УДК 681.324

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

АБСТРАКТНЫЙ И СТРУКТУРНЫЙ СИНТЕЗ РАСПРЕДЕЛЕННЫХ СИСТЕМ ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ МУЛЬТИПАРАДИГМАЛЬНОГО ПОДХОДА[1]

Аннотация.

Актуальность и цели. Объектом исследования являются распределенные системы обработки данных, работа которых основана на новых принципах, названных в статье парадигмами. 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 новой гибридной технологии распределенного программирования.

Результаты. Главным результатом работы является то, что 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.

Выводы. Сопоставление описательных возможностей двух моделей распределенных вычислений позволило сделать вывод о том, что новая гибридная модель распределенной обработки данных в большей степени соответствует сетевой среде и отличается от известных тем, что в ней передача управления осуществляется путем передачи сообщений через сетевое инфокоммуникационное пространство, а функциональные связи реализуются через структурированное виртуальное пространство памяти, что во многих случаях ускоряет обработку данных. Предложенные новые концептуальные, логические и логико-алгебраические модели распределенных вычислений в системах с гибридной архитектурой, которые 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, концептуальные, логические и логико-алгебраические модели, мультипарадигмальный подход, виртуальное пространство, управляющие и функциональные межмодульные связи.

[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).

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

ABSTRACT AND STRUCTURAL SYNTHESIS OF DISTRIBUTED SYSTEMS OF DATA PROCESSING ON THE BASIS OF THE MULTIPARADIGM APPROACH

Abstract.

Background. The research object is the distributed systems of data processing, the functioning of which is based on new principles called paradigms in the article. The research subject is the problems of abstract and structural design of distributed network applications on the basis of the logic-algebraic approach and some methods of artificial intelligence. The aim of the works is to improve the design methods of distributed applications on the basis of conceptual, logical and logic-algebraic models, forming the base of a new hybrid technology of distributed programming.

Results. The main result of the study is that the suggested methods allow to develop distributed application for data processing, corresponding to a certain formal specification. For realization of distributed application of data processing the authors chose the methods that require no special syntactical primitive elements at organizing parallelism in distributed systems. Description and support of parallelism in distributed systems is provided by the means, realized on the basis of the transition from primary conceptual representations of the processes, based on the rules of output and conceptual graphs, to the direct programming through the direct use of logic-algebraic expressions in the form of formalized specifications.

Conclusions. Comparison of descriptive potentials of two models of distributed calculations allowed to conclude that the new hybrid model of distributed date processing to a large extent corresponds to a network environment and differs from the existing ones by the fact that the transfer of control is executed by transmission of messages through the network infocommunicative environment, and functional links are realized through the structured virtual memeory space, which in many cases accelerates data processing. The authors suggested new conceptual, logical and logic-algebraic models of distributed calculations in systems with hybrid architecture that differ from the existing ones by its' relation to the class of the directly executed (realized) ones, the application of which allows to lower labor expenditures in the process of distributed network application creation.

Key words: distributed data processing, conceptual, logical and logic-algebraic models, multiparadigm approach, virtual space, control and functional intermodule links.

Введение

Для того чтобы распределенное приложение соответствовало выбранной разработчиком парадигме, на начальных стадиях проекта целесообразно использовать концептуальные модели и логические модели искусственного интеллекта. Под парадигмой здесь подразумевается 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.

Например, в работах [1, 2] использован неформализованный архитектурный подход к параллельному и распределенному программированию

с акцентом на определении естественного параллелизма в самих задачах, что отражается в программных моделях решений. В настоящей работе архитектурный подход основан 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.

Обычно используемые логико-алгебраические модели описывают некоторые свойства программного обеспечения и редко дают точное представление о том, за счет чего они изменяются; различие между логическими и алгебраическими моделями условно: в логических моделях изучаются суждения о свойствах предметной области, а в алгебраических моделях большее внимание уделяется термам и операциям [3–5].

Такие модели, как, например, автоматы и сети Петри, относятся к исполнимым операционным моделям [6–8]. К подобным моделям отнесем также системы алгоритмических алгебр (САА) В. М. Глушкова [9, 10] и машины абстрактных состояний (МАС) Ю. Гуревича [11, 12]. В САА под состоянием понимается состояние информационного множества, а в МАС – состояние абстрактного пространства предикатов и функций, т.е. текущая интерпретация предикатных и функциональных символов. Промежуточное положение между логико-алгебраическими и исполнимыми моделями занимают логики Ч. Хоара [13] и программные логики [14].

Из теории САА для дальнейшего использования в данной работе выбраны нотации операций α-дизъюнкции и α-итерации для всюду определенных логических условий, а также операция композиции операторов, а из теории МАС – абстракция пространства памяти и операции изменения его состояний путем модификации интерпретации предикатных и функциональных символов. Такой выбор был впервые осуществлен в работах [15, 16]. Подобную модель мы отнесем к «непосредственно исполнимым», подразумевая под этим тот факт, что составление логико-алгебраических выражений является последним этапом перед интерпретацией кодированием программных модулей на каком-либо языке программирования.

Новизна результатов, полученных в данной главе, связана с тем, что будут получены формализованные логико-алгебраические спецификации для распределенных систем с генеративной связью (так называемые Linda-подобные системы Д. Джелернтера [17–19]), систем с ориентацией на передачу сообщений и, главное, гибридных систем, использующих положительные особенности первых двух систем. Полученные формализованные логико-алгебраические спецификации позволят унифицировать представление распределенных операторных сетей и в существенной степени упростить их реализацию в виде приложений, исполняемых в сетевой распределенной среде.

1. Операторные сети для определения и формализации гибридной парадигмы распределенных вычислений

В области теоретического программирования изучаются следующие традиционные направления [20, 21]: математические основы программирования; теория схем программ (схематология); семантическая теория программ; теория вычислительных процессов и структур (теория параллельных вычислений); решение прикладных задач теоретического программирования. Цен-

тральным понятием современной теории программирования является схема программы. Из операторных схем, используемых для описания программ, выберем схемы, определенные С. С. Лавровым.

Определение 1 [22]. Последовательная операторная схема (ПОС) определена как пятерка (U, X, A, R, T), где $U = \{S_0, S_1, S_2, \ldots, S_m\}$ – множество операторов; $X = \{x_0, x_1, x_2, \ldots, x_n\}$ – множество переменных; $A \subseteq X \times U$ – отношение «быть аргументом»; $R \subseteq U \times X$ – отношение «иметь результатом»; $T \subseteq U \times U$ – отношение между оператором-предшественником и оператором-преемником, т.е. множество возможных переходов, реализующих связи по управлению.

Заметим, что в определение ПОС в явном виде не входит множество логических условий. Данное определение для сосредоточенных операторных схем программ без параллелизма, основанное на теоретико-множественном подходе, удобно использовать для последующего перехода к определению и описанию принципов построения распределенной и, возможно, параллельной операторной сети (РОС).

В основу нового определения главным образом будут положены особенности сетевой инфокоммуникационной среды, в которой программные модули для операторов, размещенные на удаленных друг от друга узлах сети, при передаче управления используют модель передачи сообщений, а при обработке данных (возможно, структурированных) – виртуальное общее структурированное пространство памяти.

Рисунок 1,***а*** иллюстрирует выбранную абстракцию среды реализации распределенных операторных сетей. В коммуникационной среде реализуются связи операторов по управлению путем передачи сообщений, причем каждый оператор при выполнении предписанных ему операций использует общее виртуальное пространство памяти *VS* (рис. 1,***б***) по принципу, впервые реализованному в системе параллельного программирования Linda [17] и развитому в [23, 24] и др. Построение такой среды на абстрактном уровне может быть представлено выделением из «генеративных» связей несвязных по времени и по ссылкам процессов связей между процессами-предшественниками и процессами-последователями и последующей реализацией этих связей явно, путем передач управляющих сообщений. Такой подход позволит сократить затраты времени на организацию управления в распределенной среде и использовать развитое программное обеспечение промежуточного уровня (*message oriented middleware*), ориентированное на обмен сообщениями в распределенном окружении.

При обработке же данных реализуемые операторами процессы будут вести себя аналогично процессам в Linda-подобной системе, коллективно используя структурированное пространство кортежей. Отличие предлагаемой модели вычислений будет заключаться в том, что связи по управлению будут осуществляться не через виртуальное пространство памяти, а через коммуникационную среду, что значительно ускорит работу сетевых приложений. Функциональные связи, или связи по данным, будут реализовываться через *VS*-пространство. Тем не менее формализованное описание инфокоммуникационных процессов и процессов обработки данных мы будем выполнять на общей логико-алгебраической основе.

