УДК 681.324

В. И. Волчихин, А. В. Дубравин, С. А. Зинкин

АБСТРАКТНЫЙ И СТРУКТУРНЫЙ СИНТЕЗ СИСТЕМ ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ ПАРАДИГМЫ ПЕРЕДАЧИ СООБЩЕНИЙ В РАСПРЕДЕЛЕННОМ ОКРУЖЕНИИ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ[1]

Аннотация.

Актуальность и цели. Объектом исследования являются распределенные системы обработки данных, работа которых основана на принципах, названных в статье парадигмами. Рассмотрена парадигма, используемая для проектирования программного обеспечения промежуточного слоя, ориентированного на обмен сообщениями в распределенном окружении. Предметом исследования являются вопросы абстрактного и структурного проектирования распределенных сетевых приложений на основе логико-алгебраического подхода и некоторых методов искусственного интеллекта. Цель работы – совершенствование методов проектирования распределенных приложений на основе концептуальных, логических и логико-алгебраических моделей, положенных в основу технологии распределенного программирования в сетях ЭВМ, основанной на передаче сообщений.

Материалы и методы. Предлагаемые методы позволяют разрабатывать распределенные приложения для обработки данных, соответствующие некоторой формальной спецификации. Для реализации распределенных приложений обработки данных выбраны методы, не требующие специальных синтаксических примитивов при организации параллелизма в распределенных системах. Описание и поддержка параллелизма в распределенных системах осуществляется средствами, реализуемыми на основе перехода от первоначальных концептуальных представлений процессов, базирующихся на правилах вывода и концептуальных графов, к непосредственному программированию путем прямого использования логико-алгебраических выражений в качестве формализованных спецификаций.

Результаты и выводы. Предложенные новые концептуальные, логические и логико-алгебраические модели распределенных вычислений в системах с передачей сообщений отличаются от известных тем, что они относятся к классу непосредственно исполнимых (реализуемых), применение которых позволяет снизить трудозатраты при создании распределенных сетевых приложений.

Ключевые слова: распределенная обработка данных, сетевое окружение, концептуальные, логические и логико-алгебраические модели, парадигма обмена сообщениями, распределенное окружение, управляющие и функциональные межмодульные связи.

V. I. Volchikhin, A. V. Dubravin, S. A. Zinkin

ABSTRACT AND STRUCTURAL SYNTHESIS OF DATA PROCESSING SYSTEMS ON THE BASIS OF THE PARADIGM

[1] Работа выполнена в рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014–2020 годы» (Соглашение № 14.574.21.0045 от 19.06.14, UIN: RFMEFI57414X0045).

OF MESSAGE TRANSMISSION IN THE DISTRIBUTED ENVIRONMENT OF A COMPUTER NETWORK

Abstract.

Background. The research object is distributed systems of data processing based on the principles, designated as “paradigms” in the article. The considered paradigm, used at designing of intermediate software, oriented towards messaging in the distributed environment. The research subject is the problems of abstract and structural design of distributed network applications based on the logic-algebraic approach and some methods of artificial intelligence. The aim of the work is to improve the design methods of distributed applications on the basis of conceptual, logic and logic-algebraic models, forming the foundation of the technique of distributed programming in computer networks based on message transmission.

Materials and methods. The suggested methods allow to develop distributed applications for data processing, corresponding to a certain specifications. For realization of distributed applications of data processing the authors chose methods that require no special syntactical primitive elements when organizing parallelism in distributed systems. Description and maintenance of parallelism in distributed systems are carried out by means based on the transition from initial conceptual process representations, based on input rules and conceptual graphs, to immediate programming through direct usage of logic-algebraic expressions as formalized specifications.

Results and conclusions. The suggested new conceptual, logic and logic-algebraic models of distributed calculations in systems with message transmission differ from the knows ones by the relation to a class of the immediately executable (realized), implementation of which allows to reduce working hours at creation of distributed network applications.

Key words: distributed data processing, network environment, conceptual, logic and logic-algebraic models, paradigm of messaging, distributed environment, control and functional intermodule communications.

Введение

Настоящая работа является продолжением работы [1], в которой были предложены концептуальные и логико-алгебраические модели для двух *Linda*-подобных парадигм распределенных вычислений. Под парадигмой подразумевалась та или иная концепция, выбранная для программного решения, например: принцип организации связей между процессами, способ именования и синхронизации процессов, подход к распределению объектов и согласованию их функционирования.

В распределенном программировании известны, например, агентно-ориентированная парадигма, парадигма классной доски, парадигма пространства кортежей и отношений и др. Ранее отмечалось, что в работах [2, 3] использованы неформализованные архитектурные подходы к параллельному и распределенному программированию с акцентом на определении естественного параллелизма в самих задачах, что отражается в программных моделях решений. В настоящей работе рассматривается проектирование программного обеспечения промежуточного уровня, ориентированного на обмен сообщениями в распределенном окружении. Подобное программное обеспечение в англоязычной литературе получило название *Message-Oriented Middleware* (*MOM*). Как и ранее, в работе [1] предлагаемый архитектурный подход осно-

ван на представлении структурных связей между компонентами, предикатами и функциями, а вычислительные процессы представляются каузально связанными модулями, работа которых задается выражениями в логике предикатов первого порядка.

1. Представление знаний эксперта-программиста концептуальными графами

Согласно работам [4, 5] «концептуальный граф представляет логическую формулу. Имена и аргументы предикатов представлены в нем соответственно двумя типами узлов. Дуги графа соединяют имена предикатов с их аргументами». Далее в настоящей работе будут использованы концептуальные графы, представляющие бинарные предикаты. При построении концептуальных графов (КГ) в настоящей работе использована система поддержки и графического представления КГ *CharGer* [6, 7].

Сетевое представление абстрактных понятий (концептов) программирования и отношений между ними в виде концептуального графа приведено на рис. 1. Этот граф построен на основе небольшой модификации ассоциативной сети структуры знаний эксперта-программиста и описания связей между понятиями из работы [8]. Данный пример демонстрирует традиционное использование концептуальных графов в качестве моделей в системах искусственного интеллекта, например, при построении экспертных систем. Особенностью таких моделей является их неполнота, они использовались при исследовании эволюции знаний программистов по мере приобретения опыта.

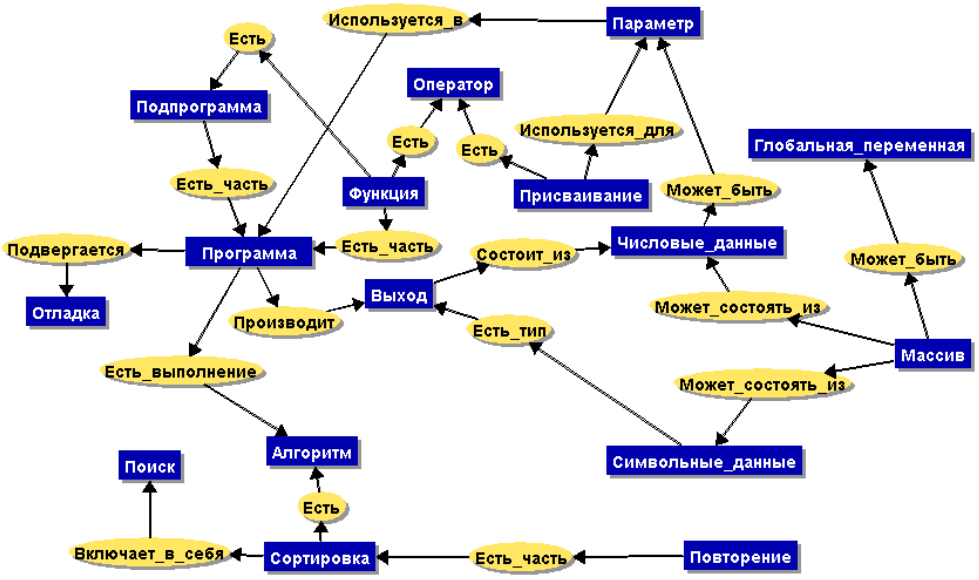


Рис. 1. Концептуальное представление знаний эксперта-программиста

Расширенное концептуальное представление знаний эксперта-программиста приведено на рис. 2. Здесь добавлены концепты и отношения, связанные с представлением алгоритма сортировки-слияния для выполнения на ЭВМ в форме программы на каком-либо алгоритмическом языке. В графе используется специальная вершина-актор, символизирующая выполнение

операции слияния двух заранее отсортированных массивов *a* и *b* в один отсортированный массив *c*.

