других термов нет.

Все операции композиции ассоциативны, операции композиции – коммутативны. Следующий порядок старшинства операций композиции: позволяет опускать ряд скобок в записи термов.

Данный язык является вариантом языка функционального параллельного программирования FPTL, реализованного на многоядерных компьютерах [5] и языка процессов, предложенного в [6].

В качестве примера приведем описание процесса параллельного выполнения значений функции , определяемой системой рекурсивных функциональных уравнений:

*;*

*.*

Процесс абсолютно параллельного выполнения функции , заданной этой системой уравнений, имеет следующее описание на данном языке:

*;*

*;*

**1.2. Сетевое представление процессов.** Для формулирования правил параллельного выполнения процессов введём модель для их сетевого представления.

1. Сеть любого акта имеет представление, изображённое на рис.1*а*.
2. Если и – термы, то композиции имеют сетевое представление *б* и *в* на рис.1.
3. Сетевое представление терма имеет вид *г* на рис.1. Сетевое представление терма отличается от приведённого удалением блока терма .
4. Процесс, заданный в виде системы уравнений , имеет представление в виде последовательности сетевых равенств: , где – сеть терма .

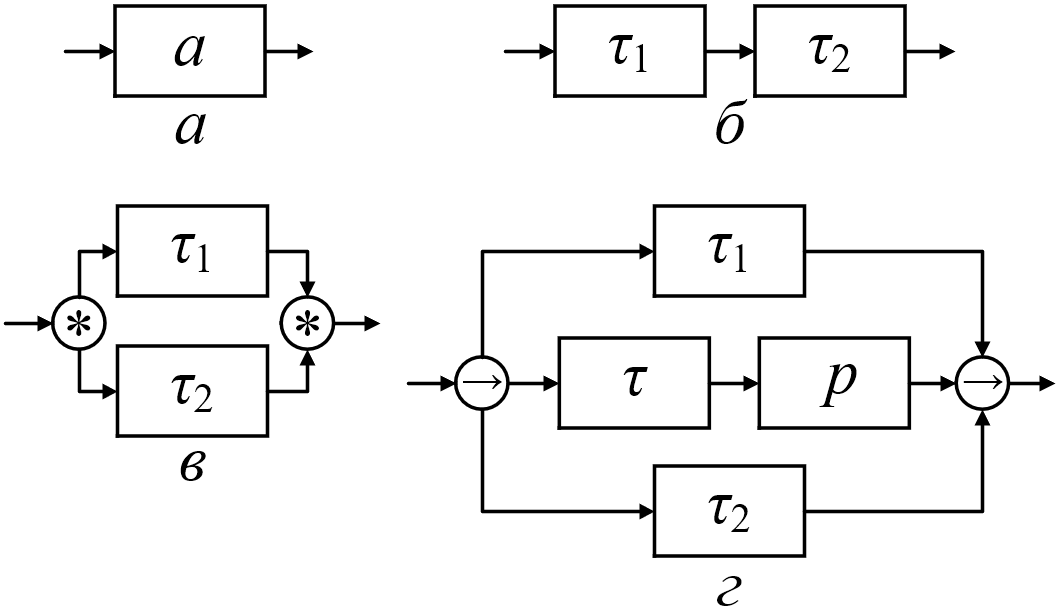


Рис.1. Сетевые представления простейших термов.

На рис. 2 приведён пример сетевого представления, описанного выше процесса.