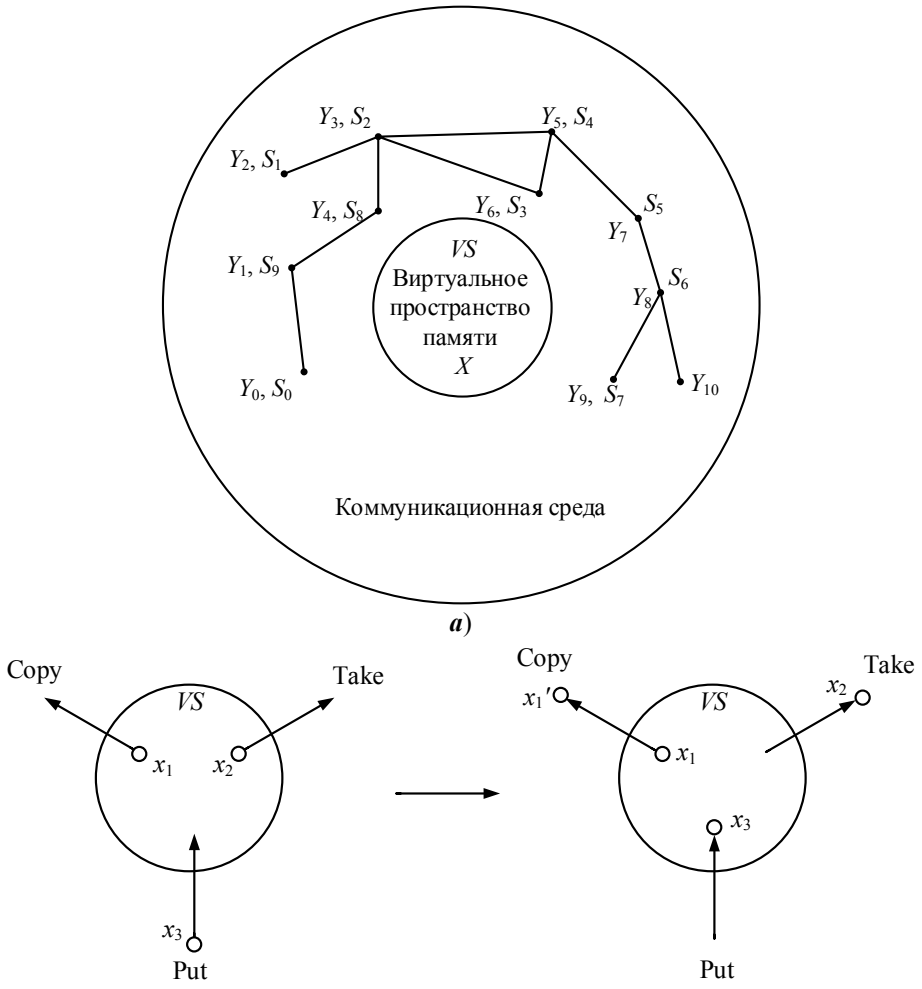

Рис. 1. Абстракция среды реализации распределенных операторных сетей (***а***) и иллюстрация выполнения операций в виртуальном пространстве памяти (***б***)

Отношения, представленные в **определении 1**, мы будем далее задавать областями истинности одноименных предикатов. Будут также изменены некоторые имена математических объектов.

Определение 2. Распределенная операторная сеть (РОС, или *DON – distributed operator net*) задается кортежем

$$DON = (S, X, Y, Op, Depl, Comm, Arg, Oper, Res, Trans, Work,$$
$$VS, L, Q, Form, CG, Rules, M), \quad (1)$$

где

$S = \{s_0, s_1, \ldots, s_m\}$ – множество операторов;

$X = \{x_0, x_1, \ldots, x_n\}$ – множество элементов данных (переменных, массивов, списков, кортежей, отношений);

$Y = \{y_0, y_1, \ldots, y_k\}$ – множество вершин (узлов) графа связей инфокоммуникационной среды передачи данных;

$Op = \{Op_0, Op_1, \ldots, Op_q\}$ – множество конкретных операций или блоков операций;

$Depl: S \to Y$ – функция развертывания операторов в инфокоммуникационной среде;

$Comm: Y \times Y \to \{true, false\}$ – бинарный предикат связности узлов инфокоммуникационной среды;

$Arg: S \times X \to \{true, false\}$ – бинарный предикат «являться аргументом»;

$Oper: S \times Op \to \{true, false\}$ – бинарный предикат «конкретизация оператора»;

$Res: S \times X \to \{true, false\}$ – бинарный предикат «являться результатом»;

$Trans: S \times S \to \{true, false\}$ – бинарный предикат, описывающий связи по управлению между операторами;

$Work: S \to \{true, false\}$ – унарный предикат, задающий состояния выполнения операторов;

VS – виртуальное структурированное пространство памяти;

$L = \{\text{Put, Take, Copy}\}$ – множество операций, связанных с абстрактным пространством виртуальной структурированной памяти VS;

$Q: S \to \{true, false\}$ – унарный предикат, сохраняющий значения α-условий после выполнения операторов из множества S;

$Form: S \to \{true, false\}$ – унарный предикат, определяющий сформированность условий $Q(s_i)$ после выполнения соответствующих операторов $s_i \in S$, $i = 0, 1, \ldots, m$;

CG – множество концептуальных графов для описания семантики операторов;

$Rules$ – множество правил выполнения операторов;

M – множество абстрактных модулей, описанных логико-алгебраическими выражениями, специфицирующими работу операторов РОС.

Далее для предикатов $Trans$ и $Form$ для краткости будут также использоваться сокращенные имена T и F соответственно.

Данное определение обладает некоторой избыточностью и включает понятия, необходимые или рекомендуемые для проектирования приложения по распределенной операторной сети. Определение дано для РОС, выполняющихся в абстрактной среде, представленной на рис. 1. Это определение не включает операторы распараллеливания ветвей вычислений, а также разветвления вычислительных процессов на число ветвей, больших двух. Однако понятий, связанных с **определением 2**, будет достаточно и для формализованных спецификаций указанных дополнительных операторов.

При составлении логико-алгебраических выражений для модулей из множества M мы будем использовать элементы систем алгебраических алгебр В. М. Глушкова [9, 10], например, операции α-дизъюнкции и α-итерации для всюду определенных логических условий, а также специальные виды операторов – тождественный оператор E и нигде не определенный, или невозможный, оператор N. Так, при реализации выражения $[\alpha](A \vee B)$ и при истинном значении условия α реализуется оператор A, а при ложном – B. В САА определена некоммутативная и ассоциативная операция умножения (последовательная композиция) операторов $A*B$, которая в данной работе обозначается как «*». Известно, что множество операторов является полугруппой,

а E и N играют роль единицы и нуля этой полугруппы соответственно: $A * E = E * A = A$; $A * N = N * A = N$.

В рамках предлагаемой в работе гибридной технологии распределенного программирования оператор E мы будем интерпретировать как пустой оператор, а оператор N – как недопустимый оператор. Введем дополнительный оператор R (*Return*) «возврата». При реализации модулем m выражения $m = [\alpha](A \vee R)$ при ложном значении условия α управление передается оператору R, который возвращает модуль m к повторной проверке условия α. Это действие повторяется до тех пор, пока условие α ложно. Как только условие α станет истинным, будет выполнен оператор A. При выполнении «возврата» оператор R инициирует обновление условия α путем установления связи с каким-либо другим вычислительным процессом – локальным или сетевым, запрашивая и ожидая от него ответного значения α.

Далее используются абстрактные модули, реализующие выражения $r = [\alpha](A \vee E)$ и $s = [\alpha](A \vee N)$. Модулем вида r при истинном α выполняется оператор A, а при ложном – пустой оператор E. На практике это означает, что при ложном значении α модуль r не выполняет никаких действий. Модулем вида s при истинном α выполняется оператор A, а при ложном – нигде не определенный, или невозможный, оператор N, что возможно интерпретировать как возникновение ошибочной ситуации, приводящей к остановке процесса.