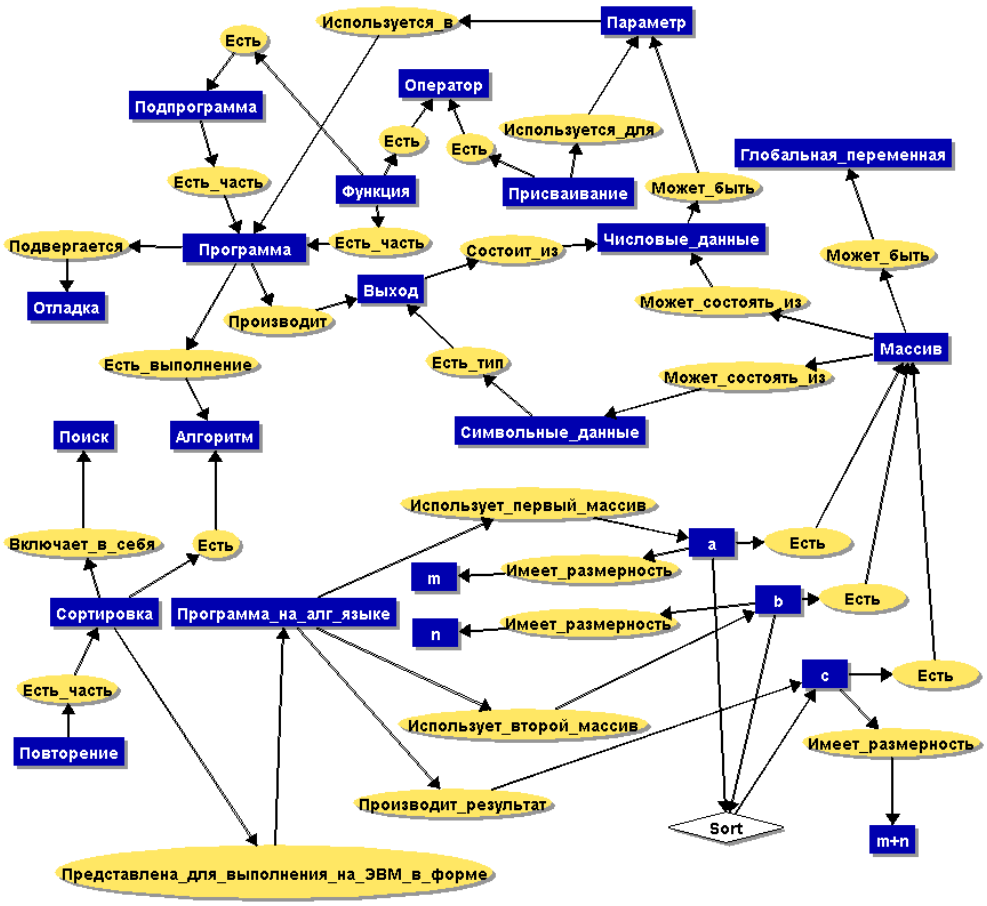


Рис. 2. Расширенное концептуальное представление знаний эксперта-программиста

2. Использование концептуальных графов в качестве математических моделей распределенных программ

Как следует из работы [9], целью построения математических моделей программ является замена сложных реальных объектов более простыми абстрактными моделями; при этом данные модели должны удовлетворять следующим требованиям: при помощи модели можно изучать свойства достаточно широких классов программ, а не отдельных конкретных программ; модель должна сохранять все интересующие исследователя свойства и особенности рассматриваемого класса программ и позволять игнорировать несущественные для данной проблемы свойства, например, синтаксические детали; модель должна быть модифицируемой и допускать нововведения, отслеживающие развитие языков программирования; удобно, если структура модели изобразительно подобна структуре программ, что дает возможность привлекать на некоторых этапах исследований программистскую интуицию.

Концептуальные схемы программ, построенные на основе концептуальных графов, удовлетворяют большинству из перечисленных требований; недостающие требования могут быть учтены путем дополнительного исполь-

зования логико-алгебраических моделей при их соответствующей интерпретации. Данные модели предлагается использовать при проектировании схем распределенных программ, выполняемых в сетевой среде. В основу проектирования должны быть положены логический и логико-алгебраический аппараты. Логические модели при этом могут быть представлены концептуальными графами и дополнительными логическими правилами, а логико-алгебраические модели могут быть представлены сетями абстрактных машин, примененными ранее в предыдущей работе авторов [1].

В дальнейшем при описании предлагаемых методов будет использован пример программы сортировки-слияния массивов, выбранный, например, из работы [10] не в силу его каких-то специфических особенностей, а для иллюстрации практически всех черт предлагаемого в настоящей работе подхода, основанного на последовательном использовании концептуальных, логических и логико-алгебраических моделей распределенных программ. Так, на данном примере будет проиллюстрирован переход от схемы сосредоточенной программы к схеме ее распределенной реализации в виде сети алгоритмических модулей.

Концептуальное представление программы, соответствующей актору Sort на рис. 2 как при сосредоточенной, так и при распределенной реализации операторов S_0, S_1, …, S_9, иллюстрирует рис. 3. Здесь запись вида $+Q(S_i)$ & *Trans* означает, что связь по управлению операторов S_i и S_j существует при истинном составном высказывании $Q(S_i)$ & $Trans(S_i, S_j)$, где значение высказывания $Q(S_i)$ формируется в результате выполнения оператора S_i, а запись $-Q(Si)$ & Trans означает, что связь по управлению операторов S_i и S_k существует при истинном значении составного высказывания $\neg Q(S_i)$ & $Trans(S_i, S_k)$, т.е. при ложном значении $Q(S_i)$. Здесь Q – унарный предикатный символ, а *Trans* – бинарный предикатный символ, S_i, и S_j – имена операторов, или предметные константы. Все атомарные константные формулы вида $Trans(S_i, S_j)$, $S_i \in S$, $S_j \in S$, где S – множество операторов, определяемые концептуальной схемой программы, представленной на рис. 3, предложено использовать при формировании сообщений, передающих управление от одних операторов к другим. Областью истинности предиката *Trans* является одноименное отношение $Trans \subseteq S \times S$ между оператором-предшественником и оператором-следователем. Такие же обозначения использовались для других парадигм распределенных вычислений в работе [1].

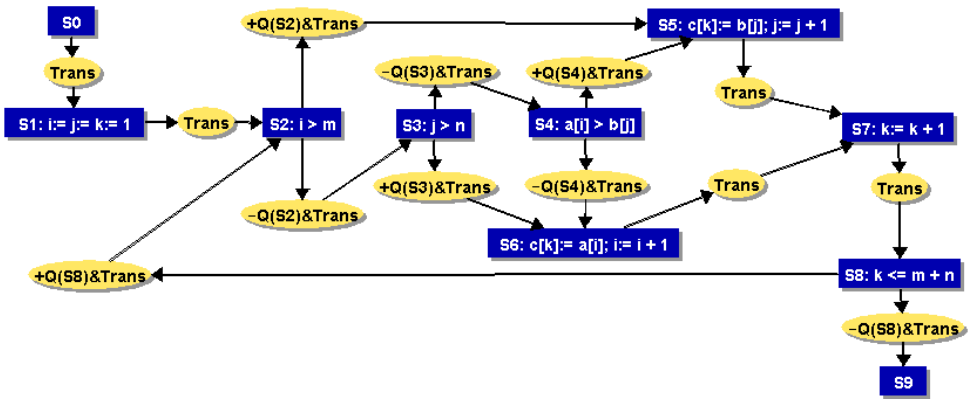


Рис. 3. Пример концептуальной схемы программы

Управляющие связи, соответствующие отношению *Trans*, предлагается реализовывать путем передачи специальных управляющих сообщений между узлами вычислительной сети, на которой размещены программные модули, реализующие операторы S_0, S_1, …, S_9. Для реализации дополнительных функциональных межмодульных связей на основании концептуальной схемы программы построим совмещенный управляющий и функциональный граф распределенной программы (рис. 4).