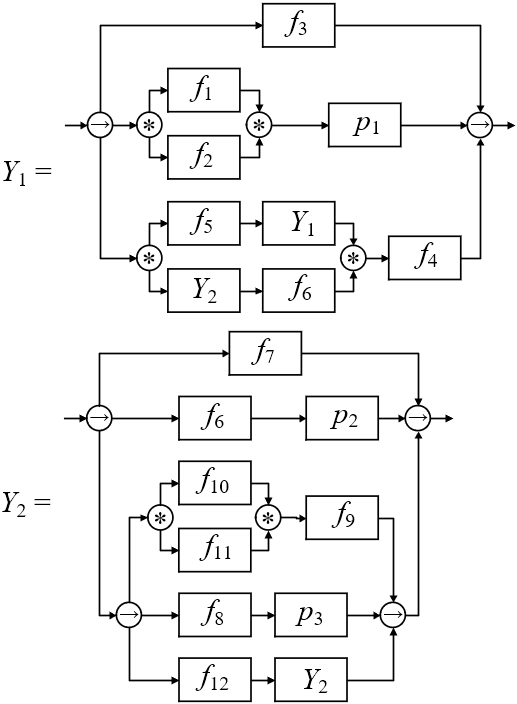


Рис. 2. Сетевое представление уравнений *Y*1 и *Y*2.

**1.3. Правила выполнения процесса.** Правила параллельного выполнения процесса на сети формулируются таким образом, чтобы сохранялась возможность абсолютно параллельного выполнения процесса. При этом мы предполагаем, что выполняются следующие условия.

* Длительность выполнения любого акта, за исключением акта , является конечной и ненулевой, длительность акта равна 0.
* Инициализация любого акта процесса осуществляется без задержек по его готовности для выполнения.
* Завершение выполнения любого акта, не являющегося переменным, сопровождается посылкой инициализирующих сигналов другим актам, на которые акт непосредственно влияет, то есть по дугам на сети, ведущим от акта, завершившего выполнение, к другим актам.
* Выполнение переменного акта сопровождается порождением нового процесса, описанного термом , а в сетевой интерпретации подстановкой сети терма вместо акта и инициализацией выполнения без задержек процесса представленного .
* Выполнения акта условия завершается выбором одного из двух возможных его значений, обозначаемых true или false. При этом при истинном значении процесс в сетевом представлении терма продолжает свое выполнение, если он ещё не завершён, а процесс, представленный в сети термом , прерывается. При ложном значении процесс продолжает выполнение, если он не завершился в момент завершения акта , а процесс, представленный термом , прерывается.
* Парные узлы сети, представляющие пары операций и , выполняют роль управляющих актов. Открывающий узел после его инициализации активизирует одновременное выполнение непосредственно следующих за ним актов по обеим исходящим из него связям на сети, а закрывающий парный ему узел активизируется только по завершении актов на обоих его входах. Открывающий акт операции после своей инициализации ведёт себя аналогично открывающему акту , а парный ему закрывающий акт инициализируется без задержек после поступления хотя бы на один из его входов сигнала завершения непосредственного предшествующего ему акта.
* Выполнение акта завершается без задержек после его инициализации и следующий за процесс не может быть инициализирован.

Не нарушая общности, будем предполагать, что выполнение процесса, заданного системой уравнений (), начинается с выполнения акта , который инициализирует выполнение процесса, описанного .

Каждая конкретная реализация или траектория выполнения процесса, основанного на перечисленных правилах, определяется функциями: и (замкнутый интервал действительных чисел), где определяет для каждого акта в момент его инициализации длительность его выполнения (T-заданная шкала временных отсчётов с отношениями , , , и на ней), а – задаёт вероятность в момент завершения выполнения акта принимать значение «истина» или «ложь», причём .

В реальности длительность выполнения актов процесса является случайной величиной. Простой пример этого – длительность выполнения арифметических операций компьютера, которая зависит от распределения единиц и нулей в мантиссах чисел, для которых выполняется арифметическая операция. Поэтому полное представление о поведении процесса мы можем получить, рассматривая множество всех траекторий выполнения процесса (историй, протоколов, трасс – другие эквивалентные термины), каждая из которых является следствием реализованных функций и .

Таким образом, можно считать, что о характере поведения процесса мы можем судить, рассматривая функцию , порождающую множество всех траекторий его выполнения для заданного множества пар в общем случае случайных функций и .

