ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2025, № 3, с. 60–69

КОМПЬЮТЕРНЫЕ МЕТОДЫ

УДК 519.687

ДИСКРЕТНЫЕ ДЕТЕРМИНИРОВАННЫЕ ПРОЦЕССЫ: РЕАЛЬНОЕ ВРЕМЯ, ПАРАЛЛЕЛИЗМ, СЛОЖНОСТЬ, ЯЗЫКИ ОПИСАНИЯ

© 2025 г. В. П. Кутепов

Национальный исследовательский университет "МЭИ", Москва, Россия

e-mail: kutepov@appmat.ru

Поступила в редакцию 25.06.2024 г.

После доработки 26.10.2024 г.

Принята к публикации 24.02.2025 г.

Рассмотрены модели и языки, предназначенные для описания и управления выполнением в реальном времени дискретных детерминированных процессов. Процессы разделены на свободные и зависимые от реального времени, введены характеристики сложности процессов, учитывающие фактор одновременности (параллелизма) при их выполнении. Предложены унифицированные операции композиции вычислимых функций, на основе которых создан язык, позволяющий в естественной форме описывать параллелизм алгоритмических процессов. Показано, каким образом фактор реального времени может быть отражен в процессах путем расширения языка примитивами управления изменением состояний процесса в зависимости от контролируемого времени.

Ключевые слова: параллельные процессы, реальное время, модели и языки параллельных процессов, сложность процессов

DOI: 10.31857/S0002338825030066, **EDN:** BGCGGJ

DISCRETE ALGORITHMIC PROCESSES: REAL TIME, PARALLELISM, COMPLEXITY, LANGUAGES

V. P. Kutepov

National Research University "Moscow Power Engineering Institute", Moscow, Russia

e-mail: kutepov@appmat.ru

Models and languages for descriptions and control of real time execution of discrete algorithmic processes are considered in the paper. Processes are divided into two groups: free and of real time processes. New operations for composition of parallel processes and language for their descriptions are proposed. The methods of parallel processes complexity evaluation are introduced.

Keywords: parallel processes, real time, models and languages for parallel processes, complexity of parallel processes

Введение. Понятие процесса является центральным при исследовании различных физических и природных явлений, проектировании различных систем, описании деятельности людей и т.п. Привычная классификация процессов разделяет их на непрерывные и дискретные, детерминированные и случайные, конструктивные, т.е. поддающиеся строгому языковому описанию в виде алгоритмов, дифференциальных уравнений и др., и открытые, к конечному описанию которых можно только приближаться.

В самом общем виде дискретный процесс (в статье рассматриваются дискретные процессы) может быть определен как частично упорядоченное следование во времени актов (неделимых и специфичных действий), подчиненных определенным причинно-следственным отношениям. Глубина детализации этих наблюдений прямо зависит от точности, с которой можно измерять реальное время.

Точность измерения времени, будь это привычные часы или другие измерительные приборы, всегда ограничена как интервалом (квантом) их собственной периодичности, так и в силу

известного принципа Гейзенберга о невозможности одновременно с абсолютной точностью контролировать изменения состояний процесса и моменты времени их изменений.

Для физических процессов на атомном уровне эта точность может достигать 10^{-15} с. и выше. Например, измеренное время прохождения рентгеновского луча через атом водорода составляет порядка 2.47×10^{-19} с. В компьютерах и компьютерных системах точность измерения времени определяется частотой задающих ее генераторов. Оперируя процессами в реальном времени, надо также учитывать неизбежные задержки, связанные со снятием показателей с соответствующих “часов”, реально измеряющих время.

Кроме проблемы создания новых или расширения старых языков с целью обеспечение их средствами описания процессов, зависимых от реального времени, не менее сложной и не до конца решенной является проблема создания языков и операционных средств для описания и эффективной реализации управления такими факторами реальных процессов, как одновременность (параллелизм) выполнения множества актов, их синхронизация, контроль асинхронного завершения, регулируемый определенными условиями, и др.

Акторные модели [1], сети Петри [2, 3], схемы параллельных программ [4], алгебраические модели и языки параллельных процессов [5–8] и многие другие работы по-разному решают данную проблему. Эти работы во многом предопределили направленность и выразительную силу созданных на их основе языков параллельного программирования [9–12].

В статье основной акцент сделан на решении проблемы поиска в определенном смысле унифицированных средств для описания параллелизма алгоритмических процессов, зависимых от реального времени, и создании на этой основе языков параллельного программирования, процессы выполнения программ которых эту зависимость от реального времени реализуют.

В разд. 1 описана обобщенная акторная модель дискретных процессов, состояния которых идентифицируются как множества одновременно выполняемых актов в конкретный момент времени, а условием переходов является завершение одного или одновременно нескольких актов процесса. Эта модель позволяет осуществить прямой переход к оцениванию динамических характеристик сложности процессов: степени их параллелизма, характера изменения подмножеств актов процесса, выполняемых на заданных временных интервалах. В разд. 2 рассматриваются шкалы времени, отношения между ними, примитивы управления изменением состояний актов процесса в зависимости от значений реально контролируемого времени. В разд. 3 предложен универсальный набор операций композиции вычислимых функций, процессная интерпретация которых одновременно придает им универсальный характер при описании параллелизма алгоритмических процессов.

Предложен вариант языка параллельного программирования, обогащенный множеством примитивов, позволяющих в процессах описывать их зависимость от контролируемого реального времени. В Приложении рассмотрены общие черты прикладной временной логики, которая строится на базе классического исчисления предикатов и может быть инструментом формального доказательства утверждений о процессах, зависимых от реального времени.

