УДК 519716.35
DOI: 10.15827/0236-235X.131.375-384
Дата подачи статьи: 10.06.20
2020. Т. 33. № 3. С. 375–384

Параллельные процессы и программы: модели, языки, реализация на системах

В.П. Кутепов [1], *д.т.н., профессор, kutepov@appmat.ru*
А.А. Ефанов [1], *аспирант, 1134togo@gmail.com*

[1] *Национальный исследовательский университет «МЭИ», кафедра прикладной математики, г. Москва, 111250, Россия*

Предлагаются модели и языки параллельных процессов и программ, имеющие более широкие возможности для описания параллелизма по сравнению с известными моделями и языками, созданными для этой цели.

Понятие актора, выполнение актов и взаимодействие акторов при выполнении процесса – базовые элементы описания процессов. В качестве общей формы задания процессов используются системы рекурсивных уравнений, правыми частями которых являются группы взаимодействующих при выполнении процесса акторов. Графическая и текстовая формы описания процессов и принцип инициализации выполнения акта по готовности поступающих на входы соответствующего ему актора сигналов (данных) от влияющих на него акторов создает необходимые условия для эффективного описания процессов, их анализа, модификации и параллельного выполнения. Интерпретация акторов процесса как процедур обеспечивает прямой переход от языка процессов к высокоуровневому языку модульного потокового параллельного программирования.

Рассмотрены главные проблемы реализации предложенных языков на компьютерных системах: однозначное именование порождаемых при взаимодействии процессов актов, организация управления процессами на компьютерных системах.

В статье рассматриваются более общие (в отличие от классических) модель и язык процессов. Они позволяют описывать группы взаимодействующих процессов при задании общего процесса в виде системы процессных уравнений. Динамически порождаемое структурное описание причинно-следственных зависимостей между акторами процесса в группах и использование механизма активизации актора при поступлении необходимых сигналов (данных) на соответствующие его входы обеспечивают без ограничений реализацию параллелизма и асинхронности при выполнении актов процесса. Использование рекурсивных определений переменных в системе уравнений, описывающей процесс, создает условия для динамического порождения групп процессов.

На основе описанной в статье модели процессов создан язык параллельного программирования, реализуемый на компьютерных системах.

Ключевые слова: *параллельные процессы, параллельное программирование, компьютерные системы, реализация параллельных процессов, модель акторов.*

При организации выполнения работ, программ и прочего специалисты имеют дело с процессами, главными критериями оптимальности которых являются время и используемые ресурсы. Совмещение во времени и одновременность (параллелизм) выполнения неделимых действий процесса или его актов, асинхронность, то есть способность реагировать на произвольную длительность актов, – основные факторы, от которых зависит оптимальность реализации процессов [1, 2].

Параллельное программирование является хорошим примером, отражающим различные этапы развития средств описания параллелизма и его реализации на компьютерных системах. От простых языковых конструкций явного указания на то, что в программе может или должно выполняться одновременно (par begin … end, fork ... join), сегодня созданы разнообразные языки и среды, предназначенные для описания и реализации параллелизма.

Примерами широко применяемых языков и сред параллельного программирования являются Multithreading, PVM, MPI, Erlang [3, 4]. В основу их создания положено понятие процесса, базовыми конструкциями которого являются акторы или исполнители конкретных действий процесса и способы их композиции и взаимодействия при описании и выполнении процесса. MPI построен на простой формальной модели, в которой конечное множество акторов или процессов выполняют независимые фрагменты программы и взаимодействуют друг с другом путем передачи сообщений.

В отличие от MPI в PVM есть средства группирования процессов, а в процессной модели Erlang допускается использование рекурсивно порождаемых процессов.

В данной статье предложены акторная модель [5], язык для описания параллельных процессов и созданный на их основе высокоуровневый язык модульного потокового параллельного программирования. Главное отличие модели параллельных процессов, описанной в статье, состоит в том, что в ней используются графическая и эквивалентная текстовая версии описания параллельных процессов.

Общая форма задания процесса в виде системы в общем случае рекурсивных уравнений вместе с графическим представлением процесса обеспечивают условия для наглядного отображения декомпозиционной структуры процесса, упрощают его проектирование, анализ, модификацию. Явно указываемые в описании процесса структурированные связи между его акторами позволяют использовать известный принцип инициализации выполнения акта по готовности сигналов (данных), поступающих от других актов, и реализовать присущий процессу параллелизм без задержек.

В то же время интерпретация акторов процесса как процедур, для описания которых могут использоваться известные языки программирования, превращает язык процессов в язык параллельного программирования, сохраняя прямую связь между их процессной и интерпретационной семантиками [6].

С точки зрения теории главным вопросом является универсальность модели процессов, что определяется способностью ее средствами описывать различные формы параллелизма [7]. Алгебраическая модель параллельных процессов, описанная в статье [8] и исследованная на предмет выразительной силы представлять ее средствами различные формы параллелизма, расширяет эти способности по сравнению с широко известными алгебраическими моделями параллельных процессов [9–12].