Множество всех возможных траекторий выполнения процесса конструктивно порождаемо. Для этого в [6] введена автоматная модель параллельного выполнения процесса, в которой состояние процесса определено как множество всех актов, которые одновременно выполняются на некотором шаге (в некоторый момент времени при отображении выполнения процесса на «ось» реального времени). Условием перехода из в состояние , является завершение одного или одновременно нескольких актов в момент , выполняемых в состоянии . При этом множество актов в состоянии включает не завершившие выполнение акты в состоянии и все акты процесса, которые получают право на выполнение после завершивших выполнение актов в соответствии с функцией для условных актов.

**Определение**. Траекторией процесса для заданных функций и назовём в общем случае неограниченную последовательность следования состояний при выполнении процесса, начальным состоянием которого является выполнение в системе уравнений (), описывающей процесс.

Рассматривая все возможные отношения «раньше» или «одновременно» между временем завершения подмножества актов в состоянии, определённых функцией , и все возможные сочетания значений «истина» и «ложь», присваиваемых завершившим выполнение актам-условиям функцией , мы можем определить множество всех возможных последующих состояний процесса. Повторяя эту процедуру для каждого состояния, можно конструктивно воспроизвести все возможные траектории выполнения процесса.

На рис. 3 приведен простой пример, иллюстрирующий построение всех возможных траекторий абсолютно параллельного выполнения процесса . Завершившие выполнение акты указаны на переходах между состояниями. Рассматривается случай, когда длительность акта, выполняемого в различных состояниях, может быть произвольной.

Отметим, что после завершения акта-условия в любом состоянии процесса в зависимости от его значения «истина» или «ложь» прерывается выполнение всех актов в этом состоянии, которые были ранее активизированы и не влияют на продолжение выполнения процесса.

