УДК 004.9
DOI 10.21685/2072-3059-2020-4-6

В. И. Волчихин, С. А. Зинкин, Н. С. Карамышева

КОНЦЕПТУАЛЬНЫЕ МОДЕЛИ ФУНКЦИОНАЛЬНОЙ АРХИТЕКТУРЫ МОБИЛЬНЫХ РЕКОНФИГУРИРУЕМЫХ АГЕНТНО-ОРИЕНТИРОВАННЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Аннотация.

Актуальность и цели. На современном уровне организации распределенных вычислительных систем необходимо учитывать мобильность компонент – вычислительных узлов, размещенных на мобильных платформах, и модулей программного обеспечения в форме мобильных агентов. Актуальной и сложной является проблема организации взаимодействий компонент в виртуализированных облачно-сетевых РВС (ОС РВС), программное обеспечение которых базируется на платформах мобильных и стационарных агентов, а аппаратное обеспечение включает стационарные и мобильные вычислительные узлы. Объектом исследования является функциональная архитектура ОС РВС. Предметом исследования является методика построения концептуальной модели распределенных вычислений в ОС РВС. Целью исследования является создание простой концептуальной модели облачно-сетевых распределенных вычислительных систем, объединяющих в себе свойства собственно облачных и грид-систем со свойствами мультиагентных систем и пригодной для последующего создания программного обеспечения прикладного и промежуточного (*middleware*) уровней ОС РВС путем последовательной детализации концептуальной модели.

Материалы и методы. В процессе исследований выполняется построение модели сети многоленточных машин Тьюринга и разрабатываются формализованные спецификации узлов сети на основе теории сетей абстрактных модулей и исполнимых логико-алгебраических моделей.

Результаты. Предложена обобщенная концептуальная модель функциональной архитектуры агентно-ориентированной облачно-сетевой РВС с переменной структурой и мобильным программным обеспечением на базе сети многоленточных машин Тьюринга. Предложены новые унифицированные описания элементов концептуальной модели – детерминированных, недетерминированных и вероятностных машин Тьюринга с переменной конфигурацией на базе аппарата формализованных спецификаций – сетей абстрактных модулей.

Выводы. Предложена методика построения обобщенной концептуальной модели функциональной архитектуры агентно-ориентированной ОС РВС с переменной структурой и мобильным программным обеспечением на базе сети машин Тьюринга, позволяющая разработчику оценивать свойства и определять состав программного обеспечения РВС данного типа. Предположено, что на практике такая модель пригодна также для реализации на ее основе прототипного программного обеспечения систем распределенной и параллельной символьной мультиобработки данных.

Ключевые слова: облачно-сетевые распределенные вычислительные системы, агенты, сервисы, мультиагентные системы, сети многоленточных машин Тьюринга, логико-алгебраические спецификации.

V. I. Volchikhin, S. A. Zinkin, N. S. Karamysheva

CONCEPTUAL MODELS OF FUNCTIONAL ARCHITECTURE OF MOBILE RECONFIGURABLE AGENT-ORIENTED DISTRIBUTED COMPUTING SYSTEMS

Abstract.

Background. At the current level of organization of distributed computing systems (DCS), it is necessary to take into account the mobility of components - computing nodes located on mobile platforms and software modules in the form of mobile agents. The organizing of the interactions of components in virtualized cloud-network DCS (CN DCS), the software of which is based on the platforms of mobile and stationary agents, and the hardware includes stationary and mobile computing nodes, is an urgent and complex problem. The object of the research is the functional architecture of the CN DCS. The subject of the research is a methodology for constructing a conceptual model of distributed computing in CN DCS. The aim of the study is to create a simple conceptual model (CM) of CN DCS, combining the properties of cloud and grid systems with the properties of multi-agent systems and suitable for the subsequent creation of software for applied and middleware levels of CN DCS by sequential detailing of the CM.

Materials and methods. During the research process, a model of a network of multi-tape Turing machines has being built and formalized specifications of network nodes are developed based on the theory of networks of abstract modules and executable logical-algebraic models.

Results. A generalized conceptual model of the functional architecture of an agent-oriented cloud-network DCS with variable structure and mobile software based on a network of multi-tape Turing machines is proposed. The new unified descriptions of the elements of the conceptual model - deterministic, non-deterministic and probabilistic Turing machines with variable configuration based on the apparatus of formalized specifications - networks of abstract modules are proposed.

Conclusions. A method for constructing a generalized conceptual model of the functional architecture of an agent-based CN DCS with a variable structure and mobile software based on a network of Turing machines, which allows a developer to evaluate the properties and determine the composition of DCS software of this type is proposed. It is assumed that, in practice, such a model is also suitable for the implementation on its basis of prototype software for systems of distributed and parallel symbolic multiprocessing of data.

Keywords: cloud-network distributed computing systems, agents, services, multi-agent systems, networks of multi-tape Turing machines, logical-algebraic specifications.

Введение

Организация распределенных вычислений при работе вычислительных систем как в замкнутом объеме беспроводных компьютерных сетей, так и при соединении в глобальную сеть удаленных локальных сетей нередко связана с необходимостью учета мобильности объектов – вычислительных узлов, размещенных на мобильных платформах и модулей программного обеспечения в форме мобильных агентов. Существуют несколько библиотек для раз-

работки мобильных агентов. Большинство таких библиотек разработаны на платформенно-независимом объектно-ориентированном языке Java, который предназначен в первую очередь для использования в Интернете, так как стандартная библиотека Java поддерживает известные протоколы и интерфейсы, использующиеся в различных сетях.

В международной практике получили применение ряд платформ для мобильных и стационарных агентов: Agent Factory, AgentBuilder, AgentScape, Aglets, AGLOBE, Cougaar, CybelePro, EMERALD, JACK, JADE, Jadex, Jason, JIAC, Jas, MASON, SeSAm, Swarm и др. [1–5]. Агенты, разработанные в рамках данных платформ, разнообразны по целям и возможностям применения. Одними из наиболее развитых и апробированных платформ мобильных и интеллектуальных агентов являются платформы Aglets, JADE и Jason [6–14].

Обычно используется следующее определение: мобильные агенты – это программы, которые можно отправить с одного компьютера и доставить на удаленный компьютер для выполнения. По прибытии на удаленный компьютер они представляют свои учетные данные и получают доступ к локальным службам и данным. Они также предоставляют единую унифицированную парадигму для работы распределенных объектов, охватывающую синхронность и асинхронность, передачу сообщений и объектов, а также стационарные объекты и мобильные объекты. Мобильный агент имеет возможность перемещаться на любой доступный ему узел сети. При перемещении агент может перенести с собой все остальные объекты, связанные с ним в текущий момент времени, а также свое состояние, в котором он находился в момент отправки. При прибытии на место назначения мобильный агент восстанавливает свое состояние и продолжает свою работу, либо принимает собственное решение о дальнейших действиях. Мобильные агенты поддерживают также неустойчивые соединения, портативные компьютеры, периферийные устройства, мобильные средства связи [6–8, 12–14].

Мобильный агент создается в специальной серверной среде, поставляемой от разработчиков. Серверная среда или сервер агентов предоставляет базовые сервисы для управления агентами: создание, уничтожение, клонирование, отправка, деактивация, назначение и смена контекста (виртуального рабочего пространства). Сервер агентов также служит для управления глобальными политиками безопасности, правами доступа агентов к ресурсам и другим агентам, регистрацией агентов, правилами межконтекстных взаимодействий [14].

Все эти особенности делают использование технологии мобильных агентов привлекательным для работы в сетях. Данная технология позволяет проектировать и строить эффективные распределенные системы [9–11].

В работе [15] была предложена организация функционирования облачно-сетевых распределенных вычислительных систем (РВС), основанная на формализации перехода от известной облачной архитектуры «функция как сервис» (FaaS – Function-as-a-Service) к новой архитектуре «агент как сервис» (AaaS – Agent-as-a-Service). Однако, несмотря на достигнутый прогресс в области глобальных распределенных и агентно-ориентированных вычислений, технологии глобального программирования требуют дополнительного теоретического обоснования.

В настоящей работе предложена обобщенная концептуальная модель функциональной архитектуры РВС с переменной структурой и мобильным

агентно-ориентированным программным обеспечением на базе сети машин Тьюринга, позволяющая разработчику оценивать свойства и определять состав программного обеспечения РВС. На практике такая модель может быть пригодна для реализации на ее основе прототипного программного обеспечения вычислительных систем для распределенной и параллельной символьной мультиобработки данных.

Новое унифицированное описание на базе аппарата формализованных спецификаций – сетей абстрактных модулей, позволит специфицировать ряд математических абстракций: детерминированных, недетерминированных и вероятностных машин Тьюринга с переменной конфигурацией и, в дальнейшем, реконфигурируемых конечных автоматов различных видов: детерминированных, недетерминированных и вероятностных. Описание позволяет строить исполнимые модели функционирования распределенных вычислительных систем с переменной структурой, пригодные для создания элементов функциональной архитектуры распределенных вычислительных систем в виде прототипного и рабочего программного обеспечения промежуточного (*middleware*) уровня.

1. Концептуальная модель распределенных вычислений в глобальных вычислительных сетях

Предлагается концептуальная модель распределенных вычислений (КМ РВ) в глобальных вычислительных сетях. Для определения близкой к КМ РВ модели коллектива вычислителей (МКВ) в связи с эквивалентностью различных алгоритмических систем, в работе [16] были выбраны машины Тьюринга. Предложенный в этой работе подход позволяет на концептуальном уровне рассмотреть системную и функциональную архитектуры однородных вычислительных систем, структур и сред. Этот же подход с существенными изменениями и дополнениями в настоящей работе предлагается использовать для построения концептуальной модели глобальных агентно-ориентированных вычислений в вычислительных сетях на основе сети многоленточных машин Тьюринга.

Согласно описанию МКВ из работы [16] структурная схема коллектива вычислителей представлена в виде решетки, представляющей однородную вычислительную среду. Для машин Тьюринга, находящихся в узлах решетки, введены две специальные операции – передачи ($П_1$) и приема ($П_2$). Передающая машина Тьюринга, находящаяся в состоянии $q_{П1}$, передает обозреваемый головкой символ для записи в обозреваемые головками ячейки всех остальных машин, которые должны в данный момент выполнять операции приема, находясь в состояниях $q_{П2}$. Все машины Тьюринга ведут вычисления независимо друг от друга в соответствии со своими программами. Организация фазы управления основана на введенных ранее в теории однородных вычислительных систем операциях обобщенного условного ($У_0$) и безусловного ($У_1$) переходов.