2. Сетевая интерпретация логико-алгебраических выражений

Реализация выполнения операторов в распределенной инфокоммуникационной среде тесно связана также с реализацией «синхронизатора» и близких к нему механизмов «спусковой функции» и «операции ожидания», известных в параллельном и распределенном программировании: «синхронизатор, установленный в некотором месте параллельной регулярной схемы, осуществляет задержку вычислений в данном месте схемы вплоть до момента, когда его условие синхронизации (сопряженное с прохождением соответствующих контрольных точек) станет истинным» [10]. Соответствующее абстрактному модулю вида $m = [\alpha](A \vee R)$ применение «синхронизатора» дано ниже. Продолжая определять гибридную технологию, рассмотрим работу модуля p, реализующего следующее логико-алгебраическое выражение:

$$p = [Trans(S_i, S_j)](\{Trans(S_i, S_j) \leftarrow false; S_j; Trans(S_j, S_k) \leftarrow true\} \vee R),$$

где S_j – оператор, который должен быть выполнен на узле y_q вычислительной сети, т.е. $Depl(S_j) = y_q$; S_i – оператор, передающий управление оператору S_j по сети; *Trans* – бинарный предикат, описывающий причинно-следственные связи между операторами РОС, т.е. истинность высказывания $Trans(S_i, S_j)$ означает, что передано управление оператору S_j от оператора S_i. При ложном значении высказывания $Trans(S_i, S_j)$ в модуле p управление передается оператору возврата R, который переводит модуль в режим ожидания истинности высказывания $Trans(S_i, S_j)$ путем опроса входных портов данного сетевого узла либо, проявляя активность, запрашивает значение этого высказывания, отправляя соответствующее служебное сообщение. При получении истинного значения высказывания $Trans(S_i, S_j)$, т.е. после успешной передачи управления

оператору S_j от оператора S_i, в модуле p выполняются действия, описанные в фигурных скобках. Сначала выполняется модификация предиката *Trans*: $Trans(S_i, S_j) \leftarrow false$ для предотвращения повторного запуска модуля. Далее выполняется оператор S_j, а затем путем модификации бинарного предиката *Trans*: $Trans(S_j, S_k) \leftarrow true$ управление передается оператору S_k. Точками с запятыми в фигурных скобках обозначены операции конкатенации, или умножения, операторов.

Сетевую интерпретацию абстрактного модуля p, принимающего управление от оператора S_i, затем выполняющего оператор S_j и, наконец, передающего управление оператору S_k, можно задать явно в следующем виде, используя функцию *Depl*:

$$p' = [Trans(Depl(S_i), Depl(S_j))](\{Trans(Depl(S_i), Depl(S_j)) \leftarrow false;\ S_j;$$

$$Trans(Depl(S_j), Depl(S_k)) \leftarrow true\} \vee R),$$

что соответствует методу реализации передачи управления через коммуникационную среду для случая, когда всем программным модулям известны адреса размещения операторов, которым передается управление.

3. Операции, связанные с виртуальным *VS*-пространством, и особенности передачи управления в гибридной модели

Связи по управлению. В распределенной Linda-подобной системе управление может быть передано через виртуальное *VS*-пространство следующим образом. Пусть, например, модуль r, реализующий оператор S_r, собирается передать управление модулю q, реализующему оператор S_q. Модуль r выполняет модификацию предиката $Trans(S_r, S_q) \leftarrow true$, что соответствует помещению кортежа <“*Trans*”, “S_r”, “S_q”> в виртуальное пространство *VS*. Модуль q, готовый к приему управления, отыскивает в *VS*-пространстве одноименный кортеж “*Trans*”, второй элемент которого равен “S_r”, а третий – “S_q”. В логико-алгебраической модели этот ассоциативный поиск соответствует проверке истинности высказывания $Trans(S_r, S_q)$ соответствующим модулем q. Такая передача управления требует значительных затрат времени и не всегда оправдана. Поэтому при помощи **определения 2** нами сформулировано предложение по использованию гибридного варианта распределенной технологии, когда связи по управлению реализуются через коммуникационное пространство, а функциональные связи – через *VS*-пространство.

В дальнейшем, когда в логико-алгебраических выражениях не будет явно использоваться функция *Depl*, будут специально оговариваться методы передачи управления – с использованием явной адресации модулей (или операторов) с последующей передачей управляющих сообщений либо с использованием ассоциативного поиска нужного кортежа в *VS*-пространстве.

Функциональные связи. Рассмотрим множество операций L = {Put, Take, Copy}, связанных с абстрактным *VS*-пространством виртуальной структурированной памяти. Описанные выше действия по организации связей по управлению могут быть реализованы при помощи этих операторов. Перейдем теперь к рассмотрению реализации функциональных связей и свяжем формальное описание работы программных модулей с операциями из множества L. В логико-алгебраической модели операциям Put, Take и Copy поставлены в соответствие одноименные бинарные предикаты *Put*, *Take* и *Copy*, значения

которых могут модифицироваться (для различения имен будем использовать курсив). Рассмотрим выполнение операции Put. В формальной модели операция размещения элемента данных *d* в *VS*-пространстве при выполнении программного модуля для оператора S_i отмечается модификацией предиката $Put(S_i, d) \leftarrow true$, а модификация $Put(S_i, d) \leftarrow false$ соответствует завершению операции размещения. В реальной системе этой модификации соответствует размещение в *VS*-пространстве кортежа <"*d*", *Val*(*d*)>, где в кавычки заключено имя *d* элемента данных (переменной, кортежа или отношения), а в качестве второго элемента указывается значение *Val*(*d*) этого элемента (значение переменной, конкретный кортеж или конкретное отношение). Поскольку *VS*-пространство является виртуальным, кортеж можно «хранить» в нем постоянно, если в него добавить третий элемент – логическую константу *true* или *false* и хранить этот кортеж в следующем виде: <"*d*", *Val*(*d*), *true*>. Значению *true* в качестве третьего элемента кортежа будет соответствовать его виртуальное «наличие» в *VS*-пространстве, а *false* – его «отсутствие». Теперь модификации бинарного предиката $Take(S_i, d) \leftarrow true$ будет соответствовать последующее состояние кортежа <"*d*", *Val*(*d*), *false*>, означающее его виртуальное «изъятие» из *VS*-пространства. После выполнения модификации бинарного предиката $Copy(S_i, d) \leftarrow true$ имя "*d*" и значение *Val*(*d*) будут скопированы, если третьим элементом кортежа было *true*, при этом состояние самого кортежа <"*d*", *Val*(*d*), *true*> осталось прежним.

Если вторым элементом кортежа является *Ref*(*d*), т.е. кортеж имеет вид <"*d*", *Ref*(*d*), *true*>, то это означает, что указано не значение *d*, а ссылка на место (адрес) хранения этого значения в сети. Мнемонику операций Put, Take и Copy дополняет рис. 1,***б***.

В силу неинтерпретированности модели определенных кортежем распределенных операторных сетей при реализации распределенного сетевого приложения программист может выбирать удобную сетевую технологию программирования, ориентированную на распространенные модели взаимодействия: удаленный вызов процедур, удаленное обращение к методам, ориентированный на сообщения промежуточный уровень и потоки данных, описанные, например, в работе [2]. Для связи посредством сообщений возможно, например, использовать следующие примитивы для интерфейса сокетов: Socket, Bind, Listen, Accept, Connect, Send, Receive, Close; для другой распространенной модели – интерфейса передачи сообщений MPI, для высокоскоростных взаимодействующих сетей используются примитивы: MPI_bsend, MPI_send, MPI_ssend, MPI_sendrecv, MPI_isend, MPI_issend, MPI_recv, MPI_irecv [2]. Естественно, при передаче информации по сети задействованы стеки сетевых протоколов, например, стек TCP/IP.

Назначение и обоснование использования других введенных в **определении 2** понятий будут даны на последующем примере.

4. Логико-алгебраическое описание распределенных операторных сетей

В качестве показательного полнофункционального примера, иллюстрирующего все основные особенности построения и формализации РОС, рассмотрим задачу перехода от сосредоточенной операторной схемы (рис. 2) из работы [22, рис. 3 на стр. 211] к ее распределенному варианту в виде РОС.