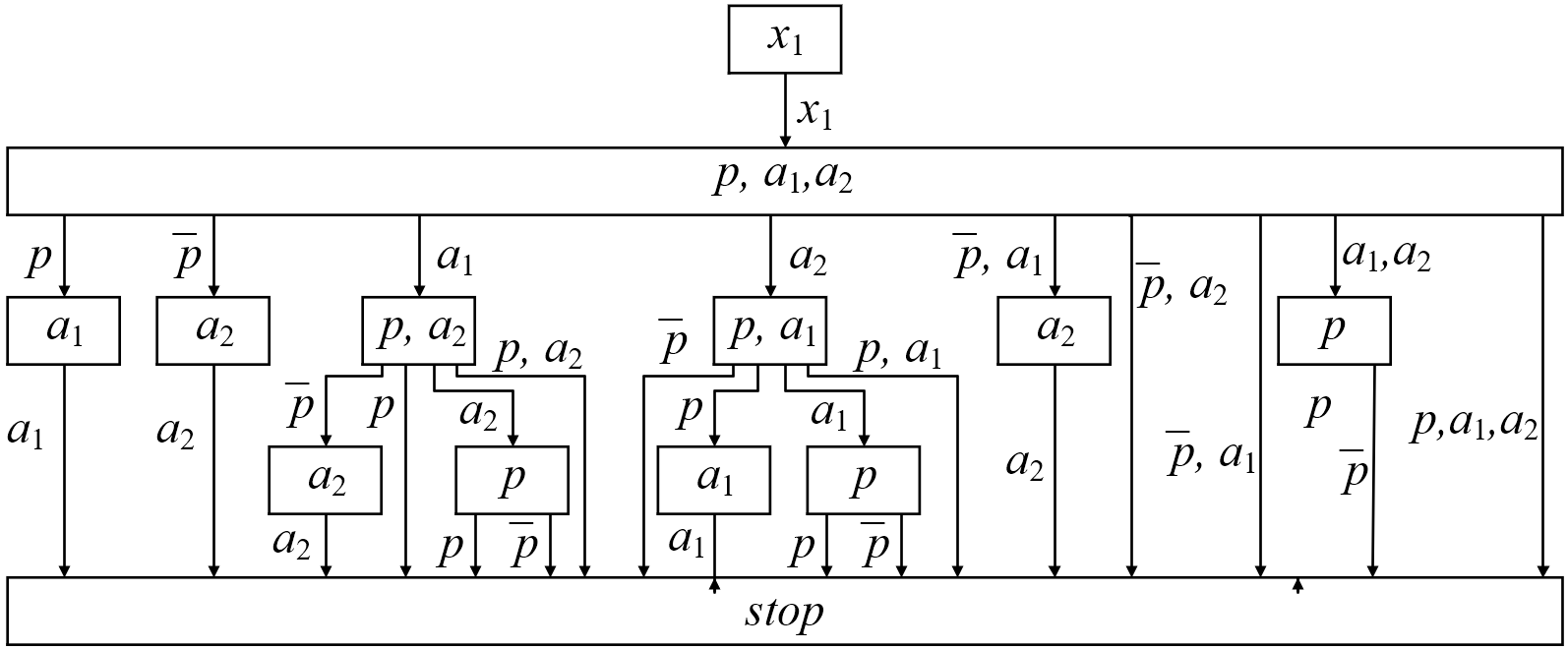


Рис. 3. Множество траекторий процесса.

Перейдем к математической трактовке процессов, определенных в виде системы процессных уравнений (), которая должна дать ответ на вопрос, что является решением для каждого в этой системе.

Для этого определим последовательность процессов для каждого и , положив , , где в правой части указан результат одновременной подстановки , вместо всех вхождений в .

Очевидно, что множество всех траекторий для любого конечно.

**Определение**. Введем отношение строгого порядка на множестве траекторий процессов и , полагая , если для каждой траектории существует траектория такая, что является начальным отрезком траектории (префиксом последовательности состояний ).

Здесь и далее обозначает множество траекторий процесса для заданных функций , – пустое множество.

**Утверждение**. Для любого выполняется .

**Следствие 1**. Последовательность для , образует цепь, наименьшим элементом которой является пустое множество. Эта цепь имеет в качестве предела наименьшую верхнюю грань , .

Согласно известному результату Д. Скотта, данный предел является наименьшим (относительно операции ⊆ включение множеств) решением системы процессных уравнений ().

Это утверждение остается в силе, если одновременную подстановку в определении , заменить на последовательное выполнение в произвольном порядке подстановок вместо в терм .

**Следствие 2**. Процесс, реализуемый в соответствии с описанными выше правилами его выполнения, порождает для каждого в качестве начального состояния множество траекторий, являющееся наименьшим решением для для системы процессных уравнений.

Отношение включения, определенное на множестве траекторий процессов, позволяет формально сравнивать их процессные возможности и говорить об их эквивалентности. Выше мы определили множество траекторий процесса, рассматривая все возможные отношения раньше или одновременно между моментами завершения выполнения актов в каждом состоянии процесса. В реальности время выполнения актов процесса часто заранее фиксировано, что уменьшает множество его траекторий.

Процессы, как и программы, создаются для определенной цели и при их создании главные усилия сосредоточены на поиске такой их семантически эквивалентной формы, которая обеспечивает необходимую степень параллелизма и эффективность (время выполнения процесса и используемые ресурсы). Созданный нами язык функционального параллельного программирования, который реализован на многоядерных компьютерах, имеет средства приведения программы путем её эквивалентных преобразований к эквивалентной форме с большей или меньшей степенью параллелизма в процессе его выполнения [5]. Самое, возможно, интересное в нём заключается в том, что рассматриваемые выше операции композиции процессов изначально были введены как операции композиции функций, что чрезвычайно важно, когда мы пытаемся определить семантическое значение изучаемых процессов.

Изучая процессы, связанные с физическими явлениями и др., мы вынуждены иметь дело с реальным тем или иным способом и с определенной точностью измеряемым временем. При этом условие инициализации акта процесса в общем случае зависит от отношения между моментами времени поступления воздействующих на него «сигналов». Однако для формализации этого необходимо существенное расширение языка процессов, связанное с заданием иной семантики процесса [6].

