УДК 681.324

С. А. Зинкин

ИЕРАРХИЧЕСКИЕ СЕТИ АБСТРАКТНЫХ МАШИН И ВИРТУАЛИЗАЦИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ ВНЕШНЕГО ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ

Аннотация. Определяются формализмы для описания иерархий абстрактных машин с целью последующей виртуализации систем внешнего хранения и обработки данных. Дано понятие об иерархической эволюционирующей алгебраической системе, на основании которого построена конкретная иерархия абстрактных машин, легко реализуемая программно и аппаратно.

Ключевые слова: иерархия абстрактных машин, система обработки данных, формальная спецификация, сети абстрактных машин.

Abstract. A new formalism based on hierarchical abstract machines is given. The purpose of the description is virtualization of external data processing systems. Definition of hierarchical evolving algebraic systems is given and an example of concrete abstract machines hierarchy is presented. Such a hierarchy provides a simple software and hardware implementation.

Keywords: abstract machines hierarchy, data processing systems, formal specifications, abstract machines network.

Введение

Две тенденции – достижение высокой производительности и логической гибкости – во многом определяют развитие современной вычислительной техники. Они во многом противоречат друг другу – в ряде случаев приходится жертвовать скоростью выполнения рабочих операций для повышения функциональности. С другой стороны, сохраняются стимулы к разработке и использованию перспективных высокопроизводительных устройств, например, ассоциативной памяти, систолических матриц, твердотельных «дисковых» устройств, «интеллектуальных» дисков и других устройств, построенных на основе технологий больших интегральных схем (БИС). В качестве интеллектуального «периферийного» устройства может быть рассмотрена и вся локальная или глобальная сеть, с которой работает пользователь. Посредством использования технологий виртуализации здесь можно реализовать, например, сеть внешнего хранения структурированных данных или внешний «сетевой сопроцессор» (метакомпьютер).

Многими исследователями в области новых компьютерных технологий предложены методы по преодолению семантического разрыва между структурой «классического» процессора (иногда условно называемого фоннеймановским) и используемыми алгоритмическими структурами. Однако при организации вычислений в современных сетевых архитектурах возникают новые проблемы, связанные с использованием средств синхронизации процессов, составлением параллельных программ, разработка которых качественно и количественно отличается от разработки последовательных программ, коллективным использованием распределенных ресурсов. Требование специализации ЭВМ, связанное с необходимостью разработки и последующего использования аппаратных средств для реализации механизмов логического вывода, поиска в базе знаний, реализации других алгоритмов искусст-

венного интеллекта зачастую вступает в противоречие с необходимостью использования в сетях по различным причинам преимущественно «классических» ЭВМ. Поэтому распространены методы реализации различных новых логических архитектур на доступной физической инфраструктуре вычислительной сети.

Некоторые известные принципы, принятые ранее в процессе развития вычислительной техники и организации вычислений (см., например, [1]), можно перенести и на сетевые структуры: принцип изменяемой хранимой программы, которая может быть модифицирована другими программами; принцип полиморфизма, заключающийся в возможности изменения архитектуры вычислительной системы таким образом, чтобы она соответствовала назначению системы; применение многопортового механизма взаимодействия с блоками памяти; применение блокировки данных вместо блокировки критических участков программы для сокращения временных затрат на синхронизацию в многопроцессорных системах; ресурсный принцип – при его реализации для каждого процесса устанавливается размер того или иного ресурса, который этот процесс может удерживать в состоянии ожидания в точке синхронизации и др. В сетевой среде с применением перечисленных принципов можно создать некоторое виртуальное пространство структурированной памяти, разделяемое несколькими процессами. Данные процессы могут осуществлять согласованные взаимодействия по выполнению какого-либо приложения при работе системы как единого целого.

1 Многоуровневые абстракции и виртуальные системы

Система, или сеть хранения и обработки данных, представляется пользователю в виде виртуальной вычислительной системы с многочисленными ресурсами и функциями. Концепция многоуровневой виртуальной системы базируется на понятии уровней абстракции. Как вертикальные, так и горизонтальные межмодульные связи являются результатами причинно-следственных (каузальных) связей. Как правило, связываются только смежные уровни, но при необходимости связи могут распространяться на произвольное число уровней. В реальных системах обычно устанавливаются жесткие ограничения на глубину связей. Нередко ограничиваются даже связи внутри одного и того же уровня. В качестве формальной модели структурной организации виртуальной системы нами выбираются логико-алгебраические описания, или спецификации сетей абстрактных машин. Такие описания удобны для их практического использования в качестве основы для непосредственно исполняемых спецификаций, которые могут быть реализованы в рамках произвольных, в том числе объектно-ориентированных технологий. В итоге реализуется иерархическая система вложенных друг в друга виртуальных машин. Процесс проектирования аппаратных и микропрограммных (на самом нижнем уровне иерархии), а также программно-реализованных модулей, полученных на основе использования сценарных представлений и формальных логико-алгебраических спецификаций, хорошо автоматизируется.

Абстрактной машиной принято называть математическую формализацию, предназначенную для моделирования программы реальной вычислительной машины; при практической же реализации используются аналоги абстрактных машин в виде виртуальных. На практике нередко термин «абстрактная машина» используется и для обозначения виртуальной машины. Тех-

нология виртуализации, базирующаяся на сетях абстрактных машин (СеАМ), наследует ряд общих черт от известных технологий.

На практике виртуальные машины создаются в виртуальной сетевой инфраструктуре в виде программного слоя в операционной системе или непосредственно над аппаратным обеспечением компьютера. Для приложений систем и сетей хранения и обработки данных при использовании непосредственно выполняемых спецификаций СеАМ «толщина» этого программного слоя невелика.