Однако вычислительная среда, реализуемая на основе глобальной вычислительной сети, имеет чаще всего нерегулярную структуру из-за сложной системы коммуникаций, включающей каналы связи, коммутаторы, концентраторы, маршрутизаторы и другие устройства. Глобальные вычисления ча-

ще всего требуют организации логической связи всех вычислительных узлов по принципу «каждый с каждым». Особенно это важно для пиринговых [7] (рис. 1) и метакомпьютерных [9, 10] распределенных вычислительных систем, построенных на платформе мобильных агентов. При отсутствии непосредственных связей используются физические связи через промежуточные (транзитные) узлы, т.е. непосредственные логические связи устанавливаются как виртуальные соединения.

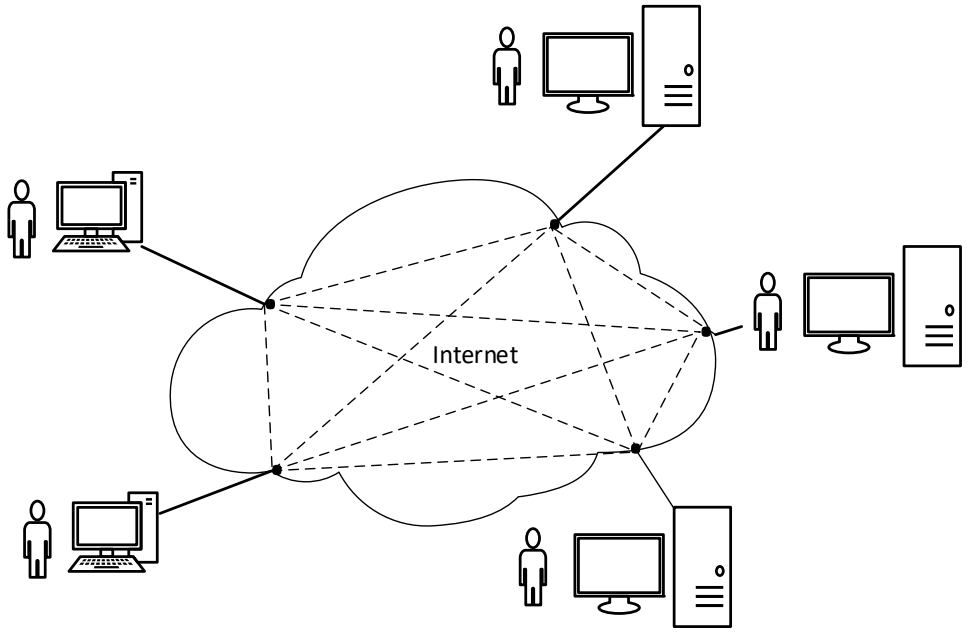

Рис. 1. Коммуникации в пиринговой (peer-to-peer) компьютерной сети

На абстрактном уровне детализацией этих связей можно пренебречь. Поэтому предлагаемая концептуальная модель распределенных вычислений в глобальных вычислительных сетях имеет следующие отличия от модели коллектива вычислителей из [16]: граф, связывающий машины Тьюринга, является полным ориентированным графом без петель. Тем самым будет учтена сетевая специфика модели: полный граф реализуется в логической сети, а машины Тьюринга реализуются на уровне распределенного приложения. Далее в новой модели сети машин Тьюринга доопределяются операции фаз обмена и получения результатов вычислений: вводятся операции, которые отсутствовали в прежней модели коллектива вычислителей. В новой модели машины Тьюринга, размещенные в узлах, являются многоленточными, а различные программы для различных лент и головок одной и той же машины выполняются последовательно. При необходимости может быть также установлен основной режим работы многоленточной машины Тьюринга, при котором согласно исходному определению [16] машина использует все ленты и головки, выполняя одну (свою) программу. Более подробное описание многоленточной машины Тьюринга будет дано далее.

Для описания некоторых из новых операций потребуется понятие о конфигурации многоленточной машины Тьюринга. Формальное определение

многоленточной машины Тьюринга приведено, например, в работе [17]. Для описания ряда новых операций для сети машин Тьюринга потребуется дополнительно использовать понятие о конфигурации этих машин из указанной работы:

«Конфигурация N-ленточной машины Тьюринга – это следующий кортеж:

$$(q,\ \alpha_1\uparrow\beta_1, \alpha_2\uparrow\beta_2, \ldots, \alpha_i\uparrow\beta_i,\ \ldots, \alpha_N\uparrow\beta_N),$$

где q – состояние управляющего устройства; $\alpha_i\uparrow\beta_i$ (i = 1, 2, …, N) – непустая часть i-й ленты; $\uparrow$ – специальный символ, не являющийся символом ленты и указывающий, что i-я головка в данный момент обозревает ячейку, расположенную непосредственно справа от этого символа».

Структурная схема концептуальной модели глобальных вычислений представлена на рис. 2.

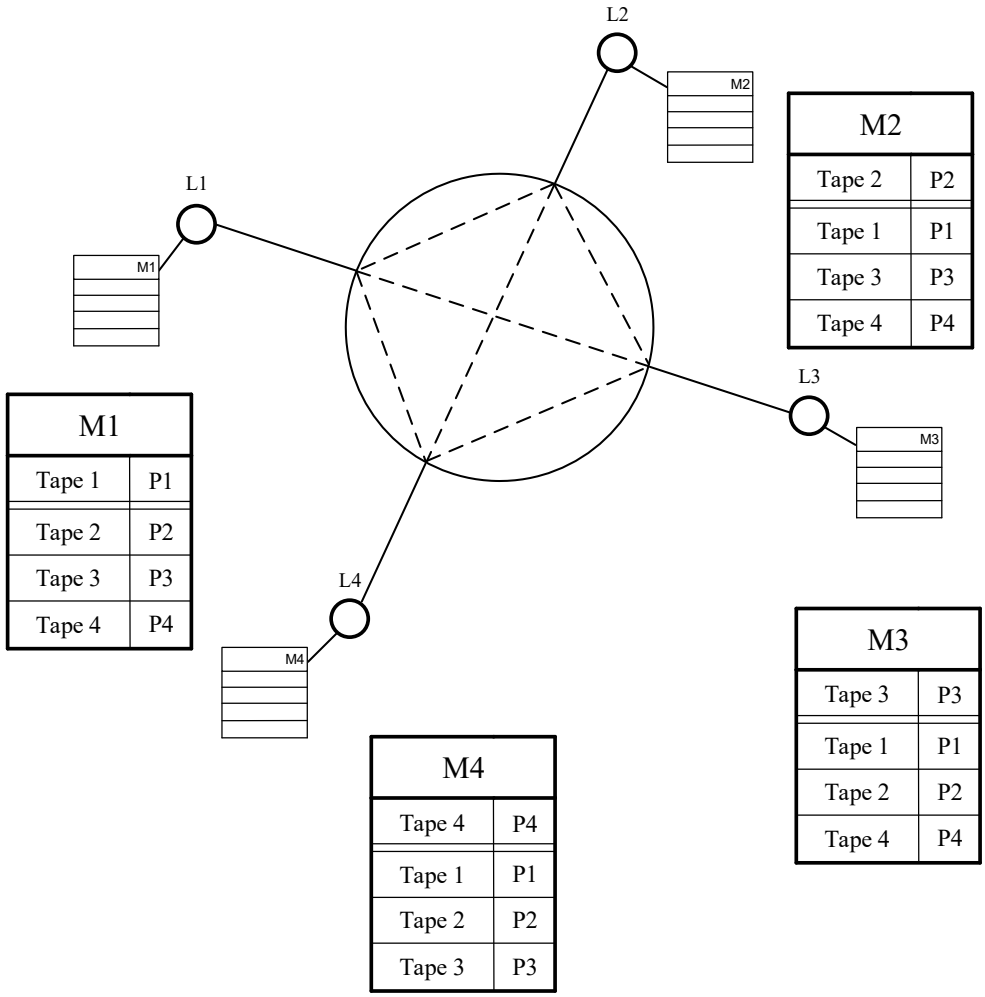

Рис. 2. Концептуальная модель распределенных вычислений в глобальной вычислительной сети, построенная на основе сети многоленточных машин Тьюринга

Здесь для простоты вместо двунаправленных дуг изображены пунктиром ребра полного неориентированного графа, образующие связи по принципу «каждый с каждым». Каждый *i*-й узел сети включает машину Тьюринга со считывающими-записывающими головками на собственной *i*-й ленте $Tape_i$ и на остальных дополнительных $(N-1)$ лентах. Дополнительные ленты используются для хранения копий содержимого лент смежных машин, а также могут иметь самостоятельное значение для узла. Абстрактное управляющее устройство каждой *i*-й машины Тьюринга имеет память для хранения программ $P_1, P_2, \ldots, P_N$, в том числе собственной программы P_i и копий программ смежных машин. Расширенный набор сетевых операций обмена и управления (формирования результатов) в сети Тьюринга представлен в табл. 1.

Таблица 1

Сетевые операции обмена и управления
в концептуальной модели распределенных вычислений
на основе сети многоленточных машин Тьюринга

Глобальные операции	Комментарии
1	2
Send-to-all	Аналог операции $П_1$ в МКВ: передающая машина Тьюринга (МТ), находясь в специальном состоянии $q_{Send\text{-}to\text{-}all}$, передает обозреваемый головкой символ для записи в обозреваемые головками ячейки всех остальных машин
Receive-from(i)	Аналог операции $П_2$ в МКВ: принимающая МТ, находясь в состоянии $q_{Receive\text{-}from(i)}$, принимает символ от передающей *i*-й машины и записывает его в обозреваемую ячейку
Send-to(i)	МТ, находясь в состоянии $q_{Send\text{-}to(i)}$, передает обозреваемый головкой символ для записи его в обозреваемую ячейку машины *i*
Send-tape-to-all	Содержимое ленты МТ передается всем остальным машинам
Send-tape-to(i)	Содержимое ленты МТ передается *i*-й машине
Receive-tape-from(i)	МТ принимает содержимое ленты от *i*-й машины
Send-data-to-all	Содержимое части ленты (данные) МТ передается всем остальным машинам
Send-data-to(i)	Содержимое части ленты (данные) МТ передается *i*-й машине
Receive-data-from(i)	МТ принимает содержимое части ленты (данные) от *i*-й машины
Send-config-to-all	Текущая конфигурация МТ передается всем остальным машинам
Send-config-to(i)	Текущая конфигурация МТ передается *i*-й машине
Receive-config-from(i)	МТ принимает текущую конфигурацию от *i*-й машины
Send-prog-to-all	Текущая программа МТ передается всем остальным машинам
Send-prog-to(i)	Текущая программа МТ передается *i*-й машине
Receive-prog-from(i)	МТ принимает текущую программу от *i*-й машины