**2. Однозначное именование инициализируемых актов.** Есть еще один важный аспект процессной реальности, который связан с однозначным именованием порождаемых актов при выполнении процесса. Во-первых, акт с одним и тем же именем может несколько раз входить в термы в описании процесса (). Хотя такие акты рассматриваются в предполагаемой их интерпретации как эквивалентные, в процессе они считаются различными. Операция инициализации терма при готовности для выполнения переменного акта также, как правило, приводит к активизации актов, имена которых совпадают с именами выполняемых актов. Формально можно разными способами ввести однозначное именование такого рода актов. Однако в реализации процессов на практике, например при выполнении параллельных программ на компьютерной системе, при решении проблемы однозначного именования инициализируемых актов время и требуемая память являются основными критериями оптимальности. В реализации языка граф-схемного потокового программирования на компьютерных системах [18] каждому инициализируемому акту присваивается уникальный идентификатор, представляющий собой пару: номер узла, где порождается акт, и уникальный номер (натуральное число), фиксируемый в специальной таблице номеров для исключения повторений. Чтобы уменьшить размер этой таблицы, после завершения выполнения акта его номер считается доступным для нового использования. Отметим, что в системах параллельного программирования PVM [7] и ERLANG [9] алгоритмы однозначного именования процессов реализованы, однако вопрос об их оптимальности остается открытым, более точно, не рассматривался.

**3. Сложность параллельных процессов.** Основными критериями сложности параллельных процессов являются время выполнения процесса и требуемые ресурсы для достижения этого времени. Мы будем исходить из того, что задано константное время выполнения каждого акта процесса, определяемое функцией , и вероятность акта-условия принимать истинное или ложное значение, заданное функцией .

**3.1. Временная сложность процессов.** Для всякого процесса, при заданной длительности его актов, множество траекторий его выполнения однозначно определяется только значениями вероятности актов-условий принимать истинное или ложное значение. Для термов и оба «слагаемых» в них являются ортогональными процессами в том смысле, что только один из них может завершиться успешно. Последнее зависит от того, истинное или ложное значение принимает акт-предикат . Если известна длительность всех актов процесса и вероятность принимать истинное значение для предиката при выполнении процесса, то среднее время выполнения рассматриваемых термов определяется функцией : и .

Существует процедура эквивалентного преобразования любого процесса, сохраняющего среднее время выполнения любого терма , в форму такую, что термы в этом разложении попарно ортогональны и не содержат вхождений операции .

Следующие аксиомы эквивалентности термов из [5] позволяют выполнить это преобразование:

, ,

, ,

, .

Приведём пример преобразования терма к эквивалентной форме ортогонального разложения:

.

Очевидно, что среднее время выполнения терма , приведённого к эквивалентной форме ортогонального разложения , определяется в виде , где – вероятность успешного завершения процесса, представленного термом . Эта вероятность есть произведение вероятностей всех входящих в актов-условий (предикатов) при условии, что они получат значение «истина».

Следующие правила позволяют вычислить среднее время выполнения терма , не содержащего вхождений операции , где – время выполнения терма .

1. Если , , .
2. Если , *.*
3. Если или , *.*
4. Если , .

Возвращаясь к общей форме задания процесса в виде системы уравнений (), для любого среднее время выполнения , можно определить, следуя описанной процедуре. Среднее время вычисления процесса для , определяется очевидным способом как . Для заданной точности вычисления ε можно использовать стандартную в таких случаях процедуру вычисления, применяя метод последовательных приближений и контролируя её окончание по формуле .

Для уравнений с правосторонней рекурсией, то есть сводимых к циклическим определениям, среднее время можно определить, используя решение систем линейных уравнений.

В качестве примера рассмотрим функцию , вариант процесса параллельного выполнения которой можно описать в виде уравнения с правосторонней рекурсией: . Определение среднего времени вычисления процесса сводится к решению линейного уравнения: .

Заметим, что для языка функционального параллельного программирования [5] этот метод вычисления среднего времени параллельного выполнения программ реализован в виде специального инструмента, используемого при разработке и оптимизации программы.