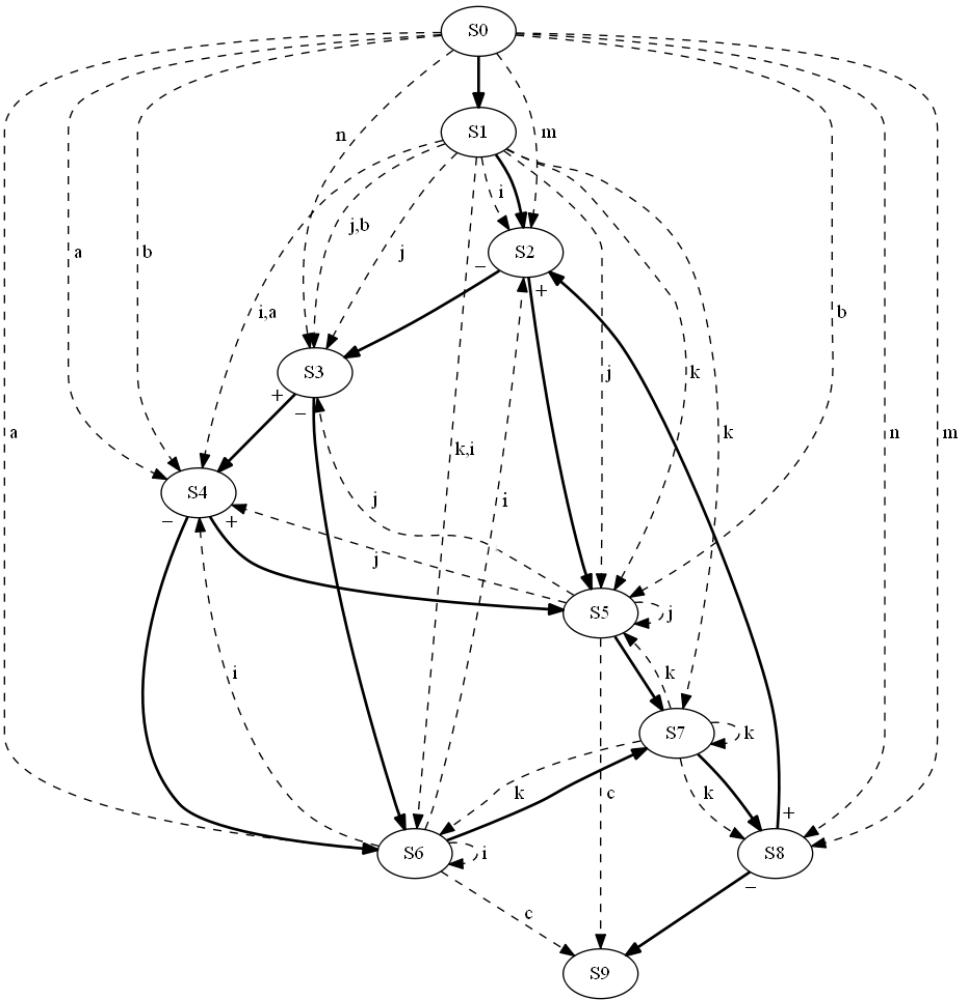


Рис. 4. Совмещенный управляющий и функциональный граф распределенной программы

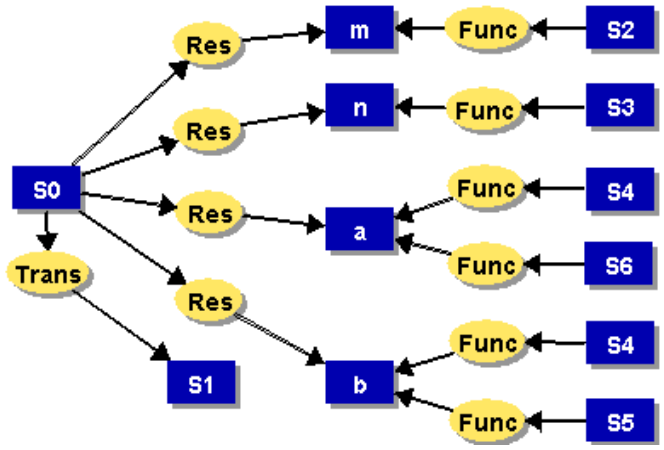
Сплошные дуги в графе программы на рис. 4 соответствуют управляющим связям, или связям операторов по управлению, а пунктирные – функциональным связям, или связям по данным. Вершины в графе отождествлены с функциональными операторами.

При распределенной реализации программы управляющие и функциональные связи реализуются при помощи соответствующих управляющих и функциональных сообщений, передаваемых между узлами вычислительной сети, на которых размещены программные модули, реализующие функциональ-

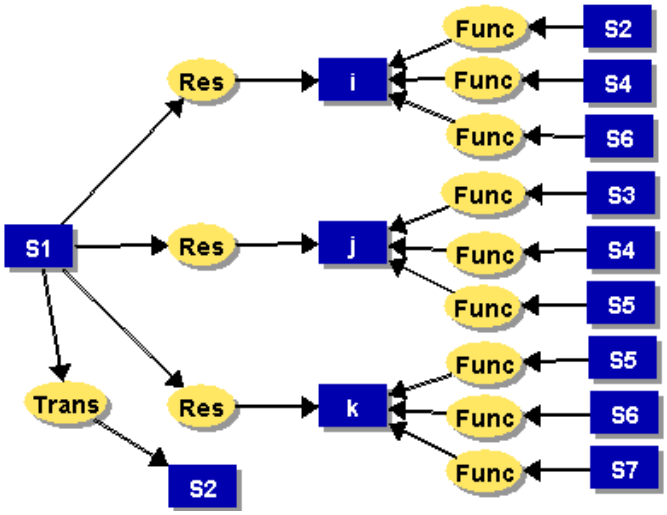
ные операторы. На этапе разработки распределенного приложения предварительно формируются два отношения *Trans* и *Data*, представляющие собой области истинности одноименных предикатов. Бинарное отношение *Trans* необходимо для последующего формирования управляющих сообщений, а тернарное отношение *Data* – для формирования функциональных сообщений. При формировании отношения *Trans* учитываются сплошные дуги, а при формировании отношения *Data* – пунктирные дуги и их веса, т.е. значения передаваемых переменных или ссылок на них (на рис. 4 каждой паре весов на одной пунктирной стрелке соответствуют две совмещенные функциональные дуги графа).

В настоящей работе предлагается далее на этапе проектирования распределенного приложения при определении состава передаваемых управляющих и функциональных сообщений использовать концептуальные графы.

Концептуальные графы операторов, приведенные на рис. 5, дополняют концептуальную схему программы, представленную на рис. 3, т.е. полный граф концептуальной схемы программы определяется путем объединения всех концептуальных графов для операторов.