1. Общее представление дискретных процессов и их сложность. Неформально процесс определяют как упорядоченное следование фиксированного множества событий, действий (далее – актов), подчиненных причинно-следственным отношениям между ними. Если причинно-следственные отношения между актами зависят от временного фактора, то процесс относят к зависимым от тем или способом измеряемого времени. Практически это означает, что условия инициирования выполнения актов процесса зависят не только от значений, воздействующих на них “сигналов”, которые поступают от завершивших выполнение актов, но и от моментов времени их поступления и отношений между ними.

Как следствие этого, при изучении процессов используется событийная и временная модели. При этом для ниже определенных свободных процессов часто достаточна более простая событийная модель.

О п р е д е л е н и е 1. Процесс назовем свободным, если множество всех допустимых его реализаций (трасс, см. далее) не изменяется при любой конечной отличной от нулевого значения длительности выполняемых им актов.

Практически это означает, что условие выполнения актов процессов не зависит от времени поступления завершивших выполнения актов, воздействующих на его инициирование. Процессы выполнения последовательных программ традиционных языков программирования (C, C++, других языков), языков параллельного программирования (MPI, ERLANG, FPTL и других языков) относятся к классу свободных процессов.

Как следствие, изменение технических характеристик компьютерной платформы сохраняет неизменным семантическое значение программы как функционального преобразования входных данных в результат.

Свободный дискретный процесс можно определить в виде четверки $<A,A_0,R,EC>$, где $A = \{a_i | i = \overline{1,n}\}$, n > 0, – множество актов, которые участвуют в процессе, $R \subseteq A \times A$ – отношение причинно-следственных отношений между актами, $A_0 \subseteq A$ – подмножество актов, с которых начинается выполнение процесса, EC – правила выполнения процесса, определяющие на событийном уровне условия инициирования выполнения акта по завершению выполнения воздействующих на него актов, согласно отношению R.

Для процессов, зависимых от времени, модели которых рассматриваются в следующем разделе статьи, условия инициирования актов существенно зависят от применяемой модели времени и временных отношений между поступающими на входы акта воздействий от завершивших выполнения актов процесса. Эти акты определяются отношением R, которые в общем случае, в свою очередь, могут изменяться с течением времени.

В качестве примера на рис. 1 приведена временная диаграмма процесса абсолютно параллельного вычисления значений функции, заданной в виде рекурсивного уравнения: $F(x) = if\ p(f_1(g_1(x)))\ then\ g_2(x)\ else\ F\ (f_2(g_3(x)))$.

На рис. 1 порядок инициирования функций сопровождается приписыванием им порядкового номера (индекса сверху), позволяющего идентифицировать их как различные акты при выполнении. Возможно другое, более общее графическое представление выполнения процесса в виде множества всех его трасс, которое позволяет явно отобразить не только следование актов процесса во времени, но и причинно-следственные зависимости между актами (рис. 2). На рис. 2 на дугах должно быть указано время вычисления значений функций (время выполнений соответствующих им актов).

Определение 2. Часть трассы выполнения процесса, наблюдаемой в момент времени t, множество всех актов, выполняемых и завершивших выполнение в этот момент, будем обозначать соответственно $h(t)$, $S(t)$ и $S_{end}\ (t)$.

2. Автоматное представление и сложность процессов. Введем автоматное описание выполнения процесса, которое далее используется при рассмотрении проблемы оценивания динамических характеристик сложности процессов. Пусть процесс выполняется на множестве актов $A = \{a_i |\ i = \overline{1,k}\}$, $k > 0$.

Определение 3. Автоматная модель процесса на множестве актов A определяется совокупностью следующих компонент: (S, S_o, S_k, T, F), где S, S_o, S_k – множества всех, начальных и конечных состояний процесса, T – “шкала” времени, по которой с определенной точностью

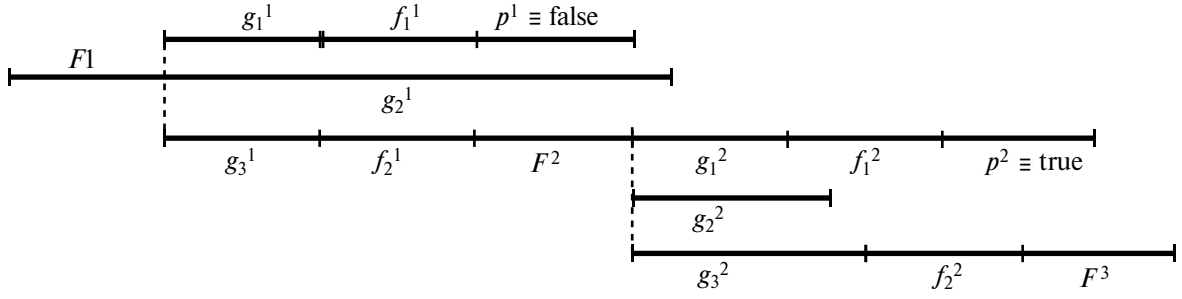

Рис. 1. Временная диаграмма абсолютно параллельного вычисления значения функции $F(X)$.

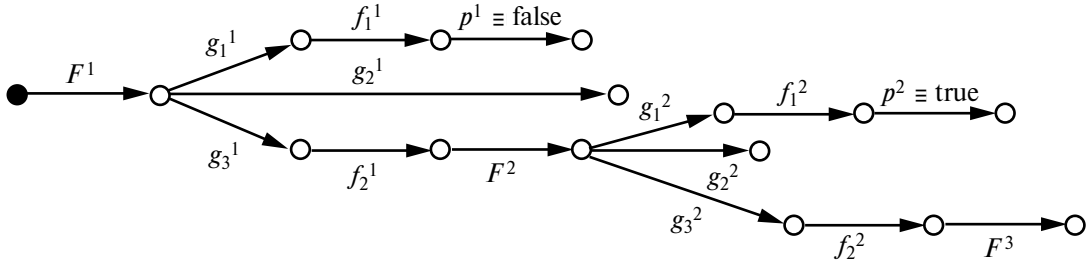

Рис. 2. Трасса выполнения процесса.

