УДК 519.876.5

Б. Б. Нестеренко, М. А. Новотарский
Институт математики НАН Украины
ул. Терещенковская, 3, 01601 Киев-4, ГСП, Украина
тел. (044) 234-04-07, e-mail: model@imath.kiev.ua, novot@ukr.net

Алгебра процессов для моделирования сложных систем с реальной рабочей нагрузкой

Приведен краткий обзор современных средств для описания моделей сложных систем. Показано, что для повышения адекватности моделей актуальна задача развития новых низкоуровневых формальных средств описания объектов — алгебр процессов. Коротко описаны основы известных алгебр процессов и предложена алгебра процессов, ориентированная на имитационное моделирование сложных систем с реальной рабочей нагрузкой. Рассмотрены вопросы определения эквивалентности системы и модели. Приведен пример описания однородной вычислительной среды в терминах алгебры процессов.

Ключевые слова: *алгебра процессов, процесс, активность, поведенческая эквивалентность, строгое взаимное подобие.*

Введение

Одним из самых мощных и эффективных средств исследования сложных дискретных систем является компьютерное моделирование. Особую актуальность оно приобретает в том случае, когда необходимо проводить исследование дискретных систем, натурные эксперименты с которыми нецелесообразны из соображений безопасности или дороговизны. К настоящему времени накоплен огромный опыт создания компьютерных моделей, различающихся природой формального описания объекта исследования. Среди прочих широкое распространение получил объектно-ориентированный подход к построению моделей. Известным представителем средств объектно-ориентированного моделирования является язык Modelica [1], созданный для описания сложных систем, содержащих механические, гидравлические, электрические, электронные и другие компоненты. Язык обеспечивает поддержку классов и экземпляров блоков, а также наследование и полиморфизм блоков. Класс определяет некоторый шаблон или прототип блока. Оперируя классом, нельзя говорить о конкретных значениях переменных, так как в определении класса присутствует только информация об их типах и именах. Экземпляр блока — это конкретный представитель класса блоков, вклю-

© Б. Б. Нестеренко, М. А. Новотарский

чающий свои собственные значения переменных. В функциональную схему могут входить несколько статических или динамических экземпляров одного и того же класса. Статические экземпляры создаются при создании модели и уничтожаются при ее уничтожении, а динамические могут создаваться и уничтожаться в ходе моделирования. Механизм наследования позволяет создавать новые классы путем использования уже имеющихся в качестве предков, а полиморфизм означает возможность использования вместо экземпляра блока некоторого базового класса экземпляра любого его производного класса.

Следует отметить, что упомянутый принцип объектно-ориентированного моделирования больше отражает внутреннюю организацию модели. В то же время чрезвычайно важно обеспечить комфортные условия взаимодействия с ней пользователей. В связи с этим большой стимул к развитию получил визуальный подход, разрабатываемый с целью упрощения технологии создания моделей. Визуализация чаще всего происходит за счет применения технологии «блочного моделирования», использующей графический язык иерархических блок-схем. Функции элементарных блоков могут быть предопределенными или описываться с помощью языка более низкого уровня. Подход «блочного моделирования» использован, например, в пакетах SIMULINK (MATLAB) [2], VisSim [3] и др.

Современные средства моделирования стремятся сочетать в себе объектно-ориентированный и визуальный подходы. Примером удачной реализации является унифицированный язык моделирования (UML) [4]. Первоначально разработанный для определения, визуализации, проектирования и документирования программных систем, сейчас UML используется также при моделировании бизнес-процессов, в системном проектировании и отображении организационных структур.

Использование упомянутых подходов чаще всего оправдано в том случае, когда речь идет об анализе качественных параметров объектов исследования. В случае необходимости получения точных характеристик сложной системы возникает проблема обоснования адекватности модели. Избыточный характер визуального описания и неточная семантика зачастую становятся непреодолимыми препятствиями для решения этой задачи. Решение проблемы лежит в сфере использования низкоуровневых средств формального описания. Долгое время фаворитом низкоуровневых средств являлись сети Петри [5] и их модификации [6]. Однако громоздкость сетевого формального представления стала существенным препятствием при описании сложных систем. Современные низкоуровневые средства, наряду с компактностью представления модели, должны обеспечивать хорошо формализованный механизм установления ее адекватности. Именно таким требованиям соответствуют алгебры процессов.