а)



б)

Рис. 5. Примеры концептуальных графов для операторов S_0, S_1, …, S_9

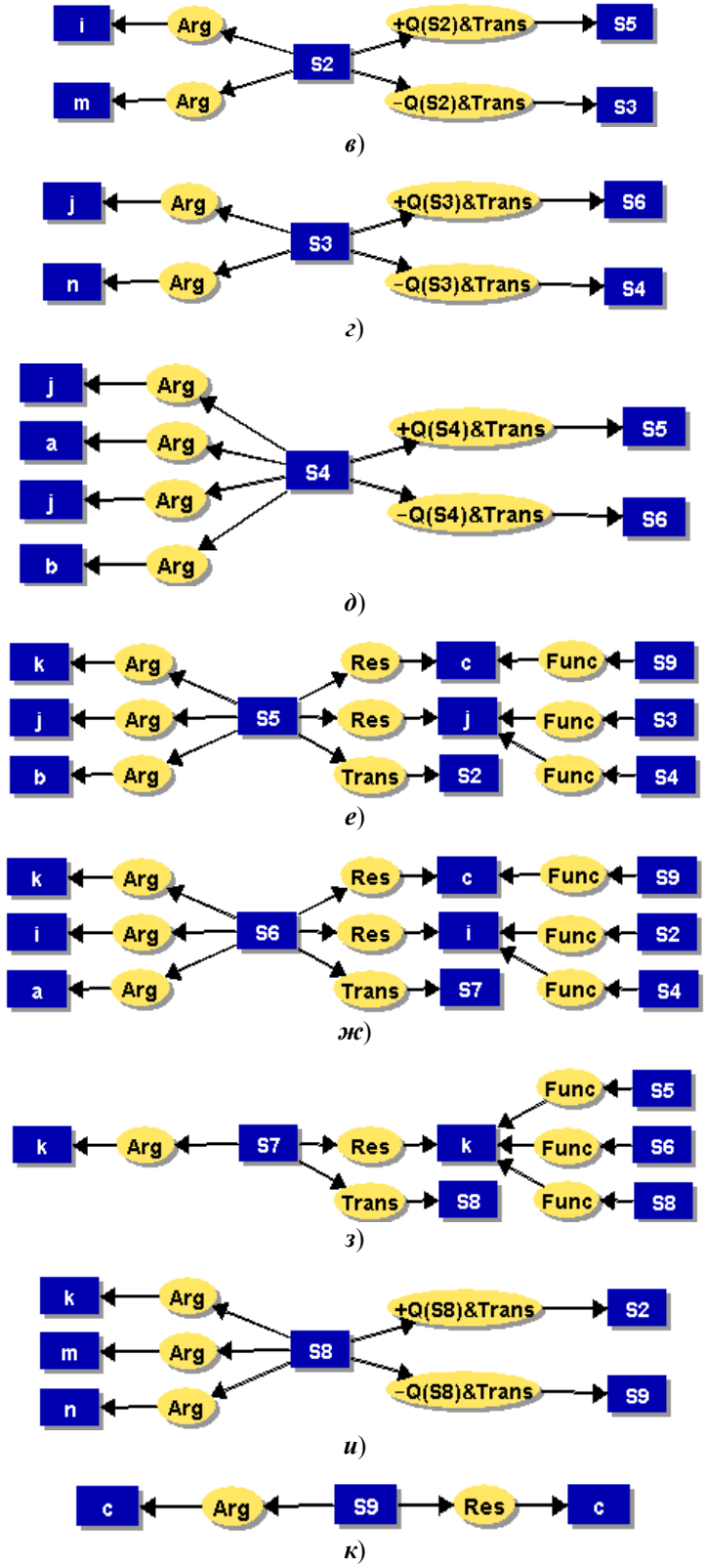


Рис. 5. Окончание

В графах овалами представлены следующие отношения: $Arg \subseteq S \times X$ – отношение «быть аргументом», где X – множество переменных; $Res \subseteq S \times X$ – отношение «являться результатом»; $Func \subseteq S \times X$ – отношение «являться функциональной переменной», т.е. переменной, передаваемой от одного оператора к другому для реализации функциональной связи, или связи по данным. Бинарные отношения *Trans*, *Arg* и *Res* образуют экстенсиональную базу данных, а единственное тернарное отношение *Data* – интенсиональную базу данных.

Правила формирования тернарного отношения *Data*, или области истинности одноименного тернарного предиката, используемого при формировании сообщений для реализации функциональных межоператорных связей, имеют следующий вид:

$$Res(S_0, m)\ \&\ Func(S_2, m) \supset Data(S_0, S_2, m);$$

$$Res(S_0, n)\ \&\ Func(S_3, n) \supset Data(S_0, S_3, n);$$

$$Res(S_0, a)\ \&\ Func(S_4, a) \supset Data(S_0, S_4, a);$$

$$Res(S_0, a)\ \&\ Func(S_6, a) \supset Data(S_0, S_6, a);$$

$$Res(S_0, b)\ \&\ Func(S_4, b) \supset Data(S_0, S_4, b);$$

$$Res(S_0, b)\ \&\ Func(S_5, b) \supset Data(S_0, S_5, b);$$

$$Res(S_1, i)\ \&\ Func(S_2, i) \supset Data(S_1, S_2, i);$$

$$Res(S_1, i)\ \&\ Func(S_4, i) \supset Data(S_1, S_4, i);$$

$$Res(S_1, i)\ \&\ Func(S_6, i) \supset Data(S_1, S_6, i);$$

$$Res(S_1, j)\ \&\ Func(S_3, j) \supset Data(S_1, S_3, j);$$

$$Res(S_1, j)\ \&\ Func(S_4, j) \supset Data(S_1, S_4, j);$$

$$Res(S_1, j)\ \&\ Func(S_5, j) \supset Data(S_1, S_5, j);$$

$$Res(S_1, k)\ \&\ Func(S_5, k) \supset Data(S_1, S_5, k);$$

$$Res(S_1, k)\ \&\ Func(S_6, k) \supset Data(S_1, S_6, k);$$

$$Res(S_1, k)\ \&\ Func(S_7, k) \supset Data(S_1, S_7, k);$$

$$Res(S_5, c)\ \&\ Func(S_9, c) \supset Data(S_5, S_9, c);$$

$$Res(S_5, j)\ \&\ Func(S_3, j) \supset Data(S_5, S_3, j);$$

$$Res(S_5, j)\ \&\ Func(S_4, j) \supset Data(S_5, S_4, j);$$

$$Res(S_6, c)\ \&\ Func(S_9, c) \supset Data(S_6, S_9, c);$$

$$Res(S_6, i)\ \&\ Func(S_2, i) \supset Data(S_6, S_2, i);$$

$$Res(S_6, i)\ \&\ Func(S_4, i) \supset Data(S_6, S_4, i);$$

$$Res(S_7, k)\ \&\ Func(S_5, k) \supset Data(S_7, S_5, k);$$

$$Res(S_7, k)\ \&\ Func(S_6, k) \supset Data(S_7, S_6, k);$$

$$Res(S_7, k)\ \&\ Func(S_8, k) \supset Data(S_7, S_8, k);$$

Например, правило $Res(S_0, m)$ & $Func(S_2, m) \supset Data(S_0, S_2, m)$ можно интерпретировать следующим образом: «если значение переменной m является результатом выполнения оператора S_0 и должно участвовать в реализации функциональной связи, ведущей к оператору S_2, то для организации этой связи следует передать значение переменной m от оператора S_0 оператору S_2». Истинность высказывания $Data(S_0, S_2, m)$ в логико-алгебраической модели вычислительного процесса интерпретируется как необходимость передачи сообщения со значением переменной m от узла вычислительной сети, на котором выполнен оператор S_0, к узлу, на котором должен быть выполнен оператор S_2. Аналогичным образом интерпретируются и все остальные правила.

Атомарные константные формулы, описывающие непосредственные связи операторов по управлению, составляются непосредственно по концептуальной схеме на рис. 3: $Trans(S_0, S_1)$, $Trans(S_1, S_2)$, $Trans(S_2, S_3)$, $Trans(S_2, S_5)$, $Trans(S_3, S_4)$, $Trans(S_3, S_6)$, $Trans(S_4, S_5)$, $Trans(S_4, S_6)$, $Trans(S_5, S_7)$, $Trans(S_7, S_8)$, $Trans(S_8, S_2)$, $Trans(S_8, S_9)$, и используются далее при формировании управляющих сообщений.

3. Формализованные спецификации на основе логико-алгебраических представлений для программного обеспечения распределенных систем, ориентированных на передачу сообщений

Концептуальные схемы распределенных программ относятся к классу универсальных вычислительных моделей, поскольку они позволяют описывать все структурные дейкстровские конструкции: следование, разветвление и циклы. Однако они не относятся к классу непосредственно исполнимых моделей в том смысле, что они не могут использоваться непосредственно для построения программ – требуется дополнительная интерпретация используемых понятий и отношений между ними. Например, концептуальные модели оставляют программисту свободу в выборе парадигмы распределенных вычислений, например – гибридной, *Linda*-подобной или ориентированной на передачу сообщений [1].