Однако алгебраические модели имеют свои ограничения по сравнению с предложенными в статье моделями и языками параллельных процессов.

Модель и язык параллельных процессов

Процесс – это наблюдаемое во времени частично упорядоченное следование актов (событий, состояний), подчиненное определенным правилам.

В формальном смысле модель процессов представляет собой триаду $<ACT, L, R>$, где ACT – счетное множество акторов, из которых создается процесс; L – язык описания процесса; R – правила выполнения процесса.

Акторы являются исполнителями актов (далее авторы часто используют понятие акта в обоих смыслах, которые следуют из контекста), представляющими неделимые действия процесса и обладающими следующими основными характеристиками:

– способность акта выполнять конкретное действие при воздействии на него других актов процесса путем посылки соответствующих сигналов (в более широком контексте – данных);

– возможность инициализации выполнения акта не только в результате получения сигналов от других актов, но и между моментами поступления сигналов и их значений (конкретных данных);

– способность акта воздействовать на другие акты (на соответствующие им акторы) с целью инициализации их выполнения;

– мобильность акта, предполагающая возможность его выполнения различными исполнителями при реализации процесса на системе.

Процесс должен обладать способностью группирования выполняемых им актов как необходимым условием эффективной организации коллективного выполнения работ.

Циклы и возможность рекурсивного порождения процессов – неотъемлемая характеристика языка процессов, определяющая способность динамического порождения процессов и их сложность.

Типичная и широко применяемая форма конвейерной обработки информации и выполнения потоков работ – еще одна важная для практики особенность процессов.

Структурное задание процесса. В определении языка процессов использовано представление актора, которое отражает его наиболее общие структурные и поведенческие особенности и обеспечивает простой способ перехода от описания процесса к описанию строящейся на его основе параллельной программы и наоборот.

Определение. Актор $a \in ACT$ определяется следующими параметрами:

– пара натуральных чисел (m, n), где m и n – количество групп входов и выходов актора;

– упорядоченные множества входов и выходов (кортежи), сопоставляемые каждой группе входов и выходов;

– множество процедур, однозначно сопоставляемых каждой группе входов, с количеством параметров, равным количеству входов группы.

На рисунке 1 графически представлен актор с m группами входов, на которые поступают сигналы от выполняемых актов процесса, и n группами выходов, на одну из которых процедура, соответствующая группе входов процесса, посылает при своем выполнении воздействующие на инициализацию других актов процесса сигналы (данные).

Входы каждой группы входов актора разделяются на три типа в соответствии с той ролью, которую выполняют поступающие на них данные при инициализации и выполнении в общем случае потока тегированных данных генерируемыми копиями процедуры соответствующей группы входов актора.

Знаками □, ○ и × обозначены входы, на которые поступают данные трех типов: управляющие, рабочие и интерфейсные. По управляющим логическим данным определяется необходимость активизации соответствующей процедуры актора, данные на рабочих и интерфейсных входах используются в качестве фактических параметров активизируемой процедуры. При этом по интерфейсным данным копия процедуры при ее инициализации получает необходимую информацию о том, каким акторам процесса должны быть переданы вычисленные ею данные.

Перейдем к формальному определению процессов, язык которых задан в виде триады $<ACT, L, R>$. Разделим множество акторов ACT на два подмножества: $ACT = ACT' \cup ACT''$, где ACT' – переменные акторы, обозначаемые X, X_i, $i = 1, 2, \ldots, n$, и ACT'' – константные акторы, обозначаемые a, a_i, $i = 1, 2, \ldots, n$.

Определение. Структурное описание процесса на подмножестве акторов $ACT''' \subset ACT$ задается в виде пары $<EQ, CON>$, где EQ – система процессных уравнений $X_i = \tau_i$, $i = 1, 2, \ldots, n$, τ_i – конечное подмножество индексированных акторов из ACT'''; $CON = <CON_1, CON_2, \ldots, CON_N>$, CON_i: $OUTPTS_i \times INPTS_i \rightarrow \rightarrow \{0, 1\}$ – функция, которая задает связи между выходами и входами всех выходов и входов акторов в τ_i (0 означает отсутствие связи, 1 – ее наличие).

Индексация акторов, входящих в правые части τ_i в системе процессных уравнений, необходима для однозначного именования порождаемых ими актов при выполнении процесса и осуществляется путем приписывания однозначного индекса сверху для каждого вхождения актора во все правые части τ_i системы процессных уравнений.

Предполагается выполнение замкнутости для термов τ_i в системе процессных уравнений, означающее, что в них могут входить только индексированные переменные из множества $\{X_i \mid i = 1, 2, \ldots, n\}$.

Задание интерпретации процесса. Рассмотрим задание интерпретации процессов применительно к процедурной форме описания параллельных программ [6].

Интерпретация есть функция, сопоставляющая каждой группе входов каждого актора процесса процедуру на некотором языке программирования, параметрами которой являются тег из натурального ряда чисел, рабочие и интерфейсные параметры. Тег в реализации процесса сопровождает данные, пересылающие закончившие выполнение инициированные копии процедуры акторов другим акторам процесса, обеспечивая таким образом однозначность при обработке процессом потоков поступающей извне и генерируемой самим процессом информации.