Основы описания алгебры процессов

Алгебра процессов — это инструмент формального описания, который моделирует параллельные системы посредством их алгебры и обеспечивает аппарат анализа структуры и поведения модели. Основополагающими разработками в данной области принято считать такие алгебры процессов: Calculus of Communicating Systems (CCS) [7], Communicating Sequential Processes (CSP) [8], Algebra of

Communicating Processes (ACP) [9] и Performance Evaluation Process Algebra [10]. Все упомянутые алгебры включают описание активных компонент и механизмы взаимодействия между ними. В большинстве случаев активные компоненты представлены *процессами* или *агентами,* состоящими из *активностей,* каждая из которых задает определенные дискретные действия системы. Любая активность может быть внутренней по отношению к процессу, или задавать взаимодействие между процессами. Такое взаимодействие носит бинарный характер. Поэтому каждая активность $\alpha \in \Delta$ всегда имеет сопряженную активность $\overline{\alpha} \in \overline{\Delta}$, и это именно та сопряженная активность, с которой готова синхронизироваться активность α. В алгебре процессов можно скрыть поведение фрагмента процесса, представив его как черный ящик. При этом возникает возможность построить сложную имитационную модель, исключив более простые части, поведение которых не представляет интереса при исследовании. Невидимое поведение моделируют с помощью специальной активности σ, представляющей внутренние действия процесса. Таким образом, полный алфавит активностей $Act = \Lambda \cup \{\sigma\}$ включает активности взаимодействия $\Lambda = \Delta \cup \overline{\Delta}$ и множество $\{\sigma\}$ внутренних активностей.

Простейший константный процесс не может выполнять ни одной активности и приводит к освобождению всех ресурсов посредством выполнения активности nil. Если процесс включает более одной активности, последовательность их свершения регулируется префиксной записью $\alpha.P$, смысл которой состоит в том, что она представляет процесс, сначала выполняющий активность α, а затем функционирующий, как *P*. Примером элементарного процесса P_1 может служить выражение:

$$P_1 := begin.end.nil .$$

Символ $:=$ означает присвоение выражения в правой части процессу P_1.

Действия процесса согласно данному выражению состоят из прямой последовательности трех активностей. Активность *begin* выполняет действия, связанные с началом функционирования процесса P_1, *end* завершает процесс P_1, *nil* освобождает ресурсы и удаляет процесс P_1 из модели.

Для моделирования систем с параллельным функционированием компонентов синтаксис алгебры дополняют операциями выбора. Операция *P* + *Q* представляет процесс, который ждет готовности процессов *P* или *Q* принять участие в некоторой активности. После этого происходит *недетерминированный выбор* между ними. Применив префиксную запись, приведем пример процесса P_4, использующего данную операцию:

$$P_4 := begin_1.P_1 + begin_2.P_2 .$$

Процесс P_4 предполагает выбор между двумя альтернативами. Этот выбор контролируется начальными активностями составляющих его процессов и не зависит от внешнего воздействия.

Для того чтобы конструировать сложные процессы, необходимо также уметь параллельно запускать несколько подпроцессов, обеспечивая механизмы взаимодействия и синхронизации. С этой целью используют параллельную конструкцию $P \| Q$, которая представляет два процесса, развивающихся параллельно. Процессам *P* и *Q* позволено любое индивидуальное поведение, однако при наличии в них сопряженных активностей выполняется синхронизация путем одновременного свершения активностей.

К общепринятым операциям, используемым в различных алгебрах процессов, относят переименование $P[K]$ и сжатие $P \setminus A$. Процесс $P[K]$ ведет себя как *P*, но во всех своих активностях переименовывается функцией *K*. Процесс $P \setminus A$ ведет себя как *P*, но не позволяет выполнять никаких активностей из *A*.