измеряются моменты времени изменения состояний процесса, F: $S \times T \times S_{end} \to S$ – функция изменения состояний процесса.

Будем предполагать, что изменение состояний процесса происходит в моменты времени $t \in T$ завершения подмножества $S_{end}(t)$ множества всех выполняемых актов процесса в этот момент. Обозначим через $S_{new}(t)$ подмножество всех актов процесса, которые могут начать выполнение после завершения актов $S_{end}(t)$.

Новое состояние процесса $S(t + \Delta t)$ определяется следующей формулой:

$S(t + \Delta t) = S(t) \backslash S_{end}(t) \cup S_{new}(t)$.

На функцию F накладывается условия независимости от будущего времени и непрерывности [6]:

$F\ (S, t_0, \emptyset) = S_0$,

где $\emptyset$ – пустое множество, S_0 – начальное состояние,

$F\ (S, t_k, S_{end}) = F\ (F\ \ldots F(F(F(\ S_0, \emptyset, t_0), S_{end}, t_1)), S_{end}, t_2) \ldots t_{k-1}), S_{end}, t_k)$,

где $t_0, t_1, \ldots, t_k$ – моменты времени изменения состояний процесса.

Уточним понятие состояния процесса, определяемое через состояния актов. Поскольку при выполнении процесса, как было отмечено, несколько его актов с одним и тем же именем могут быть одновременно инициированы и одновременно выполняться (рис. 1), его состояния логично определить, используя понятие мультимножества [13].

Определение 4. Пусть задано множество $A = \{a_i \mid i = \overline{1,k}\}$, $k > 0$.

Мультимножество множества A – это множество, в котором элементы множества A могут повторяться конечное число раз с присвоением им отличающих идентификаторов [13]. Для процесса его актам присваиваются номера в порядке их инициирования, а состояние процесса $S(t)$ определяется как мультимножества актов, выполняемых в момент времени t.

Определенная в этом разделе формула изменения состояний процесса $S(t + \Delta t) = S(t) \backslash S_{end}(t) \cup S_{new}(t)$ позволяет перейти к определению функций сложности процесса.

Определение 5. Функции $n(t) = |S(t)|, n_{end}(t) = |S_{end}(t)|, new\ (t) = |S_{new}(t)|$ будем называть функциями сложности процесса. Первая из них характеризует общую сложность процесса, измеряемую количеством актов, выполняемых в различные моменты времени. Ее максимальное значение характеризует максимальное количество ресурсов, которые должны быть в системе для абсолютно параллельного выполнения процесса. Функции $new(t)$ и $n_{end}(t)$ дают представление о нагрузке, которая ложится на средства управления (операционную систему и др.) выполнением процесса и транзакции в системе. Другие динамические характеристики процесса также могут вычисляться по этим функциям [14–16].

Естественно, что контролировать моменты одновременного завершения нескольких актов можно только с ограниченной точностью. Реально изменение состояний контролируются индивидуально в моменты завершения выполнения каждого акта, что не исключает реальной одновременности выполнения этого действия несколькими актами процесса.

3. Модели процессов, зависимых от реального времени. Общая форма модельного описания различных классов процессов, зависимых от времени, состоит из следующих элементов: $< A, SYN, F_{log}, I, MEC >$, где $A = \{a_i | i = \overline{1,n}\}$, $n>0$, – множество акторов-исполнителей одноименных актов в процессе, SYN – синтаксис языка, которым порождается определяемый класс схем процессов SCH, SYN: $A \to SCH$, F_{log} – множество формул прикладной временной логики (она приведена в Приложении статьи), которыми определяются зависимые от времени условия инициирования и завершения выполнения акторами процесса соответствующих им актов, I – интерпретация функций или отношений, реализуемых актами при выполнении процесса, MEC – модель выполнения процессов.

Заметим, что известная классификация языков программирования по их выразительным возможностям и в соответствии с их синтаксисом (регулярные, контекстно-свободные и контекстно-связанные) остается в силе и для языков процессов с параллельной семантикой их выполнения.

Конкретный процесс в рассматриваемой модели задается на множестве акторов $A' \subseteq A$ в виде следующих элементов: $< S_0, S_{fin}, F_{in}, F_{out}, F_{do} >$, где S_0 и S_{fin} – подмножества акторов множества A', с которых начинается и которыми завершается выполнение процесса; F_{in}: $A' \to F_{log}$ – функция, сопоставляющая акторам A' логические временные функции из множества F_{log}, которыми определяются условия и моменты времени инициирования акторами из A' актов процесса; F_{out}: $A' \to F_{send}$ – функция, которая сопоставляет акторам функции из множества F_{send}, определяющие время и акторы адресаты, которым передается инициирующая их выполнение информация завершившими выполнение актами; F_{do}: $A' \times T \to T$ – функция, которой определяется длительность выполнения актов в момент их инициирования.

Уточним семантику этих функций. Синтаксисом SYN определяется для каждого процесса отношение $R \subseteq A' \times A'$ воздействия завершивших выполнение актов процесса на инициирование выполнения новых актов. Обозначим через $in(a_i) = \{(a_j | (a_j, a_i) \in R\}$ и $out(a_i) = \{(a_j | (a_i, a_j) \in R\}$ множество актов, которые влияют на инициирование акта a_i, и множество актов, на инициирование которых влияет акт a_i соответственно.