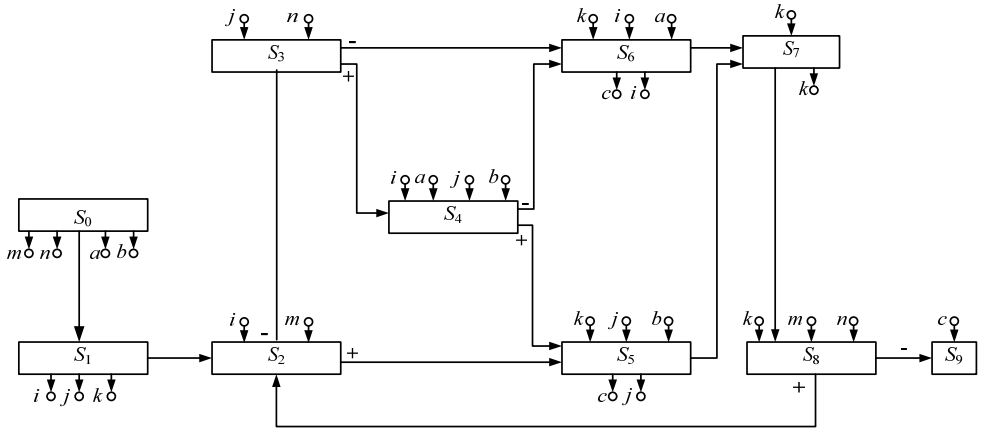

Рис. 2. Пример операторной схемы РОС-1 (SEQUENTIAL)

Операторная схема, представленная на рис. 2, построена в соответствии с **определением 1** для следующего фрагмента сосредоточенной программы из [22], записанной на псевдокоде:

	Read(*m*, *n*, *a*, *b*);	{S_0}
	$i := j := k := 1$	{S_1}
C:	**if** $i > m$ **then go to** *B*;	{S_2}
	if $j > n$ **then go to** *A*;	{S_3}
	if $a[i] > b[j]$ **then**	{S_4}
B:	**begin** $c[k] := b[j]$; $j := j + 1$ **end**	{S_5}
	else	
A:	**begin** $c[k] := a[i]$; $i := i + 1$ **end**	{S_6}
	$k := k + 1$;	{S_7}
	if $k <= m + n$ **then** **go to** *C*;	{S_8}
	Write(*c*)	{S_9}

Данная программа позволит продемонстрировать многие особенности предлагаемой «гибридной» технологии распределенного программирования, основанной на предложенном определении распределенной операторной сети и концепции абстрактной среды для ее реализации.

Рассмотрим распределенную операторную сеть РОС-1 (SEQUENTIAL), построенную по операторной схеме, представленной на рис. 2). Логико-алгебраические выражения, описывающие связи по управлению, т.е. логику управления ходом выполнения операторов для РОС-1, имеют следующий вид:

$m_{S0} = [Start](\{Start \leftarrow false;\ S_0;\ T(S_0, S_1) \leftarrow true\} \vee R);$
$m_{S1} = [T(S_0, S_1)](\{T(S_0, S_1) \leftarrow false;\ S_1;\ T(S_1, S_2) \leftarrow true\} \vee R);$
$m_{S2} = [T(S_1, S_2) \vee T(S_8, S_2)](\{T(S_1, S_2) \leftarrow false;\ T(S_8, S_2) \leftarrow false;\ S_2;$
$[Q(S_2)](T(S_2, S_5) \leftarrow true \vee T(S_2, S_3) \leftarrow true)\} \vee R);$
$m_{S3} = [T(S_2, S_3)](\{T(S_2, S_3) \leftarrow false;\ S_3;\ [Q(S_3)](T(S_3, S_4) \leftarrow true \vee$
$T(S_3, S_6) \leftarrow true)\} \vee R);$
$m_{S4} = [T(S_3, S_4)](\{T(S_3, S_4) \leftarrow false;\ S_4;\ [Q(S_4)](T(S_4, S_5) \leftarrow true \vee$
$T(S_4, S_6) \leftarrow true)\} \vee R);$
$m_{S5} = [T(S_2, S_5) \vee T(S_4, S_5)](\{T(S_2, S_5) \leftarrow false;\ T(S_4, S_5) \leftarrow false;\ S_5;$
$T(S_5, S_7) \leftarrow true\} \vee R);$
$m_{S6} = [T(S_3, S_6) \vee T(S_4, S_6)](\{T(S_3, S_6) \leftarrow false;\ T(S_4, S_6) \leftarrow false;\ S_6;$
$T(S_6, S_7) \leftarrow true\} \vee R);$
$m_{S7} = [T(S_5, S_7) \vee T(S_6, S_7)](\{T(S_5, S_7) \leftarrow false;\ T(S_6, S_7) \leftarrow false;\ S_7;$
$T(S_7, S_8) \leftarrow true\} \vee R);$
$m_{S8} = [T(S_7, S_8)](\{T(S_7, S_8) \leftarrow false;\ S_8;\ [Q(S_8)](T(S_8, S_2) \leftarrow true \vee$
$T(S_8, S_9) \leftarrow true)\} \vee R);$
$m_{S9} = [T(S_8, S_9)](\{T(S_8, S_9) \leftarrow false;\ S_9;\ Finish \leftarrow true\ \} \vee R).$

Данные выражения описывают выполнение действий программно реализуемыми абстрактными модулями из множества M согласно **определению 2** и в соответствии с операторной схемой РОС-1, реализуемой в распределенной среде с архитектурой, представленной на рис. 1.

5. Формализованные спецификации на основе концептуальных и логико-алгебраических представлений гибридных моделей распределенных систем

Для иллюстрации методики построения распределенного приложения рассмотрена операторная схема, представленная на рис. 2. Построение новых логико-алгебраических моделей в соответствии с **определением 2** для новой распределенной операторной сети РОС-1* (DISTRIBUTED-DON) похоже на построение модели РОС-1 (SEQUENTIAL), но отличается от нее тем, что при составлении логических правил учтены соответствующие операторам концептуальные графы, а логико-алгебраические выражения (формализованные спецификации) строятся непосредственно по правилам. Главной же отличительной особенностью новых алгебраических выражений для модулей распределенного приложения здесь является учет особенностей работы со структурированным виртуальным *VS*-пространством и со связанными с ним операциями Put, Take и Copy.

Для упрощения процесса создания распределенных приложений зададим правила использования виртуального пространства данных в виде формул в исчислении предикатов первого порядка. Операциям Put, Take и Copy поставим в соответствие одноименные бинарные предикаты $Put(s, x)$, $Take(s, x)$ и $Copy(s, x)$, определенные на предметных переменных $s \in S$ и $x \in X$, где S и X – определенные ранее множества операторов и переменных соответственно. Эти же предметные переменные, как следует из **определения 2** для сетей класса *DON*, определены также и для бинарных предикатов: $Arg(s, x)$ – x является аргументом для оператора s, $Res(s, x)$ – x является результатом выполнения оператора s. Добавим к перечисленным предикатам унарные предика-

ты $Work(s)$, $Q(s)$ и $Trans(s', s'')$, $s' \neq s''$, $s' \in S$, $s'' \in S$. Истинность высказывания $Work(s_i)$ свидетельствует о выполнении оператора s_i. Значение высказывания $Q(s_i)$ формируется по завершении выполнения оператора s_i; тот факт, что это значение сформировано, обозначим при помощи унарного предиката $Form(s)$, $s \in S$. Так, истинность высказывания $Form(s_i)$ будет означать, что истинное или ложное значение высказывания $Q(s_i)$ сформировано по завершении выполнения оператора s_i. Назначение бинарного предиката $Trans(s', s'')$ было дано в **определении 2**.

Правила для описания логики работы программного модуля, реализующего оператор S_0 в распределенной среде, представим в виде следующих формул:

$Start \supset Work(S_0)$;

$Work(S_0)$ & $\neg Arg(S_0, m)$ & $Res(S_0, m)$ & $\neg Arg(S_0, n)$ & $Res(S_0, n)$ & & $\neg Arg(S_0, a)$ & $Res(S_0, a)$ & $\neg Arg(S_0, b)$ & $Res(S_0, b) \supset$ $\supset Oper(S_0, Read(m, n, a, b))$;

$Oper(S_0, Read(m, n, a, b)) \supset Put(S_0, m)$ & $Put(S_0, n)$ & $Put(S_0, a)$ & &$Put(S_0, b)$;

$Put(S_0, m)$ & $Put(S_0, n)$ & $Put(S_0, a)$ & $Put(S_0, b) \supset Trans(S_0, S_1)$ & & $\neg Work(S_0)$.

Приведенные формулы, каждая из которых содержит антецедент (посылку) и консеквент (заключение), связаны символом импликации. Истинность формул соответствует успешным завершениям конкретных действий в операторной сети. Логические выражения содержат атомарные константные формулы исчисления предикатов первого порядка и пропозициональную переменную *Start*. Антецеденты описывают условия, при которых возможно совершение дальнейших действий, а консеквенты описывают сами эти действия.

Дадим содержательное описание, предварявшее составление данных формул, представляющих логическую модель представления знаний о процессе выполнения оператора S_0 при учете «накладных расходов» на его выполнение в распределенной среде.

Первая формула: после получения инструкции *Start* о начале работы операторной сети начинается работа по выполнению оператора S_0, отмечаемая истинностью высказывания $Work(S_0)$.

Вторая формула: если работа по выполнению оператора S_0 началась и операнды – переменные m, n и одномерные массивы a и b – должны быть результатами, но не являются аргументами при выполнении оператора, то на узле сети $Depl(S_0) = Y_0$ (здесь и далее значение унарной функции *Depl* согласно **определению 2** должно соответствовать распределению операторов по узлам, заданному на рис. 1,***а***; в распределенной программе это значение интерпретируется как адрес узла Y_0) должна быть выполнена операция чтения $Read(m, n, a, b)$, чему соответствует истинность высказывания $Oper(S_0, Read(m, n, a, b))$.