По аналогии с англоязычными терминами *software* (программное обеспечение), *firmware* (микропрограммное обеспечение) и *hardware* (аппаратное обеспечение) в последнее время нашел применение термин *vmware* (технология виртуальных машин, где *vm – virtual machine*). Известно, что иерархия программных средств систем и сетей хранения и обработки данных может рассматриваться как реализация иерархии абстрактных машин. Программные средства, реализующие абстрактную машину, выполняют вычислительные и управляющие функции.

Желательно, чтобы при реализации абстрактных машин на разных уровнях иерархии использовались сходные технологии, основанные на ограниченном выборе формальный моделей. В приложениях информатики обычно рассматривают некоторое множество Σ представлений с интерпретацией I в множестве S элементов; интерпретация I данному представлению σ ставит в соответствие некоторое абстрактное информационное содержание $I(\sigma)$, т.е. интерпретации соответствует отображение I: $\Sigma \rightarrow S$ [2]. Пусть Σ – множество функциональных и предикатных символов различных арностей (в многосортных, или многоосновных, системах тип n-арного функционального символа – это кортеж $(i_1, i_2, \ldots, i_n, j)$, а тип n-арного предикатного символа – это кортеж $(i_1, i_2, \ldots, i_n)$, где $i_1, i_2, \ldots, i_n$ (j – названия (сорта) для основ, или носителей), а S – множество конкретных функций и предикатов. СеАМ использует «модули-продукции» и «модули-процедуры», которые модифицируют, или «обновляют», интерпретацию I, выполняя сгруппированные в блоки так называемые правила обновления вида $I(\sigma_i) \leftarrow s_j$. В машинах абстрактных состояний, описанных в работах [3, 4], определены используемые нами специальные операции – элементарные обновления функций и предикатов. Элементарное правило обновления функции, или предиката, записывается в виде правила вывода:

$$\frac{t_1, t_2, ..., t_k, t_{k+1}}{s(t_1, t_2, ..., t_k) \leftarrow t_{k+1}},$$

где $t_1, t_2, ..., t_k$ – термы различных сортов; s – функциональный или предикатный символ. В случае, если s – функциональный символ, t_{k+1} – суть терм любого сорта, а если s – предикатный символ, то t_{k+1} – булево выражение.

Сеть СеАМ функционирует, переходя от одной интерпретации $I(t_i)$ к другой $I(t_k)$, где t_i и t_k – последовательные моменты времени. В алгебре абстрактных машин мы включаем подобные правила обновления в систему образующих.

Наиболее удобным подходом к внутренней интеллектуализации был бы такой подход, при котором разработчик формулировал условия получения

результата обработки данных, определял минимальные требования к архитектуре системы или сети, а затем данная архитектура «настраивалась» на решение конкретной задачи. Реализуются связи между программными модулями, задается режим использования ресурсов, причем последовательность операций, реализующих эти действия, задается определенными заранее формальными исполняемыми спецификациями. Развивая данную концепцию, представим четыре уровня представления реальной системы. Виртуальная машина проектировщика позволяет задавать архитектуру с требуемыми свойствами, уровнями параллельности, ресурсами. Виртуальная машина пользователя позволяет вводить необходимое ему описание структуры и настраивать аппаратное и программное обеспечение. Виртуальная машина системного программиста представляет инструментарий для настройки модулей операционной системы и систем управления базами данных и знаний для распределения памяти. Аппаратная виртуальная машина реализует язык низкого уровня для аппаратуры, которым пользуется системный программист. Здесь считаем, что низший уровень – микропрограммный.

2 Концептуальные поведенческие модели и базовый формализм иерархических эволюционирующих алгебраических систем

Рассмотрим подход к построению формализованного описания предметной области методами концептуального поведенческого моделирования, в большей степени пригодными к внутренней интеллектуализации систем и сетей хранения и обработки данных и интегрированными с технологиями, базирующимися на многоуровневых семантических сетях и абстрактных машинах [5–8].

Под моделью обычно подразумевается такое представление системы, которое построено с целью ее последующего изучения. В связи с большим числом проблем, которые нужно разрешить при проектировании системы, естественно строить некоторую иерархию моделей. При многоуровневом проектировании «сверху вниз» верхние уровни модели нередко рассматриваются как концептуальные, иногда «существующие» лишь в умах исследователей. Концептуальные модели играют фундаментальную роль в проектировании систем; в них определяются основные виды отношений между объектами: структуризации, функциональные, причинно-следственные, семантические и др. [9]. Семантика отношений может иметь декларативный или процедурный характер. Иерархические связи, как правило, определяются отношениями структуризации. Реальные системы хранения и обработки данных функционируют в окружающей среде. Для эффективной работы системы должны быть разработаны основные поведенческие процедуры, которые определили бы ее взаимодействие с окружающей средой, с пользователями, с другими системами.

Технологии интеллектуальных систем, основанные на моделях представления знаний, часто базируются на использовании принципов управления, отличных от обычных фон-неймановских. Использование принципа «активные команды – пассивные данные» во многих приложениях нецелесообразно [9]; актуализацию действий логичнее осуществлять на основе знаний. Так, источником активности системы может быть изменение состояния информационной базы, отображающее появление в базе новых фактов, событий, в том числе установление связей между объектами.

Нами рассматриваются поведенческие, или имитационные, модели, которые работают точно так же, как и моделируемые системы: поведение модели во времени соответствует переходам между состояниями в реальной системе. Между определенной таким образом поведенческой моделью и реальной системой существует концептуальное соответствие. Концептуальное проектирование поведенческой модели следует начать с построения статической информационной модели системы, не зависящей от физических условий реализации. Под физической реализацией подразумевается состав реализующих программ, используемый язык программирования, аппаратная платформа, вопросы производительности. Однако при выборе в качестве начального формализма многоосновных алгебраических систем возможно сразу определить не только информационную базу, но и поведенческий аспект модели, т.е. ее функционирование во времени, для чего задается счетное множество констант (или нульарных операций), принадлежащих сорту Time (время), предикаты сравнения и арифметические операции для выполнения действий с натуральными числами. На концептуальном уровне выбираются также абстрактные исполнители – агенты, функционирующие в некотором абстрактном пространстве. Как и при проектировании базы данных, в проектировании поведенческой модели можно выделить концептуальный, логический и физический уровни. Использование формальных методов при определении операционной семантики поведенческой модели позволяет сделать логический и физический уровни проектирования «тонкими» в том смысле, что подготовленные на концептуальном уровне и реализуемые на нижних уровнях формальные спецификации можно отнести к классу «непосредственно исполняемых», т.е. реализующие их виртуальные машины просты в разработке и применении.