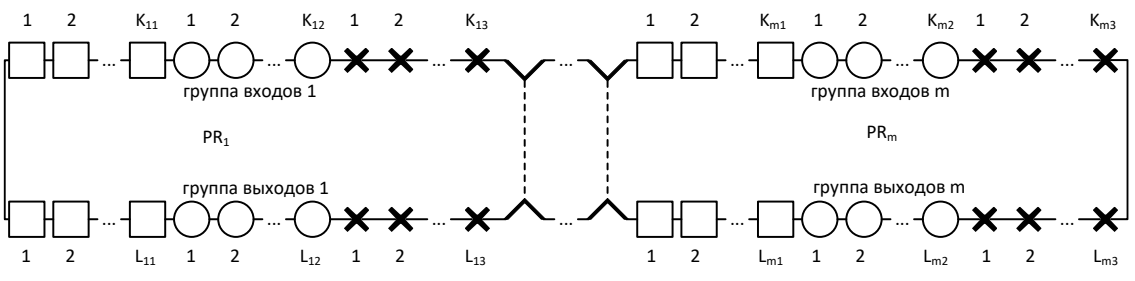

Рис. 1. Графическое представление актора

Fig. 1. The actor graphic representation

Структурное описание процесса вместе с заданием интерпретации обеспечивают простой переход от описания процесса к параллельной программе и наоборот [6].

Грамматика языка процессов и параллельных программ приведена рисунке 2.

Рассмотрим следующий пример описания процесса параллельного вычисления значений функции $F_1(x)$, заданной в виде двух рекурсивных уравнений:

$F_1(x) = if\ P_1(x)\ then\ f_1(x)\ else\ f_2(F_1(f_1(x)), F_2(f_3(x)))$;

$F_2(x) = if\ P_2(x)\ then\ g_1(x)\ else\ F_2(g_2(x))$.

На рисунке 3 приведены два варианта графического представления структурного описания процесса, описывающего параллельные вычисления значений функции $F_1(x)$. Текстовое описание процесса легко воспроизвести в соответствии с грамматикой языка процессов (рис. 2).

Заметим, что каждое повторное вхождение актора в описание процесса сопровождается приписыванием ему верхнего индекса (номера), таким образом обеспечивая однозначную идентификацию эквивалентных акторов, взаимодействие с другими акторами которых в процессе различно. Идентификаторы всех переменных акторов в левой части уравнений по умолчанию в качестве верхнего индекса имеют значение 0.

В структурном представлении процесса также участвуют акторы $F_1^{(0)}$ и $F_2^{(0)}$, что связано с правилами параллельного выполнения процесса. Эти акторы имеют две группы входов.

Имена акторов в блоках можно рассматривать как ссылки на соответствующие процедуры после приписывания к ним номера соответствующей группы входов актора.

‘PROCESS’ <имя процесса> <описание процесса> ‘END PROCESS’

<описание процесса> ::= ‘EQUATIONS’ <описание системы уравнений> ‘END EQUATIONS’ ‘;’ ‘CONNECTIONS’ <описание связей выход-вход актов процесса> ‘END CONNECTION’

<описание системы уравнений> ::= <описание уравнения> ‘,’ <описание системы уравнений>

<описание уравнения> ::= <имя переменного акта> ‘(’ <имя процедуры> ‘)’ ‘=’ <список актов в правой части уравнения>

<описание связей вход-выход актов процесса> ::= <описание связей вход-выход уравнения> ‘;’ <описание связей вход-выход актов процесса>

<описание связей выход-вход актов уравнения> ::= ‘VAR’ <имя переменного акта в левой части уравнения> ‘:’ <описание связей выход-вход актов в правой части уравнения>

<описание связей выход-вход актов в правой части уравнения> ::= <описание связей выход-вход акта> ‘;’ <описание связей выход-вход актов в правой части уравнения>

<описание связей выход-вход акта> ::= ‘OUT’ ‘(’ <имя акта> ’)’ ‘GROUP’ ‘(’ <номер группы выходов акта-отправителя данных> ‘,’ <количество выходов> ‘)’ ‘TO WHOM’ <описание входных связей актов-приемников данных> ‘END OUT’

<описание входных связей актов-приемников даннах> ::= <описание входных связей акта-приемника данных> ‘;’ <описание входных связей актов-приемников данных>

<описание входных связей акта-приемника данных> ::= <имя акта-приемника данных> ‘,’ <описание входных связей групп входов акта-приемника данных>

<описание входных связей групп входов акта-приемника данных> ::= ‘GROUP’ ‘(’ <номер группы входов> ‘)’ ‘:’ <описание входных связей группы входов акта-приемника данных>

<описание входных связей группы входов акта-приемника данных> ::= <описание связи входа группы входов акта-приемника данных> ‘;’ <описание входных связей группы входов акта-приемника данных>

<описание связи входа группы входов акта-приемника данных> ::= ‘(’ <номер входа в группе входов акта-приемника данных> ‘,’ <номер выхода в группе выходов акта-отправителя данных> ‘,’ <тип передаваемых данных> ‘,’ <тип входа> ‘)

Примечания

<тип передаваемых данных> – любой тип данных используемого языка программирования процедур актов;

<тип входа> – принимает значения 0 или 1 и служит для указания, должна (значение 0) процедура применяться к поступившим тегированным данным или (значение 1) не должна;

все определения, которые содержат слово «имя», представляются в виде идентификатора языка;

определения, содержащие слово «номер», представляются натуральными числами.