Для последующего рассмотрения в качестве основного этапа технологии проектирования программного обеспечения распределенных систем рассмотрим исполнимые логико-алгебраические модели в интерпретации, данной в работе [1]. Другие приложения логических и логико-алгебраических методов для описания моделей программ и производственных процессов рассматривались, например, в работах [11–13]. До настоящего времени актуальными и мало исследованными остаются вопросы объема накладных расходов при декомпозиции программ для выполнения в параллельных и распределенных средах [2, 3, 14].

Большой проблемой является также отображение распределенных программ на обрабатывающие модули распределенной вычислительной среды. Логико-алгебраические методы проектирования позволяют выбирать нужную степень декомпозиции программы на модули, вплоть до элементарных операций языка программирования, а методы искусственного интеллекта, основанные на концептуальных схемах программ, позволят эффективно согласовать архитектурные и языковые вопросы.

Построим логико-алгебраическую (исполнимую) модель для программного обеспечения распределенных систем, полностью ориентирован-

ных на передачу сообщений. Необходимость построение подобной модели вызвана тем, что обычно используемые на практике методы формализации распределенных программных систем практически не затрагивают обработку данных, либо не относятся к классу непосредственно исполнимых. Последнее обстоятельство часто вынуждает практического программиста к отказу от использования формализованных моделей, что может привести к наличию большого числа ошибок в логике работы распределенного приложения либо к усложнению процесса его проектирования.

Абстрактным объектам логико-алгебраической модели – множествам предметных переменных и констант, функциям, предикатам, правилам модификации предикатов – соответствуют реальные объекты, с которыми работают распределенные приложения, реализуемые на физической вычислительной сети. Логика работы с этими объектами напоминает сложную семафорную технику. При описании работы модели далее, как это часто принято, будут отождествляться ее абстрактные элементы и элементы предметной области (там, где это не вызывает противоречий).

В то время как согласно определению из [10] для определения «сосредоточенной» операторной схемы, реализуемой на одном узле вычислительной сети, при описании *связей по управлению* использован бинарный предикат *Trans*: $S \times S \to \{true, false\}$ (сокращенно T), для описания распределенной реализации операторной схемы понадобится тернарный (трехместный) предикат *Data*: $S \times S \times X \to \{true, false\}$ (сокращенно D), с помощью которого формализуются *функциональные связи*, т.е. передачи элементов данных или ссылок на них по сети. Здесь $S = \{S_0, S_1, \ldots, S_m\}$ – множество операторов; $X = \{x_0, x_1, \ldots, x_n\}$ – множество элементов данных (переменных, массивов, списков, кортежей, отношений). Добавив множество $Y = \{Y_0, Y_1, \ldots, Y_k\}$ вершин (узлов) графа связей инфокоммуникационной среды передачи данных, унарную функцию *Depl*: $S \to Y$ развертывания операторов в инфокоммуникационной среде и бинарный предикат *Comm*: $Y \times Y \to \{true, false\}$ связности узлов инфокоммуникационной среды, модифицируем определение из [10], получая следующее определение.

Определение. Распределенная операторная сеть (РОС, или *MOON – Message-Oriented Operator Net*) для спецификации распределенной вычислительной системы с программным обеспечением, ориентированным на передачу сообщений при реализации связей по управлению и функциональных связей по данным (путем передачи самих данных либо ссылок на них), описывается кортежем

$$MOON = (S, X, Y, Depl, Comm, Arg, Res, Trans, Data),$$

где областью истинности бинарного предиката *Arg*: $S \times X \to \{true, false\}$ задано отношение «быть аргументом», а областью истинности бинарного предиката *Res*: $S \times X \to \{true, false\}$ – отношение «иметь результатом». Остальные элементы кортежа определены выше.

Распределенные вычислительные системы, построение распределенных приложений для которых основано на данном определении, содержат сетевые вычислительные узлы, обладающие собственной локальной памятью и, возможно, частично или полностью разделяемую общую память. Функциональ-

ные связи реализуются здесь путем передачи сообщений с элементами данных или ссылок на них по сети. Связи по управлению между узлами реализуются путем передачи служебных сообщений.

Например, как было определено ранее, модификация предиката

$$Trans(S_1, S_2) \leftarrow true$$

означает, что от оператора S_1 реализуется связь по управлению, направленная к оператору S_2, путем передачи управляющего сообщения от узла сети, на котором реализуется оператор S_1, узлу, на котором реализуется оператор S_2.

Аналогично, модификация предиката

$$Data(S_0, S_2, m) \leftarrow true$$

означает реализацию функциональной связи между операторами S_0 и S_2 путем передачи «функционального» сообщения со значением переменной m или со ссылкой на нее, т.е. с ее адресом в сети. Строго говоря, сообщения обоих типов должны содержать адреса узлов, на которых реализуются соответствующие операторы, т.е. правила модификации предикатов должны иметь такой вид:

$$Trans(S_1, Depl(S_1), S_2, Depl(S_2)) \leftarrow true);$$

$$Data(S_0, Depl(S_0), S_2, Depl(S_2), m) \leftarrow true,$$

где функция *Depl* определяет адреса узлов-источников и узлов-приемников. Но мы далее для краткости записи атомарных константных формул будем опускать адресную информацию и полагать, что она существует по умолчанию и входит в формат передаваемых сообщений. В сообщениях также следует явно указывать, передается ли значение элемента данных (например, $Val(m)$) или ссылка на него ($Ref(m)$). Для краткости будем указывать просто имя элемента данных (например, m).

Положим, что модификации предикатов вида

$$Trans(S_1, S_2) \leftarrow false;$$

$$Data(S_0, S_2, m) \leftarrow false$$

означают, что передача соответствующих сообщений завершена. Кроме того, предполагаем, что конфигурация сети коммуникаций, определяемая областью истинности бинарного предиката *Comm*, известна всем узлам сети (на практике каждый узел может иметь лишь ограниченную информацию о топологии сети, поскольку при перемещении сообщений в реальной сети участвуют программно или аппаратно реализованные маршрутизаторы).

Рассмотрим переход от сосредоточенной операторной схемы к распределенной сети типа РОС (*MOON*). Разработаем новую систему логико-алгебраических выражений для сети типа РОС (*MOON*), в которой будут дополнительно учтены функциональные связи, реализуемые по мере выполнения распределенного вычислительного процесса. Как следует из определения сети РОС (*MOON*), каждый оператор сети может быть выполнен только тогда, когда по отношению к нему будет реализована не только связь по управлению, но и все функциональные связи, т.е. связи по всем данным, которые он должен использовать. Новая система логико-алгебраических выражений имеет следующий вид:

$q_{S0} = [Start](\{Start \leftarrow false;\ S_0;\ D(S_0, S_2, m) \leftarrow true;\ D(S_0, S_3, n) \leftarrow true;$
$D(S_0, S_4, a) \leftarrow true;\ D(S_0, S_4, b) \leftarrow true;\ D(S_0, S_5, b) \leftarrow true;$
$D(S_0, S_6, a) \leftarrow true;\ D(S_0, S_8, m) \leftarrow true;\ D(S_0, S_8, n) \leftarrow true;$
$T(S_0, S_1) \leftarrow true\} \vee R);$

$q_{S1} = [T(S_0, S_1)](\{T(S_0, S_1) \leftarrow false;\ S_1;\ D(S_1, S_2, i) \leftarrow true;$
$D(S_1, S_3, j) \leftarrow true;\ D(S_1, S_4, i) \leftarrow true;\ D(S_1, S_4, j) \leftarrow true;$
$D(S_1, S_5, k) \leftarrow true;\ D(S_1, S_5, j) \leftarrow true;\ D(S_1, S_6, k) \leftarrow true;$
$D(S_1, S_6, i) \leftarrow true;\ D(S_1, S_7, k) \leftarrow true;\ T(S_1, S_2) \leftarrow true\} \vee R);$