Приведенная оценка среднего времени выполнения процесса предполагает, что достаточно ресурсов для абсолютного выполнения любого множества инициализируемых на разных шагах актов процесса. В реальности при ограниченных ресурсах эта наилучшая оценка времени выполнения процесса позволяет оценить, как «далеко» реальный результат находится от оптимального.

Есть еще один важный аспект оценки организационной сложности процесса, которая непосредственно влияет на его временную сложность, и, что не менее актуально, на сложность описания процесса, его анализ и оптимизацию. В [21] мы исследовали эту проблему для случая параллельного выполнения функциональных программ, в основе которой лежит анализ количества и «глубины» рекурсивных определений в описании процесса, степени взаимной рекурсивности и другое.

Рассмотрение процессов с учётом их интерпретации, то есть конкретной работы или задачи, которую они должны выполнять или решать, существенно расширяет «круг» проблем, с которыми приходится сталкиваться на практике. Это и выбор или построение языка процессов, адекватного решаемой проблеме, и поиск такого описания процесса, который удовлетворяет временным требованиям и ресурсным ограничениям при его выполнении. Язык функционального параллельного программирования интересен тем [5], что в нём между функциональным описанием и процессом параллельного выполнения соответствующей этому описанию программы существует строго устанавливаемая взаимосвязь. Другими словами, множество всех возможных порождаемых при выполнении программы траекторий конструктивно воспроизводится по её описанию [5], кроме того, существует система эквивалентных преобразований функциональных программ [22,23], позволяющая приводить программу к максимально параллельной форме.

**3.2. Ресурсная сложность процессов.** Объем ресурсов, необходимых для реализации процесса, еще одна, возможно центральная, проблема, когда мы говорим о реализации процессов на системах. Эта проблема формулируется в общей постановке как требование минимизации времени выполнения процесса на заданном количестве ресурсов (компонентов) некоторой системы и её решение базируется на создании эффективных методов управления процессами, главными из которых являются методы планирования процессов и распределения ресурсов [3]. При этом для приближения решения этой проблемы к реальной практике приходится учитывать специфические особенности поведения процессов, являющихся в общем случае случайными, возможность достаточно точного прогнозирования изменения их характеристик [4] и др.

Ниже для рассматриваемого языка процессов мы введем еще один критерий их сложности, по которому можно определить предельные значения ресурсов, необходимых для их абсолютно параллельной реализации.

Для этого мы используем рассмотренную в п.1.3 автоматную модель представления траекторий выполнения процесса, состояниями которых являются множества актов, которые одновременно выполняются в определённые моменты времени.

Пусть – состояние процесса в момент времени t, а – количество одновременно выполняемых актов в этом состоянии.

**Определение**. Для каждой траектории определим величины, которые характеризуют максимальное и среднее количество одновременно выполняемых актов в состояниях траектории: и , где – длительность траектории. Максимальное значение первой величины, вычисляемое относительно всех траекторий процесса, дает информацию о максимальном количестве узлов системы, необходимой для абсолютно параллельной его реализации.

Суммарное значение среднего количества актов для каждой траектории процесса , умноженное на вероятность траектории, характеризует среднее значение количества используемых узлов системы. Известно, что эффективность (время и используемые ресурсы) выполнения процесса на системе также существенно зависит от интенсивности обменных взаимодействий между узлами системы. В автоматном представлении выполнения процесса переходы между состояниями взвешены множествами актов, одновременно завершающих выполнение в исходном состоянии, а в новое состояние добавляются инициализируемые после перехода акты процесса. Все эти действия так или иначе определяют нагрузку на каналы и управление системы при выполнении процессов. По вычисляемым аналогичным образом этим параметрам можно получить данные о нагрузке на каналы и управление системы при выполнении процессов.