Рис. 2. Грамматика языка процессов

Fig. 2. The grammar of the process language

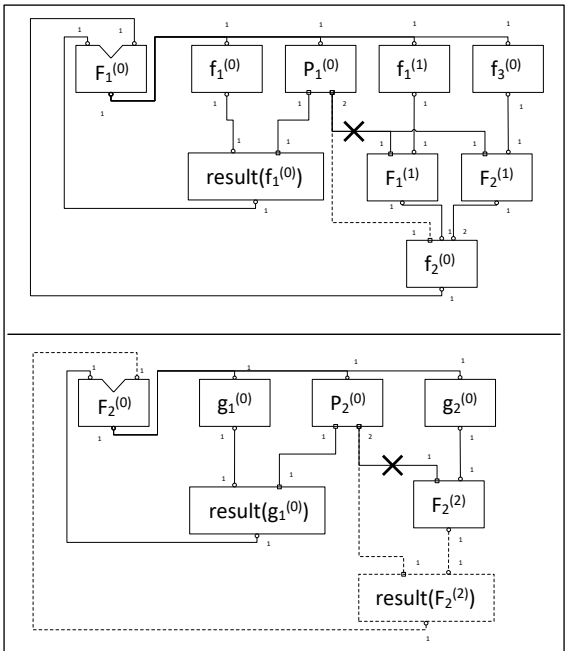

Рис. 3. Структурное описание процесса

Fig. 3. Process structural description

Процесс описан таким образом, чтобы увеличить степень параллелизма в нем [7] за счет вычислений с упреждением значений функций в блоках $f_1^{(0)}$, $f_2^{(0)}$, $g_1^{(0)}$, $g_2^{(0)}$. Пунктирными линиями на схемах обозначен второй вариант описания параллельного процесса, в котором с упреждением опережающим образом осуществляется инициализация переменных акторов, обеспечивая таким образом абсолютное распараллеливание при вычислении значений функции $F_1(x)$. Убираемые связи помечены крестиком. Однако при этом возникает нетривиальная проблема прерывания динамически порождаемых актов (инициализируемых процедур акторов) процесса при рекурсивных вызовах (инициализации копий процедур) переменных акторов и инициализации копий процедур константных актов при обработке потоковой информации.

Правила выполнения процесса

Процесс в общем случае оперирует с потоком тегированных данных, поступающих на входы акторов процесса. Условием инициализации акта является поступление данных с одним и тем же тегом на все рабочие и интерфейсные входы некоторой группы входов актора при дополнительном условии, что на все управляющие входы группы поступили логические значения *true* с тем же тегом. Если хотя бы одно из значений на управляющем входе группы есть *false*, поступившие тегированные данные аннулируются. Инициализация акта константного актора означает вызов для выполнения копии процедуры, прописанной для данной группы входов актора в задании процесса. После завершения процедуры результаты ее выполнения должны быть переданы на входы акторов, связи с которыми указаны в описании процесса для конкретной группы выходов актора, инициализировавшего выполнение процедуры. Это может быть реализовано путем присвоения соответствующей адресной информации в виде значений интерфейсных параметров процедуры.

Например, после завершения процедуры актора $P_1^{(0)}$ со значением *true* на управляющий вход номер 1 акта result($f_1^{(0)}$) должно быть передано значение *true*, а на вход 1 актора $F_1^{(1)}$ и на вход 1 актора $F_2^{(1)}$ – значение *false*.

Рассмотрим правила выполнения процессов, в которых учитывается возможность их реализации на реальных компьютерных системах с их ограничениями на пропускную способность каналов передачи данных.

Каждый переменный актор при инициализации его управляющей процедуры выполняет все необходимые функции по управлению выполнением актов, порождаемых акторами в правой части соответствующего ему уравнения в задании процесса. Эти функции включают следующие действия:

– прием и формирование очередей асинхронно поступающих тегированных данных на входы акторов в правой части переменного актора;

– определение готовности актора для выполнения после поступления на все входы одной из его групп входов данных с одним и тем же тегом;

– аннулирование поступивших данных на входы групп входов акта, если хотя бы на один его управляющий вход поступило значение *false*;

– назначение копии процедуры актора для выполнения, что в реализации процесса на конкретной системе означает определение конкретного менее загруженного узла системы, на котором должна выполняться процедура актора;

– контроль окончания выполнения процедуры актора, который завершается передачей полученных результатов (тегированных данных) на соответствующую группу входов

управляющего актора, инициировавшего выполнение процедуры.