$q_{S2} = [T(S_1, S_2) \& D(S_1, S_2, i) \& D(S_0, S_2, m) \vee T(S_8, S_2) \& D(S_6, S_2, i)]$
$(\{T(S_1, S_2) \leftarrow false;\ T(S_8, S_2) \leftarrow false;\ S_2;\ D(S_1, S_2, i) \leftarrow false;$
$D(S_6, S_2, i) \leftarrow false;\ [Q(S_2)](T(S_2, S_5) \leftarrow true \vee T(S_2, S_3) \leftarrow true)\} \vee R);$

$q_{S3} = [T(S_2, S_3) \& D(S_0, S_3, n) \& (D(S_1, S_3, j) \vee D(S_5, S_3, j))]$
$(\{T(S_2, S_3) \leftarrow false;\ S_3;\ D(S_1, S_3, j) \leftarrow false;\ D(S_5, S_3, j) \leftarrow false;$
$[Q(S_3)](T(S_3, S_4) \leftarrow true \vee T(S_3, S_6) \leftarrow true)\} \vee R);$

$q_{S4} = [T(S_3, S_4) \& D(S_0, S_4, b) \& D(S_0, S_4, a) \& (D(S_1, S_4, i) \& D(S_1, S_4, j) \vee$
$D(S_5, S_4, j) \vee D(S_6, S_4, i))](\{T(S_3, S_4) \leftarrow false;\ S_4;\ D(S_1, S_4, i) \leftarrow false;$
$D(S_1, S_4, j) \leftarrow false;\ D(S_5, S_4, j) \leftarrow false;\ D(S_6, S_4, i) \leftarrow false;$
$[Q(S_4)](T(S_4, S_5) \leftarrow true \vee T(S_4, S_6) \leftarrow true)\} \vee R);$

$q_{S5} = [((T(S_2, S_5) \vee T(S_4, S_5)) \& D(S_0, S_5, b) \& (D(S_1, S_5, k) \& D(S_1, S_5, j) \vee$
$D(S_7, S_5, k) \& D(S_5, S_5, j))](\{T(S_2, S_5) \leftarrow false;\ T(S_4, S_5) \leftarrow false;\ S_5;$
$D(S_1, S_5, k) \leftarrow false;\ D(S_1, S_5, j) \leftarrow false;\ D(S_7, S_5, k) \leftarrow false;$
$D(S_5, S_5, j) \leftarrow false;$
$T(S_5, S_7) \leftarrow true;\ D(S_5, S_9, c) \leftarrow true;\ D(S_5, S_3, j) \leftarrow true;\ D(S_5, S_4, j) \leftarrow true;$
$D(S_5, S_5, j) \leftarrow true\} \vee R);$

$q_{S6} = [(T(S_3, S_6) \vee T(S_4, S_6)) \& D(S_0, S_6, a) \& (D(S_1, S_6, k) \& D(S_1, S_6, i) \vee$
$D(S_7, S_6, k) \& D(S_6, S_6, i))](\{T(S_3, S_6) \leftarrow false;\ T(S_4, S_6) \leftarrow false;\ S_6;$
$D(S_1, S_6, k) \leftarrow false;\ D(S_1, S_6, i) \leftarrow false;\ D(S_7, S_6, k) \leftarrow false;$
$D(S_6, S_6, i) \leftarrow false;$

$T(S_6, S_7) \leftarrow true$; $D(S_6, S_9, c) \leftarrow true$; $D(S_6, S_2, i) \leftarrow true$; $D(S_6, S_4, i) \leftarrow true$;
$D(S_6, S_6, i) \leftarrow true\} \vee R)$;

$q_{S7} = [(T(S_5, S_7) \vee T(S_6, S_7))\&(D(S_1, S_7, k) \vee D(S_7, S_7, k))]$
$(\{T(S_5, S_7) \leftarrow false$; $T(S_6, S_7) \leftarrow false$; S_7; $D(S_1, S_7, k) \leftarrow false$;
$D(S_7, S_7, k) \leftarrow false$;
$T(S_7, S_8) \leftarrow true$; $D(S_7, S_8, k) \leftarrow true$; $D(S_7, S_5, k) \leftarrow true$; $D(S_7, S_6, k) \leftarrow true$;
$D(S_7, S_7, k) \leftarrow true\} \vee R)$;

$q_{S8} = [T(S_7, S_8)\&D(S_7, S_8, k)\&D(S_0, S_8, m)\&D(S_0, S_8, n)]$
$(\{T(S_7, S_8) \leftarrow false$; S_8; $D(S_7, S_8, k) \leftarrow false$;
$[Q(S_8)](T(S_8, S_2) \leftarrow true \vee T(S_8, S_9) \leftarrow true\} \vee R)$;

$q_{S9} = [T(S_8, S_9)\&(D(S_5, S_9, c) \vee D(S_6, S_9, c))](\{T(S_8, S_9) \leftarrow false$;
S_9; $D(S_5, S_9, c) \leftarrow false$; $D(S_6, S_9, c) \leftarrow false$; $Finish \leftarrow true\} \vee R)$.

Подставляя на место операторов конкретные операции, выполняемые в программе, получим следующие окончательные выражения для модулей сети виртуальных машин:

$q_{S0} = [Start](\{Start \leftarrow false$; $Read(m, n, a, b)$; $D(S_0, S_2, m) \leftarrow true$;
$D(S_0, S_3, n) \leftarrow true$; $D(S_0, S_4, a) \leftarrow true$; $D(S_0, S_4, b) \leftarrow t\ rue$;
$D(S_0, S_5, b) \leftarrow true$; $D(S_0, S_6, a) \leftarrow true$; $D(S_0, S_8, m) \leftarrow true$;
$D(S_0, S_8, n) \leftarrow true$; $T(S_0, S_1) \leftarrow true\} \vee R)$;

$q_{S1} = [T(S_0, S_1)](\{T(S_0, S_1) \leftarrow false$; $i := j := k := 1$; $D(S_1, S_2, i) \leftarrow true$;
$D(S_1, S_3, j) \leftarrow true$; $D(S_1, S_4, i) \leftarrow true$; $D(S_1, S_4, j) \leftarrow true$;
$D(S_1, S_5, k) \leftarrow true$; $D(S_1, S_5, j) \leftarrow true$; $D(S_1, S_6, k) \leftarrow true$;
$D(S_1, S_6, i) \leftarrow true$; $D(S_1, S_7, k) \leftarrow true$; $T(S_1, S_2) \leftarrow true\} \vee R)$;

$q_{S2} = [T(S_1, S_2)\&D(S_1, S_2, i)\&D(S_0, S_2, m) \vee T(S_8, S_2)\&D(S_6, S_2, i)]$
$(\{T(S_1, S_2) \leftarrow false$; $T(S_8, S_2) \leftarrow false$; $Q(S_2) := i > m$; $D(S_1, S_2, i) \leftarrow false$;
$D(S_6, S_2, i) \leftarrow false$; $[Q(S_2)](T(S_2, S_5) \leftarrow true \vee T(S_2, S_3) \leftarrow true)\} \vee R)$;

$q_{S3} = [T(S_2, S_3)\&D(S_0, S_3, n)\&(D(S_1, S_3, j) \vee D(S_5, S_3, j))]$
$(\{T(S_2, S_3) \leftarrow false$; $Q(S_3) := j > n$; $D(S_1, S_3, j) \leftarrow false$; $D(S_5, S_3, j) \leftarrow false$;
$[Q(S_3)](T(S_3, S_4) \leftarrow true \vee T(S_3, S_6) \leftarrow true)\} \vee R)$;

$q_{S4} = [T(S_3, S_4)\&D(S_0, S_4, b)\&D(S_0, S_4, a)\&(D(S_1, S_4, i)\&D(S_1, S_4, j) \vee$
$D(S_5, S_4, j) \vee D(S_6, S_4, i))](\{T(S_3, S_4) \leftarrow false; Q(S_4):=a[i] > b[j];$
$D(S_1, S_4, i) \leftarrow false; D(S_1, S_4, j) \leftarrow false; D(S_5, S_4, j) \leftarrow false;$
$D(S_6, S_4, i) \leftarrow false; [Q(S_4)](T(S_4, S_5) \leftarrow true \vee T(S_4, S_6) \leftarrow true)\} \vee R);$