Определяя иерархическую поведенческую модель, вначале рассмотрим определенную в [10] упорядоченную тройку, или конечную сигнатуру $\Sigma = < P, F, \mu >$, где P – множество предикатных символов; F – множество функциональных символов; μ – отображение арности, или отображение множества $P \cup F$ в множество натуральных чисел ω. Если $X \subseteq P \cup F$, то сигнатура $\Sigma_1 = < P \cap X, F \cap X, \mu \restriction X >$ называется ограничением сигнатуры Σ на множество X и обозначается $\Sigma_1 = \Sigma \restriction X$. Рассмотрим далее пару $\mathfrak{A} = < A, I^{\mathfrak{A}} >$, где $A = \{A_1, A_2, \ldots, A_m\}$ – множество основных множеств (носителей), а $I^{\mathfrak{A}}$ – отображение множества $P \cup F$ в множество предикатов и функций, определенных на основных множествах из множества A, или интерпретация сигнатуры Σ в A [10].

В основу построения концептуальной поведенческой модели положим иерархическое проектирование «сверху вниз». Каждому уровню i ($i = 1, 2, \ldots, n$) поведенческой модели на начальном этапе ее построения поставим в соответствие алгебраическую систему $\mathfrak{A}_i$. В процессе иерархического проектирования при переходе от уровня i к уровню $(i + 1)$ осуществляется «сильное» обогащение многоосновной алгебраической системы $\mathfrak{A}_{i+1}$ сигнатуры Σ_{i+1} системой $\mathfrak{A}_i$ сигнатуры Σ_i, т.е. выполняются следующие условия:

1) $A_i = A_{i+1} \restriction (\mathrm{A}_1^{(i)} \cup \mathrm{A}_2^{(i)} \cup \ldots \cup \mathrm{A}_{m_i}^{(i)})$;

2) $\Sigma_i = \Sigma_{i+1} \restriction (P_i \cup F_i)$;

3) $I^{\mathfrak{A}_i} = I^{\mathfrak{A}_{i+1}} \restriction (P_i \cup F_i)$,

где применение операции $\restriction$ в условии 1 к множеству основ A_{i+1} и к объединению $\mathrm{A}_1^{(i)} \cup \mathrm{A}_2^{(i)} \cup ... \cup \mathrm{A}_{m_i}^{(i)}$ основных множеств на i-м уровне иерархии означает, что

$$A_i = \{\mathrm{A}_1^{(i+1)} \cap \mathrm{A}_1^{(i)}, \mathrm{A}_2^{(i+1)} \cap \mathrm{A}_2^{(i)}, ..., \mathrm{A}_{m_i}^{(i+1)} \cap \mathrm{A}_{m_i}^{(i)}\},$$

а остальные основные множества $\mathrm{A}_{m_i+1}^{(i+1)}, \mathrm{A}_{m_i+2}^{(i+1)}, ..., \mathrm{A}_{m_{i+1}}^{(i+1)}$, возможно добавленные для системы $\mathfrak{A}_{i+1}$ и не определенные для системы $\mathfrak{A}_i$, в данной операции не участвуют. Условия 2 и 3 соответствуют принятым в [10].

Введенное нами понятие «сильного» обогащения алгебраической системы означает, что в процессе построения модели по методике «сверху вниз» происходит поэтапное добавление новых элементов к основным множествам, а также расширение состава самих основных множеств (при $m_{i+1} \geq m_i$, $i = 1, 2, \ldots, n - 1$), что сопровождается обогащением сигнатуры алгебраической системы данного уровня новыми предикатными и функциональными символами, а также их новой интерпретацией.

Зафиксируем иерархию (направленное множество) алгебраических систем

$$\mathfrak{A} = < \mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_n >$$

с соответствующими:

– вектором сигнатур:

$$\boldsymbol{\Sigma} = < \Sigma_1, \Sigma_2, \ldots, \Sigma_n >;$$

– вектором множеств основ:

$$\boldsymbol{A} = < A_1, A_2, \ldots, A_n >;$$

– вектором интерпретаций сигнатур:

$$\boldsymbol{I}^{\mathfrak{A}} = < I^{\mathfrak{A}_1}, I^{\mathfrak{A}_2}, \ldots, I^{\mathfrak{A}_n} >.$$

От направленного множества алгебраических систем, определенного в [10], данное множество отличается тем, что здесь все системы имеют различную сигнатуру, причем мощность каждой последующей системы больше или равна предыдущей. Иерархическая система $\mathfrak{A}$ функционирует, переходя от одного вектора $\boldsymbol{I}^{\mathfrak{A}}(t_i)$ интерпретаций сигнатур к другому вектору $\boldsymbol{I}^{\mathfrak{A}}(t_{i+1})$, $t_{i+1} > t_i$, при этом текущий вектор интерпретаций сигнатур $\boldsymbol{I}^{\mathfrak{A}}(t)$ соответствует текущему абстрактному состоянию иерархии алгебраических систем; на каждом уровне элементы вектора интерпретации сигнатур изменяются согласованно друг с другом). Все уровни модели при этом остаются различимыми (видимыми) для пользователя, что позволяет в дальнейшем построить иерархию сети абстрактных машин и затем перейти к ее реализации иерархической сетью виртуальных машин.