Сопоставленная актору a_i логическая функция из $F_{\log}$ имеет следующую форму ее параметров:$(val(a_{i1}), t_1), (val(a_{i2}), t_2), ..., (val(a_{ik}) t_k), clock(m)$, где пара $(val(a_{ij}), t_j)$ представляет значение инициирующего сигнала, поступающего на i-й вход актора a_i, и время его поступления t_j. Параметр $clock(m)$ есть функция, "снимающая" реальное время, которое измеряется часами с номером m в момент поступления последнего инициирующего актор a_i сигнала. Для различных процессов точностью измерения времени определяется степень приближения контролируемых динамических характеристик процесса к их реальным значениям. Практически функция инициирования актором акта представляет собой конъюнкцию двух логических функций, одной из которых определяет отношение между величинами поступающих на входы актора сигналами, а второй – отношение между моментами времени их поступления и реальным временем.

Функцией F_{out} каждому актору процесса сопоставляется конечное подмножество множества $F_{\log}$ ортогональных логических функций, одна из которых принимает значение "истина" в момент завершения выполнения акта a_i и которой определяется подмножество множества $out(a_j)$ акторов, которым передается инициирующая их выполнение информация.

Общий вид функций множества F_{send}, реализующих процедуру передачи данных, может быть таким:

$if\ \varphi_1(\vartheta al(out(a_i)), t)\ then\ send(subset_1(\vartheta al(out(a_i))), t)\ else\ if\ \varphi_2(\vartheta al(out(a_i)), t)\ then\ send(subset_2(\vartheta al\ (out(a_i))), t)\ else\ ...\ if\ \varphi_k(\vartheta al(out(a_i)), t)\ then\ send\ (subset_k(\vartheta al(out\ (a_i))), t)$, $k > 0$.

Здесь ортогональными функциями φ_i, $i = \overline{1,k}$, которые активируются в момент времени t завершения выполнения акта a_i, определяется подмножество акторов множества $out(a_i)$, которым функциями $subset_i$ и $send_i$ передаются значения сигналов для их инициирования.

Естественное требование минимизации времени реализации таких функций в управлении процессом является важной задачей.

4. Язык параллельных процессов. Описание одновременности и асинхронности выполнения множества актов в процессах является не тривиальной задачей при создании языка для этой цели. Это особенно хорошо прослеживается, когда проводится анализ перехода от последовательного к параллельному программированию. Простые примитивы описания параллелизма $begin\ O_1, O_2, ..., O_n\ end, fork <$ множество нитей $> join$, где нить – линейная последовательность операторов программы, остаются базовыми в известных языках параллельного программирования MPI, HASKEL, ERLANG и др.

Статьи [5, 6] являются основополагающими с точки зрения поиска универсальных средств описания параллелизма дискретных детерминированных процессов. К традиционным и необходимым операциям последовательной композиции и выбора продолжения процесса в [5, 6] предложены две новые операции: параллельной композиции процессов и их взаимодействия. Однако анализ разнообразных семантических форм параллелизма, которые присущи вычислительным процессам, связанным с вычислением значений функций и логическими преобразованиями, показывает, что для его описания необходим более широкий класс процессных примитивов [8, 11–13].

Ниже представлен язык, который существенно расширяет возможности описания параллелизма дискретных процессов по сравнению с языками, предложенными в [5, 6]. В нем используется более широкий набор операций композиции процессов, позволяющих явно указывать на одновременность и асинхронность выполняемых процессом актов. Принципиальная особенность языка состоит в том, что при функциональной интерпретации его можно считать универсальным языком описания параллельных вычислительных процессов. Полное формальное определение и реализация этого языка приведены в [7, 11, 12].

4.1. Синтаксис языка. Базовыми элементами языка являются следующие множества A актов, выполняющих определенные функции в процессе:

$$A_1 = \{a_i^{(m_i,n_i)} \mid i = \overline{1,k_1}\}, A_2 = \{p_i^{(m_i,1)} \mid i = \overline{1,k_2}\} \cup \left\{\neg^{(1,1)}\right\},$$

$$A_3 = \{[m,k] | m \geq k > 0\}, A_4 = \left\{X_i^{(m_i,n_i)} | i = \overline{1,k_3}\right\},$$

где m, m_i, n_i, k, k_1, k_2, k_3 – элементы натурального ряда чисел, $[m,k]$ – функция выбора k –го элемента из кортежа данных длины m.

Элементы этих множеств будем называть безусловными, условными, выбора элемента из кортежа и переменными актами. Пара чисел (m, n) в обозначении акта указывают на количество его входов m и количество выходов n соответственно, которая далее называется арностью акта.

Процесс задается в виде системы в общем случае рекурсивных уравнений:

$(*)$ $X_i^{(m_i,n_i)} = \tau_i^{(m_i,n_i)}$, $i = \overline{1,k}, k \in NAT$, где $r_i^{(m_i,n_i)}$ – процессные термы указанной арности, определяемые ниже на множестве операций композиции процессов: $\{\cdot, *, \to, \bigoplus\}$.

1. Любой акт, принадлежащий множествам A_1, A_3,A_4, – безусловный терм. Акты множества A_2 – условные термы. Арность этих термов равна арности соответствующих им актов.

2. Если$\tau_1^{(m_1,n_1)}$ и $\tau_2^{(m_2,n_2)}$ – безусловные термы, то $(\tau_1 \cdot \tau_2)^{(m_1,n_2)}$ при условии $n_1 = m_2$ и $(\tau_1 * \tau_2)^{(m_1,n_1+n_2)}$ при условии $m_1 = m_2$ – безусловные термы указанной арности.

3. Если $\tau^{(m,n)}$ – безусловный терм, то $(\tau \cdot p_i^{(n,1)})^{(m,1)}$ – условный терм указанной арности, где $p_i^{(n,1)}$) акт множества A_2. Если $\tau^{(m,1)}$ – условный терм, то $(\tau^{(m,1)} \cdot \urcorner)^{(m,1)}$ – условный терм.