Третья формула: после выполнения операции $Read(m, n, a, b)$ значения или ссылки на переменные m, n и одномерные массивы a и b должны быть помещены в *VS*-пространство, т.е. после выполнения операции в данном пространстве должны находиться следующие кортежи:

<“*m*”, *Val*(*m*), *true*>;
<“*n*”, *Val*(*n*), *true*>;
<“*a*”, *Val*(*a*), *true*>;
<“*b*”, *Val*(*b*), *true*>,

или, в случае хранения ссылок, кортежи:

<“*m*”, *Ref*(*m*), *true*>;
<“*n*”, *Ref*(*n*), *true*>;
<“*a*”, *Ref*(*a*), *true*>;
<“*b*”, *Ref*(*b*), *true*>.

Факты присутствия данных кортежей в *VS*-пространстве подтверждаются в логической модели истинностью высказываний $Put(S_0, m)$, $Put(S_0, n)$, $Put(S_0, a)$ и $Put(S_0, b)$.

Четвертая формула: если результаты помещены в *VS*-пространство, то необходимо передать управление следующему оператору, т.е., как следствие, должно быть истинным высказывание $Trans(S_0, S_1)$ и ложным высказывание $Work(S_0)$.

Рассмотренная логическая модель представления знаний о выполнении оператора S_0 в распределенной среде имеет в основном декларативный характер, а процедурная составляющая здесь задана неявно. Хотя подобная модель полезна на ранних этапах проектирования программного продукта, обладая ясной формальной семантикой и операционной поддержкой в виде механизмов вывода, для непосредственного программирования такой модели недостаточно, так как она имеет высокий уровень абстракции и не относится к классу исполнимых (операционных) моделей. Для усиления процедурной составляющей модели представления знаний далее будет использоваться язык сетей абстрактных машин в интерпретации работ [15, 16].

Далее для новых выражений для модулей РОС-1* будут использоваться имена M_{Si} вместо прежних имен в РОС-1 m_{Si}, $i = 0, 1, \ldots, 9$. Логико-алгебраическое выражение (формализованная спецификация) для рабочей программы, реализующей модуль M_{S0} и построенное с учетом приведенных выше правил, имеет следующий вид:

$$M_{S0} = \{[Start](\{Start \leftarrow false;\ Work(S_0) \leftarrow true\} \vee R);$$
$$[Work(S_0)\ \&\ \neg Arg(S_0, m)\ \&\ Res(S_0, m)\ \&\ \neg Arg(S_0, n)\ \&\ Res(S_0, n)\ \&$$
$$\&\ \neg Arg(S_0, a)\ \&\ Res(S_0, a)\ \&\ \neg Arg(S_0, b)\ \&\ Res(S_0, b)]$$
$$(Oper(S_0, Read(m, n, a, b)) \leftarrow true \vee N);$$
$$[Oper(S_0, Read(m, n, a, b))](Put(S_0, m) \leftarrow true;\ Put(S_0, n) \leftarrow true;$$
$$Put(S_0, a) \leftarrow true;\ Put(S_0, b) \leftarrow true;\ Oper(S_0, Read(m, n, a, b)) \leftarrow false \vee N);$$
$$[Put(S_0, m)\ \&\ Put(S_0, n)\ \&\ Put(S_0, a)\ \&\ Put(S_0, b)](\{Put(S_0, m) \leftarrow false;$$
$$Put(S_0, n) \leftarrow false;\ Put(S_0, a) \leftarrow false;\ Put(S_0, b) \leftarrow false;$$
$$Trans(Depl(S_0), Depl(S_1)) \leftarrow true;\ Work(S_0) \leftarrow false\} \vee N)\}.$$

Данное выражение построено на основе операции α-дизъюнкции и композиции операторов. Операционная семантика задается операциями модификации предикатов. Присваивание значения *true* при этом соответствует началу определенной операции, а присваивание значения *false* – ее окончанию. Семантика операций соответствует предикатным именам. Данное выражение легко программируется, результатом чего является структурированный программный модуль.

Остальные концептуальные графы, правила и логико-алгебраические выражения для модулей распределенного приложения формируются таким же образом.

Правила для оператора S_1:

$Trans(S_0, S_1) \supset Work(S_1)$;

$Work(S_1)$ & $\neg Arg(S_1, i)$ & $\neg Arg(S_1, j)$ & $\neg Arg(S_1, k)$ & $Res(S_1, i)$ &

& $Res(S_1, j)$ & $Res(S_1, k) \supset Oper(S_1, \{i{:=}j{:=}k :=1\})$;

$Oper(S_1, \{i{:=}j{:=}k{:=}1\}) \supset Put(S_1, i)$ & $Put(S_1, j)$ & $Put(S_1, k)$;

$Put(S_1, i)$ & $Put(S_1, j)$ & $Put(S_1, k) \supset Trans(S_1, S_2)$ & $\neg Work(S_1)$.

Логико-алгебраическое выражение для модуля M_{S1}:

$M_{S1} = \{[Trans(Depl(S_0), Depl(S_1))](\{Trans(Depl(S_0), Depl(S_1)) \leftarrow false;$

$Work(S_1) \leftarrow true\} \vee R)$;

$[Work(S_1)$ & $\neg Arg(S_1, i)$ & $\neg Arg(S_1, j)$ & $\neg Arg(S_1, k)$ & $Res(S_1, i)$ &

& $Res(S_1, j)$ & $Res(S_1, k)](Oper(S_1, \{i{:=}j{:=}k :=1\}) \leftarrow true \vee N)$;

$[Oper(S_1, \{i{:=}j{:=}k{:=}1\})](\{Put(S_1, i) \leftarrow true; Put(S_1, j) \leftarrow true;$

$Put(S_1, k) \leftarrow true; Oper(S_1, \{i{:=}j{:=}k :=1\}) \leftarrow false\} \vee N)$;

$[Put(S_1, i)$ & $Put(S_1, j)$ & $Put(S_1, k)](\{Put(S_1, i) \leftarrow false; Put(S_1, j) \leftarrow false;$

$Put(S_1, k) \leftarrow false; Trans(Depl(S_1), Depl(S_2)) \leftarrow true; Work(S_1) \leftarrow false\} \vee N)\}$.

Правила для оператора S_2:

$Trans(S_1, S_2) \vee Trans(S_8, S_2) \supset Work(S_2)$;

$Work(S_2)$ & $Arg(S_2, i)$ & $\neg Res(S_2, i) \supset Copy(S_2, i)$;

$Work(S_2)$ & $Arg(S_2, m)$ & $\neg Res(S_2, m) \supset Copy(S_2, m)$;

$Copy(S_2, i)$ & $Copy(S_2, m) \supset Oper(S_2, Q(S_2) := i > m)$;

$Oper(S_2, Q(S_2) := i > m) \supset Form(S_2)$;

$Form(S_2)$ & $Q(S_2) \supset Trans(S_2, S_5)$ & $\neg Work(S_2)$;

$Form(S_2)$ & $\neg Q(S_2) \supset Trans(S_2, S_3)$ & $\neg Work(S_2)$.

Логико-алгебраическое выражение для модуля M_{S2}:

$M_{S2} = \{[Trans(Depl(S_1), Depl(S_2)) \vee Trans(Depl(S_8), Depl(S_2))]$

$(\{Trans(Depl(S_1), Depl(S_2)) \leftarrow false; Trans(Depl(S_8), Depl(S_2)) \leftarrow false;$

$Work(S_2) \leftarrow true\} \vee R)$;

$[Work(S_2)$ & $Arg(S_2, i)$ & $\neg Res(S_2, i)](Copy(S_2, i) \leftarrow true \vee N)$;

$[Work(S_2)$ & $Arg(S_2, m)$ & $\neg Res(S_2, m)](Copy(S_2, m) \leftarrow true \vee N)$;

$[Copy(S_2, i)$ & $Copy(S_2, m)](Oper(S_2, Q(S_2) := i > m) \leftarrow true \vee N)$;

$[Oper(S_2, Q(S_2) := i > m)](Form(S_2) \leftarrow true \vee N)$;

$[Form(S_2)](\{Copy(S_2, i) \leftarrow false; Copy(S_2, m) \leftarrow false;$

$Oper(S_2, Q(S_2) := i > m) \leftarrow false; Form(S_2) \leftarrow false;$

$[Q(S_2)](\{Trans(Depl(S_2), Depl((S_5)) \leftarrow true;$

$Work(S_2) \leftarrow false\} \vee \{Trans(Depl(S_2), Depl(S_3)) \leftarrow true;$

$Work(S_2) \leftarrow false\})\} \vee N)\}$.