$q_{S5} = [((T(S_2, S_5) \vee T(S_4, S_5))\&D(S_0, S_5, b)\&(D(S_1, S_5, k)\&D(S_1, S_5, j) \vee$
$D(S_7, S_5, k)\&D(S_5, S_5, j))](\{T(S_2, S_5) \leftarrow false; T(S_4, S_5) \leftarrow false;$
$c[k] := b[j]; j := j + 1; D(S_1, S_5, k) \leftarrow false; D(S_1, S_5, j) \leftarrow false;$
$D(S_7, S_5, k) \leftarrow false; D(S_5, S_5, j) \leftarrow false; T(S_5, S_7) \leftarrow true;$
$D(S_5, S_9, c) \leftarrow true; D(S_5, S_3, j) \leftarrow true; D(S_5, S_4, j) \leftarrow true;$
$D(S_5, S_5, j) \leftarrow true\} \vee R);$

$q_{S6} = [(T(S_3, S_6) \vee T(S_4, S_6))\&D(S_0, S_6, a)\&(D(S_1, S_6, k)\&D(S_1, S_6, i) \vee$
$D(S_7, S_6, k)\&D(S_6, S_6, i))](\{T(S_3, S_6) \leftarrow false; T(S_4, S_6) \leftarrow false; c[k] := a[i];$
$i := i + 1; D(S_1, S_6, k) \leftarrow false; D(S_1, S_6, i) \leftarrow false;$
$D(S_7, S_6, k) \leftarrow false; D(S_6, S_6, i) \leftarrow false;$
$T(S_6, S_7) \leftarrow true; D(S_6, S_9, c) \leftarrow true; D(S_6, S_2, i) \leftarrow true; D(S_6, S_4, i) \leftarrow true;$
$D(S_6, S_6, i) \leftarrow true\} \vee R);$

$q_{S7} = [(T(S_5, S_7) \vee T(S_6, S_7))\&(D(S_1, S_7, k) \vee D(S_7, S_7, k))]$
$(\{T(S_5, \mathrm{S}_7) \leftarrow false; T(\mathrm{S}_6, \mathrm{S}_7) \leftarrow false; k := k + 1; D(S_1, S_7, k) \leftarrow false;$
$D(S_7, S_7, k) \leftarrow false; T(S_7, S_8) \leftarrow true; D(S_7, S_8, k) \leftarrow true;$
$D(S_7, S_5, k) \leftarrow true; D(S_7, S_6, k) \leftarrow true; D(S_7, S_7, k) \leftarrow true\} \vee R);$

$q_{S8} = [T(S_7, S_8)\&D(S_7, S_8, k)\&D(S_0, S_8, m)\&D(S_0, S_8, n)]$
$(\{T(S_7, S_8) \leftarrow false; Q(S_8):= k <= m + n; D(S_7, S_8, k) \leftarrow false;$
$[Q(S_8)](T(S_8, S_2) \leftarrow true \vee T(S_8, S_9) \leftarrow true\} \vee R);$

$q_{S9} = [T(S_8, S_9)\&(D(S_5, S_9, \mathrm{c}) \vee D(S_6, S_9, \mathrm{c}))](\{T(S_8, S_9) \leftarrow false;$
$Write(c); D(S_5, S_9, \mathrm{c}) \leftarrow false; D(S_6, S_9, \mathrm{c}) \leftarrow false; Finish \leftarrow true\} \vee R).$

Нотация логико-алгебраических выражений соответствует нотации, принятой в работах [1, 15–17]. Символом R обозначен оператор возврата

к проверке условия, описанного в квадратных скобках выражения для каждого абстрактного модуля сети абстрактных машин. В рассмотренном примере реализация функциональных связей и связей по управлению между операторами (в абстрактной модели) или между программными модулями (в реальном приложении) происходит с помощью механизмов передачи сообщений – управляющих и функциональных. При реализации распределенной программы необходимо выбрать, передавать значения элементов данных или только ссылки на места хранения этих значений. От этого выбора зависит объем передаваемой информации и, следовательно, эффективность работы приложения в сетевой среде. При получении логико-алгебраических выражений учтены представления распределенных вычислений концептуальными графами (рис. 3, 5), а также основанные на них правила формирования предикатов *Trans* и *Data*. Дополнительные концептуальные схемы для описания других управляющих структур распределенного программирования приведены в работах [18–20].

При абстрактном и структурном синтезе распределенных приложений от программиста дополнительно требуется наличие знаний эксперта-разработчика, представленных концептуальными графами на рис. 1, 2, а также знание принципов построения концептуальных схем распределенных программ. В настоящей работе показано, что проектирование экспертной модели знаний программиста и проектирование самого распределенного приложения может быть выполнено на основе общей концептуальной схемы.

Заключение

1. Предложена новая концептуальная модель распределенной обработки данных в сетевой среде, в которой учитывается реализация связей по управлению и данным через сетевое инфокоммуникационное пространство, что упрощает абстрактный синтез систем распределенной обработки данных.

2. Формально определен новый класс распределенных операторных сетей, причем в основу определения положены как известные теоретико-множественные понятия, так и понятия, связанные с использованием инфокоммуникационной среды, что в существенной степени облегчает составление и применение системы формализованных спецификаций для программных модулей, выполняемых в распределенной сетевой среде.

3. Формально определена операционная семантика распределенной модели вычислений, базирующаяся на концептуальном представлении выполняемых операций и логико-алгебраических моделях.

4. Предложены новые концептуальные, логические и логико-алгебраические модели распределенных вычислений в системах с архитектурой, основанной на передаче сообщений, отличающиеся от известных тем, что они относятся к классу непосредственно исполнимых (реализуемых), применение которых позволяет снизить трудозатраты при создании распределенных сетевых приложений.

Список литературы

1. **Волчихин, В. И.** Абстрактный и структурный синтез распределенных систем обработки данных на основе мультипарадигмального подхода / В. И. Волчихин, А. В. Дубравин, С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2015. – № 1 (33). – С. 60–69.

2. **Хьюз, К.** Параллельное и распределенное программирование на C++ / К. Хьюз, Т. Хьюз. – М. : Вильямс, 2004. – 672 с.
3. **Таненбаум, Э.** Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – СПб. : Питер, 2003. – 877 с.
4. Логический подход к искусственному интеллекту: от классической логики к логическому программированию / А. Тейз, П. Грибомон, Ж. Луи и др. – М. : Мир, 1990. – 429 с.
5. **Sowa, J. F.** Conceptual Graphs: Draft Proposed American National Standard / J. F. Sowa // Proceedings of the 7th International Conference on Conceptual Structures: Standards and Practices (July 12–15, 1999). – M., 1999. – P. 1–65.
6. **Delugach, H.** CharGer: Some Lessons Learned and New Directions. Working with Conceptual Structures: Contributions to ICCS 2000 / G. Stumme ; ed., Shaker Verlag, 2000. – P. 306–309.
7. CharGer Manual. – 2005. – Vol. 3.5, b. 1. – P. 1–58. – URL: http://charger.sourceforge.net (дата обращения: 04.04.2015).
8. **Гаврилова, Т. А.** Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб. : Питер, 2001. – 384 с.
9. **Котов, В. Е.** Теория схем программ / В. Е. Котов, В. К. Сабельфельд. – М. : Наука, 1991. – 248 с.
10. **Лавров, С. С.** Программирование. Математические основы, средства, теория / С. С. Лавров. – СПб. : БХВ-Петербург, 2001. – 320 с.
11. **Ломакина, Л. С.** Теория и практика структурного тестирования программных систем / Л. С. Ломакина, А. С. Базин, А. Н. Вигура, А. В. Киселев. – Воронеж : Научная книга, 2013. – 220 с.
12. **Плесневич, Г. С.** Логические модели / Г. С. Плесневич // Искусственный интеллект : в 3 кн. Кн. 2. Модели и методы : справочник / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – С. 14–28.
13. **Курганский, В. И.** Алгебраические преобразования программ и порождаемых ими отношений / В. И. Курганский // Системы управления и информационные технологии. – 2006. – № 3.1 (25). – С. 139–144.
14. Элементы параллельного программирования / В. А. Вальковский, В. Е. Котов, А. Г. Марчук, Н. Н. Миренков ; под ред. В. Е. Котова. – М. : Радио и связь, 1983. – 240 с.
15. **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.
16. **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. 37–50.
17. **Зинкин, С. А.** Элементы новой объектно ориентированной сетевой технологии для моделирования и реализации систем и сетей хранения и обработки данных / Зинкин С. А. // Информационные технологии. – 2008. – № 10. – С. 20–27.
18. **Дубравин, А. В.** Элементы концептуального распределенного программирования в сетях / А. В. Дубравин, С. А. Зинкин // Университетское образование (МКУО-2015) : сб. ст. XIX Междунар. науч.-метод. конф., посвящ. 70-летию Победы в Великой Отечественной войне (г. Пенза, 9–10 апреля 2015 г.) : в 2 т. / под ред. А. Д. Гулякова, Р. М. Печерской. – Пенза : Изд-во ПГУ, 2015. – Т. 1. – С. 222–225.
19. **Дубравин, А. В.** Формальное определение гибридной модели распределенных вычислений в сетях / А. В. Дубравин, С. А. Зинкин // Университетское образование (МКУО-2015) : сб. ст. XIX Междунар. науч.-метод. конф., посвящ. 70-летию