Принято различать проектирование иерархической модели и собственно иерархическое проектирование. Иерархии моделей некоторой системы

соответствуют различным способам деления ее на компоненты, а иерархическое проектирование связано с пошаговым уточнением модели. Однако при использовании одних и тех же формализмов – сценарных сетей или сетей абстрактных машин – может оказаться полезным на этапах иерархического проектирования «зафиксировать» построенные модели в том смысле, что они будут соответствовать некоторому «полезному» уровню абстракции и останутся различимыми для проектировщика. Пользователь, как обычно, может иметь дело лишь с верхним уровнем абстракции.

Зафиксированную при этом сигнатуру и установленные правила модификации ее интерпретации удобно использовать при определении операционной семантики сети абстрактных машин данного уровня. В результате формируется иерархия сетей абстрактных машин. Следовательно, формализм, основанный на использовании определенной нами иерархии многоосновных алгебраических систем, пригоден одновременно для обоих видов проектирования. Поскольку при задании формализма учтены изменения вектора интерпретаций сигнатур каждого уровня во времени, определенную таким образом иерархию многоосновных алгебраических систем назовем «эволюционирующей».

В нашем случае работу моделируемой системы целесообразно описать системой логико-алгебраических выражений, по которым в дальнейшем прослеживается ее эволюция. В данную систему выражений, составляющих описание имитационной модели, необходимо включать также выражения для логических условий, при которых возникают те или иные переходы в моделируемой системе. Применение методов имитационного моделирования в общем случае далее связано с исследованием хронологической последовательности событий: захвата и освобождения ресурсов, поступления запросов и окончания их обработки и др. Структура проектируемой системы «выводится» непосредственно из проектных спецификаций, которые целесообразно построить по иерархическому принципу. В процессе проектирования спецификации поэтапно уточняются в соответствии с определенными уровнями абстракции.

3 Иерархические сети абстрактных машин

Из описания сетей абстрактных машин СеАМ, данного в работах [5, 6], следует, что на основе применения алгебры модулей можно строить произвольные иерархические сети. Рассмотрим пример иерархии абстрактных машин, входящих в сеть N_0, который представлен на рис. 1. В имени произвольной подсети N_{ij} первый индекс указывает на уровень иерархии, на котором она находится, а второй индекс является номером подсети на данном уровне. Дуги указывают на связь подсетей по управлению, построенному также по иерархическому принципу. Пространство функций и предикатов FS здесь структурировано и представлено возможно пересекающимися (для организации иерархических каузальных связей) подпространствами FS_i, $i = 0, 1, 2, \ldots, 5$.

Детальнее иерархические связи в сети N_0 показаны на рис. 2. Стрелками на данных рисунках обозначены виртуальные связи; реальные же физические связи реализуются через смежные уровни структурированного FS-пространства. Описание каждой подсети может включать как обычные, так и иерархические символы для обозначения модулей. Иерархические символы обозна-

чают целые фрагменты иерархической сети. Подсети, как обычно в иерархических системах, различаются по степени детализации.

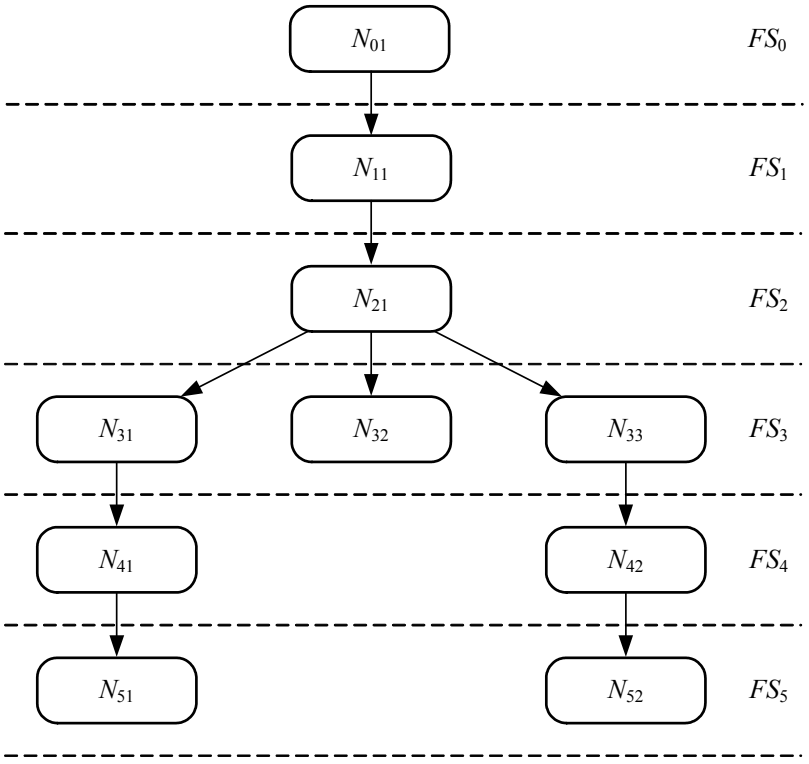

Рис. 1 Иерархия сетей абстрактных машин (иерархическая сеть N_0)

Отметим, что подсети не являются макроопределениями – это реально функционирующие и запускаемые с верхних уровней иерархии части общей иерархической сети, которые разделяют общее FS-пространство функций и предикатов (точнее, информационных объектов, представляющих данные функции и предикаты в общем информационном пространстве).

В иерархической сети возможна ситуация, когда несколько модулей, конкурируя, могут пытаться инициировать работу одной и той же подсети (рис. 3). Такие модули назовем конкурирующими. Разрешение конфликтов в сетях СеАМ осуществляется с помощью известных методов, описанных, например, в работе [11], или в рамках *FS*-технологии путем блокировки стартовых функций и предикатов подсети перед ее запуском.