4. Если $\tau_1^{(m_1,1)}$– условный терм,$\tau_2^{(m_1,n_2)}$ – безусловный терм, то $(\tau_1 \to \tau_2)^{(m_1,n_2)}$ – безусловный терм. Если $\tau_2^{(m_1,1)}$– условный терм, то $(\tau_1 \to \tau_2)^{(m_1,1)}$ – условный терм.

5. Если $\tau_1^{(m_1,n_1)}$и $\tau_2^{(m_1,n_1)}$ – безусловные термы, то $(\tau_1 \oplus \tau_2)^{(m_1,n_1)}$– безусловный терм. Если $\tau_1^{(m,1)}$ и $\tau_2^{(m,1)}$ – условные термы, то $(\tau_1 \bigoplus \tau_2)^{(m,1)}$ – условный терм.

6. Других термов в языке нет.

Следующее уменьшающееся слева направо старшинство операций композиции процессов $\cdot, *, \to, \bigoplus$ позволяет опускать скобки в записи термов.

4.2. Интерпретация актов. 1. Акты процесса интерпретируются как однозначные преобразователи сигналов данных, поступающих на их входы. Акт арности $(m,n), m > 0$, начинает выполнение без задержек после поступления на все его m входов сигналов от других актов, завершивших выполнение. Если хотя бы на один вход акта поступает сигнал со значением ω (см. далее), инициирование выполнения акта запрещается. Акты, для которых $m = 0$, начинают выполнение без задержек. После завершения выполнения акт отправляет n сигналов другим актам с целью инициирования их выполнения.

2. Условные акты $p_i^{(mi,1)}$ завершают выполнение одним из двух значений 1 или 0, этими двумя значениями реализуется выбор продолжения или прерывания процесса. Акт $\urcorner^{(1,1)}$ изменяет значение поступившего на его вход сигнала 1 на 0 и 0 на 1.

3. Акт $[m, k]$ выбирает k–е поступившее значение сигнала из m сигналов, могущих поступить на его m входов.

4. После поступления сигналов, отличных от ω, на все входы переменного акта $X_i^{m_i,n_i}$ инициируется переход к выполнению терма $\tau_i^{m_i,n_i}$ в системе уравнений, определяющий процесс.

4.3. Правила выполнения процесса. Не нарушая общности, будем считать, что завершение процесса, определенного системой уравнений (*), начинается с переменного акта $X_1^{(m_1,n_1)}$ путем перехода к выполнению всех начальных актов терма $\tau_1^{(m_1,n_1)}$.

Предполагается, что выполняются требования конечной длительности выполнения актов и передачи сигналов другим актам после своего завершения. Операции композиции процессов выполняют роль операций управления, контролируя последовательное или одновременное выполнение актов процесса, условия их инициирования и завершения.

1. Операция последовательной композиции в процессе $(PR_1 \cdot PR_2)$ предполагает, что инициирование любого акта процесса PR_2 всегда следует после завершения актов процесса PR_1, воздействующих на акт.

2. Операция $*$ предполагает одновременное выполнение актов процессов PR_1 и PR_2 в композиции $(PR_1 * PR_2)$ по их готовности (поступлению воздействующих на их инициирование сигналов). Это означает, что допускается разновременное инициирование выполнения параллельных процессов PR_1 и PR_2 (см. пример ниже**).**

3. Акт $[m, k]$ начинает выполняться после поступления на k-й его вход сигнала от завершившего выполнение акта процесса не зависимо от того, поступили или нет сигналы на другие его входы. Это означает, что до завершения параллельного выполнения процессов PR_1 и PR_2 в композиции $(PR_1 * PR_2)$ возможно инициирование актом $[m, k]$ новых актов, одновременно выполняемых с актами процессов PR_1 и PR_2. Например, в процессе $\left(a_1^{(1,1)} * a_2^{(1,1)}\right) \cdot \left([2,2] \cdot a_3^{(1,1)} * a_4^{(2,1)}\right)$ возможно одновременное выполнение акта $a_3^{(2,1)}$ с еще не завершившим выполнение актом $a_1^{(1,1)}$ при условии, что акт $a_2^{(1,1)}$ завершил выполнение.

4. Процессы PR_1 и PR_2 в $(PR_1 \to PR_2)$, где PR_1 – процесс-условие, начинают выполнение по готовности и выполняются параллельно. Если процесс PR_1 завершается раньше процесса или одновременно с ним, то результат выполнения $(PR_1 \to PR_2)$ определяется PR_2 при условии, что процесс PR_1 завершается условным значением 1. Если процесс PR_1 завершается со значением 0, то процесс $(PR_1 \to PR_2)$ завершается прерыванием и неопределенным значением ω. Если PR_2 завершается раньше PR_1, то после завершения PR_1 по его значению 0 или 1 определяется успешное завершение процесса или процесс завершается со значением ω. Процессы вида $(\tau \to \tau_{2)}$ и $(\tau \cdot \urcorner \to \tau_2)$ назовем ортогональными, поскольку только один из них может завершиться со значением, отличным от ω.

5. Процессы вида $(PR_1 \oplus PR_2)$ начинают выполнение по готовности PR_1 или PR_2 и выполняются одновременно. Результатом выполнения $(PR_1 \oplus PR_2)$ является успешное завершение первым одного из процессов или одновременное успешное завершение обоих процессов. Операция $\oplus$ позволяет реализовать параллельное выполнение ортогональных или содержательно эквивалентных процессов одновременно, обеспечивая минимизацию общего времени выполнения.

Отметим, что все операции композиции и логика определения процессов зарождались в поисках универсальных средств для описания параллелизма алгоритмических процессов, в частности процессов параллельного вычисления значений функций [7, 11, 12].

5. Решение систем процессных уравнений. Автоматная модель параллельного выполнения процессов, введенная в разд.2 статьи, предполагает, что состояние автомата ассоциируется с множеством всех допускающих одновременное выполнения актов процесса, а переходы из одного состояния в другое осуществляются в моменты одновременного завершения любого подмножества актов в состоянии.