Поскольку переменный актор может быть определен в задании процесса рекурсивно и инициализирован многократно, возникает необходимость однозначного именования таким образом динамически порождаемых им актов (инициализируемых копий процедур).

Для последовательных программ рекурсивные вызовы процедур реализуются посредством известного механизма линейного упорядочения контекстов (состояний выполнения программы) и сохранения их в стековой памяти. При параллельном выполнении процесса его состояние представляет собой древовидную структуру [8], на разных ветвях которой выполняются управляющие процедуры инициализированных переменных акторов и процедур акторов, содержащихся в правой части соответствующих им уравнений в задании процесса.

Поэтому при инициализации переменного актора $X_i^{(k)}$ ему присваивается уникальный номер, а уникальным идентификатором порожденного акта становится триада <X_i, k, номер акта>. Далее процедура инициализированного переменного актора выполняет все функции управления актами, инициируемыми акторами в правой части уравнения для $X_i^{(0)}$ при обработке потока тегированных данных. В свою очередь, константным акторам в правой части уравнения приписывается тот же номер, что и переменному актору. Это позволяет однозначно идентифицировать порожденную переменным актором и подчиненную ему группу акторов (константных и переменных акторов для $X_i^{(0)}$).

Так как константные акторы могут быть готовы для выполнения многократно, при инициализации их процедур для конкретных тегированных данных поступающего на входы актора потока данных им присваивается дополнительный уникальный номер. Это позволяет однозначно идентифицировать их при выполнении на соответствующих узлах системы. Таким образом, идентификатором инициализируемого константного актора актором $a_i^{(n)}$ становится четверка: <a_i, n, идентификатор переменного акта, уникальный номер акта $a_i^{(n)}$>.

Инициализированный константным актором акт далее выполняет соответствующую поступившим данным процедуру, зависящую от группы входов, на которые поступили данные с определенным тегом. Получая в качестве интерфейсного параметра идентификатор переменного актора, инициированная процедура константного актора пересылает вычисленные результаты управляющей процедуре переменного актора.

Данная логика присваивания идентификаторов актам, с одной стороны, обеспечивает однозначность их именования, а с другой – позволяет сохранить обратную связь с акторами-прародителями, что необходимо для «возвращения» им результатов выполнения актов.

Не нарушая общности, можно считать, что выполнение процесса начинается с процедуры управляющего актора $X_1^{(0)}$, на одну из групп входов которого передаются определенные тегированные данные для обработки. Эти данные могут поступать в виде потока динамически, например, путем считывания данных с определенного носителя или канала.

С теоретической точки зрения каждое выполнение процесса характеризуется траекторией, отражающей временное следование актов, описанных генерируемыми правилами. Понятие траектории процесса является основополагающим при исследовании сложности выполнения процессов [12], их формализации средствами логики [13], построения алгоритмических моделей процессов реального времени [14–16], интерпретации их как процессов, реализующих вычисления [17, 18].

Реализация процессов на системах

Хотя описанные модель и язык параллельных процессов имеют общий характер, рассмотрим их реализацию применительно к параллельному программированию и реализации параллельных программ на компьютерных системах [6, 19]. Вначале затронем аспект программирования процесса.

Процесс как программный продукт. Прежде всего должна быть обеспечена возможность максимальной инвариантности процедур константных и переменных акторов, позволяющая применять модульную сборочную технологию процесса из уже созданных программных продуктов. Для константных и переменных акторов в качестве их программных объектов естественным образом может использоваться процедурная форма представления со свободным выбором языка программирования [6]. Тот факт, что у актора в общем случае может быть более одной процедуры, которая «привязывается» к определенной группе входов, не усложняет проектирование процесса.

Управляющие процедуры переменных акторов выполняют указанные ранее общие функции и могут быть программно реализованы как стандартный программный продукт (например, в виде класса в терминологии языка C++).

Конкретизация функций этого программного объекта в процессе определяется сопоставляемым ему описанием акторов в правой части соответствующего уравнения для переменного актора и связей между ними.

Системный аспект реализации процесса. Будем предполагать, что для выполнения процесса используется компьютерная система или сеть с операционными средствами организации передачи данных между узлами при выполнении процедур акторов на соответствующих узлах. Последнее означает, что операционная система такого кластера способна тем или иным способом контролировать загруженность узлов и в соответствии с этим назначать порождаемые акторами процедуры актов на наименее загруженные узлы при выполнении процесса.

Естественно, что на каждом узле управляет процессами его операционная система.

В соответствии с описанными правилами выполнения процесса при инициализации переменного актора $X_i^{(k)}$, $i = 1, 2, \ldots, n$, управление группой акторов, содержащихся в правой части уравнения $X_i^{(0)}$, осуществляется процедурой $X_i^{(0)}$. Эту процедуру следует рассматривать как самостоятельный системный процесс, аналогичный процессу выполнения программы на узле.