Рассмотренное неформальное описание основных синтаксических операций, характерных для различных алгебр процессов, дает общее представление об этом инструменте формализации. Высокая эффективность данного подхода стимулирует разработчиков применять алгебры процессов к самым различным моделям. На сегодняшний день хорошо разработаны подходы к использованию алгебры процессов для построения стохастических моделей параллельных структур [10] и для имитационных моделей с синтетическими рабочими нагрузками, базирующимися на вероятностном распределении параметров входных заданий [11]. Однако такие подходы требуют предварительных знаний о системе и ее рабочей нагрузке для получения адекватных результатов моделирования. При использовании моделей на этапе разработки сложной системы такие знания, как правило, отсутствуют. Поэтому представляет значительный интерес создание алгебры процессов для имитационного моделирования систем с реальной рабочей нагрузкой.

Алгебра процессов для описания моделей сложных систем с реальной нагрузкой

Кроме рассмотренных особенностей рассматриваемая в данной работе версия алгебры процессов отличается применением в ней технологии «блочного моделирования». Новизна этого подхода состоит в том, что алгебраическая модель имеет двухуровневый характер. На верхнем уровне с помощью выражений алгебры процессов формируется структура модели и определяется характер взаимодействия между компонентами. Нижний уровень содержит алгоритмы, описывающие процесс обработки реальной рабочей нагрузки отдельными компонентами. Такой подход позволяет достичь высокой адекватности модели при реализации конкретных вычислительных алгоритмов. Для его успешной реализации полный алфавит активностей $Act = \Lambda \cup \Phi$ включает активности взаимодействия $\{\lambda_j\}_{j=1}^{n} \subseteq \Lambda$ и множество внутренних активностей $\{\varphi_i\}_{i=1}^{m} \subseteq \Phi$. Активности взаимодействия моделируют обмен информацией между компонентами сложной системы, а внутренние активности определяют алгоритмы функционирования этих компонент на заданном уровне абстрагирования.

Синтаксис алгебры процессов для имитационного моделирования сложных систем с реальной рабочей нагрузкой основан на использовании двух типов переменных: *Pvar и Var*. Переменные типа *Pvar* предназначены для описания значе-

ний параметров процессов системы, а переменные типа *Var* задают значения параметров рабочей нагрузки. *Pvar* — множество переменных для временного хранения и обработки экземпляров процессов, включающее такие переменные процессов:

$$X, \quad X_{[i]}, \quad X_{key}, \quad nil.$$

Формальное описание алгебры процессов допускает наличие конечного множества переменных, значения которых определяются при помощи операции $X := P$. Данная операция присваивает тривиальной переменной X поведение процесса P. Переменные сложного типа могут отображать индексные и списковые структуры. Индексная переменная процесса имеет вид $X_{[i]}$, где $i \in I$ — значение индекса. Списковая переменная X_{key} всегда содержит ссылку key, указывающую на следующий элемент. Последний элемент списка имеет ссылку nil.

Выражения для процессов объединены во множество $Pexp$, а выражения для рабочей нагрузки составляют множество выражений *Exp*. *Pexp* — множество, которое включает такие выражения:

$$\alpha.P,\ \langle t\rangle.P,\ \langle s \leftarrow t\rangle.P,\ (s \leftarrow y(t)).P;$$

$$P_1 + \cdots + P_i + \cdots + P_I,\ v_1!P_1 + \cdots + v_i!P_i + \cdots + v_I!P_I,\ w_1\#P_1 + \cdots + w_i\#P_i + \cdots + w_I\#P_I;$$

$$P \mid Q \mid \cdots \mid C,\ P \triangleright Q \mid \cdots \mid C,\ P \leftrightarrow Q \mid \cdots \mid C,\ P \lhd\rhd Q \mid \cdots \mid C;$$

$$P\langle K \mid;\ P \setminus A;$$

$$k \times P, C\{P\}, \varphi_P(C);$$

$$\Re(X = P), \quad cycle[x](X = P),$$

где $X \in Pvar$, $P, Q \in Pexp$, $t, s, w, x \in Var$, $y \in Exp$, $\alpha \in Act$; I — конечное множество индексов; K — функция переименования $K : Act \to Act$, $A \subset \Lambda$ — подмножество активностей взаимодействия.

Переменные типа *Var* соответствуют простым и структурированным типам переменных, описываемым в алгоритмических языках программирования. Поэтому правила для выражений *Exp* напоминают правила составления выражений в упомянутых языках.