**Заключение.** Рассмотренные в статье методы оценивания временной и ресурсной сложности параллельных процессов являются «оптимистичными». В действительности при выполнении процессов на реальных системах существенное влияние на время выполнения процесса оказывает ресурсная среда (количество узлов системы и их технические характеристики), время, затрачиваемое на управление процессами и реализацию обменных взаимодействий [3,4]. Для параллельных программ принципиальным факторами, влияющими на время их выполнения, являются организация программы и степень распараллеливания [1,2]. Последний фактор существенно влияет на время выполнения программы, как показывает практика [5,19], существует оптимальный уровень распараллеливания, при котором достигается минимальное время выполнения программы на используемой системе с заданным в ней количеством узлов и ядер.

**Список литературы**

1. Кутепов В.П. О параллелизме с разных сторон. 5-я Международная конференция «Параллельные вычисления и проблемы управления», М: Институт проблем управления, 2010. С. 41-52.
2. Кутепов В.П., Фальк В.Н. Формы, языки представления, критерии и параметры сложности параллелизма. Программные продукты и системы. 2010, №3. С.16-25.
3. Кутепов В.П. Интеллектуальное управление процессами и загруженностью в параллельных системах. Изд. РАН, ТиСУ, 2007, №5. С. 58-73.
4. Бражникова Ю.Н., Горицкий Ю.А., Кутепов В.П., Панков Н.А. Исследование методов прогнозирования загруженности компьютеров и компьютерных систем. Программные продукты и системы, 2015, №2. С. 135­-147.
5. Кутепов В.П., Шамаль П.Н. Реализация языка функционального параллельного программирования FPTL на многоядерных компьютерах. Изд. РАН, ТиСУ, 2014, №3. С. 46-60.
6. Кутепов В.П. Модели и языки для описания параллельных процессов. Изв. РАТ, ТиСУ, 2018, №3. С. 116-127.
7. Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy Sunderam. PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. IEEE, 1996. 84 p.
8. Хьюз К., Хьюз Т. Параллельное и распределенное программирование с использованием C++. М.:Вильямс, 2004. 667 с.
9. Cesarini F., Thompson S. ERLANG programming: a concurrent approach to software development. O'Really, 2009. 498 p.
10. RICHARD H. CARVER, KUO-CHUNG TAI. Modern Multithreading: Implementing, Testing, and Debugging Multithreaded Java/C++. Wiley-Interscience, 2005. 480 p.
11. Burstall, R.M., Sannella, D.T. HOPE: An experimental applicative language. //LFP '80 Proceedings of the 1980 ACM conference on LISP and functional programming, 1980, P. 136-143.
12. Саймон Марлоу. Параллельное и конкурентное программирование на языке Haskell. М.: ДМК Пресс, 2014. 372 c.
13. Общая теория систем, М.: Мир, 1966. 188 c.
14. Заде Л. Понятие состояния в теории систем. Сб. статей «Общая теория систем», М.: Мир, 1966, С. 49-65.
15. Майхилл Дж. Абстрактная теория самовоспроизведения. Сб. статей «Общая теория систем», М.: Мир, 1966, С. 121-140.
16. Milner R. A calculus of communicating systems Lecture notes in computer science, Springer-Verlag, New York, 1980. 174 p.
17. Хоар Ч. Взаимодействующие последовательные процессы. М.: Мир, 1989, 264 с.
18. Кутепов В.П., Маланин В.Н., Панков Н.А. Граф-схемное потоковое параллельное программирование: язык, процессная модель, реализация на компьютерных системах. Изд. РАН, ТиСУ, 2012, №1, С. 87-102.
19. Кутепов В.П., Зубов М.И. Реализация и экспериментальное исследование эффективности упреждающего параллелизма. Вестник МЭИ, 2019, №4, (в печати).
20. Питерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 1984, 264 с.
21. Бажанов С.Е., Воронцов М.М., Кутепов В.П., Шестаков Д.А. Структурный анализ и планирование процессов параллельного выполнения функциональных программ. Изв. РАН, ТиСУ, 2005, №6, С. 111-126.
22. Кутепов В.П., Фальк В.Н. Функциональные системы: теоретический и практический аспекты. Кибернетика, 1979, №1, С.46-58.
23. Кутепов В.П. Исчисление эквивалентности функциональных схем и параллельные алгоритмы. Программирование, 1979, №6.