Рассмотрим два варианта конструктивного определения множества трасс выполнения процесса, заданного в виде системы процессных уравнений $X_i = \tau_i, i = \overline{1,k}$.

Первый вариант следует стандартной модели пошаговой реализации рекурсивных вычислений путем подстановки вместо переменных X_i их правых частей в системе уравнений.

Определение 6. Положим $\vartheta_i^{n+1} = X_i^n \cdot \tau_i^{n+1}$, $i = \overline{1,k}$, где τ_i^n есть терм, полученный из терма τ_i путем приписывания всем входящим в него константам и переменным актам в качестве верхнего индекса натурального числа n. Определим последовательность $X_i^1 = X_i^0 \cdot \vartheta_i^1, X_i^{n+1} = [\vartheta_i^{n+1} / X_j^n | i = \overline{1,k}] X_j^n$ для $n = 1,2,...$

Например, для $X_1 = (p \to a) \cdot X_1$ имеем последовательность $X_1 = X_1^0 \cdot (p^1 \to a^1) \cdot X_1^1, X_1^2 = X_1^0 \cdot (p^1 \to a^1) \cdot X_1^1 \cdot (p^2 \to a^2) \cdot X_1^2, \ldots, X_1^{n+1} = X_1^0 \cdot (p^1 \to a^1) \cdot X_1^1 \cdot (p^2 \to a^2) \cdot X_1^2 \cdot \ldots \cdot (p^n \to a^n) \cdot X_1^{n+1}$ для $n = 1,2,...$ Верхние индексы для актов введены с целью явного указания порядка порождения одноименных актов при выполнении процесса.

Второй вариант определяется следующей последовательностью равенств: $X_i^0 = S_i, X_i^1 = S_i \cdot \tau_i, X_j^2 = \left[X_j^1 / X_j \middle| j = \overline{1,k} \right](S_i \cdot \tau_i), \ldots, X_j^{n+1} = \left[X_j^n / X_j \middle| j = \overline{1,k} \right](S_i \cdot \tau_i)$, где $S_i, i = \overline{1,k}$, – специальные символы, отличные от актов множества A.

Для примера выше имеем следующую последовательность равенств: $X_1^0 = S_1, X_1^1 = S_1 \cdot (p \to a) \cdot X_1, X_1^2 = S_1 \cdot (p \to a) \cdot S_1 \cdot (p \to a) \cdot X_1, \ldots, X_i^{n+1} = S_1 \cdot (p \to a) \cdot S_1 \cdot (p \to a) \cdot X_1 \cdot \ldots \cdot S_1 \cdot (p \to a) \cdot X_1$.

На рис. 3 приведен пример построения всех трасс выполнения процесса для приближения, $X_1^1 = X_1^0 \cdot (p^1 \to a^1) \cdot X_1^1$, где указатель end в состоянии процесса означает завершение трассы.

Обозначим через $PR(X_i^n)$ множество всех трасс выполнения процесса для X_i^n, $i = \overline{1,k}$. Это множество в графическом смысле представляет собой дерево с корневой вершиной X_i^0. Введем отношение строгого частичного порядка $\subseteq$ на последовательности множества трасс $PR(X_i^n)$ для $n = 0,1,2,\ldots$

Определение 7. Обозначим $PR(X_i^n) \subseteq PR(X_i^{n+1})$, если каждая трасса $PR(X_i^{n+1})$ имеет в качестве начального отрезка одну и только одну трассу $PR(X_i^n)$.

Справедливо следующее утверждение.

Утверждение. Последовательность $PR(X_i^n)$ для $n = 0,1,2,\ldots$ образует цепь с минимальным элементом $PR(X_i^0), i = \overline{0,k}$.

Обозначим $\widetilde{X}_i = \lim\limits_{n \to \infty} \left[X_j^n / X_j \mid j = \overline{1,k} \right] \tau_i, i = \overline{0,k}$.

Следствие. Предельные значения $\widetilde{X}_i$ являются решением системы уравнений $X_i = \tau_i, i = \overline{1,k}$.

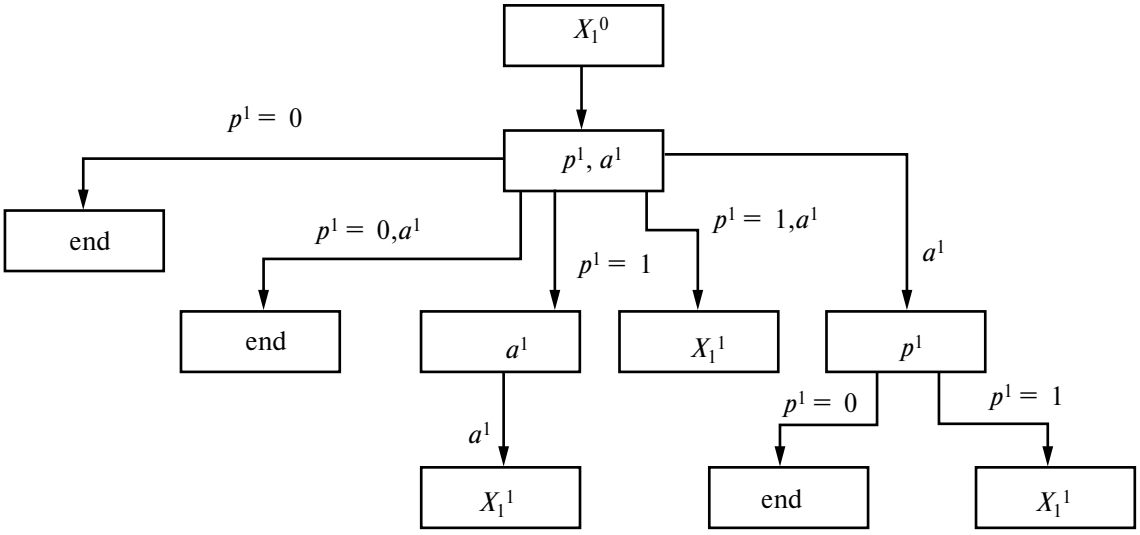

Рис. 3. Все трассы выполнения процесса $X_1^0 \cdot \left(p^1 \to a^1\right) \cdot X_1^1$.