Окончание табл. 1

1	2
Send-config-prog-to-all	Текущая конфигурация и программа МТ передаются всем остальным машинам
Send-config-prog-to(i)	Текущая конфигурация и программа МТ передаются *i*-й машине
Receive-config-prog-from(i)	МТ принимает текущую конфигурацию и программу от *i*-й машины
Generalized-cond-go-to-all	Обобщенный условный переход, аналог операции $У_0$ в МКВ, но эта операция выполняется не по цепочке, последовательно, а широковещательно. Исходная выделенная машина (например, *i*-я) после завершения собственных вычислений заставляет все остальные машины после завершения вычислений в каждой из них перейти в состояния $q_{Result(j)}$, $j \neq i$. Фиксируется содержимое всех лент и данные с них (результаты вычислений) могут быть затем переданы исходной *i*-й машине. Для этого каждая машина должна выполнить подпрограмму передачи символов со своих лент на одноименные ленты *i*-й машины, после чего каждая машина переходит в состояние q_{0j}, $j \neq i$, и останавливается. Выделенная *i*-я машина переходит в состояние q_{0i} и останавливается после завершения обработки полученных результатов
Generalized-go-to(i)	Обобщенный безусловный переход первого типа, аналог операции $У_1$ в МКВ: из данной МТ переводится в нужное состояние $q_{Result(i)}$ любая другая *i*-я машина. Это состояние может интерпретироваться как признак завершения вычислений. Дальнейшие действия машин аналогичны операции *Generalized-cond-go-to-all* для исходной и *i*-й машины
Generalized-begin-to-all	Обобщенный безусловный переход второго типа: выполняя данную операцию, данная машина инициирует начало работы всех остальных машин, переводя их в состояния q_{1i}
Generalized-end-to-all	Обобщенный безусловный переход третьего типа: выполняя данную операцию, данная машина инициирует окончание работы всех остальных машин, переводя их в состояния q_{0i}

Выполнение каждой операции осуществляется в машине Тьюринга в том случае, когда она переходит в специальное состояние, имя которого соответствует этой операции. Например, выполнение операции обобщенного безусловного перехода третьего типа происходит при нахождении машины Тьюринга в состоянии $q_{Generalized\text{-}end\text{-}to\text{-}all}$.

Каждая машина Тьюринга, расположенная в узле сети, обрабатывает содержимое собственных лент последовательно, поэтому можно применять основные приемы программирования, описанные в работах [17–22] и основанные на использовании в программах операций суперпозиции, композиции и разветвления. Данные способы сочетания машин Тьюринга облегчают про-

цесс программирования: можно разбивать задачу на подзадачи, последовательно или параллельно соединять программы для отдельных подзадач (суперпозиция или композиция машин соответственно), организовывать ветвления и циклы, т.е. язык тьюрингова программирования достаточно богат [20–21].

Дополнительно определенные локальные операции обмена и управления в сети машин Тьюринга описаны в табл. 2.

Таблица 2

Локальные узловые многоленточные операции обмена и управления в концептуальной модели распределенных вычислений на основе сети машин Тьюринга

Локальные операции	Комментарии
Switch-local(i,j)	Управляющее устройство *i*-й машины, обрабатывающее содержимое ячеек своей основной *i*-й ленты при помощи *i*-й головки, попав в состояние $q_{Switch\text{-}local(i,j)}$, передает обозреваемый *i*-й головкой символ для записи в обозреваемую *j*-й головкой ячейку на *j*-й ленте и переключается на *j*-ю головку для обработки содержимого ячеек *j*-й ленты
Return-local(j,i)	Управляющее устройство *i*-й машины, обрабатывающее содержимое ячеек *j*-й ленты (одной из дополнительных лент) при помощи *j*-й головки, попав в состояние $q_{Return\text{-}local(j,i)}$, передает обозреваемый *j*-й головкой символ для записи в обозреваемую *i*-й головкой ячейку на *i*-й (основной) ленте и переключается на *i*-ю головку для обработки содержимого ячеек *i*-й ленты
Local-mode-to-all(i)	Для управляющего устройства *i*-й машины при попадании в состояние $q_{Local\text{-}mode\text{-}to\text{-}all(i)}$ становится возможным использование всех лент и головок при работе по одной *i*-й программе P_i. При этом обозреваемый *i*-й головкой символ передается всем остальным головкам для записи в обозреваемые ими ячейки

Отличительной особенностью концептуальной модели распределенных вычислений является то, что в ней предусмотрен не только обмен данными между машинами Тьюринга в виде содержимого лент, но и обмен программами, а также текущими вычислениями в виде конфигураций машин Тьюринга. Это соответствует ряду распространенных парадигм организации распределенных вычислений: взаимодействующих последовательных процессов, агентно-ориентированного программирования и параллельной обработки структурированных данных.

Данная модель распределенных вычислений в глобальных вычислительных сетях имеет концептуальный характер и может быть использована для непосредственной программной реализации сети машин Тьюринга. Она дает представление о функциональной архитектуре распределенных вычислительных систем, о комплексе программ, который должен быть реализован, о плане распределения памяти, а также о виде передаваемых управляющих сообщений, данных и мобильных программ, что облегчает разработку проек-

та распределенного приложения, предназначенного для реализации на сети компьютеров.

2. Динамически модифицируемые формальные исполнимые модели инфокоммуникационных систем и процессов

Рассматриваются некоторые динамически модифицируемые формальные модели для спецификации инфокоммуникационных систем и процессов. Декларируемая в работе концепция «непосредственной исполнимости» формальных моделей – это метафора, означающая, что разработка формализованных спецификаций является последним шагом для программиста перед непосредственным созданием сетевого программного обеспечения.

Рассматриваемые формальные динамически реконфигурируемые модели предназначены для описания и проектирования функциональной и системной архитектуры распределенных систем, создаваемых на основе инфраструктуры глобальных и локальных вычислительных сетей.

Используемые формализмы основаны на логических и алгебраических моделях. Эти модели возможно использовать для представления тех или иных свойств сетевого программного обеспечения. Однако в литературе не достаточно подробно освещены вопросы использования формальных моделей в случаях, когда функциональная или системная архитектура моделируемых систем подвержена изменениям в силу ряда объективных причин: увеличение или уменьшение числа средств обработки, хранения и передачи данных, изменение режима функционирования вследствие уменьшения или увеличения нагрузки на вычислительные мощности, корректировка программного обеспечения.

Динамически модифицируемые формальные модели для спецификации инфокоммуникационных систем и процессов могут относительно легко программироваться и реализовываться для распределенных вычислительных систем – облачных сред, грид-систем, беспроводных мобильных сетей. Используемая в работе концепция «непосредственной исполнимости» формальных моделей предполагает, что разработка на их основе формализованных спецификаций является последним шагом для программиста перед непосредственным созданием сетевого программного обеспечения. В противоположность этому подходу, например, в Университете Дьюка, США, шт. Северная Каролина (USA, Durham, North Carolina, Duke University), разработана программа FLAP (англ. Java Formal Languages and Automata Package) – свободная кроссплатформенная программа для экспериментов с различными объектами, встречающихся в теории формальных языков [23]. Возможности программы FLAP: имитирует машину Тьюринга, в том числе многоленточную; имитирует автоматы Мили и Мура; имитирует магазинный автомат; демонстрирует лемму о разрастании для регулярных и контекстно-свободных грамматик; схематично рисует недетерминированные конечные автоматы (НДКА) и детерминированные конечные автоматы (ДКА); «умеет» пошагово проводить преобразование регулярного выражения в НДКА и детерминизацию НДКА, а также минимизацию ДКА [23].

В отличие от данной и других подобных разработок, в настоящей работе введено новое понятие сети машин Тьюринга и ее описание, а далее также

будут рассмотрены формализованные спецификации основных реконфигурируемых дискретных моделей на основе теории алгебраических систем.

Алгебраические системы изучались многими исследователями. Одной из основополагающих работ является монография А. И. Мальцева [24]. Многоосновные, или многосортные, алгебраические системы детально описаны в монографии Б. И. Плоткина [25]. Родственная концепция эволюционирующих алгебр, в современной интерпретации автора – машин абстрактных состояний (МАС), была предложена в работах Ю. Гуревича; особенно важны введенные им понятия модификации предикатов и функций [26, 27]. Хорошо известно, что МАС могут моделировать другие типы машин, таких как машины Тьюринга, равнодоступные адресные машины и другие абстрактные машины. Главным отличием предлагаемой в настоящей работе концепции является заложенная в определение ряда формальных моделей возможность их реконфигурации. Другой отличительной особенностью предлагаемого подхода является структурирование сетей абстрактных модулей, или машин на основе использования некоторых элементов алгебры алгоритмов В. М. Глушкова [28, 29]. В частности, в данных работах определены суперпозиции альфа-дизъюнкций (для всюду определенных условий) вида $[\alpha](A \vee B)$ – аналога оператора “if” и композиция вида: $A*B$ или просто A; B – последовательное выполнение операторов. Следует далее отличать символ операции “;” от знака препинания. Операция “$\leftarrow$” применяется для модификации значений переменных, функций и предикатов.

Концепция «непосредственно исполнимых» формальных моделей используется в смысле систем алгоритмических алгебр (САА) В. М. Глушкова. Совместное использование математического аппарата машин абстрактных состояний, алгоритмических алгебр и алгебраических систем в виде сетей абстрактных модулей (САМ) было предложено в работах [30–33].

Для обозначения понятия сети абстрактных модулей, или машин, будет использоваться аббревиатура САМ (англоязычный вариант – NAM, Network of Abstract Modules).

Для того чтобы использовать единый подход к построению непосредственно исполнимых моделей, используется понятие динамически модифицируемой многоосновной, или многосортной, алгебраической системы AS, определяющей сеть абстрактных модулей (данное определение AS основано на интеграции упомянутых выше абстракций МАС, САА и САМ согласно [30–33]):

$$AS = (\boldsymbol{A}, P, F, I_{F0}, I_{P0}, Rules, \boldsymbol{M}, Z), \quad (1)$$

где $\boldsymbol{A} = \{A_1, A_2, \ldots, A_n\}$ – конечное множество основных множеств, или сортов $A_1, A_2, \ldots, A_n$; P – конечное множество предикатных символов; F – конечное множество функциональных символов; I_{F0} – начальная интерпретация функциональных символов; I_{P0} – начальная интерпретация предикатных символов; $Rules$ – конечное множество правил модификаций, или обновлений, интерпретации предикатных и функциональных символов; $\boldsymbol{M}$ – конечное множество абстрактных модулей, построенных на основе операций и суперпозиций операций САА при всюду определенных логических условиях; Z: $\boldsymbol{M} \rightarrow \mathbf{P}(Rules)$ – отображение множества $\boldsymbol{M}$ на множество подмножеств множества $Rules$, где $\mathbf{P}$ – символ булеана множества.