На рисунке 4 приведены структура, блоки и информация, показывающие организацию управляющей процедуры $X_i^{(k)}$, $i = 1, 2, \ldots, n$.

В таблице состояний $n_г$ – двоичный признак готовности процедуры актора для выполнения. При инициализации акта по готовности вызываемая для его выполнения процедура с соответствующим списком поступивших данных (фактических параметров процедуры) назначается на выполнение, пройдя перед этим этап планирования.

Входные данные процедуры константного акта, которые выполняют роль условий необходимости применения к ним процедуры, вынесены в отдельную таблицу, содержащую значения входов-условий.

Отметим, чем надо руководствоваться при планировании процессов-процедур, порождаемых при выполнении процесса на системе. Ясно, что управляющая процедура переменного актора выполняет более сложную коллективную работу по сравнению с процедурой константного акта. Учитывая ограничения систем по пропускной способности их каналов, становится оправданным присвоение более высоких приоритетов переменным акторам при планировании. На следующем, более низком уровне предпочтение при планировании должны получать процедуры актов, результат выполнения которых непременно будет использован. Например, для процесса на рисунке 3 предпочтение при планировании должно быть отдано акту $P_1^{(0)}$ для $F_1^{(0)}$ и акту $P_2^{(0)}$ для $F_2^{(0)}$ по сравнению с другими актами, которые могут выполняться одновременно с ними.

Что касается выбора узла для выполнения процедур актов процесса, то это должен быть наименее загруженный узел, к которому существует наиболее быстрая доставка необходимых данных. Алгоритмы, предназначенные для этой цели и используемые в кластерах, скрыты,

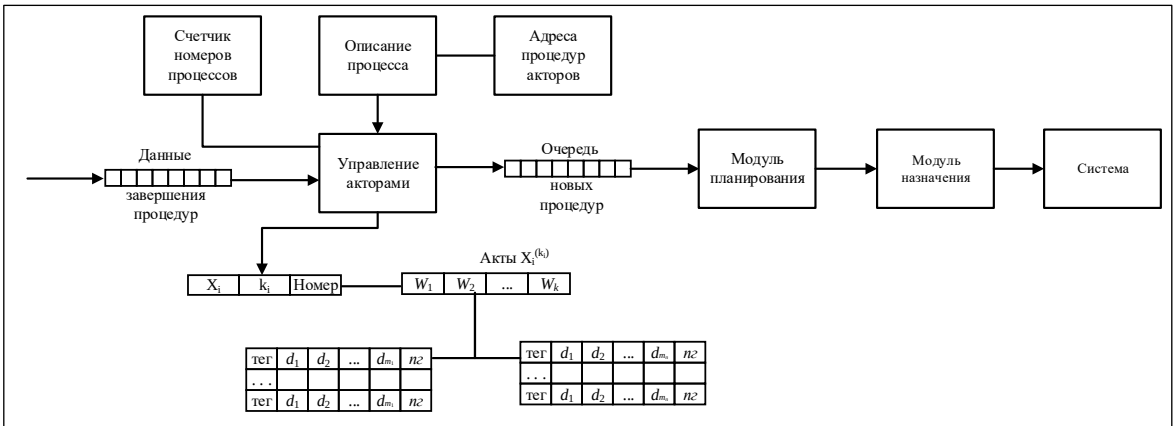

Рис. 4. Структура управления актами $X_i^{(k_i)}$

Fig. 4. The actor control structure for $X_i^{(k_i)}$

информации о том, насколько они эффективны, нет.

В работе [19] предложены и исследованы эффективные алгоритмы для решения этой задачи. Они без особой сложности могут быть реализованы, что должно повысить общую эффективность выполнения параллельных процессов на системах.

Заключение

Предложенный в статье язык может одновременно рассматриваться как язык параллельных процессов и как высокоуровневый язык модульного параллельного программирования. Простые средства группирования актов, рекурсия как более мощное средство задания процессов по сравнению с циклическими определениями, модульный характер организации процесса и параллельной программы, возможность их схемного представления – это те особенности языка, которые увеличивают его выразительные возможности на стадии как проектирования процессов и параллельных программ, так и организации их выполнения.

Возможны и, очевидно, будут полезны следующие расширения языка процессов: введение специальных переменных акторов, используемых в правых частях системы процессных уравнений при описании процесса, которые при их инициализации могут вызывать для выполнения новые процессы из соответствующей библиотеки процессов; обеспечение возможности динамически определяемого взаимодействия между акторами в правых частях системы процессных уравнений.

Работа выполнена при финансовой поддержке РФФИ, грант № 18-01-00548.

Литература

1. Варшавский В.И., Кишиневский М.А., Мараховский В.Б. [и др.]. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах. М.: Наука, 1986. 400 с. DOI: 10.13140/RG.2.1.2230.6644.

2. Эндрюс Г.Р. Основы многопоточного параллельного и распределенного программирования; [пер. с англ.]. М.: Вильямс, 2003. 505 с.