Другой возможностью разрешения конфликтных ситуаций является введение приоритетов между агентами-серверами, интерпретирующими логико-алгебраические выражения, составленные для модулей, и рассматривающими их в качестве сценариев своей работы.

Во многих случаях удается избежать значительного числа блокировок функций и предикатов в *FS*-пространстве: достаточно на этапе подготовки некоторого модуля к выполнению «захватить» только такие функции или предикаты, посредством которых организуется каузальная связь данного модуля с другими. Рисунок 4 иллюстрирует организацию каузальных межмодульных связей в ограниченных «окрестностях» *FS*-пространства. Простую

физическую реализацию данных взаимодействий в мультипроцессорной системе иллюстрирует структура, в которой процессоры разделяют по известной схеме общую многопортовую память (возможно, виртуальную). Косвенные связи между программными модулями реализуются через общее пространство памяти. Для ускорения обменов процессоры могут передавать и принимать сообщения через быстродействующий коммутатор. Процессоры могут иметь собственную локальную память. Общая память может содержать как быстродействующие модули основной памяти, так и внешнюю память. Вся система, включая общую память, может быть построена в виде составной вычислительной сети.

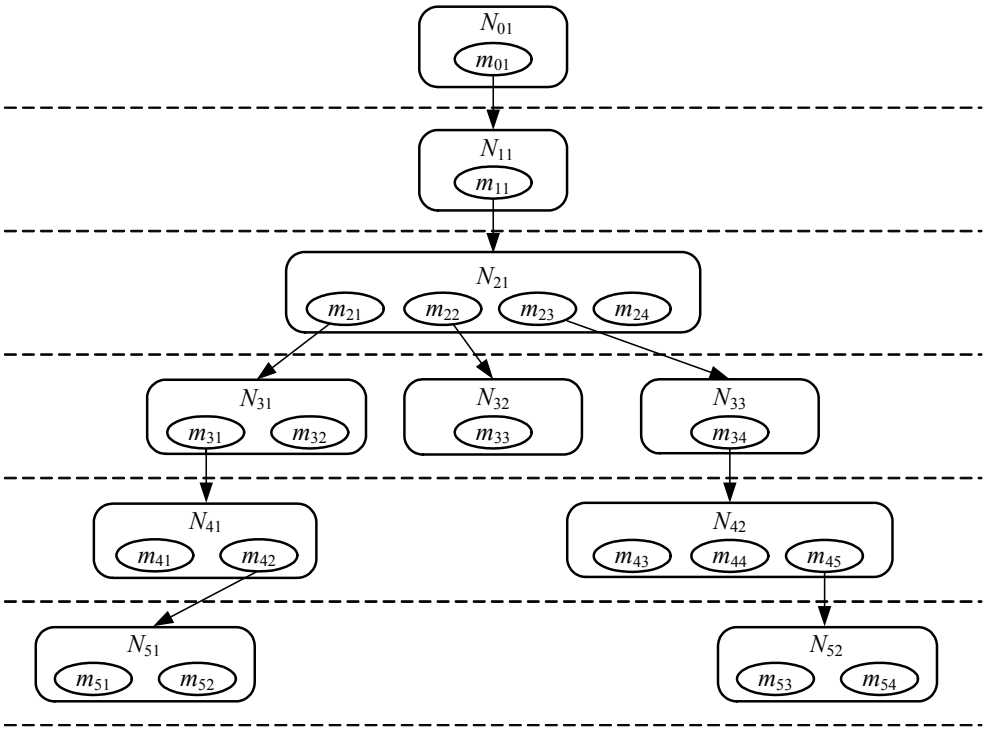

Рис. 2 Иерархические связи в сети абстрактных машин N0

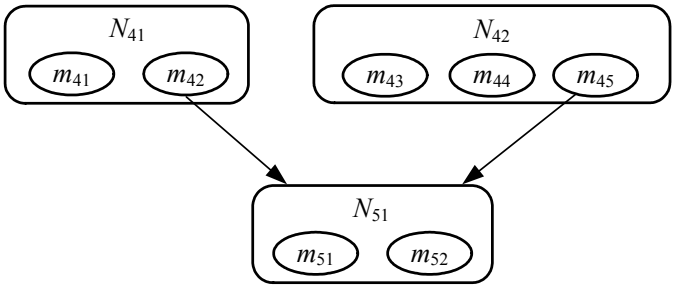

Рис. 3 Пример фрагмента иерархической сети с конфликтующими модулями

Активизация некоторой подсети N_{ij} производится после выполнения правила обновления предиката $P_{S\_ij}(N_{ij}) \leftarrow$ true, при этом активизируется соответствующий агент-сервер λ_{ij} и разблокируются локальные функции и пре-

дикаты, составляющие локальное LFS_{ij}-пространство данной подсети. Далее агент-сервер λ_{ij} блокирует локальное LFS_{ij}-пространство и в подсети N_{ij} выполняются каузально связанные модули. По завершении работы подсети N_{ij} ее локальное пространство разблокируется. Локальные пространства подсетей могут пересекаться. Вместе с тем, по аналогии с распределенным программированием, для каждой подсети иерархической сети могут быть определены локальные функции и предикаты и определены правила их видимости на нижних уровнях иерархии.