Операции префиксации $\alpha.P,\ \langle t\rangle.P,\ \langle s \leftarrow t\rangle.P,\ (s \leftarrow y(t)).P$ образуют базовый механизм конструирования поведения процесса. Тривиальная префиксация $\alpha.P$ представляет процесс, поведение которого соответствует описанному ранее. Префиксная задержка $\langle t\rangle.P$ — это операция, которая задает процесс с начальной задержкой на t единиц модельного времени, а параметрическая префиксная за-

держка $\langle s \leftarrow t \rangle.P$ описывает действия процесса, для которого предварительно устанавливается значение временного параметра s путем присвоения ему значения t. Функциональная префиксация $(s \leftarrow y(t)).P$ определяет действия процесса, задержанного на величину s, принадлежащую области значений функции y.

Осуществление префиксаций $\alpha.P$ и $\langle t \rangle.P$ не зависит от состояния внешней среды и определяется исключительно внутренними параметрами процесса. Поэтому их используют для описания независимых процессов. Префиксации $\langle s \leftarrow t \rangle.P$ и $(s \leftarrow y(t)).P$ отображают характер взаимодействия процессов с внешней средой. Такое взаимодействие может быть выражено в виде параметрической синхронизации. Синхронизацию по времени задают путем параметрической замены величины задержки начала развития процесса P в одном из параллельных процессов. При функциональной задержке развитие процесса P начинается только при условии срабатывания ключевой функции y.

Операции выбора $P_1 + \cdots + P_i + \cdots + P_I$, $v_1!P_1 + \cdots + v_i!P_i + \cdots + v_I!P_I$, $w_1\#P_1 + \cdots + w_i\#P_i + \cdots + w_I\#P_I$ предназначены для определения одного процесса из множества процессов с целью его дальнейшего развития. Недетерминированный выбор обеспечивает определение одного из процессов $P_1 + ... + P_i + ... + P_I$ в момент времени, который называется моментом выбора. Управляемый выбор $v_1!P_1 + \cdots + v_i!P_i + \cdots + v_I!P_I$ осуществляет выбор того процесса P_i, для которого соответствующая логическая переменная v_i принимает истинное значение.

Вероятностный выбор $w_1\#P_1 + \cdots + w_i\#P_i + \cdots + w_I\#P_I$ производит выбор процесса P_i с вероятностью w_i при условии $w_1 + ... + w_i + ... + w_I = 1$.

Множество активностей процесса выбора формируется путем объединения множеств составных процессов:

$$Act\left(\sum_{i \in I} P_i\right) := Act(P_1) \cup ... \cup Act(P_i) \cup ... \cup Act(P_I).$$

Условие выполнения активности $\alpha \in Act\left(\sum_{i \in I} P_i\right)$ требует принадлежности данной активности к множеству активностей одного из процессов $\alpha \in Act(P_k),\ k \in I$. Суть операции выбора состоит в отображении конкуренции элементов системы за владение скрытым ресурсом. Поэтому в случае наличия одинаковых активностей во множествах разных процессов срабатывает только та активность, которая первой получила доступ к скрытому ресурсу.

Операции параллельной композиции и взаимодействия $P \,|\, Q \,|\cdots|\, C$, $P \triangleright Q \,|\cdots|\, C$, $P \leftrightarrow Q \,|\cdots|\, C$, $P \triangleleft\triangleright Q \,|\cdots|\, C$ определяют взаимное поведение параллельно развивающихся процессов. Операция параллельной композиции

$P|Q|\cdots|C$ показывает, что упомянутые процессы в ходе своего развития либо не взаимодействуют вообще, либо характер их взаимодействия не влияет на результаты моделирования. Операция управляемой передачи $P \triangleright Q|\cdots|C$ определяет процедуру передачи информации от процесса P к параллельной композиции $Q|\cdots|C$ в момент свершения активности вывода в процессе P. Операции синхронного взаимодействия $P \leftrightarrow Q|\cdots|C$ и асинхронного взаимодействия $P \lhd\rhd Q|\cdots|C$ формируют процедуры обмена информацией, отличающиеся тем, что для синхронного обмена необходимо одновременное выполнение комплементарных активностей передачи и приема, а для асинхронного обмена не требуется ожидать момента возникновения комплементарных активностей.