Таким образом, под воздействием множества $\boldsymbol{M}$ абстрактных модулей алгебраическая система AS эволюционирует, переходя от одних интерпретаций предикатов и функций к другим. При формализации абстрактных модулей будут также использоваться определенные в алгебре алгоритмов Глушкова операторы-константы E (тождественный, или пустой, оператор) и N (нигде не определенный, или невозможный, оператор), а также дополнительно вводимые операторы H (оператор останова) и оператор Ret (оператор возврата к проверке начального логического условия в абстрактном модуле).

Реконфигурируемая динамически модифицируемая многоосновная, или многосортная, алгебраическая система для сети абстрактных модулей теперь определяется на основе определения (1) следующим образом:

$$RAS = (\boldsymbol{A}, P, F, I_{F0}, I_{P0}, Rules, \boldsymbol{M}, Z, Rrules, \boldsymbol{R}, W), \qquad (2)$$

где дополнительно определены следующие понятия: $Rrules$ – множество правил реконфигурации, которые могут изменять структуру сети абстрактных модулей, а также режим функционирования самих абстрактных модулей; $\boldsymbol{R}$ – конечное множество абстрактных модулей реконфигурации, построенных на основе операций и суперпозиций операций САА при всюду определенных логических условиях; W: $\boldsymbol{R} \to \mathbf{P}(Rrules)$ – отображение множества модулей реконфигурации $\boldsymbol{R}$ на множество подмножеств множества $Rrules$, где $\mathbf{P}$ – символ булеана множества.

Модули сети, таким образом, бывают двух типов: операционные модули $\boldsymbol{M}$ и модули реконфигурации $\boldsymbol{R}$. Операционные модули при необходимости могут выполнять функции реконфигурации путем изменения значений логических условий, проверяемых в самом модуле или в других модулях.

Для обозначения реконфигурируемой сети абстрактных модулей, или машин, используется аббревиатура РСАМ (англоязычный вариант – RNAM, Reconfigurable Network of Abstract Modules).

Правила реконфигурации из множества $Rrules$ строятся аналогично правилам из множества $Rules$, но они могут в существенной степени изменять структурные и операционные свойства формальной модели. Например, правило

$$r_1 = \boldsymbol{M} \leftarrow \boldsymbol{M} \cup \{m_i\}$$

позволяет добавить к множеству $\boldsymbol{M}$ сети абстрактных модулей новый модуль m_i, а правило

$$r_2 = \boldsymbol{M} \leftarrow \boldsymbol{M} \setminus \{m_j\}$$

позволяет удалить модуль m_j из сети. Поэтому реконфигурацию сети САМ можно осуществлять, удаляя одни модули и заменяя их другими модулями с новой функциональностью. Подобные реконфигурации должны сопровождаться корректировкой составных логических выражений для условных частей абстрактных модулей.

В выражениях для правил реконфигурации, выполняющих операции над множествами, используются характеристические функции множеств. Пусть, например, $f_{\boldsymbol{M}}$: $\boldsymbol{M} \to \{\mathbf{true}, \mathbf{false}\}$ – характеристическая функция (унарный предикат); $\boldsymbol{M} = \{m_1, m_2, \ldots, m_n\}$ – множество возможных абстрактных модулей в РСАМ. Тогда реализация правила $f_{\boldsymbol{M}}(m_i) \leftarrow \mathbf{true}$ (прежнее значение

высказывания $f_M(m_i) = \mathbf{false}$) соответствует добавлению нового модуля m_i к сети, а реализация правила $f_M(m_j) \leftarrow \mathbf{false}$ (при прежней истинности высказывания $f_M(m_i) = \mathbf{true}$) соответствует удалению модуля m_j из сети. Развивая пример, введем условия α и β и составим выражения для двух модулей реконфигурации РСАМ:

$$R_1 = [\alpha](f_M(m_i) \leftarrow \mathbf{true} \vee E);$$

$$R_2 = [\beta](f_M(m_j) \leftarrow \mathbf{false} \vee E),$$

где описанные выше правила $f_M(m_i) \leftarrow \mathbf{true}$ и $f_M(m_j) \leftarrow \mathbf{false}$ выполняются модулями $R_1 \in \boldsymbol{R}$ и $R_2 \in \boldsymbol{R}$ реконфигурации РСАМ при истинности условий α и β – простых или составных высказываний, возможно, с предикатными символами и предметными константами. В общем случае α и β – замкнутые (без свободных предметных переменных) выражения в логике предикатов.

Одно из правил $f_M(m_i) \leftarrow \mathbf{true}$ или $f_M(m_j) \leftarrow \mathbf{false}$ в зависимости от условий α и β может быть выполнено следующим модулем реконфигурации:

$$R_{12} = [\alpha](f_M(m_i) \leftarrow \mathbf{true} \vee E) * [\beta](f_M(m_j) \leftarrow \mathbf{false} \vee E),$$

где символ операции "*" композиции (последовательного выполнения) модулей можно заменить простой точкой с запятой ";".

В основу построения модулей сетей САМ и РСАМ положена логика предикатов первого порядка. В многоосновном (многосортном) исчислении предикатов первого порядка каждому терму однозначно приписывается сорт данного терма (сорта в приложениях данной работы определены для таких классов объектов, как агенты, сетевые узлы, наборы данных, передаваемые сообщения). Имеет место следующее правило вывода [34]:

$$\frac{t_1, t_2, \ \dots\ , t_i, \ \dots\ , t_k}{f(t_1, t_2, \ \dots\ , t_i, \ \dots\ , t_k)},$$

где t_1, t_2, … , t_i, … , t_k – термы сортов π_1, π_2, … , π_i, … , π_k соответственно; f – k-арный функциональный символ вида (π_1, π_2, … , π_i, … , $\pi_k \rightarrow \pi$); $f(t_1, t_2, \dots, t_i, \dots, t_k)$ – терм сорта π.

Для эффективного отображения динамики предметной области путем модификации баз данных добавим к приведенному правилу вывода пару дополнительных правил модификации функций и предикатов:

$$\frac{t_1, t_2, \ \dots\ , t_i, \ \dots\ , t_k, t_{k+1}}{f(t_1, t_2, \ \dots\ , t_i, \ \dots\ , t_k) \leftarrow t_{k+1}}$$ – правило модификации функции;

$$\frac{t_1, t_2, \ \dots\ , t_i, \ \dots\ , t_k, b}{p(t_1, t_2, \ \dots\ , t_i, \ \dots\ , t_k) \leftarrow b}$$ – правило модификации предиката,

здесь t_{k+1} – терм сорта π; p – k-арный предикатный символ вида (π_1, π_2, … , π_i, …, π_k); b – булев терм, принимающий значения в множестве {*true*, *false*}. Правила данного вида активно использованы в работах [26, 27], посвященных использованию аппарата машин с абстрактными состояниями.

Аналогично САА В. М. Глушкова модули САМ и РСАМ формируются на основе алгебр условий и модулей. Система образующих алгебры модулей

включает дополнительно, в отличие от САА, элементарные правила модификации предикатов и функций. Алгебра всюду определенных условий аналогична алгебре предикатов первого порядка.

3. Базовые реконфигурируемые модели: логико-алгебраические спецификации реконфигурируемых машин Тьюринга

В качестве базового примера, иллюстрирующего применение РСАМ при реконфигурации формальной модели некоторой дискретно-событийной системы, выбраны машины Тьюринга. В разд. 1 машины Тьюринга были выбраны в качестве узлов сети Тьюринга в концептуальной модели распределенных вычислений, моделирующей глобальные коммуникации и вычисления в вычислительной сети, поэтому вопросы реализации моделей данных машин на основе сетей абстрактных модулей САМ являются актуальными. Одним из важнейших приложений машин Тьюринга является обработка строк символов, а приложением сети Тьюринга, определенной в разд. 1, может быть распределенная символьная мультиобработка.

Описание машины Тьюринга, соответствующее эквивалентным описаниям из [19–22] и других работ, приведено в целях согласования дальнейших обозначений:

> «Машина Тьюринга состоит из потенциально бесконечной в обе стороны ленты, управляющего устройства и головки. Лента разбита на ячейки, которые могут содержать символы из алфавита $A = \{a_1, a_2, \ldots, a_m\}$. Этот алфавит содержит и пустой символ #, а любая ячейка, содержащая в данный момент этот символ, называется пустой ячейкой. Управляющее устройство может находиться в одном из состояний, принадлежащих множеству $Q = \{q_1, q_2, \ldots, q_n\}$, где q_1 – начальное состояние; в множестве Q выделено множество $Q_0 \subset Q$ заключительных состояний. Головка может считывать символы с ленты, записывать их на ленту и перемещаться в одну из сторон вдоль ленты. В каждый момент времени головка обозревает одну ячейку ленты. При работе машины Тьюринга повторяется следующая последовательность действий:
>
> 1) считывание символа, находящегося в ячейке напротив головки;
>
> 2) поиск единственной (для детерминированной машины Тьюринга) команды, имеющей вид $qa \to q'a'w$, где q – текущее состояние управляющего устройства, a – считанный символ (для недетерминированных машин таких команд с одинаковой левой частью qa может быть несколько;
>
> 3) выполнение выбранной команды: перевод управляющего устройства в новое состояние q', запись в текущую обозреваемую ячейку символа a' вместо стираемого символа a и последующее перемещение головки вправо, если $w = R$, и влево, если $w = L$, или отсутствие движения головки при $w = S$.
>
> Машина останавливается, если для пары qa нет команды вида $qa \to q'a'w$ или новое состояние q' принадлежит множеству Q_0 заключительных состояний. Результат работы остановившейся машины – заключительное слово на ленте. Машина Тьюринга перерабатывает начальные слова на ленте в заключительные, задавая тем самым словарную функцию, для которой начальные слова являются значениями аргумента, а заключительные – значениями функции».

Известны различные способы моделирования машин Тьюринга: частично-рекурсивными функциями, системами подстановок и др. Некоторые приемы программирования на машинах Тьюринга приведены, например, в

работах [35, 36]. Известно [19, 37], что модель машины Тьюринга допускает расширения: можно рассматривать машины Тьюринга с произвольным числом лент и многомерными лентами с различными ограничениями; все эти машины являются полными по Тьюрингу и моделируются обычной машиной Тьюринга. Из других модификаций машин Тьюринга известны недетерминированные машины Тьюринга, вероятностные машины Тьюринга, квантовые машины Тьюринга (абстрактные машины, используемые для моделирования квантовых компьютеров), нейронные машины Тьюринга, в которых нейросети используют внешнюю память для записи и последующего чтения информации так же, как это делает машина Тьюринга [19, 37, 38].