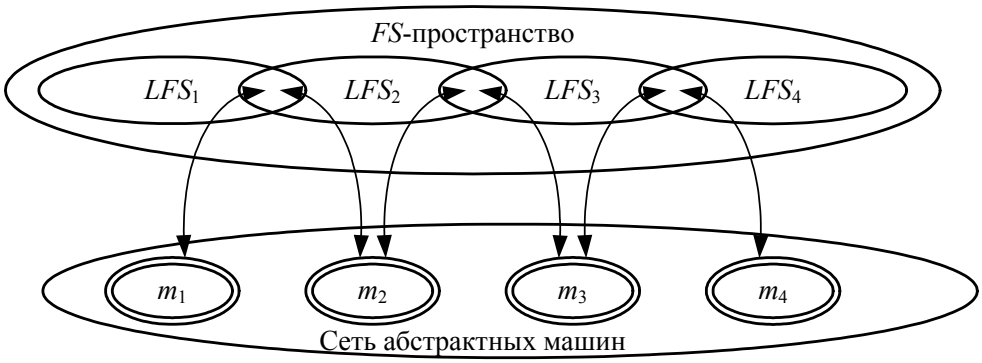

Рис. 4 Межмодульные каузальные связи
в ограниченных окрестностях *FS*-пространства

Для каждой подсети определим n_{ij}-арный предикат P_{N_ij}; его арность равна числу включенных в подсеть N_{ij} модулей. Модули m_1, m_2, …, m_{nij} считаются входящими в подсеть N_{ij} после того, как будет выполнено обновление $P_{N\_ij}(m_1, m_2, \ldots, m_{nij}) \leftarrow$ true данного предиката. С помощью унарных предикатов r задаются межмодульные каузальные связи в подсетях.

Для каждой подсети введем также унарную функцию F_{N_ij}. В результате обновления данной функции $F_{N\_ij}(m_k) \leftarrow N_{qr}$ имя модуля m_k связывается с именем подсети N_{qr}; таким образом, в иерархической сети реализуются связи по управлению, показанные на рис. 2. Символ N_{ij} при этом становится иерархическим. В отличие от многих других иерархических сетей, определенная нами иерархическая СеАМ (ИСеАМ) имеет динамическую структуру, которая может настраиваться и изменяться в процессе функционирования. Например, после выполнения правила обновления унарного предиката $P_{S\_ij}(N_{ij}) \leftarrow$ false подсеть N_{ij} становится пассивной и тем самым исключается из иерархии подсетей вместе с управляемыми ею подсетями на нижних уровнях иерархии.

Выражения для модулей, составляющих представленную на рис. 1, 2 иерархическую сеть СеАМ, представим в следующем виде:

$$m_0(\lambda_0) = [\alpha_{start}(\lambda_0)](\{P_{S\_11}(N_{11}) \leftarrow \text{true}, P_{N\_11}(m_{11}) \leftarrow \text{true}, F_{N\_21}(\text{m}_{11}) \leftarrow N_{11}\} \vee R^E);$$

$$m_{11}(\lambda_{11}) = [r_1(\lambda_{11})](\{r_1(\lambda_{11}) \leftarrow \text{false}, P_{S\_21}(N_{21}) \leftarrow \text{true},$$
$$P_{N\_21}(m_{21}, m_{22}, m_{23}, m_{24}) \leftarrow \text{true}, F_{N\_21}(m_{11}) \leftarrow N_{21}, r_2(\lambda_{11}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{21}(\lambda_{21}) = [r_3(\lambda_{21})](\{r_3(\lambda_{21}) \leftarrow \text{false}, P_{S\_31}(N_{31}) \leftarrow \text{true},$$
$$P_{N\_31}(m_{31}, m_{32}) \leftarrow \text{true}, F_{N\_31}(m_{21}) \leftarrow N_{31}, r_4(\lambda_{21}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{22}(\lambda_{21}) = [r_4(\lambda_{21})](\{r_4(\lambda_{21}) \leftarrow \text{false}, P_{S\_32}(N_{32}) \leftarrow \text{true}, P_{N\_32}(m_{33}) \leftarrow \text{true}, F_{N\_32}(m_{22}) \leftarrow N_{32}, r_5(\lambda_{21}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{23}(\lambda_{22}) = [r_7(\lambda_{22})](\{r_7(\lambda_{22}) \leftarrow \text{false}, P_{S\_33}(N_{33}) \leftarrow \text{true}, P_{N\_33}(m_{34}) \leftarrow \text{true}, F_{N\_33}(m_{23}) \leftarrow N_{33}, r_8(\lambda_{22}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{24}(\lambda_{21}) = [r_5(\lambda_{21}) \& r_8(\lambda_{22})](\{r_5(\lambda_{21}) \leftarrow \text{false}, r_8(\lambda_{22}) \leftarrow \text{false}, r_6(\lambda_{21}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{31}(\lambda_{31}) = [r_9(\lambda_{31})](\{r_9(\lambda_{31}) \leftarrow \text{false}, P_{S\_41}(N_{41}) \leftarrow \text{true}, P_{N\_41}(m_{41}, m_{42}) \leftarrow \text{true}, F_{N\_41}(m_{31}) \leftarrow N_{41}, r_{10}(\lambda_{31}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{32}(\lambda_{31}) = [r_9(\lambda_{31})](\{r_9(\lambda_{31}) \leftarrow \text{false}, r_{10}(\lambda_{31}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{33}(\lambda_{32}) = [r_{11}(\lambda_{32})](\{r_{11}(\lambda_{32}) \leftarrow \text{false}, r_{12}(\lambda_{32}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{34}(\lambda_{33}) = [r_{13}(\lambda_{33})](\{r_{13}(\lambda_{33}) \leftarrow \text{false}, P_{S\_33}(N_{33}) \leftarrow \text{true}, P_{N\_42}(m_{43}, m_{44}, m_{45}) \leftarrow \text{true}, F_{N\_42}(m_{34}) \leftarrow N_{42}, r_{14}(\lambda_{33}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{41}(\lambda_{41}) = [r_{15}(\lambda_{41})](\{r_{15}(\lambda_{41}) \leftarrow \text{false}, r_{16}(\lambda_{41}) \leftarrow \text{true}, r_{17}(\lambda_{42}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{42}(\lambda_{41}) = [r_{16}(\lambda_{41}) \& r_{17}(\lambda_{42})](\{r_{16}(\lambda_{41}) \leftarrow \text{false}, r_{17}(\lambda_{42}) \leftarrow \text{false}, P_{S\_51}(N_{51}) \leftarrow \text{true}, P_{N\_51}(m_{51}, m_{52}) \leftarrow \text{true}, F_{N\_51}(m_{42}) \leftarrow N_{51}, r_{18}(\lambda_{41}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{43}(\lambda_{43}) = [r_{19}(\lambda_{43})](\{r_{19}(\lambda_{43}) \leftarrow \text{false}, r_{20}(\lambda_{43}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{44}(\lambda_{43}) = [r_{19}(\lambda_{43})](\{r_{19}(\lambda_{43}) \leftarrow \text{false}, r_{20}(\lambda_{43}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{45}(\lambda_{43}) = [r_{19}(\lambda_{43})](\{r_{19}(\lambda_{43}) \leftarrow \text{false}, P_{S\_52}(N_{52}) \leftarrow \text{true}, P_{N\_52}(m_{53}, m_{54}) \leftarrow \text{true}, F_{N\_52}(m_{45}) \leftarrow N_{52}, r_{20}(\lambda_{43}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{51}(\lambda_{51}) = [r_{21}(\lambda_{51})](\{r_{21}(\lambda_{51}) \leftarrow \text{false}, r_{22}(\lambda_{51}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{52}(\lambda_{51}) = [r_{21}(\lambda_{51})](\{r_{21}(\lambda_{51}) \leftarrow \text{false}, r_{22}(\lambda_{51}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{53}(\lambda_{52}) = [r_{23}(\lambda_{52})](\{r_{23}(\lambda_{52}) \leftarrow \text{false}, r_{24}(\lambda_{52}) \leftarrow \text{true}, r_{25}(\lambda_{53}) \leftarrow \text{true}\} \vee R^E);$$