3. Хьюз К., Хьюз Т. Параллельное и распределенное программирование с использованием C++; [пер. с англ.]. М.: Вильямс, 2004. 667 с.

4. Cesarini F., Thompson S. ERLANG Programming: a Concurrent Approach to Software Development. O'Reilly Media, 2009, 496 p.

5. Agha G.A., Mason I.A., Smith S.F., Talcott C.L. Towards a theory of actor computation. Proc. Int. Conf. Concurrency Theory, 1992, pp. 565–579. DOI: 10.1007/BFb0084816.

6. Кутепов В.П., Маланин В.Н., Панков Н.А. Граф-схемное потоковое параллельное программирование: язык, процессная модель, реализация на компьютерных системах // Изв. РАН. ТСУ. 2012. № 1. С. 87–102. DOI: 10.1134/S1064230712010108.

7. Кутепов В.П., Фальк В.Н. Формы, языки представления, критерии и параметры сложности параллелизма // Программные продукты и системы. 2010. № 3. С. 16–25.

8. Кутепов В.П. Модели и языки для описания параллельных процессов // Изв. РАН. ТСУ. 2018. № 3. С. 116–127. DOI: 10.7868/S0002338818030113.

9. Milner R. A calculus of communicating systems. LNCS, 1980, vol. 92, 184 p.

10. Хоар Ч. Взаимодействующие последовательные процессы; [пер. с англ.]. М.: Мир, 1989. 264 с. DOI: 10.13140/RG.2.1.3474.7129.

11. Bergstra J.A., Klop J.W. Process algebra for synchronous communication. Information and Control. 1984, vol. 60, iss. 1-3, pp. 109–137. DOI: 10.1016/S0019-9958(84)80025-X.

12. Кутепов В.П., Фальк В.Н. Алгоритмические параллельные процессы и их сложность // Вестн. МЭИ. 2020. № 3. С. 102–110. DOI: 10.24160/1993-6982-2020-3-102-110.

13. Lamport L. The temporal logic of actions. ACM TOPLAS, 1994, vol. 16, iss. 3, pp. 872–923. DOI: 10.1145/177492.177726.

14. Middelburg C.A. Revisiting timing in process algebra. The Journal of Logic and Algebraic Programming, 2003, vol. 54, iss. 1-2, pp. 109–127. DOI: 10.1016/S1567-8326(02)00029-2.

15. Montanari A., Policriti A. Decidability results for metric and layered temporal logics. Notre Dame Journal of Formal Logic, 1996, vol. 37, no. 2, pp. 260–282. DOI: 10.1305/ndjfl/1040046089.

16. Baeten J.C.M., Middelburg C.A. Real time process algebra with time-dependent conditions. The Journal of Logic and Algebraic Programming, 2001, vol. 48, no. 1-2, pp. 1–38. DOI: 10.1016/S1567-8326(01)00004-2.

17. Baeten J.C.M., Bergstra J.A. Discrete time process algebra. Formal Aspects of Computing, 1996, vol. 8, no. 2, pp. 188–208.

18. Ковалев С.П. Архитектура времени в распределенных информационных системах // Вычислительные технологии. 2002. Т. 7. № 6. С. 38–53.

19. Кутепов В.П. Интеллектуальное управление процессами и загруженностью в вычислительных системах // Изв. РАН. ТСУ. 2007. № 5. С. 58–73. DOI: 10.1134/S1064230707050061.

Software & Systems Received 10.06.20
DOI: 10.15827/0236-235X.131.375-384 2020, vol. 33, no. 3, pp. 375–384

Parallel processes and programs: models, languages, implementation in systems

V.P. Kutepov [1], *Dr.Sc. (Engineering), Professor, kutepov@appmat.ru*
A.A. Efanov [1], *Postgraduate Student, 1134togo@gmail.com*

[1] *National Research University "Moscow Power Engineering Institute", Moscow, 111250, Russian Federation*

Abstract. The authors propose models and languages of parallel processes and programs, which have wider possibilities for describing parallelism in comparison with well-known models and languages created for this purpose.

The concept of an actor, the execution of acts, and the interaction of actors in the performance of a process are the basic elements of a process description. The general form of defining a process is a set of recursive equations, the right parts of which are groups of actors, which interact during the process execution. The graphic and textual form of the description of processes and the principle of initializing the execution of an act by the readiness of the signals (data) arriving at the inputs of the corresponding actor from the actors influencing it creates the necessary conditions for an effective description of the processes, their analysis, modification, and parallel execution. The interpretation of process actors as procedures provides a direct transition from the process language to the high-level language of modular multi-flow parallel programming.

The paper considers the main implementation problems of the proposed languages on computer systems: the unambiguous naming of the acts generated during the interaction of the process, the organization of process control on computer systems.