Победы в Великой Отечественной войне (г. Пенза, 9–10 апреля 2015 г.) : в 2 т. / под ред. А. Д. Гулякова, Р. М. Печерской. – Пенза : Изд-во ПГУ, 2015. – Т. 1. – С. 226-228.

20. **Дубравин, А. В.** Развитие логико-алгебраического подхода к созданию распределенных приложений для обработки данных в вычислительных сетях / А. В. Дубравин, С. А. Зинкин // Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации (Распознавание-2015) : сб. ст. XII Междунар. науч.-техн. конф. (г. Курск, 12–15 мая 2015 г.). – Курск : Изд-во ЮЗГУ, 2015.

References

1. Volchikhin V. I., Dubravin A. V., Zinkin S. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2015, no. 1 (33), pp. 60–69.
2. Kh'yuz K., Kh'yuz T. *Parallel'noe i raspredelennoe programmirovanie na C++* [Parallel and distributed C++ programming]. Moscow: Vil'yams, 2004, 672 p.
3. Tanenbaum E., M. van Steen *Raspredelennye sistemy. Printsipy i paradigmy* [Distributed systems. Principles and paradigms]. Saint-Petersburg: Piter, 2003, 877 p.
4. Teyz A., Gribomon P., Lui Zh. et al. *Logicheskiy podkhod k iskusstvennomu intellektu: ot klassicheskoy logiki k logi-cheskomu programmirovaniyu* [Logic approach to artificial intelligence: from classical logic to logical programming]. Moscow: Mir, 1990, 429 p.
5. Sowa J. F. *Proceedings of the 7th International Conference on Conceptual Structures: Standards and Practices (July 12–15, 1999)*. Moscow, 1999, pp. 1–65.
6. Delugach H., Stumme G. *CharGer: Some Lessons Learned and New Directions. Working with Conceptual Structures: Contributions to ICCS 2000*. 2000, pp. 306–309.
7. *CharGer Manual*. 2005, vol. 3.5, b. 1, pp. 1–58. Available at: http://charger.sourceforge.net (accessed April 4, 2015).
8. Gavrilova T. A., Khoroshevskiy V. F. *Bazy znaniy intellektual'nykh sistem* [Knowledge bases of intelligence systems]. Saint-Petersburg: Piter, 2001, 384 p.
9. Kotov V. E., Sabel'fel'd V. K. *Teoriya skhem programm* [Theory of program schemes]. Moscow: Nauka, 1991, 248 p.
10. Lavrov S. S. *Programmirovanie. Matematicheskie osnovy, sredstva, teoriya* [Programming. Mathematical bases, means, theory]. Saint-Petersburg: BKhV-Peterburg, 2001, 320 p.
11. Lomakina L. S., Bazin A. S., Vigura A. N., Kiselev A. V. *Teoriya i praktika strukturnogo testirovaniya programmnykh sistem* [Theory and practice of structural testing of software systems]. Voronezh: Nauchnaya kniga, 2013, 220 p.
12. Plesnevich G. S. *Iskusstvennyy intellekt: v 3 kn. Kn. 2. Modeli i metody: spravochnik* [Artificial intelligence: in 3 books. Book 2. Models and methods: reference book]. Moscow: Radio i svyaz', 1990, pp. 14–28.
13. Kurganskiy V. I. *Sistemy upravleniya i informatsionnye tekhnologii* [Control systems and information technologies]. 2006, no. 3.1 (25), pp. 139–144.
14. Val'kovskiy V. A., Kotov V. E., Marchuk A. G., Mirenkov N. N. *Elementy parallel'nogo programmirovaniya* [Elements of parallel programming]. Moscow: Radio i svyaz', 1983, 240 p.
15. Zinkin S. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2007, no. 3, pp. 13–22.
16. Zinkin S. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2007, no. 4, pp. 37–50.
17. Zinkin S. A. *Informatsionnye tekhnologii* [Information technologies]. 2008, no. 10, pp. 20–27.

18. Dubravin A. V., Zinkin S. A. *Universitetskoe obrazovanie (MKUO-2015): sb. st. XIX Mezhdunar. nauch.-metod. konf., posvyashch. 70-letiyu Pobedy v Velikoy Otechestvennoy voyne (g. Penza, 9–10 aprelya 2015 g.): v 2 t.* [University education (MKUO-2015): proceedings of XIX International scientific and methodological conference devoted to 70th Anniversary of the Victory in the Great Patriotic War (Penza, 9-10 April 2015): in 2 volumes]. Penza: Izd-vo PGU, 2015, vol. 1, pp. 222–225.
19. Dubravin A. V., Zinkin S. A. *Universitetskoe obrazova-nie (MKUO-2015): sb. st. XIX Mezhdunar. nauch.-metod. konf., posvyashch. 70-letiyu Pobedy v Velikoy Otechestvennoy voyne (g. Penza, 9–10 aprelya 2015 g.): v 2 t.* [University education (MKUO-2015): proceedings of XIX International scientific and methodological conference devoted to 70th Anniversary of the Victory in the Great Patriotic War (Penza, 9-10 April 2015): in 2 volumes]. Penza: Izd-vo PGU, 2015, vol. 1, pp. 226–228.
20. Dubravin A. V., Zinkin S. A. *Optiko-elektronnye pribory i ustroystva v sistemakh raspoznavaniya obrazov, obrabotki izobrazheniy i simvol'noy informatsii (Raspoznavanie-2015): sb. st. XII Mezhdunar. nauch.-tekhn. konf. (g. Kursk, 12–15 maya 2015 g.)*. [Optoelectronic devices and equipment in systems of image recognition, image processing and symbolic information (Recognition-2015): proceedings of XII International scientific and technical conference (Kursk, 12-15 May 2015)]. Kursk: Izd-vo YuZGU, 2015.

Волчихин Владимир Иванович
доктор технических наук, профессор, президент Пензенского государственного университета (Россия, г. Пенза, ул. Красная, 40)

E-mail: cnit@pnzgu.ru

Volchikhin Vladimir Ivanovich
Doctor of engineering sciences, professor, President of Penza State University (40 Krasnaya street, Penza, Russia)

Дубравин Алексей Викторович
старший преподаватель, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: radamsa@yandex.ru

Dubravin Aleksey Viktorovich
Senior lecturer, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Зинкин Сергей Александрович
доктор технических наук, профессор, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: zsa49@yandex.ru

Zinkin Sergey Aleksandrovich
Doctor of engineering sciences, professor, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

УДК 681.324

Волчихин, В. И.

Абстрактный и структурный синтез систем обработки данных на основе парадигмы передачи сообщений в распределенном окружении вычислительной сети / В. И. Волчихин, А. В. Дубравин, С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2015. – № 2 (34). – С. 104–122.