Практически $\widetilde{X_i}$ содержит все возможные трассы абсолютно параллельного выполнения процесса для $X_i, i = \overline{1,k}$.

Процессы, представляемые в языке, относятся к свободным процессам. Язык достаточно просто расширить на случай описания процессов, зависимых от реального времени. Для этого достаточно ввести в него приведенные в разд. 3 спусковые функции, определяющие условия инициирования каждого акта процесса с учетом реального времени, и функции воздействия акта на другие акты процесса после своего завершения.

Заключение. Отметим основные отличия моделей дискретных параллельных процессов, предложенных в известных работах [5, 6], от модели и языка параллельных процессов, представленных в статье. В [5, 6] реальное время в процессах не рассматривается, а параллелизм в процессах трактуется как возможность чередования независимых актов процесса при их выполнении. В статье явно используются не только отношения “раньше” и “позже” между выполняемыми актами процесса, но также отношение одновременности между моментами начала и завершения актов. Это является существенным расширением процессной модели, а также необходимым условием для реального оценивания сложности параллельных процессов [16].

При поиске сигнатуры операций композиции процессов в работе [6] основной акцент сделан на выборе минимальных средств для описания параллелизма и взаимодействия параллельных процессов. По сути, к традиционным двум операциям последовательной композиции и весьма упрощенного выбора в модели процессов [6] добавлены операция композиции параллельных процессов и оператор взаимодействия двух параллельных процессов. Как уже было отмечено, сигнатура операций композиции в модели параллельных процессов, предложенная в статье, в большой степени была продиктована поиском универсальных средств описания одновременности и асинхронности алгоритмических процессов вычисления значений функций и процессов доказательства формул в работах [7, 8, 11, 12]. Можно показать, что язык параллельных процессов существенно расширяет возможности описания различных форм параллелизма, в частности параллелизма алгоритмических процессов, по сравнению с моделями параллельных процессов в статьях [5, 6].

Кроме проблемы оценивания сложности параллельных процессов и приведения ее к оптимальным значениям путем эквивалентных преобразований [16], за пределами статьи осталась еще одна не менее важная с практической точки зрения проблема, касающаяся формализации доказательства свойств процессов и отношений между ними (проблемы завершения процессов, эквивалентности и т.д.). Основополагающие работы по этой проблеме достаточно полно отражены в разд. 1, 2 статьи. Результаты построения различных логических теорий для этих целей, предложенные в работах [17–26], могут естественным образом использоваться для описанного языка процессов. Сопоставление каждому акту трассы процесса логических функций, определяющих временные условия его инициирования и завершения, являются

необходимыми условиями применения этих моделей для формального доказательства свойств и отношений процесса.

ПРИЛОЖЕНИЕ

Прикладная логика дискретного времени. С практической точки зрения время можно рассматривать как отображение последовательности событий на строго упорядоченное подмножество вещественных чисел, значениями которых с требуемой точностью измеряется определенное отношением “после” время между событиями в наблюдаемой последовательности. Логические утверждения, сопровождающие самостоятельные участки программы (последовательности команд, операторы и др.), обычно используются для доказательства правильности программ [17–19]. При этом реальное время в этих работах выражается косвенно в виде отношения S_{i+1} “после” S_i, $i = 0,1,...,$ заданного для наблюдаемой последовательности состояний процесса.

Отображение отношения “после” на определенную модель времени $MT = (T,\rightarrow)$, где T – множество временных отсчетов, $\rightarrow$ – рефлексивное строгого порядка отношение на T, можно рассматривать как общий подход к формализации понятия времени. Модели (N, <), (W, <), (R, <), где N, W, R – множества натуральных, рациональных и действительных чисел с соответствующими им отношениями порядка – известные примеры модельного представления времени. Модель (N, <) представляет формализацию дискретного времени, а модель (R, <) – непрерывного времени.

В работе [20] с обширной библиографией содержится подробный анализ различных точек зрения на сущность времени и подходов к построению его моделей. Там же можно найти описание различных вариантов линейной, ветвящейся точечных логик и интервальной логики [21]. В работе [22] приведен обзор языков логического программирования с учетом фактора времени, в [23, 24] предложены методы доказательства свойств бесконечных дискретных процессов.

Если попытаться найти общую логическую базу перечисленных работ, то их можно рассматривать как специальные расширения логики первого или более высоких порядков с использованием в формулах временного параметра и дополнительной аксиоматики – определенным множеством формул (так называемых модальностей), которые характеризуют рассматриваемые временные отношения [25, 26].

Один из вариантов прикладной временной логики с ориентацией на практическое применение для конструктивных процессов может быть построен как специализация логики первого порядка с использованием в ее формулах в качестве специального параметра параметр времени вместе с его аксиоматикой и правилами доказательства (вывода).

Подобную модель MT можно рассматривать как совокупность определяющих ее элементов: $MT = (T, \Delta t, \leq, AT, RT)$, где T – счетное упорядоченное отношением $\leq$ числовое множество, элементы которого могут быть получены суммированием определенного временного кванта Δt, характеризующего точность измеряемого реально времени, AT – арифметические операции: $+, -, \times, /$ на T, RT – операции отношения на T: $=, \neq, \geq, \leq, >, <$. Аксиоматика арифметических операций и отношений детально изучена в работе [27].