$$m_{54}(\lambda_{52}) = [r_{24}(\lambda_{52}) \& r_{25}(\lambda_{53})](\{r_{24}(\lambda_{52}) \leftarrow \text{false}, r_{25}(\lambda_{53}) \leftarrow \text{false}, r_{26}(\lambda_{52}) \leftarrow \text{true}\} \vee R^E).$$

Запись вида $m_{ij}(\lambda_{pq})$ здесь означает, что модуль m_{ij} интерпретируется агентом-сервером λ_{pq} и используется вместо правила обновления $f_{\text{start}}(m_{ij}) \leftarrow \lambda_{pq}$ стартовой функции f_{start}. Каждое из данных выражений для модулей построено на основе операции α-дизъюнкции; в выражениях используется символ R^E тождественного или пустого правила обновления. Работа иерархической сети начинается с того момента, как станет истинным высказывание $\alpha_{\text{start}}(\lambda_0)$. Затем выполняется принадлежащий подсети N_{01} высшего (нулевого) уровня стартовый модуль m_{01}, инициирующий подсеть N_{11} первого уровня. Затем инициируется работа остальных подсетей и составляющих их модулей. Для простоты здесь не применяются и не описываются операции, связанные с запуском процедур агентов-демонов, выполняющих те операции, ради которых и построена иерархическая сеть. Отметим также, что каждому модулю можно было бы назначить по «собственному» агенту-серверу, что в большей степени соответствовало бы концепции управления множеством продукций как независимыми модулями. Однако для повышения эффективности функционирования сети разработчик может на основе анализа межмодульных каузальных

связей использовать меньшее число агентов, учитывая тот факт, что ряд модулей выполняется последовательно и может быть выполнен одним и тем же агентом-сервером.

Рассмотрим подробнее построение модулей иерархической сети N_0. Иерархические сети абстрактных машин естественным образом формализуются в терминах и понятиях, похожих на термины и понятия систем алгоритмических алгебр [12]. В алгебре модулей абстрактных машин зафиксирована система образующих, включающая множество правил элементарных обновлений $\{r_\sigma, \sigma \in \Sigma\}$ функций и предикатов с символами из множества Σ и атомарные формулы (атомы). Система операций, принимающих значения в множестве модулей абстрактных машин, содержит операции α-дизъюнкции, α-итерации из алгебры алгоритмов Глушкова, а также выбранный нами набор темпоральных и пространственных операций. Логические условия формируются по обычным в логике предикатов первого и высших порядков правилам; кроме того, предполагается, что логические условия всюду определены. Отметим, что, в отличие от алгебры модулей СеАМ и РСеАМ [5, 6], в алгебре сценариев [7, 8] система образующих содержит вместо правил элементарных обновлений элементарные сценарии, или сценарии низшего уровня.

В качестве примера рассмотрим построение одного из модулей, например: m_{11} иерархической сети СеАМ N_0 на базе суперпозиции операций «|» (выполнить, возможно, одновременно) и α-дизъюнкции (для удобства данную тернарную операцию будем представлять также и в префиксной форме записи, обозначая ее символом $\mathrm{Op}_\vee$). Следующая последовательность шагов позволяет сформировать модуль m_{11} иерархической сети абстрактных машин:

$$a_1 = r_1(\lambda_{11}) \leftarrow \text{false},$$

$$a_2 = a_1 \mid (P_{S\_21}(N_{21}) \leftarrow \text{true}),$$

$$a_3 = a_2 \mid (P_{N\_21}(m_{21}, m_{22}, m_{23}, m_{24}) \leftarrow \text{true}),$$

$$a_4 = a_3 \mid (F_{N\_21}(\lambda_{21}) \leftarrow N_{21}),$$

$$a_5 = a_4 \mid (r_2(\lambda_{11}) \leftarrow \text{true}),$$

$$m_{11} = \mathrm{Op}_\vee(r_1(\lambda_{11}), a_5, R^E) = [r_1(\lambda_{11})](a_5 \vee R^E) = [r_1(\lambda_{11})](\ \{r_1(\lambda_{11}) \leftarrow \text{false},$$
$$P_{S\_21}(N_{21}) \leftarrow \text{true}, P_{N\_21}(m_{21}, m_{22}, m_{23}, m_{24}) \leftarrow \text{true},$$
$$F_{N\_21}(m_{11}) \leftarrow N_{21}, r_2(\lambda_{11}) \leftarrow \text{true}\} \vee R^E.$$