Работу одноленточных машин Тьюринга, как детерминированных, так и недетерминированных, опишем логико-алгебраическим выражением M для модуля сети абстрактных машин САМ (данное описание основано на понятиях абстракций МАС, САА и САМ; здесь и далее в настоящей работе построение динамических исполнимых моделей на языке САМ базируется на исправленных нами и дополненных формализованных спецификациях из работы [30]):

Базовая модель M для одноленточной машины Тьюринга:

$$M = Place(Head) \leftarrow 0;\ State(q_1) \leftarrow \textbf{true};$$

$$[(\exists_{Sel\_one}\ q)\ State(q)]([a \leftarrow Tape(Place(Head))]$$

$$([\exists_{Sel}(q, a, q', a', w)\ Program(q, a, q', a', w)]$$

$$(\{[S_{Q0}(q')](H \vee E);\ Tape(Place(Head)) \leftarrow a';$$

$$State(q) \leftarrow \textbf{false};\ State(q') \leftarrow \textbf{true};\ \{[w = R]$$

$$(Place(Head) \leftarrow Inc(Place(Head)) \vee$$

$$[w = L](Place(Head) \leftarrow Dec(Place(Head)) \vee E))\};\ Ret\} \vee N) \vee N) \vee N), \quad (3)$$

где q – символ, обозначающий текущее состояние управляющего устройства, $q \in Q$, $Q = \{q_1, q_2, \ldots, q_n\}$;

Q – внутренний алфавит, или множество внутренних состояний управляющего устройства;

a – символ, находящийся в текущей ячейке ленты, обозреваемой головкой, $a \in A$, $A = \{a_1, a_2, \ldots, a_m\}$;

A – внешний алфавит символов, размещаемых в ячейках ленты;

q_1 – начальное состояние управляющего устройства, $q_1 \in Q$;

q' – следующее состояние управляющего устройства, $q' \in Q$;

a' – следующий символ в ячейке ленты, обозреваемый головкой после очередного перемещения, $a' \in A$;

$w \in \{R, L, S\}$ – переменная, значения которой определяют направление перемещения головки машины Тьюринга – сдвиг головки на одну ячейку вправо (R), сдвиг на одну ячейку влево (L) и отсутствие движения головки (S), $W = \{R, L, S\}$;

$\exists_{Sel\_one}$ – оператор выбора единственного кортежа (в текущем примере кортеж содержит единственный элемент – номер состояния устройства управления), отвечающего некоторому условию, из отношения, являющегося

областью истинности предиката, записанного справа; в случае, если условию выбора отвечает не один кортеж или не удалось выбрать единственный кортеж, то работа реализации машины Тьюринга прекращается и фиксируется ошибочная ситуация;

$\exists_{Sel}$ – "псевдооператор" САМ, который следует заменить оператором выбора $\exists_{Sel\_one}$ единственного кортежа вида (q, a, q', a', w) при реализации детерминированной машины Тьюринга (ДМТ) или оператором $\exists_{Sel\_all}$ выбора нескольких кортежей, отвечающих условию выбора – то есть выбранной паре (q, a), из отношения, являющегося областью истинности предиката *Program* для недетерминированной машины Тьюринга (НМТ). Для реализации на основе выражения M вероятностной машины Тьюринга (ВМТ) на место псевдооператора $\exists_{Sel}$ следует поставить оператор выбора $\exists_{Sel\_any}$: в случае, если условию выбора отвечает не один кортеж (команда машины Тьюринга), то выбирается любой из них в соответствии с заданной вероятностью;

Program(q, a, q', a', w) – предикат, область истинности которого определяет рабочую программу машины Тьюринга:

$$Program: (Q \times A) \times (Q \times A \times W) \to \{\mathbf{true}, \mathbf{false}\};$$

Tape – унарная функция, определяющая последовательность символов на ленте:

Tape: $\mathbf{Z} \to A$, где $\mathbf{Z}$ – множество целых чисел;

Place – унарная функция, определяющая текущее положение головки:

Place: $Hd \to \mathbf{Z}$, где Hd – множество имен головок для многоленточных машин Тьюринга; в данном случае для одноленточной машины Тьюринга это множество содержит лишь одно имя $Hd = \{Head\}$, где *Head* является предметной константой, или именем единственной головки;

State(q) – унарный предикат, определяющий состояние управляющего устройства машины Тьюринга (в любой момент времени истинно высказывание $(\exists!q)State(q)$, где *State*: $Q \to \{\mathbf{true}, \mathbf{false}\}$), а $\exists!$ – известный в математической логике квантор "существует единственный элемент, для которого истинно некоторое высказывание";

Inc и *Dec* – инкрементная и декрементная функции соответственно; данные функции увеличивают или уменьшают на единицу значение переменной, указывающее на текущее положение головки;

$S_{Q0}(q)$ – характеристическая функция подмножества финальных состояний $Q_0 \subset Q$, причем $q_1 \notin Q_0$; $S_{Q0}: Q \to \{\mathbf{true}, \mathbf{false}\}$;

Ret – завершающий оператор блока, т.е. выражения, заключенного в фигурные скобки; он возвращает управление на первое условие цепочки условий перед блоком, приведшей к выполнению данного блока – в данном примере к условию $[(\exists_{Sel\_one} q)\ State(q)]$.

Одним из символов внешнего алфавита $A = \{a_1, a_2, \ldots, a_m\}$ является пустой символ, обозначаемый символом #.

Работа машин Тьюринга подробно описана в литературе, поэтому далее будут описаны в основном особенности моделирования машины Тьюринга модулем (3) сети абстрактных модулей *M*. Работа упомянутых при описании САМ-моделей разновидностей машин Тьюринга – ДМТ, НМТ и ВМТ будет конкретизирована на такой же основе.

Устройство управления может находиться в одном из состояний, определяемых внутренним алфавитом $Q = \{q_1, q_2, \ldots, q_n\}$. Непересекающиеся подмножества начальных и конечных состояний $Q_1 \subset Q$ and $Q_0 \subset Q$ соответственно определяют начало и окончание работы машины Тьюринга. Машина Тьюринга останавливается, если $q \in Q_0$ после выполнения очередной команды (при истинном высказывании $S_{Q0}(q)$). Для рассматриваемого примера $Q_1 = \{q_1\}$.

Список всех кортежей, или команд вида (q, a, q', a', w), в области истинности предиката *Program* составляет программу машины Тьюринга. В приведенном примере кортеж (q_i, a_j, q', a', w) начинается с пары символов (q_i, a_j); $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$; n – число состояний устройства управления, m – число символов внешнего алфавита A. В программе детерминированной машины Тьюринга для фиксированных q_i и a_j может присутствовать лишь один кортеж (команда), начинающийся с этих двух символов, называемых левой частью команды. Для недетерминированной машины Тьюринга таких команд, начинающихся с пары (q_i, a_j), может быть несколько. Значения элементов данной пары определяются в соответствии с приведенным выше выражением M следующим образом:

$$q_i = (\exists_{Sel\_one}\, q)\, State(q);\ a_j = Tape(Place(Head)).$$

В соответствии с первым выражением оператор $(\exists_{Sel\_one}\, q)$ выбирает значение символа q, соответствующее текущему состоянию управляющего устройства (естественно, что это значение должно быть единственным в области истинности унарного предиката *State*). В соответствии со вторым выражением сначала при помощи унарной функции *Place* определяется местоположение головки *Head* управляющего устройства (т.е. номер текущей ячейки), а затем при помощи унарной функции *Tape* определяется содержимое ячейки, обозреваемой головкой в данный момент времени.

Пара операторов в выражении M для модуля CAM

$$Place(Head) \leftarrow 0;\ State(q_1) \leftarrow \textbf{true}$$

определяет начальное местоположение головки (на нулевой ячейке) и начальное состояние управляющего устройства q_1.

Во второй строке

$$[(\exists_{Sel\_one}\, q)\, State(q)]([a \leftarrow Tape(Place(Head))]$$

проверяются два условия. Первое условие формулируется следующим образом: “состояние q управляющего устройства определено”, а второе – “символ с текущей ячейки, обозреваемой головкой *Head*, определен и стал значением предметной переменной a”. На то, что таким образом определены условия, указывают квадратные скобки. Ранее такой же синтаксис выражений с квадратными скобками был определен для суперпозиций операций α-дизъюнкции в алгебре алгоритмов Глушкова.

Таким образом, к началу выполнения оператора

$$([\exists_{Sel}(q, a, q', a', w) Program(q, a, q', a', w)]$$

значения предметных переменных q и a (т.е. левая часть команды) определены, что делает возможным выполнить команду (q, a, q', a', w) машины

Тьюринга. Как было принято ранее при определении выражения M, псевдооператор $\exists_{Sel}$ заменяется "рабочим" оператором $\exists_{Sel\_one}$ для ДМТ и оператором $\exists_{Sel\_all}$ для НМТ. Далее по ключу (q, a) в области истинности предиката *Program*, т.е. в программе машины Тьюринга, производится поиск одного кортежа-команды для ДМТ или множества кортежей-команд для НМТ. Программа, т.е. область истинности предиката *Program*, определяется самим разработчиком, т.е. сам разработчик определяет, какая машина Тьюринга моделируется – детерминированная или недетерминированная. Для ДМТ в области истинности предиката *Program* не может быть двух или более кортежей с одинаковой левой частью (q, a) команды (q, a, q', a', w). Поэтому для ДМТ с помощью оператора $\exists_{Sel\_one}$ будет выбран единственный кортеж, удовлетворяющий условию поиска. Для НМТ таких кортежей может оказаться несколько, и в таком случае оператор $\exists_{Sel\_all}$ работает, создавая копии мгновенных описаний (МО) и несколько экземпляров машин. Работа НМТ и ВМТ будет пояснена в дальнейшем изложении.