Операции переименования и сжатия $P\langle K|, P \setminus A$ относят к операциям группового редактирования, поскольку они обеспечивают модификацию группы активностей процесса одновременно. Операцией переименования $P\langle K|$ называют систему, представленную процессом P, все активности которого переименовываются с помощью функции K. Операция сжатия $P \setminus A$ представлена системой, состоящей из процесса P и множества активностей A, в которой заблокированы все активности, принадлежащие множеству сжатия A. Главным назначением данных операций является модификация поведения группы объектов модели в ходе моделирования. Например, с помощью операции сжатия можно управлять синхронизацией обмена между процессом P и параллельной композицией $Q|\cdots|C$ путем блокирования комплементарных активностей взаимодействия.

Операции клонирования, включения и сокрытия введены для придания описаниям моделей иерархических свойств. Операция клонирования $k \times P$ порождает k идентичных процессов P. Операция включения $C\{P\}$ задает конструкцию из процессов C и P, включающую процесс P в качестве субпроцесса процесса C. Операция сокрытия $\varphi_P(C)$ позволяет скрывать выполнение субпроцесса P, входящего в главный процесс C, путем замены его некоторой вычислительной активностью φ.

Операции рекурсии и цикла $\Re(X = P)$, $cycle\langle x\rangle(X = P)$ предназначены для многократного повторения некоторой последовательности активностей, входящих в процесс P. Операцией рекурсии $\Re(X = P)$ называют систему, состоящую из рекурсивно повторяемой последовательности активностей процесса P, присвоенных переменной типа *Pvar*. Операцией цикла $cycle\langle x\rangle(X = P)$ называют систему, которая состоит из циклически повторяемой x раз последовательности активностей процесса P, присвоенного переменной типа *Pvar*. Операция рекурсии может задавать бесконечную или конечную последовательность процессов. Конечная последовательность возникает в том случае, когда в ходе выполнения очередного экземпляра процесса исполняется условие завершения рекурсии, которое всегда определяется внутренним состоянием процесса. Операция цикла отличается от операции рекурсии наличием переменной x, значение которой ограничивает количество повторений выполнения процесса P.

Описание системы с помощью рассмотренных операций алгебры процессов задает только верхний уровень модели. Для исчерпывающего ее описания необходимо функционально определить элементы множества активностей Act, представив модель в виде маркированной системы с переходами: $\left(\Pi, Act, S, \left\{\xrightarrow{\alpha} \middle| \alpha \in Act\right\}\right)$, где $\Pi = \begin{Bmatrix} P_{1,1} & \cdots & P_{1,4} \\ \cdots & P_{i,j} & \cdots \\ P_{4,1} & \cdots & P_{4,4} \end{Bmatrix}$ — множество процессов; $\xrightarrow{\alpha}$ — множество переходов, формируемых внутренними активностями и активностями взаимодействия при $\left\{\xrightarrow{\alpha}\right\} \subseteq \Pi \times Act \times \Pi$; S — множество состояний системы.

Такое представление дает возможность воспользоваться подходами общей теории систем для строгого обоснования эквивалентности модели ее спецификации в случае моделирования с целью разработки.

Вопросы эквивалентности

Основой определения адекватности является поведенческая эквивалентность, которая с помощью анализа состояний системы позволяет сформулировать условия эквивалентного поведения процессов [12]. Под состоянием системы в данном случае понимают совокупность значений параметров, установленных в результате свершения активности.

Для практического определения эквивалентности используют взаимное подобие, основывающееся на отношении эквивалентности.

Пусть P — процесс, а бинарное отношение R на P представляет собой множество $P \times P$ с состояниями $(p, q) \in R$ или pRq. Тогда эквивалентным отношением R на P будем называть бинарное отношение, отвечающее таким условиям:

— R должно быть рефлексивным, т.е. pRp, qRq при $p, q \in P$;

— R должно быть симметричным, т.е. из pRq следует qRp при $p, q \in P$;

— R должно быть транзитивным, т.е. из pRq и qRz следует pRz при $p, q, z \in P$.