It is known that classical algebraic models of processes belong to R. Milner and C. Hoar. In this paper, we consider a more general model and language of processes, which, on the one hand, makes it possible to describe groups of interacting processes when the general process is defined in the form of a system of process equations. On the other hand, a dynamically generated structural description of cause-and-effect relationships between process actors in groups and the use of an actor's activation mechanism upon receipt of the necessary signals (data) to its respective inputs provide, without limitation, the implementation of parallelism and asynchrony when performing process acts. The use of variables in the system of equations describing the process creates the conditions for the dynamic generation of process groups.

The model described in the article is the basis for creating a parallel programming language implemented on computer systems.

Keywords: parallel processes, parallel programming, computer systems, implementation of parallel processes, actor model.

Acknowledgements. *This work was financially supported by RFBR, project no. 18-01-00548.*

References

1. Varshavskiy V.I., Kishinevsky M.A., Marakhovsky V.B. (Eds.). *Automata Control of Asynchronous Processes in Computers and Discrete Systems*. Moscow, 1986, 400 p. (in Russ.). DOI: 10.13140/RG.2.1. 2230.6644.

2. Andrews G.R. *Foundations of Multithreaded, Parallel, and Distributed Programming*. Addison-Wesley Publ., 1999, 688 p. (Russ. ed.: Moscow, 2003, 505 p.).

3. Hughes C., Hughes T. *Parallel and Distributed Programming Using C++*. Addison-Wesley Publ., 2003, 720 p. (Russ. ed.: Moscow, 2004, 667 p.).

4. Cesarini F., Thompson S. *ERLANG Programming: a Concurrent Approach to Software Development*. O'Reilly Media, 2009, 496 p.

5. Agha G.A., Mason I.A., Smith S.F., Talcott C.L. Towards a theory of actor computation. *Proc. Int. Conf. Concurrency Theory*, 1992, pp. 565–579. DOI: 10.1007/BFb0084816.

6. Kutepov V.P., Malanin V.N., Pankov N.A. Flowgraph stream parallel programming: Language, process model, and computer implementation. *J. Comp. Syst. Sci. Int.*, 2012, vol. 51, no. 1, pp. 65–79. DOI: 10.1134/S1064230712010108.

7. Kutepov V.P., Falk V.N. Forms, languages and complexity parameters of the parallelism. *Software & Systems*, 2010, no. 3, pp. 16–25 (in Russ.).

8. Kutepov V.P. Models and languages for description of parallel processes. *J. Comp. Syst. Sci. Int*., 2018, vol. 57, no. 3, pp. 471–481. DOI: 10.1134/S1064230718020119.

9. Milner R. A calculus of communicating systems. *LNCS*, 1980, vol. 92, 184 p.

10. Hoare C.A.R. *Communicating Sequential Processes (CSP)*. UK, Prentice Hall Publ., 1985, 260 p. DOI: 10.13140/RG.2.1.3474.7129 (Russ. ed.: Moscow, 1989, 264 p.).

11. Bergstra J.A., Klop J.W. Process algebra for synchronous communication. *Information and Control*, 1984, vol. 60, iss. 1-3, pp. 109–137. DOI: 10.1016/S0019-9958(84)80025-X.

12. Kutepov V.P., Falk V.N. Parallel algorithmic processes and their complexity. *Bull. MPEI*, 2020, no. 3, pp. 102–110 (in Russ.). DOI: 10.24160/1993-6982-2020-3-102-110.

13. Lamport L. The temporal logic of actions. *ACM TOPLAS*, 1994, vol. 16, iss. 3, pp. 872–923. DOI: 10.1145/177492.177726.

14. Middelburg C. A. Revisiting timing in process algebra. *The Journal of Logic and Algebraic Programming*, 2003, vol. 54, iss. 1-2, pp. 109–127. DOI: 10.1016/S1567-8326(02)00029-2.

15. Montanari A., Policriti A. Decidability results for metric and layered temporal logics. *Notre Dame Journal of Formal Logic*, 1996, vol. 37, no. 2, pp. 260–282. DOI: 10.1305/ndjfl/1040046089.

16. Baeten J.C.M., Middelburg C.A. Real time process algebra with time-dependent conditions. *The Journal of Logic and Algebraic Programming*, 2001, vol. 48, no. 1-2, pp. 1–38. DOI: 10.1016/S1567-8326(01)00004-2.

17. Baeten J.C.M., Bergstra J.A. Discrete time process algebra. *Formal Aspects of Computing*, 1996, vol. 8, no. 2, pp. 188–208.

18. Kovalev S.P. Time architecture in distributed information systems. *Computational Technologies*, 2002, vol. 7, no. 6, pp. 38–53 (in Russ.).

19. Kutepov V.P. Intelligent scheduling processes and controlling workload in computing systems. *J. Comp. Syst. Sci. Int*., 2007, vol. 46, no. 5, pp. 726–740. DOI: 10.1134/S1064230707050061.

Для цитирования

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. DOI: 10.15827/0236-235X.131.375-384.

For citation

Kutepov V.P., Efanov A.A. Parallel processes and programs: models, languages, implementation in systems. *Software & Systems*, 2020, vol. 33, no. 3, pp. 375–384 (in Russ.). DOI: 10.15827/0236-235X.131.375-384.