Во фрагменте $[S_{Q0}(q')](H \vee E)$ выражения M производится проверка факта достижения конечного состояния: истинность высказывания $S_{Q0}(q')$ означает, что достигнуто одно из конечных состояний в множестве Q_0 и в модуле M выполняется оператор H, останавливающий работу машины Тьюринга. В противном случае, при ложности высказывания $S_{Q0}(q')$, машина продолжает свою работу после выполнения тождественного (пустого) оператора E. Затем при помощи оператора $Tape(Place(Head)) \leftarrow a'$ в текущую ячейку $Tape(Place(Head))$ ленты с номером $Place(Head)$ записывается новый символ a', и устройство управления переходит из прежнего состояния q в новое состояние q':

$$State(q) \leftarrow \textbf{false}; State(q') \leftarrow \textbf{true}.$$

Переменная w, значение которой являлось последним в выбранной команде, используется для определения направления перемещения головки – сдвиг головки на одну ячейку вправо $(w = R)$, сдвиг на одну ячейку влево $(w = L)$ и отсутствие движения головки $(w = S)$:

$$\{[w = R](Place(Head) \leftarrow Inc(Place(Head)) \vee$$

$$[w = L](Place(Head) \leftarrow Dec(Place(Head)) \vee E))\}.$$

Функция *Inc* увеличивает номер текущей ячейки $Place(Head)$ на единицу, а функция *Dec* уменьшает его на единицу. В данном выражении использована суперпозиция двух α-дизъюнкций.

После выполнения оператора возврата *Ret* управление передается на повторное выполнение действий в модуле M, начиная с проверки условия $[(\exists_{Sel\_one} q)\ State(q)]$. Каждый из трех невозможных операторов N выполняется при последовательном возникновении ошибочных ситуаций. На этом описание работы модуля сети САМ, моделирующего детерминированную, недетерминированную или вероятностную машину Тьюринга при различных интерпретациях псевдооператора $\exists_{Sel}$, закончено.

Реконфигурации машин Тьюринга, как детерминированных (РДМТ), так и недетерминированных (РНМТ), задаются правилами реконфигурации. Эти правила могут изменять программы работы машин Тьюринга. Например,

возможно изменить программу рассмотренной ранее машины путем использования следующих правил:

$$Program(q_5, a_{10}, q_1, a_1, L) \leftarrow \textbf{false};$$

$$Program(q_5, a_{10}, q_2, a_2, R) \leftarrow \textbf{true}.$$

Данные правила модифицируют предикат *Program*: первое правило удаляет кортеж-команду (q_i, a_j, q_1, a_1, L) из области истинности предиката, а второе – добавляет кортеж-команду (q_i, a_i, q_2, a_2, R) в эту область. Включив данные правила в выражение *M*, получим следующее выражение *RM* для абстрактного модуля РСАМ:

$$RM = Place(Head) \leftarrow 0; State(q_1) \leftarrow \textbf{true};$$

$$[(\exists_{Sel\_one}\, q)\, State(q)]([a \leftarrow Tape(Place(Head))]$$

$$([\exists_{Sel}(q, a, q', a', w)\, Program(q, a, q', a', w)]$$

$$(\{[S_{Q0}(q')](H \vee E); Tape(Place(Head)) \leftarrow a';$$

$$State(q) \leftarrow \textbf{false}; State(q') \leftarrow \textbf{true}; \{[q' = q_5]$$

$$(\{Program(q_5, a_{10}, q_1, a_1, L) \leftarrow \textbf{false};$$

$$Program(q_5, a_{10}, q_2, a_2, R) \leftarrow \textbf{true}\} \vee E)\};$$

$$\{[w = R](Place(Head) \leftarrow Inc(Place(Head)) \vee$$

$$[w = L](Place(Head) \leftarrow Dec(Place(Head)) \vee E))\};$$

$$Ret\} \vee N) \vee N) \vee N). \qquad (4)$$

То есть, достигнув состояния q_5, машина модифицирует свою программу путем замены одной команды на другую.

В общем случае правила реконфигурации могут быть включены в выражения для модулей РСАМ в следующем виде:

$$\{C \leftarrow Cond; [C](\{C \leftarrow \textbf{false};$$

$$<последовательность\ правил\ реконфигурации>\} \vee E)\},$$

где *Cond* – некоторое условие (замкнутое выражение в логике предикатов; это условие может зависеть от внешней среды); *C* – булева переменная, а *<последовательность правил реконфигурации>* – это множество *Rrules* согласованных правил реконфигурации, которые изменяют программу машины Тьюринга. К числу важных случаев можно отнести такие, когда $Cond = P_{eq}(a, a_R)$, где P_{eq} – бинарный предикат сравнения на равенство, а $a_R \in A$ – символ на входной ленте, после считывания которого модифицируется программа, или $Cond = P_{eq}(q, q_R)$, где $q_R \in Q$ – состояние устройства управления.

На основе предыдущих примеров логико-алгебраических выражений *M* и *RM* построены РСАМ-модели и для многоленточной (детерминированной, недетерминированной и вероятностной) машины Тьюринга (рис. 3). Формальное определение многоленточной машины Тьюринга приведено, например, в работах [17–19].

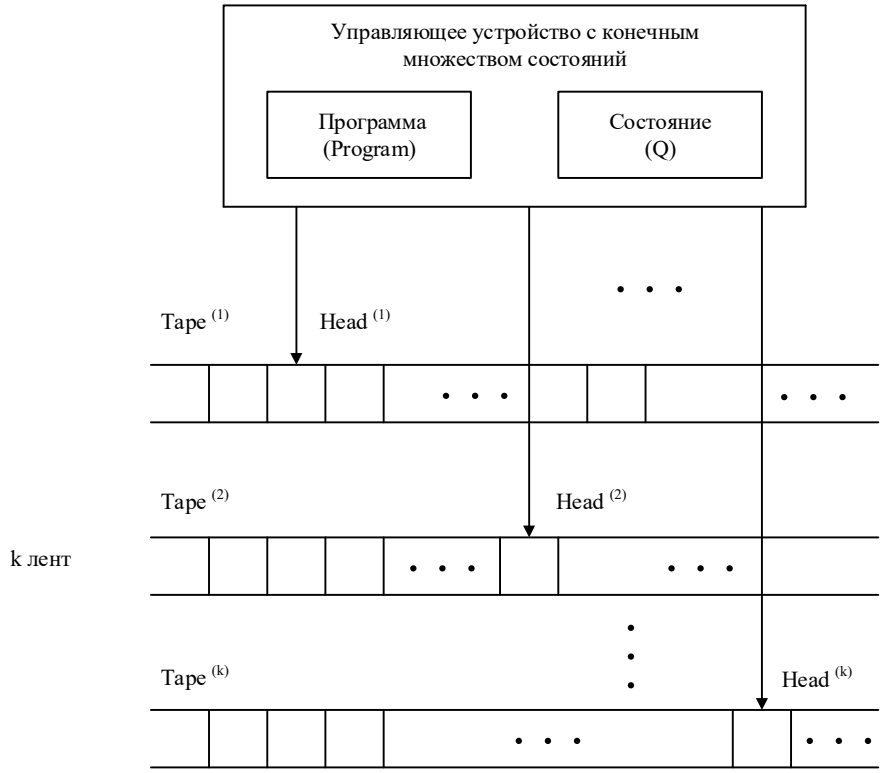

Рис. 3. Многоленточная машина Тьюринга [17]

Модель *RMK* реконфигурируемой многоленточной и многоголовочной машины Тьюринга представлена следующим РСАМ-выражением:

$$RMK = Place(Head^{(1)}) \leftarrow 0;\ Place(Head^{(2)}) \leftarrow 0;\ \ldots;\ Place(Head^{(k)}) \leftarrow 0;$$

$$State(q_1) \leftarrow \textbf{true};\ [(\exists_{One}\ q)\ State(q)]([a^{(1)} \leftarrow Tape^{(1)}(Place(Head^{(1)}))]$$

$$([a^{(2)} \leftarrow Tape^{(2)}(Place(Head^{(2)}))]\ \ldots$$

$$\ldots\ ([a^{(k)} \leftarrow Tape^{(k)}(Place(Head^{(k)}))]$$

$$([\exists_{Sel}(q, a^{(1)}, a^{(2)}, \ldots, a^{(k)},\ q', a'^{(1)}, a'^{(2)}, \ldots, a'^{(k)}, w^{(1)}, w^{(2)}, \ldots, w^{(k)})$$

$$Program(q, a^{(1)}, a^{(2)}, \ldots, a^{(k)},\ q', a'^{(1)}, a'^{(2)}, \ldots, a'^{(k)}, w^{(1)}, w^{(2)}, \ldots, w^{(k)})]$$

$$(\{[S_{Q0}(q')](H \vee E);$$

$$Tape^{(1)}(Place(Head^{(1)})) \leftarrow a'^{(1)};$$

$$Tape^{(2)}(Place(Head^{(2)})) \leftarrow a'^{(2)};\ \ldots,\ Tape^{(k)}(Place(Head^{(k)})) \leftarrow a'^{(k)};$$

$$State(q) \leftarrow \textbf{false};\ State(q') \leftarrow \textbf{true};$$

$$\{[w^{(1)} = R](Place(Head^{(1)}) \leftarrow Inc(Place(Head^{(1)})) \vee$$

$$[w^{(1)} = L](Place(Head^{(1)}) \leftarrow Dec(Place(Head^{(1)})) \vee E))\};$$

$$\{[w^{(2)} = R](Place(Head^{(2)}) \leftarrow Inc(Place(Head^{(2)})) \vee$$

$$[w^{(2)} = L](Place(Head^{(2)}) \leftarrow Dec(Place(Head^{(2)})) \vee E))\};\ \ldots$$

$$\ldots, \{[w^{(k)} = R](Place(Head^{(k)}) \leftarrow Inc(Place(Head^{(k)})) \vee$$
$$[w^{(k)} = L](Place(Head^{(k)}) \leftarrow Dec(Place(Head^{(k)})) \vee E))\};$$
$$\{C \leftarrow Cond; [C](\{C \leftarrow \textbf{false}\}; <\textit{последовательность правил}$$
$$\textit{реконфигурации}>\} \vee E)\}; Ret\} \vee N) \vee N) \ldots \vee N) \vee N) \vee N), \quad (5)$$

где верхние индексы соответствуют номерам головок и лент. Модель *RMK* многоленточной машины Тьюринга (5) предназначена для использования (при соответствующем доопределении) в качестве формализованной концептуальной модели мобильных вычислений в сетевом узле сети машин Тьюринга, описанной в разд. 1.

4. Особенности интерпретации реконфигурируемых недетерминированных машин Тьюринга сетями абстрактных модулей

Машины Тьюринга возможно использовать для распознавания языков, поиска и замены цепочек в последовательностях символов, что может использоваться при поиске информации в распределенных облачных сервисах баз данных. В отличие от обычной детерминированной машины Тьюринга недетерминированная машина Тьюринга при некоторой комбинации текущего состояния устройства управления q и считанного символа a с доступной головке текущей ячейки может допускать несколько вариантов дальнейших действий. Для одноленточной НМТ задается функция перехода $Q \times A \rightarrow \mathbf{P}(Q \times A \times W)$, соответствующая отношению $Rel \subseteq (Q \times A) \times (Q \times A \times W)$. Здесь $\mathbf{P}$ – символ булеана. Детерминированным переходам НМТ соответствует функция $Q \times A \rightarrow Q \times A \times W$.