Правила элементарных обновлений предикатов и функций

$$r_1(\lambda_{11}) \leftarrow \text{false}, P_{S\_21}(N_{21}) \leftarrow \text{true}, P_{N\_21}(m_{21}, m_{22}, m_{23}, m_{24}) \leftarrow \text{true},$$
$$F_{N\_21}(m_{11}) \leftarrow N_{21}, r_2(\lambda_{11}) \leftarrow \text{true}$$

и атомарное высказывание $r_1(\lambda_{11})$ входят в систему образующих, с помощью которых посредством суперпозиции операций $\mathrm{Op}_\vee$, $\mathrm{Op}_{\{\}}$, темпоральных $\mathrm{Op}_{\mathrm{temporal}}$ и пространственных $\mathrm{Op}_{\mathrm{space}}$ операций, а также логических операций порождаются модули сети абстрактных машин. Модуль m_{11} функционирует следующим образом. Если $r_1(\lambda_{11}) = \text{true}$, то далее в модуле выполняется блок согласованных правил обновлений предикатов и функций. Правило обновления унарного предиката $r_1(\lambda_{11}) \leftarrow \text{false}$ в дальнейшем препятствует повторному выполнению данного модуля, правило $P_{S\_21}(N_{21}) \leftarrow \text{true}$ инициии-

рует подсеть N_{21}, правило $P_{N\_21}(m_{21}, m_{22}, m_{23}, m_{24}) \leftarrow$ true включает с подсеть N_{21} модули m_{21}, m_{22}, m_{23} и m_{24}. Правило обновления унарной функции $F_{N\_21}(m_{11}) \leftarrow N_{21}$ связывает модуль m_{11} с подсетью N_{21}, а правило $r_2(\lambda_{11}) \leftarrow$ true является завершающим. Для построения всех модулей рассматриваемой иерархической сети N_0 достаточно двух многократно используемых операций $Op_\vee$ и «|». Аналогично строятся также и иерархические сети РСеАМ, основанные на использовании расширенного темпоральными операциями аппарата СеАМ. Таким образом, на конкретном примере продемонстрировано построение сложной иерархической мультиагентной сети абстрактных машин, которая далее реализуется иерархической виртуальной сетью в реальной физической среде вычислительной сети.

Заключение

Достоинством предложенного подхода как методологического инструмента является то, что логико-алгебраические методы используются здесь на всех уровнях иерархии сети, а также при описании и последующей реализации управления работой сети. Обеспечивается поддержка обеих стратегий иерархического проектирования – нисходящей и восходящей. Полученная в результате программно реализованная поведенческая или имитационная модель практически ничем не отличается от распределенного приложения или распределенной управляющей программы, выполняющей функции распределенной операционной системы сети хранения и обработки данных.

Несмотря на то, что существует большое разнообразие видов внутреннего параллелизма, логико-алгебраический подход к описанию происходящих в вычислительной системе процессов на основе исполняемых формальных спецификаций позволяет унифицировать как описание, так и реализацию сетей алгоритмических модулей на основных уровнях сети виртуальных машин.

Список литературы

1. **Хэндлер, В.** Простота и гибкость в архитектуре параллельных вычислительных систем / В. Хэндлер // Высокоскоростные вычисления. Архитектура, производительность, прикладные алгоритмы и программы суперЭВМ / под ред. Я. Ковалика. – М. : Радио и связь, 1988. – С. 61–79.
2. **Брой, М.** Информатика. Основополагающее введение : в 4-х ч. / М. Брой. – М. : Диалог-МИФИ, 1996. – Ч. 1. – 300 с.
3. **Gurevich Y.** Evolving Algebras 1993: Lipari Guide, Specification and Validation Methods / ed. E. Bцrger. – Oxford : Oxford University Press, 1995. – P. 9–36.
4. **Dexter, S.** Gurevich abstract state machines and Shцnhage storage modification machines / S. Dexter, P. Doyle, Y. Gurevich // Journal of Universal Comp. Science. – 1997. – V. 3. – № 4. – P. 279–303.
5. **Зинкин, С. А.** Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (базовый формализм и его расширения) / С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 3. – С. 13–22.
6. **Зинкин, С. А.** Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (механизмы интерпретации и варианты использования) / С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 4. – С. 37–51.
7. **Зинкин, С. А.** Самомодифицируемые сценарные модели функционирования систем и сетей хранения и обработки данных (базовый формализм и темпораль-

ные операции) / С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 1. – С. 3–12.
8. **Зинкин, С. А.** Самомодифицируемые сценарные модели функционирования систем и сетей хранения и обработки данных (реализация и свойства сценарных моделей) / С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 2. – С. 13–21.
9. Искусственный интеллект : в 3-х кн. Кн. 2. Модели и методы : справочник / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 304 с.
10. **Ершов, Ю. Л.** Математическая логика / Ю. Л. Ершов, Е. А. Палютин. – М. : Наука, 1979. – 320 с.
11. **Таненбаум, Э.** Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стен. – СПб. : Питер, 2003. – 878 с.
12. **Капитонова, Ю. В.** Математическая теория проектирования вычислительных систем / Ю. В. Капитонова, А. А. Летичевский. – М. : Наука, 1988. – 296 с.

Зинкин Сергей Александрович
кандидат технических наук, доцент,
кафедра вычислительной техники,
Пензенский государственный
университет

Zinkin Sergey Alexandrovich
Candidate of technical sciences,
associate professor, sub-department
of computer science, Penza State University

УДК 681.324

Зинкин, С. А.

Иерархические сети абстрактных машин и виртуализация интеллектуальных систем внешнего хранения и обработки данных / С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2009. – № 2 (10). – С. 25–38.