Правила для оператора S_3:

$Trans(S_2, S_3) \supset Work(S_3)$;

$Work(S_3)$ & $Arg(S_3, j)$ & $\neg Res(S_3, j) \supset Copy(S_3, j)$;

$Work(S_3)$ & $Arg(S_3, n)$ & $\neg Res(S_3, n) \supset Copy(S_3, n)$;

$Copy(S_3, j)$ & $Copy(S_3, n) \supset Oper(S_3, Q(S_3) := j > n)$;

$Oper(S_3, Q(S_3) := j > n) \supset Form(S_3)$;

$Form(S_3)$ & $Q(S_3) \supset Trans(S_3, S_4)$ & $\neg Work(S_3)$;

$Form(S_3)$ & $\neg Q(S_3) \supset Trans(S_3, S_6)$ & $\neg Work(S_3)$.

Логико-алгебраическое выражение для модуля M_{S3}:
$M_{S3} = \{[Trans(Depl(S_2), Depl(S_3))]$
$(\{Trans(Depl(S_2), Depl(S_3)) \leftarrow false; Work(S_3) \leftarrow true\} \vee R);$
$[Work(S_3) \,\&\, Arg(S_3, j) \,\&\, \neg Res(S_3, j)](Copy(S_3, j) \leftarrow true \vee N);$
$[Work(S_3) \,\&\, Arg(S_3, n) \,\&\, \neg Res(S_3, n)](Copy(S_3, n) \leftarrow true \vee N);$
$[Copy(S_3, j) \,\&\, Copy(S_3, n)](Oper(S_3, Q(S_3) := j > n) \leftarrow true \vee N);$
$[Oper(S_3, Q(S_3) := j > n)](Form(S_3) \leftarrow true \vee N);$
$[Form(S_3)](\{Copy(S_3, j) \leftarrow false; Copy(S_3, n) \leftarrow false;$
$Oper(S_3, Q(S_3) := j > n) \leftarrow false; Form(S_3) \leftarrow false;$
$[Q(S_3)](\{Trans(Depl(S_3), Depl(S_4)) \leftarrow true;$
$Work(S_3) \leftarrow false\} \vee \{Trans(Depl(S_3), Depl(S_6)) \leftarrow true;$
$Work(S_3) \leftarrow false\}) \} \vee N)\}.$
Правила для оператора S_4:
$Trans(S_3, S_4) \supset Work(S_4);$
$Work(S_4) \,\&\, Arg(S_4, i) \,\&\, \neg Res(S_4, i) \supset Copy(S_4, i);$
$Work(S_4) \,\&\, Arg(S_4, a) \,\&\, \neg Res(S_4, a) \supset Copy(S_4, a);$
$Work(S_4) \,\&\, Arg(S_4, j) \,\&\, \neg Res(S_4, j) \supset Copy(S_4, j);$
$Work(S_4) \,\&\, Arg(S_4, b) \,\&\, \neg Res(S_4, b) \supset Copy(S_4, b);$
$Copy(S_4, i) \,\&\, Copy(S_4, a) \,\&\, Copy(S_4, j) \,\&\, Copy(S_4, b) \supset$
$\supset Oper(S_4, Q(S_4) := a[i] > b[j]);$
$Oper(S_4, Q(S_4) := a[i] > b[j]) \supset Form(S_4);$
$Form(S_4) \,\&\, Q(S_4) \supset Trans(S_4, S_5) \,\&\, \neg Work(S_4);$
$Form(S_4) \,\&\, \neg Q(S_4) \supset Trans(S_4, S_6) \,\&\, \neg Work(S_4).$
Логико-алгебраическое выражение для модуля M_{S4}:
$M_{S4} = \{[Trans(Depl(S_3), Depl(S_4))]$
$(Trans(Depl(S_3), Depl(S_4)) \leftarrow false; Work(S_4) \leftarrow true\} \vee R);$
$[Work(S_4) \,\&\, Arg(S_4, i) \,\&\, \neg Res(S_4, i)](Copy(S_4, i) \leftarrow true \vee N);$
$[Work(S_4) \,\&\, Arg(S_4, a) \,\&\, \neg Res(S_4, a)](Copy(S_4, a) \leftarrow true \vee N);$
$[Work(S_4) \,\&\, Arg(S_4, j) \,\&\, \neg Res(S_4, j)](Copy(S_4, j) \leftarrow true \vee N);$
$[Work(S_4) \,\&\, Arg(S_4, b) \,\&\, \neg Res(S_4, b)](Copy(S_4, b) \leftarrow true \vee N);$
$[Copy(S_4, i) \,\&\, Copy(S_4, a) \,\&\, Copy(S_4, j) \,\&\, Copy(S_4, b)]$
$(Oper(S_4, Q(S_4) := a[i] > b[j]) \leftarrow true \vee N);$
$[Oper(S_4, Q(S_4) := a[i] > b[j])](Form(S_4) \leftarrow true \vee N);$
$[Form(S_4)](\{Copy(S_4, i) \leftarrow false; Copy(S_4, a) \leftarrow false; Copy(S_4, j) \leftarrow false;$
$Copy(S_4, b) \leftarrow false; Oper(S_4, Q(S_4) := a[i] > b[j]) \leftarrow false; Form(S_4) \leftarrow false;$
$[Q(S_4)](\{Trans(Depl(S_4), Depl(S_5)) \leftarrow true;$
$Work(S_4) \leftarrow false\} \vee \{Trans(Depl(S_4), Depl(S_6)) \leftarrow true;$
$Work(S_4) \leftarrow false\}) \} \vee N)\}.$
Правила для оператора S_5:
$Trans(S_2, S_5) \vee Trans(S_4, S_5) \supset Work(S_5);$
$Work(S_5) \,\&\, Arg(S_5, k) \,\&\, \neg Res(S_5, k) \supset Copy(S_5, k);$
$Work(S_5) \,\&\, Arg(S_5, j) \,\&\, Res(S_5, j) \supset Take(S_5, j);$
$Work(S_5) \,\&\, Arg(S_5, b) \,\&\, \neg Res(S_5, b) \supset Copy(S_5, b);$
$Copy(S_5, k) \,\&\, Take(S_5, j) \,\&\, Copy(S_5, b) \supset$
$\supset Oper(S_5, \{c[k] := b[j]; j := j + 1\});$
$Oper(S_5, \{c[k] := b[j]; j := j + 1\}) \supset Put(S_5, c) \,\&\, Put(S_5, j);$
$Put(S_5, c) \,\&\, Put(S_5, j) \supset Trans(S_5, S_7) \,\&\, \neg Work(S_5).$

Логико-алгебраическое выражение для модуля M_{S5}:
$M_{S5} = \{[Trans(Depl(S_2), Depl(S_5)) \vee Trans(Depl(S_4), Depl(S_5))]$
$(Trans(Depl(S_2), Depl(S_5)) \leftarrow false; Trans(Depl(S_4), Depl(S_5)) \leftarrow false;$
$Work(S_5) \leftarrow true\} \vee R);$
$[Work(S_5) \& Arg(S_5, k) \& \neg Res(S_5, k)](Copy(S_5, k) \leftarrow true \vee N);$
$[Work(S_5) \& Arg(S_5, j) \& Res(S_5, j)](Take(S_5, j) \leftarrow true \vee N);$
$[Work(S_5) \& Arg(S_5, b) \& \neg Res(S_5, b)](Copy(S_5, b) \leftarrow true \vee N);$
$[Copy(S_5, k) \& Take(S_5, j) \& Copy(S_5, b)]$
$(\{Oper(S_5, \{c[k] := b[j]; j := j + 1\}) \leftarrow true; Copy(S_5, k) \leftarrow false;$
$Take(S_5, j) \leftarrow false; Copy(S_5, b) \leftarrow false\} \vee N);$
$[Oper(S_5, \{c[k] := b[j]; j := j + 1\})](\{Put(S_5, c) \leftarrow true; Put(S_5, j) \leftarrow true\} \vee N);$
$[Put(S_5, c) \& Put(S_5, j)](\{Put(S_5, c) \leftarrow false; Put(S_5, j) \leftarrow false;$
$Oper(S_5, \{c[k] := b[j]; j := j + 1\}) \leftarrow false; Trans(Depl(S_5), Depl(S_7)) \leftarrow true;$
$Work(S_5) \leftarrow false\} \vee N)\}.$