Использование НМТ в различных целях подробно описано, например, в работе [17]:

> «по аналогии с недетерминированными конечными автоматами НМТ имеет конечное число возможных шагов, из которых в очередной момент выбирается какой-то один. Входная цепочка x допускается, если по крайней мере одна последовательность шагов для входа x приводит к допускающему мгновенному описанию (МО). Мгновенным описанием k-ленточной машины Тьюринга MK называется набор $\{\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots, \alpha_k\}$, где α_i для каждого i представляет собой слово вида xqy, где xy – слово на i-й ленте машины MK, q – текущее состояние машины. Головка на i-й ленте обозревает символ, стоящий справа от q в слове xqy, т.е. первый символ слова y. При данной входной цепочке x можно считать, что НМТ MK параллельно выполняет все возможные последовательности шагов, пока не достигнет допускающего МО, или пока не окажется, что дальнейшие шаги невозможны. После i шагов можно считать, что существует много экземпляров MK, причем каждый экземпляр представляет МО, в котором машина MK может оказаться после i шагов. На $(i + 1)$-м шаге экземпляр C порождает j своих экземпляров, если НМТ, находясь в МО C, может выбрать следующий шаг j способами».

В настоящей работе при описании модели НМТ в терминах сетей абстрактных модулей предлагается использовать квантифицированный оператор выбора $\exists_{Sel\_all}$ (заменяющий псевдооператор $\exists_{Sel}$ в выражениях M, RM и RMK для базовых моделей машин Тьюринга), соответствующая интерпретация которого, т.е. учет каждого состояния перехода при формировании каж-

дой соответствующей копии машины *M*, *RM* или *RMK*, позволяет определять различные параллельно работающие после очередного шага экземпляры машины, как это было определено в цитируемой работе [17].

5. Особенности интерпретации реконфигурируемых вероятностных машин Тьюринга сетями абстрактных модулей

Сетью абстрактных модулей возможно моделировать вероятностные машины Тьюринга, которые отличаются тем, что квантифицированный оператор $\exists_{Sel\_any}$ (вставляемый в соответствующее САМ-выражение *M* для обычной детерминированной машины Тьюринга вместо псевдооператора $\exists_{Sel}$) на каждом шаге работы ВМТ выполняет вероятностный выбор следующего состояния из нескольких возможных. Тип рассматриваемой здесь ВМТ выбран по аналогии с вероятностными автоматами. В случае ВМТ паре “текущее состояние – текущий символ в обозреваемой ячейке на ленте” в программе соответствуют несколько вариантов троек “следующее состояние – новый символ на ленте – направление перемещения головки”. Как известно, от недетерминированной машины Тьюринга ВМТ (здесь рассматривается одна из основных известных модификаций ВМТ) отличается тем, что она вместо недетерминированного перехода к нескольким вариантам выбирает один из вариантов с некоторой вероятностью. ВМТ, выбирая следующий вариант поведения, реализует случайную (в случае компьютерного моделирования – псевдослучайную) функцию вида *Rand*: $Q \times A \to Q \times A \times W$. При этом для всех пятерок $\forall(q, a, q', a', v) \in Q \times A \times Q \times A \times V$ значение $Rand(q, a, q', a', v)$ – это вероятность того, что управляющее устройство ВМТ, находясь в состоянии q и считав символ a из доступной головке ячейки, в ту же ячейку запишет символ a', затем перейдет в состояние q', а головка сместится на одну ячейку влево ($w = L$), вправо ($w = R$) или останется на прежнем месте ($w = S$).

Заключение

Предложено новое унифицированное описание на базе аппарата формализованных спецификаций – сетей абстрактных модулей основных типов математических абстракций: детерминированных, недетерминированных и вероятностных машин Тьюринга с переменной конфигурацией. Описание позволяет строить исполнимые модели функционирования распределенных вычислительных систем с переменной структурой, пригодные для создания элементов функциональной архитектуры распределенных вычислительных систем в виде прототипного и рабочего программного обеспечения промежуточного (*middleware*) уровня.

Предложена обобщенная концептуальная модель функциональной архитектуры агентно-ориентированной РВС с переменной структурой и мобильным программным обеспечением на базе сети машин Тьюринга, позволяющая разработчику оценивать свойства и определять состав программного обеспечения РВС. На практике такая модель пригодна для реализации на ее основе прототипного программного обеспечения систем распределенной и параллельной символьной мультиобработки данных.

Даны рекомендации по интерпретации реконфигурируемых детерминированных, недетерминированных и вероятностных машин Тьюринга сетями

абстрактных модулей, что позволяет строить концептуально-поведенческие исполнимые модели распределенных вычислений и систем, отличающихся переменной, гибкой, масштабируемой и мобильной структурой.

Библиографический список

1. **Tanenbaum, A. S.** Distributed Systems: principles and paradigms. 2nd Edition / A. S. Tanenbaum, Maarten Van Steen. – Pearson Education, Inc., 2007. – 669 p.
2. FIPA Specifications. – URL: http://www.fipa.org/specifications/index.html (дата обращения: 12.11.2019).
3. A Survey of Programming Languages and Platforms for Multi-Agent Systems / R. H. Bordini et al. // Informatica. – 2006.– Vol. 30. – P. 33–44.
4. **Kravari, K.** A Survey of Agent Platforms / K. Kravari, N. Bassiliades // Journal of Artificial Societies and Social Simulation. – 2015. – Vol. 18 (1), № 11. – P. 1–18.
5. **Cynthia, N.** Tools of the Trade: A Survey of Various Agent Based Modeling Platforms / N. Cynthia, M. Gregory // Journal of Artificial Societies and Social Simulation. – 2009. – Vol. 12 (2). – URL: http://jasss.soc.surrey.ac.uk/12/2/2.html
6. **Bellifemine, F. L.** Developing multi-agent systems with JADE / F. L. Bellifemine, G. Caire, D. Greenwood. – Wiley. – 2007. – 300 p. – DOI 10.1002/9780470058411
7. P2P Agent Platform: Implementation and Testing / V. Gorodetsky, O. Karsaev, V. Samoylov, S. Serebryakov // The AAMAS Sixth International Workshop on Agents and Peer-to-Peer. Computing (AP2PC 2007). – Honolulu, 2007. – P. 41–54.
8. Development of Mobile Agents with Aglets (A Java Based Tool) / M. Yadav, P. Sethi, D. Juneja, and N. Chauhan // Int. Journal of Innovations & Advancement in Computer Science. – 2015. – Vol. 4, Special Issue. – P. 245–251.
9. **Evripidou, P.** Metacomputing with Mobile Agents / P. Evripidou, G. Samaras // Int. Journal of Parallel Programming. – 2006. – Vol. 34, № 5. – P. 429–458.
10. **Barelos, D.** Mobile agents procedures: metacomputing in Java / D. Barelos, E. Pitoura, G. Samaras // Proc. of the ICDCS Workshop on Distributed Middleware (in conjunction with the 19th IEEE International Conference on Distributed Computing Systems (ICDCS99)). – Austin, TX USA, 1999. – P. 90–95.
11. Extendible, Mobile-Agent Based Services for the Materialization and Maintenance of Personalized and Shareable Web Views ViSMA / G. Samaras, K. Karenos, P. K. Chrysanthis, and E. Pitoura // In Proc. 11th DEXA Int. Workshop on Mobility in Databases and Distributed Systems. – 2003. – P. 974–979.
12. Web Crawler Based on Mobile Agent and Java Aglets / Md. Abu Kausar, V. S. Dhaka, Sanjeev Kumar Singh // International Journal of Information Technology and Computer Science (IJITCS). MECS Publisher. – 2013. – Vol. 5, № 10. – P. 85–91.
13. **Dada, E. G.** Performance Evaluation of AGLETS and JADE Mobile Agent Using Encryption and Decryption Time / E. G. Dada, S. B. Joseph, and M. K. Mishra // Radioelectronics&Informatics. – 2010. – № 4. – P. 16–20.
14. **Lange, D.** Programming and deploying Java mobile agents with aglets / D. Lange, M. Oshima. – Addison-Wesley Professional, 1998. – 256 p.
15. **Волчихин, В. И.** Организация функционирования облачно-сетевых распределенных вычислительных систем с архитектурой «агенты как сервисы» / В. И. Волчихин, С. А. Зинкин, Н. С. Карамышева // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2019. – № 4 (52). – С. 27–50. – DOI 10.21685/2072-3059-2019-4-3.
16. **Евреинов, Э. В.** Однородные вычислительные системы, структуры и среды / Э. В. Евреинов. – Москва : Радио и связь, 1981. – 208 с.
17. **Ахо, А.** Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – Москва : Мир, 1979. – 536 с.