Рефлексивность бинарного отношения обеспечивает внутреннюю непротиворечивость процессов, эквивалентно отображающих внутренне непротиворечивые процессы. Транзитивность позволяет сохранять эквивалентность во время пошаговой реализации модели в соответствии с ее описанием. Для того, чтобы объект моделирования и его описание были взаимно подобными, очевидна также необходимость выполнения принципа симметричности.

Для выполнения строгого взаимного подобия необходимо, чтобы активности системы соответствовали активностям модели, и состояния, достигаемые после выполнения каждой из активностей, были одинаковыми.

Пусть существует отношение $R \subseteq \Pi \times \Pi$ и процессы $P, Q \subseteq \Pi$ с допустимыми состояниями $p, p' \in P$, $q, q' \in Q$ при $(p, q) \in R$. Тогда, если:

— для каждого перехода $p \xrightarrow{\alpha} p'$ существует состояние q' такое, что

$q \xrightarrow{\alpha} q'$ при $(p', q') \in R$,

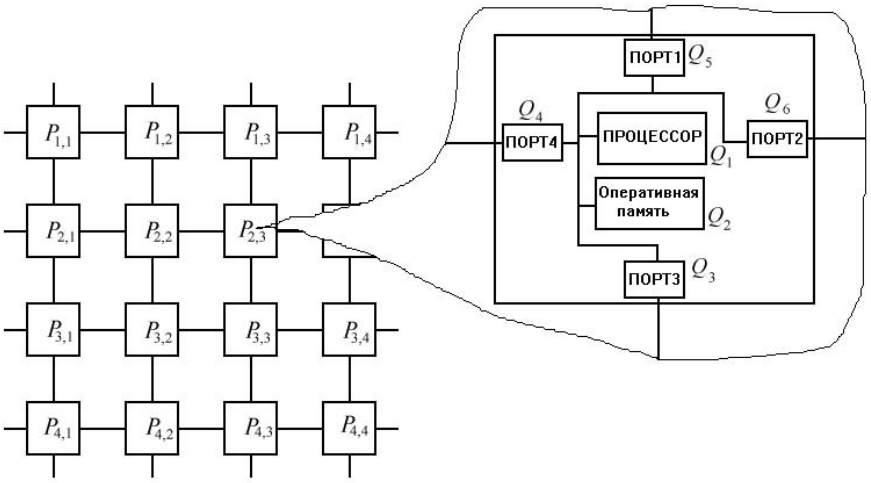
— для каждого перехода $q \xrightarrow{\alpha} q'$ существует состояние p' такое, что $p \xrightarrow{\alpha} p'$ при $(p', q') \in R$,

то отношение R будем называть строгим взаимным подобием, а состояния p и q будем называть строго взаимно подобными $p \sim q$.

В случае, когда все состояния системы и модели строго взаимно подобны, то такая модель эквивалентна исследуемой системе.

Пример описания однородной вычислительной среды

Рассмотрим, например, описание структуры однородной вычислительной среды (см. рис.), состоящей из узлов, описываемых процессами алгебры процессов.



Структура однородной вычислительной среды

Произвольный узловой процесс $P_{i,j}$ включает подпроцессы $Q_1, \cdots, Q_4$, моделирующие работу основных элементов узла в соответствии с выражением алгебры процессов:

$$P_{i,j}\{Q_1 \leftrightarrow (v_2!Q_2 + v_3!Q_3 + v_4!Q_4 + v_5!Q_5 + v_6!Q_6)\}. \quad (1)$$

Выражение (1) показывает, что в произвольный момент времени процесс Q_1 может синхронно взаимодействовать с тем процессом, который будет выбран в результате выполнения операции управляемого выбора $(v_2!Q_2 + v_3!Q_3 + v_4!Q_4 + v_5!Q_5 + v_6!Q_6)$.