Правила для оператора S_6:
$Trans(S_3, S_6) \vee Trans(S_4, S_6) \supset Work(S_6);$
$Work(S_6) \& Arg(S_6, k) \& \neg Res(S_6, k) \supset Copy(S_6, k);$
$Work(S_6) \& Arg(S_6, i) \& Res(S_6, i) \supset Take(S_6, i);$
$Work(S_6) \& Arg(S_6, a) \& \neg Res(S_6, a) \supset Copy(S_6, a);$
$Copy(S_6, k) \& Take(S_6, i) \& Copy(S_6, a) \supset$
$\supset Oper(S_6, \{c[k] := a[i]; i := i + 1\});$
$Oper(S_6, \{c[k] := a[i]; i := i + 1\}) \supset Put(S_6, c) \& Put(S_6, i);$
$Put(S_6, c) \& Put(S_6, i) \supset Trans(S_6, S_7) \& \neg Work(S_6).$

Логико-алгебраическое выражение для модуля M_{S6}:
$M_{S6} = \{[Trans(Depl(S_3), Depl(S_6)) \vee Trans(Depl(S_4), Depl(S_6))]$
$(\{Trans(Depl(S_3), Depl(S_6)) \leftarrow false;$
$Trans(Depl(S_4), Depl(S_6)) \leftarrow false; Work(S_6) \leftarrow true\} \vee R);$
$[Work(S_6) \& Arg(S_6, k) \& \neg Res(S_6, k)](Copy(S_6, k) \leftarrow true \vee N);$
$[Work(S_6) \& Arg(S_6, i) \& Res(S_6, i)](Take(S_6, i) \leftarrow true \vee N);$
$[Work(S_6) \& Arg(S_6, a) \& \neg Res(S_6, a)](Copy(S_6, a) \leftarrow true \vee N);$
$[Copy(S_6, k) \& Take(S_6, i) \& Copy(S_6, a)]$
$(Oper(S_6, \{c[k] := a[i]; i := i + 1\}) \leftarrow true \vee N);$
$[Oper(S_6, \{c[k] := a[i]; i := i + 1\})](\{Copy(S_6, k) \leftarrow false; Take(S_6, i) \leftarrow false;$
$Copy(S_6, a) \leftarrow false; Put(S_6, c) \leftarrow true; Put(S_6, i) \leftarrow true;$
$Oper(S_6, \{c[k] := a[i]; i := i + 1\}) \leftarrow false\} \vee N);$
$[Put(S_6, c) \& Put(S_6, i)](\{Put(S_6, c) \leftarrow false; Put(S_6, i) \leftarrow false;$
$Trans(Depl(S_6), Depl(S_7)) \leftarrow true; Work(S_6) \leftarrow false\} \vee N)\}.$

Правила для оператора S_7:
$Trans(S_5, S_7) \vee Trans(S_6, S_7) \supset Work(S_7);$
$Work(S_7) \& Arg(S_7, k) \& Res(S_7, k) \supset Take(S_7, k);$
$Take(S_7, k) \supset Oper(S_7, k := k + 1);$
$Oper(S_7, k := k + 1) \supset Put(S_7, k);$
$Put(S_7, k) \supset Trans(S_7, S_8) \& \neg Work(S_7).$

Логико-алгебраическое выражение для модуля M_{S7}:
$M_{S7} = \{[Trans(Depl(S_5), Depl(S_7)) \vee Trans(Depl(S_6), Depl(S_7))]$
$(\{Trans(Depl(S_5), Depl(S_7)) \leftarrow false;$

$Trans(Depl(S_6), Depl(S_7)) \leftarrow false; Work(S_7) \leftarrow true\} \vee R);$
$[Work(S_7)\ \&\ Arg(S_7, k)\ \&\ Res(S_7, k)](Take(S_7, k) \leftarrow true \vee N);$
$[Take(S_7, k)](Oper(S_7, k := k + 1) \leftarrow true \vee N);$
$[Oper(S_7, k := k + 1)](Put(S_7, k) \leftarrow true \vee N);$
$[Put(S_7, k)](\{Take(S_7, k) \leftarrow false; Oper(S_7, k := k + 1) \leftarrow false;$
$Put(S_7, k) \leftarrow false; Trans(Depl(S_7), Depl(S_8)) \leftarrow true; Work(S_7) \leftarrow false\} \vee N)\}.$

Правила для оператора S_8:

$Trans(S_7, S_8) \supset Work(S_8);$
$Work(S_8)\ \&\ Arg(S_8, k)\ \&\ \neg Res(S_8, k) \supset Copy(S_8, k);$
$Work(S_8)\ \&\ Arg(S_8, j)\ \&\ \neg Res(S_8, j) \supset Copy(S_8, j);$
$Work(S_8)\ \&\ Arg(S_8, m)\ \&\ \neg Res(S_8, m) \supset Copy(S_8, m);$
$Copy(S_8, k)\ \&\ Copy(S_8, j)\ \&\ Copy(S_8, m) \supset Oper(S_8, Q(S_8) := k <= m + n);$
$Oper(S_8, Q(S_8) := k <= m + n) \supset Form(S_8);$
$Form(S_8)\ \&\ Q(S_8) \supset Trans(S_8, S_2)\ \&\ \neg Work(S_8);$
$Form(S_8)\ \&\ \neg Q(S_8) \supset Trans(S_8, S_9)\ \&\ \neg Work(S_8).$

Логико-алгебраическое выражение для модуля M_{S8}:

$M_{S8} = \{[Trans(Depl(S_7), Depl(S_8))]$
$(\{Trans(Depl(S_7), Depl(S_8)) \leftarrow false; Work(S_8) \leftarrow true\} \vee R);$
$[Work(S_8)\ \&\ Arg(S_8, k)\ \&\ \neg Res(S_8, k)](Copy(S_8, k) \leftarrow true \vee N);$
$[Work(S_8)\ \&\ Arg(S_8, j)\ \&\ \neg Res(S_8, j)](Copy(S_8, j) \leftarrow true \vee N);$
$[Work(S_8)\ \&\ Arg(S_8, m)\ \&\ \neg Res(S_8, m)](Copy(S_8, m) \leftarrow true \vee N);$
$[Copy(S_8, k)\ \&\ Copy(S_8, j)\ \&\ Copy(S_8, m)]$
$(Oper(S_8, Q(S_8) := k <= m + n) \leftarrow true \vee N);$
$[Oper(S_8, Q(S_8) := k <= m + n)](Form(S_8) \leftarrow true \vee N);$
$[Form(S_8)](\{Copy(S_8, k) \leftarrow false; Copy(S_8, j) \leftarrow false; Copy(S_8, m) \leftarrow false;$
$Form(S_8) \leftarrow false; Oper(S_8, Q(S_8) := k <= m + n) \leftarrow false;$
$[Q(S_8)](\{Trans(Depl(S_8), Depl(S_2)) \leftarrow true;$
$Work(S_8) \leftarrow false\} \vee \{Trans(Depl(S_8), Depl(S_9)) \leftarrow true;$
$Work(S_8) \leftarrow false\})\} \vee N)\}.$

Правила для заключительного оператора S_9:

$Trans(S_7, S_8) \supset Work(S_9);$
$Work(S_9)\ \&\ Arg(S_9, c)\ \&\ \neg Res(S_9, c) \supset Copy(S_9, c);$
$Copy(S_9, c) \supset Oper(S_9, Write(c))\ \&\ \neg Work(S_9)\ \&\ Finish.$

Логико-алгебраическое выражение для модуля M_{S9}:

$[Trans(Depl(S_7), Depl(S_8))](\{Trans(Depl(S_7), Depl(S_8)) \leftarrow false;$
$Work(S_9) \leftarrow true\} \vee R);$
$[Work(S_9)\ \&\ Arg(S_9, c)\ \&\ \neg Res(S_9, c)](Copy(S_9, c) \leftarrow true \vee N);$
$[Copy(S_9, c)](\{Oper(S_9, Write(c)) \leftarrow true; Work(S_9) \leftarrow false; Copy(S_9, c) \leftarrow false;$
$Oper(S_9, Write(c)) \leftarrow false; Finish \leftarrow true\}) \vee N)\}.$

Данные логико-алгебраические выражения построены непосредственно по правилам вывода и представляют собой формализованное описание, или спецификацию, некоторой абстрактной виртуальной машины. Такая модель уже относится к классу непосредственно исполнимых и представляет собой по существу программу для некоторой виртуальной машины. Спецификации, согласно одному из этапов применения «гибридной» технологии, должны

далее использоваться на практике при реализации соответствующего программного модуля и сводят простоту создания программного продукта до уровня семафорной техники. Элементы концептуального распределенного и параллельного программирования в сетях, а также примеры концептуальных графов для описания распределенных вычислений представлены в работах [25, 26].