Следующие аксиомы являются наиболее важными с точки зрения формализации модели дискретного времени: $\forall\, t\, \forall t'(t < t' \vee t' < t \vee t' = t)$ – сравнимость t и t', $\forall t \forall t' \forall t''(t < t' \wedge t' < t'' \supset t < t'')$ – транзитивность $<$, $\forall t \forall t'(t < t' \wedge t' < t \supset t = t')$ – антисимметричность $<$, $\forall t \exists t'(t < t' \supset \exists t''(t < t'' \wedge \forall t'''(t < t''' \supset t'' \leq t''')))$ – дискретность T, $\forall t \exists t'(t < t' \wedge \forall t''(t < t'' \supset t' \leq t''))$ – каждый момент времени имеет непосредственно следующий за ним момент. Очевидно, что точечная модель дискретного времени эквивалентна модели, в которой в качестве множества T используется изоморфное T множество $NU\{0\}$, где N – натуральный ряд чисел.

Если говорить о практическом применении временной логики для доказательства свойств и отношений процессов для описанного в статье языка дискретных параллельных процессов, то содержание разд. 1–3 создает для этого необходимые условия. Это касается конструктивного определения множества всех трасс абсолютно параллельного выполнения процесса, сопоставления актам временных логических формул, предназначенных для определения условий инициирования актов, и функций, задающих условия и направления передачи, инициирующих другие акты сигналов.

СПИСОК ЛИТЕРАТУРЫ

1. *Agha G., Mason I., Smith S. Talcott C.* Towards a Theory of Actor Computations// Third International Conf. on Concurrency Theory (CONCUR ‘92). Springer-Verlag, 1992. P. 565–579.

2. *Вальковский В.А., Котов В.Е., Марчук А.Г., Миренков Н.Н.* Элементы параллельного программирования // Радио и связь. 1983. 240 с.
3. *Питерсон Дж.* Теория сетей Петри и моделирование систем. М.: Мир, 1984. 264 с.
4. *Keller R.M.* Parallel Program Schemata and Maximal Parallelism // J. ACM. 1973. V.20. № 3. P. 514–537.
5. *Milner R. A* Calculus of Communicating Systems // Lecture Notes in Computer Science. Springer-Verlag, 1980. P. 184.
6. *Хоар Ч.* Взаимодействующие параллельные процессы. М.: Мир, 1992. 184 с.
7. *Кутепов В.П., Фальк В.Н.* Функциональные системы: теоретический и практический аспекты // Кибернетика. 1979. № 1. С. 45–58.
8. *Кутепов В.П.* Модели и языки для описания параллельных процессов // Изв. РАН. ТиСУ. 2018. № 3. С. 116–127.
9. *Хьюз К., Хьюз Т.* Параллельное и распределенное программирование с использованием C++. М.: Вильямс, 2004. 667 с.
10. *Cesarini F., Thompson S.* ERLANG Programming: a Concurrent Approach to Software Development. Sebastopol, California, O'Reilly Media, 2009. 496 p.
11. *Кутепов В.П., Шамаль П.Н.* Реализация языка функционального параллельного программирования FPTL на многоядерных компьютерах // Изв. РАН. ТиСУ. № 3. С. 46–60.
12. *Кутепов В.П., Зубов М.И.* Реализация и экспериментальное исследование эффективности упреждающего параллелизма // Вестн. МЭИ. 2019. № 4. С. 119–126.
13. *Кутепов В.П., Ефанов А.А.* Параллельные процессы и программы: модели, языки, реализация на системах // Программные продукты и системы. 2020. Т. 33. № 3. С. 375–384.
14. *Кутепов В.П.* Интеллектуальное управление процессами и загруженностью в вычислительных системах // Изв. РАН. ТиСУ. 2007. № 5. С. 58–73.
15. *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. 135–146.
16. *Кутепов В.П., Фальк В.Н.* Алгоритмические параллельные процессы и их сложность // Вестн. МЭИ. 2020. № 3. С. 102–110.
17. *Apt K.* Formal Justification of a Proof System for Communicating Sequential Process // J. ACM. 1983. V. 30. № 1. P. 197–216.
18. *Pnueli A.* Temporal Logic of Programs // Proc. 18-th IEEE Sympos. of Foundation of Computer Science. Tel Aviv, Israel, 1977. P. 46–57.
19. *Lamport L.* The Temporal Logic of Actions // ACM Transactions on Programming Languages. 1993. V. 7. № 3. P. 1–52.
20. Stanford Encyclopedia of Philosophy. Temporal Logic. 2019. P. 1–17. https://plato.stanford.edu/entries/logic-temporal/
21. *Allen J.F.* Maintaining Knowledge About Temporal Intervals // Communications of ACM. 2010. V. 26. № 11. P. 832–843.
22. *Orgun M., Wadge W.* Theory and Practice of Temporal Logic Programming // J. Logic Programming. 1992. V. 13. P. 413–440.
23. *Buchi J.R.* On a Decision Method in Restricted Second Order Arithmetic // Proc. Intern. Congr. Logic, Method and Philos., Sci. Stanford University, 1960. P. 1–12.
24. *Vardi M.Y., Wolper P.* Reasoning About Infinite Computations // Information and Computation. 1994. V. 115. № 1. P. 1–37.
25. *Finger M., Gabbay Dov.* Adding a Temporal Dimension to a Logic System // J. Logic, Language and Information. 1992. V. 1. P. 203–233.
26. *Gabbay P., Pnueli A., Shelah S., Stavi J.* On the Temporal Analysis of Fairness // The ACM Sympos. on Principles of Programming Languages. Las Vegas, 1980. P. 163–173.
27. *Клини С.* Введение в метаматематику. М.: Из-во иностр. лит., 1957. 520 с.