Работа узла сводится к многократному повторению процесса $P_{i,j}$, задаваемому оператором рекурсии:

$$\Re(X = P_{i,j}). \quad (2)$$

Определив взаимодействие между узлами как асинхронное, получим выражение алгебры процессов для внешнего взаимодействия узла $P_{i,j}$:

$$P_{i,j} \lhd\rhd \left(P_{i-1,j} \middle| P_{i+1,j} \middle| P_{i,j-1} \middle| P_{i,j+1}\right). \quad (3)$$

Для получения описания однородной вычислительной среды объединим выражения (1), (2) и (3):

$$\Re\left(P_{i,j}\left\{Q_1 \leftrightarrow \left(v_2!Q_2 + v_3!Q_3 + v_4!Q_4 + v_5!Q_5 + v_6!Q_6\right)\right\}\right) \lhd\rhd \left(P_{i-1,j} \middle| P_{i+1,j} \middle| P_{i,j-1} \middle| P_{i,j+1}\right), \quad (4)$$

где $i = 1,\cdots,4;\ j = 1,\cdots,4$.

Выражение (4) определяет все возможные варианты развития процессов как на уровне взаимодействия между узлами, так и на уровне подпроцессов узла. Однако для решения вопросов эквивалентности модели необходимо активностям, формирующим данные процессы, поставить в соответствие алгоритмы обработки реальной рабочей нагрузки.

Выводы

Современные средства моделирования широко используют объектно-ориентированный подход, который в сочетании с визуальным представлением информации позволяет существенно сократить время построения модели сложной системы. Однако избыточный характер визуального описания и неточная семантика зачастую становятся непреодолимыми препятствиями для построения таких моделей, эквивалентность которых объекту моделирования должна быть строго обоснованной. Решение проблемы лежит в сфере использования низкоуровневых средств формального описания, позволяющих применять принципы определения эквивалентности, используемые в общей теории систем. Одним из современных и бурно развивающихся низкоуровневых средств такого типа являются алгебры процессов, основанные на событиях или активностях, объединяемых в процессы, которые описывают работу компонентов сложных систем. Интерпретация активностей и характер взаимодействия в рамках отдельной алгебры процессов полностью определяются ее алфавитом и совокупностью допустимых операций. Предложенная в рамках этой работы версия алгебры процессов ориентирована на имитационное моделирование сложных систем с реальной рабочей нагрузкой. Поэтому она содержит алфавит, состоящий из двух множеств активностей, объединяющих активности взаимодействия и внутренние активности. Расширенный набор операций позволяет строить иерархические модели с различными видами взаимодействия между компонентами. Определение активностей за счет применения технологии «блочного моделирования» дает возможность обработки моделью реальной рабочей нагрузки.

Решение вопросов эквивалентности сводится к установлению эквивалентного отношения между состояниями системы и модели. Строгое взаимное подобие на-

ступает при условии, что активности системы соответствуют активностям модели, и состояния, достигаемые после выполнения каждой из активностей, одинаковы.

1. *Fritzson P.* Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. — San Francisco: Wiley-IEEE Press, 2004. — 944 p.

2. *Karris T.S.* Introduction to Simulink with Engineering Applications. — Fremont: Orchard Publications, 2006. — 584 p.

3. *Дьяконов В.* VisSim+MathCad+MATLAB. Визуальное математическое моделирование. — М.: Солон-Пресс, 2004. — 384 с.

4. *Буч Г., Якобсон А.* UML. Классика CS. — С.Петербург: Из-во «Питер», 2006. — 736 с.

5. *Питерсон Дж.* Теория сетей Петри и моделирование систем. — М.: Мир, 1984. — 264 с.

6. *Нестеренко Б.Б., Новотарский М.А.* Мультипроцессорные системы. — К.: Институт математики АН Украины, 1995. — 408 с.

7. *Milner R.* Communication and Concurrency. — London: Prentice-Hall, 1989. — 260 p.

8. *Хоар Ч.* Взаимодействующие последовательные процессы. — М.: Мир, 1989. — 264 с.

9. *Bergstra J.A., Klop J.W.* Algebra for Communicating Processes with Abstraction // J. of Theoretical Computer Science. — 1985. — Vol. 37. — P. 77–121.

10. *Hilston J.* A Compositional Approach to Performance Modelling. — Cambridge University Press. —1996. — 168 p.

11. *Glynn P.W.* A GSMP Formalism for Discrete Event Simulation // Proc. of the IEEE. — 1989. — Vol. 77, N 1. — P. 14–23.

12. *Main M.* Trace, Failure and Testing Equivalences for Communicating Processes // International Journal of Parallel Programming. — 1987. — Vol.16, N 5. — P. 383–400.

Поступила в редакцию 17.10.2007