Заключение

1. Предложена гибридная модель распределенной обработки данных в сетевой среде, отличающаяся от известных тем, что в ней передача управления осуществляется путем передачи сообщений через сетевое инфокоммуникационное пространство, а функциональные связи реализуются через структурированное виртуальное пространство памяти, что во многих случаях ускоряет обработку данных.

2. Формально определены распределенные операторные сети, причем в основу определения положены как известные теоретико-множественные понятия, так и понятия, связанные с использованием инфокоммуникационной среды и структурированным виртуальным пространством памяти, 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 гибридной архитектурой, 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. **Хьюз, К.** Параллельное и распределенное программирование на C++ / К. Хьюз, Т. Хьюз. – М. : Вильямс, 2004. – 672 с.
2. **Таненбаум, Э.** Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – СПб. : Питер, 2003. – 877 с.
3. **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.
4. **Плесневич, Г. С.** Логические модели / Г. С. Плесневич // Искусственный интеллект : справочник : в 3-х кн. Кн. 2. Модели и методы / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – С. 14–28.
5. **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.
6. **Котов, В. Е.** Сети Петри / В. Е. Котов. – М. : Наука, 1984. – 160 с.
7. **Питерсон, Дж.** Теория сетей Петри и моделирование систем / Дж. Питерсон. – М. : Мир, 1984. – 264 с.
8. **Кулагин, В. П.** Моделирование структур параллельных ВС на основе сетевых моделей / В. П. Кулагин. – М. : МИЭМ, 1998. – 102 с.

9. Капитонова, Ю. В. Математическая теория проектирования вычислительных систем / Ю. В. Капитонова, А. А. Летичевский. – М. : Наука, 1988. – 296 с.
10. **Глушков, В. М.** Методы символьной мультиобработки / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. – Киев : Наукова думка, 1980. – 252 с.
11. Using Abstract State Machines at Microsoft: A Case Study / M. Barnett, E. Börger, Y. Gurevich, W. Schulte, M. Veanes // Abstract State Machines: Theory and Apps. – Springer LNCS, 2000. – P. 367–380.
12. **Gurevich, Y.** Formalizing database recovery / Y. Gurevich, N. Soparkar, C. Wallace // Journal of Universal Computer Science. – 1997. – Vol. 3, № 4 – P. 320–340.
13. **Hoare, C. A. R.** Communicating sequential processes / C. A. R. Hoare // Commun. ACM. – 1978. – Vol. 21, № 8. – P. 666–677.
14. Логика и компьютер. Моделирование рассуждений и проверка правильности программ / Н. А. Алешина, А. М. Анисов, П. И. Быстров. – М. : Наука, 1990. – 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. **Gelernter, D.** Generative communication in Linda / D. Gelernter // ACM Computing Surveys. – 1985. – Vol. 7. – P. 80–112.
18. **Ahuja, S.** Linda and friends / S. Ahuja, N. Carriero, D. Gelernter // IEEE Computer. – 1986. – Vol. 7.19 (8). – P. 26–34.
19. Официальный сайт Scientific Computing Associates, Inc. [Электронный ресурс]. – URL: http://www.lindaspaces.com. (дата обращения: 04.04.2015).
20. **Котов, В. Е.** Теория схем программ / В. Е. Котов, В. К. Сабельфельд. – М. : Наука, 1991. – 248 с.
21. 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.
22. **Лавров, С. С.** Программирование. Математические основы, средства, теория / С. С. Лавров. – СПб. : БХВ-Петербург, 2001. – 320 с.
23. **Дейтел, Х. М.** Технология программирования на Java 2. Кн. 2. Распределенные приложения / Х. М. Дейтел, П. Дж. Дейтел, С. И. Сантри. – М. : Бином-Пресс, 2003. – 464 с.
24. **Wyckoff, P.** Tspaces / P. Wyckoff // IBM System Journal. – 1998. – Vol. 37. – P. 454–474.
25. **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. – Пенза : Изд-во ПГУ, 2015. – Т. 1. – С. 222–225.
26. **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. – Пенза : Изд-во ПГУ, 2015. – Т. 1. – С. 225–228.

References

1. Kh'yuz K., Kh'yuz T. *Parallel'noe i raspredelennoe programmirovanie na C++* [Parallel and distributed C++ programming]. Moscow: Vil'yams, 2004, 672 p.
2. Tanenbaum E., M. van Steen *Raspredelennye sistemy. Printsipy i paradigmy* [Distributed systems. Principles and paradigms]. Saint-Petersburg: Piter, 2003, 877 p.

3. 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 program systems]. Voronezh: Nauchnaya kniga, 2013, 220 p.
4. Plesnevich G. S. *Iskusstvennyy intellekt: spravochnik: v 3-kh kn. Kn. 2. Modeli i metody* [Artificial intelligence: reference book: in 3 books. Book 2. Models and methods]. Moscow: Radio i svyaz', 1990, pp. 14–28.
5. Kurganskiy V. I. *Sistemy upravleniya i informatsionnye tekhnologii* [Control systems and information technologies]. 2006, no. 3.1 (25), pp. 139–144.
6. Kotov V. E. *Seti Petri* [Petri nets]. Moscow: Nauka, 1984, 160 p.
7. Piterson Dzh. *Teoriya setey Petri i modelirovanie sistem* [Theory of Petri nets and system modeling]. Moscow: Mir, 1984, 264 p.
8. Kulagin V. P. *Modelirovanie struktur parallel'nykh VS na osnove setevykh modeley* [Modeling of parallel computing systems' structures on the basis of network modeling]. Moscow: MIEM, 1998, 102 p.
9. Kapitonova Yu. V., Letichevskiy A. A. *Matematicheskaya teoriya proektirovaniya vychislitel'nykh sistem* [Mathematical theory of computing system design]. Moscow: Nauka, 1988, 296 p.
10. Glushkov V. M., Tseytlin G. E., Yushchenko E. L. *Metody simvol'noy mul'tiobrabotki* [Methods of symbol multiprocessing]. Kiev: Naukova dumka, 1980, 252 p.
11. Barnett M., Börger E., Gurevich Y., Schulte W., Veanes M. *Abstract State Machines: Theory and Apps.* Springer LNCS, 2000, pp. 367–380.
12. Gurevich Y., Soparkar N., Wallace C. *Journal of Universal Computer Science*. 1997, vol. 3, no. 4, pp. 320–340.
13. Hoare C. A. R. *Commun. ACM.* 1978, vol. 21, no. 8, pp. 666–677.
14. Aleshina N. A., Anisov A. M., Bystrov P. I. *Logika i komp'yuter. Modelirovanie rassuzhdeniy i proverka pravil'nosti programm* [Logic and computer. Modeling of reasoning and program verification]. Moscow: Nauka, 1990, 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. Gelernter D. *ACM Computing Surveys*. 1985, vol. 7, pp. 80–112.
18. Ahuja S., Carriero N., Gelernter D. *IEEE Computer*. 1986, vol. 7.19 (8), pp. 26–34.
19. *Scientific Computing Associates, Inc.* Available at: http://www.lindaspaces.com. (accessed 4 April 2015).
20. Kotov V. E., Sabel'fel'd V. K. *Teoriya skhem programm* [Theory of program schemes]. Moscow: Nauka, 1991, 248 p.
21. 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.
22. Lavrov S. S. *Programmirovanie. Matematicheskie osnovy, sredstva, teoriya* [Programming. Mathematical foundations, means, theory]. Saint-Petersburg: BKhV-Peterburg, 2001, 320 p.
23. Deytel Kh. M., Deytel P. Dzh., Santri S. I. *Tekhnologiya programmirovaniya na Java 2. Kn. 2. Raspredelennye prilozheniya* [Java 2 programming technology. Book 2. Distribution of applications]. Moscow: Binom-Press, 2003, 464 p.
24. Wyckoff P. *IBM System Journal*. 1998, vol. 37, pp. 454–474.
25. Dubravin A. V., Zinkin S. A. *Universitetskoe obrazovanie (MKUO-2015): sb. st. XIX Mezhdunar. nauch.-metod. konf.* [University education: collected articles of XIX International scientific and methodological conference]. Penza: Izd-vo PGU, 2015, vol. 1, pp. 222–225.

26. Dubravin A. V., Zinkin S. A. *Universitetskoe obrazovanie (MKUO-2015): sb. st. XIX Mezhdunar. nauch.-metod. konf.* [University education: collected articles of XIX International scientific and methodological conference]. Penza: Izd-vo PGU, 2015, vol. 1, pp. 225–228.

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 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. 60–80.