18. **Ахо, А.** Теория синтаксического анализа, перевода и компиляции. Том 1. Синтаксический анализ / А. Ахо, Дж. Ульман. – Москва : Мир, 1978. – 613 с.
19. **Хопкрофт, Д. Э.** Введение в теорию автоматов, языков и вычислений / Джон Э. Хопкрофт, Раджив Мотвани, Джеффри Д. Ульман. – 2-е изд.. : пер. с англ. – Москва : Вильямс, 2008. – 528 с.
20. **Гаврилов, Г. П.** Задачи и упражнения по дискретной математике / Г. П. Гаврилов, А. А. Сапоженко. – Москва : Физматлит, 2005. – 416 с
21. **Шоломов, Л. А.** Основы теории дискретных логических и вычислительных устройств / Л. А. Шоломов. – Москва : Наука, 1980. – 400 с.
22. **Котов, В. Е.** Теория схем программ / В. Е. Котов, В. К. Сабельфельд. – Москва : Наука, 1991. – 248 с.
23. Java Formal Languages and Automata Package [свободная кроссплатформенная программа для экспериментов с различными объектами, встречающихся в теории формальных языков. Разрабатывается Университетом Дьюка]. – URL: https://ru.wikipedia.org/wiki/JFLAP (дата доступа: 31.10.2020).
24. **Мальцев, А. И.** Алгебраические системы / А. И. Мальцев. – Москва : Наука, 1970. – 393 с.
25. **Плоткин, Б. И.** Универсальная алгебра, алгебраическая логика и базы данных / Б. И. Плоткин. – Москва : Наука, 1991. – 448 с.
26. **Gurevich, Y.** Abstract State Machines: An Overview of the Project / Y. Gurevich // Foundations of Information and Knowledge Systems. Lect. Notes Comput. Sci. – 2004. – Vol. 2942. – P. 6–13.
27. **Boerger, E.** Unifying View of Models of Computation and System Design Frameworks / E. Boerger // Annals of Pure and Applied Logic. – 2005. – Vol. 133. – P. 149–171.
28. Алгеброалгоритмические модели и методы параллельного программирования / Ф. И. Андон, А. Е. Дорошенко, Г. Е. Цейтлин, Е. А. Яценко. – Киев : Академпериодика, 2007. – 634 с.
29. **Глушков, В. М.** Алгебра. Языки. Программирование / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. – Изд. 3-е, перераб. и доп. – Киев : Наукова думка, 1989. – 376 с.
30. Directly executable formal models of middleware for MANET and Cloud Networking and Computing / D. V. Pashchenko, S. A. Zinkin, Mustafa Sadeq Jaafar, D. A. Trokoz, T. U. Pashchenko and M. P. Sinev // Proceedings of the 4th International Conference on Science & Engineering in Mathematics, Chemistry and Physics (SciTech 2016) Bandung, Indonesia, April 23–24, 2016 / Journal of Physics: Conference Series. – 2016. – Vol. 710. – P. 012024. – DOI 10.1088/1742-6596/710/1/012024.
31. **Зинкин, С. А.** Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (механизмы интерпретации и варианты использования) / С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 4. – С. 37–51.
32. **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.
33. **Зинкин, С. А.** Элементы новой объектно-ориентированной технологии для моделирования и реализации систем и сетей хранения и обработки данных / С. А. Зинкин // Информационные технологии. – 2008. – № 10. – С. 20–27.
34. **Колмогоров, А. Н.** Математическая логика / А. Н. Колмогоров, А. Г. Драгалин. – Москва : Изд-во УРСС, МГУ, 2005. – 240 с.
35. **Поликарпова, Н. И.** Автоматное программирование / Н. И. Поликарпова, А. А. Шалыто. – Санкт-Петербург : Изд-во Санкт-Петербургского гос. ун-та информационных технологий, механики и оптики, 2008. – 167 с.

36. **Туккель, Н. И.** От тьюрингова программирования к автоматному / Н. И. Туккель, А. А. Шалыто // Мир ПК. – 2002. – № 2. – С. 144–149.
37. Машина Тьюринга. – URL: https://ru.wikipedia.org/wiki/Машина_Тьюринга (дата обращения: 31.10.2020).
38. Neural Turing Machines / Alex Graves, Greg Wayne, Ivo Danihelka. – London, UK, 2014. – P. 1–26. – arXiv:1410.5401

References

1. Tanenbaum A. S., Van Steen M. *Distributed Systems: principles and paradigms. 2nd Edition*. Pearson Education, Inc., 2007, 669 p.
2. *FIPA Specifications*. Available at: http://www.fipa.org/specifications/index.html (accessed Nov. 12, 2019).
3. Bordini R. H. et al. *Informatica*. 2006, vol. 30, pp. 33–44.
4. Kravari K. A, Bassiliades N. *Journal of Artificial Societies and Social Simulation*. 2015, vol. 18 (1), no. 11, pp. 1–18.
5. Cynthia N., Gregory M. *Journal of Artificial Societies and Social Simulation*. 2009, vol. 12 (2). Available at: http://jasss.soc.surrey.ac.uk/12/2/2.html.
6. Bellifemine F. L., Caire G., Greenwood D. *Developing multi-agent systems with JADE*. Wiley. 2007, 300 p. DOI 10.1002/9780470058411
7. Gorodetsky V., Karsaev O., Samoylov V., Serebryakov S. *The AAMAS Sixth International Workshop on Agents and Peer-to-Peer. Computing (AP2PC 2007)*. Honolulu, 2007, pp. 41–54.
8. Yadav M., Sethi P., Juneja D., Chauhan N. *Int. Journal of Innovations & Advancement in Computer Science*. 2015, vol. 4, Special Issue, pp. 245–251.
9. Evripidou P., Samaras G. *Int. Journal of Parallel Programming*. 2006, vol. 34, no. 5, pp. 429–458.
10. Barelos D., Pitoura E., Samaras G. *Proc. of the ICDCS Workshop on Distributed Middleware (in conjunction with the 19th IEEE International Conference on Distributed Computing Systems (ICDCS99))*. Austin, TX USA, 1999, pp. 90–95.
11. Samaras G., Karenos K., Chrysanthis P. K., Pitoura E. *In Proc. 11th DEXA Int. Workshop on Mobility in Databases and Distributed Systems*. 2003, pp. 974–979.
12. Md. Abu Kausar, Dhaka V. S., Sanjeev Kumar Singh *International Journal of Information Technology and Computer Science (IJITCS)*. MECS Publisher. 2013, vol. 5, no. 10, pp. 85–91.
13. Dada E. G., Joseph S. B., Mishra M. K. *Radioelectronics&Informatics*. 2010, no. 4, pp. 16–20.
14. Lange D., Oshima M. *Programming and deploying Java mobile agents with aglets*. Addison-Wesley Professional, 1998, 256 p.
15. Volchikhin V. I., Zinkin S. A., Karamysheva N. S. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings Volga region. Engineering sciences]. 2019, no. 4 (52), pp. 27–50. DOI 10.21685/2072-3059-2019-4-3. [In Russian]
16. Evreinov E. V. *Odnorodnye vychislitel'nye sistemy, struktury i sredy* [Homogeneous computing systems, structures and environments]. Moscow: Radio i svyaz', 1981, 208 p. [In Russian]
17. Akho A., Khopkroft Dzh., Ul'man Dzh. *Postroenie i analiz vychislitel'nykh algoritmov* [The design and analysis of computer algorithms]. Moscow: Mir, 1979, 536 p. [In Russian]
18. Akho A., Ul'man Dzh. *Teoriya sintaksicheskogo analiza, perevoda i kompilyatsii. Tom 1. Sintaksicheskiy analiz* [The theory of parsing, translation and compiling. Volume 1. Parsing]. Moscow: Mir, 1978, 613 p. [In Russian]

19. Khopkroft D. E., Motvani R., Ul'man D. D. *Vvedenie v teoriyu avtomatov, yazykov i vychisleniy* [Introduction to automata theory, languages, and computation]. 2nd ed.: transl. from Engl. Moscow: Vil'yams, 2008, 528 p. [In Russian]
20. Gavrilov G. P., Sapozhenko A. A. *Zadachi i uprazhneniya po diskretnoy matematike* [discrete mathematics problems and exercises]. Moscow: Fizmatlit, 2005, 416 p. [In Russian]
21. Sholomov L. A. *Osnovy teorii diskretnykh logicheskikh i vychislitel'nykh ustroystv* [Fundamentals of the theory of discrete logical and computing devices]. Moscow: Nauka, 1980, 400 p. [In Russian]
22. Kotov V. E., Sabel'fel'd V. K. *Teoriya skhem programm* [Program circuit theory]. Moscow: Nauka, 1991, 248 p. [In Russian]
23. *Java Formal Languages and Automata Package [svobodnaya krossplatformennaya programma dlya eksperimentov s razlichnymi ob"ektami, vstrechayushchikhsya v teorii formal'nykh yazykov. Razrabatyvaetsya Universitetom D'yuka]* [Java Formal Languages and Automata Package [a free cross-platform program for experimenting with various objects found in the theory of formal languages. Developed by Duke University]]. Available at: https://ru.wikipedia.org/wiki/JFLAP (accessed Oct. 31, 2020). [In Russian]
24. Mal'tsev A. I. *Algebraicheskie sistemy* [Algebraic Systems]. Moscow: Nauka, 1970, 393 p. [In Russian]
25. Plotkin B. I. *Universal'naya algebra, algebraicheskaya logika i bazy dannykh* [Universal algebra, algebraic logic, and databases.]. Moscow: Nauka, 1991, 448 p. [In Russian]
26. Gurevich Y. *Foundations of Information and Knowledge Systems. Lect. Notes Comput. Sci*. 2004, vol. 2942, pp. 6–13.
27. Boerger E. *Annals of Pure and Applied Logic*. 2005, vol. 133, pp. 149–171.
28. Andon F. I., Doroshenko A. E., Tseytlin G. E., Yatsenko E. A. *Algebroalgoritmicheskie modeli i metody parallel'nogo programmirovaniya* [Algebraic algorithmic models and methods of parallel programming]. Kiev: Akademperiodika. 2007, 634 p.
29. Glushkov V. M., Tseytlin G. E., Yushchenko E. L. *Algebra. Yazyki. Programmirovanie* [Algebra. Languages. Programming]. 3rd ed., rev. and suppl. Kiev: Naukova dumka, 1989, 376 p.
30. Pashchenko D. V., Zinkin S. A., Mustafa Sadeq Jaafar, Trokoz D. A., Pashchenko T. U., Sinev M. P. *Proceedings of the 4th International Conference on Science & Engineering in Mathematics, Chemistry and Physics (SciTech 2016)* (*Bandung, Indonesia, April 23–24, 2016)*. 2016, vol. 710, p. 012024. DOI 10.1088/1742-6596/710/1/012024.
31. Zinkin S. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2007, no. 4, pp. 37–51. [In Russian]
32. Zinkin S. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2007, no. 3, pp. 13–22. [In Russian]
33. Zinkin S. A. *Informatsionnye tekhnologii* [Information technologies]. 2008, no. 10, pp. 20–27. [In Russian]
34. Kolmogorov A. N., Dragalin A. G. *Matematicheskaya logika* [Mathematical logic]. Moscow: Izd-vo URSS, MGU, 2005, 240 p. [In Russian]
35. Polikarpova N. I., Shalyto A. A. *Avtomatnoe programmirovanie* [Automated programming]. Saint-Petersburg: Izd-vo Sankt-Peterburgskogo gosudarstvennogo universiteta informatsionnykh tekhnologiy, mekhaniki i optiki, 2008, 167 p. [In Russian]
36. Tukkel' N. I., Shalyto A. A. *Mir PK* [PC world]. 2002, no. 2, pp. 144–149. [In Russian]
37. *Mashina T'yuringa* [Turing machine]. Available at: https://ru.wikipedia.org/wiki/Mashina_T'yuringa (accessed Oct. 31, 2020). [In Russian]
38. Graves A., Wayne G., Danihelka I. *Neural Turing Machines*. London, UK, 2014, pp. 1–26. arXiv:1410.5401

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

E-mail: cnit@pnzgu.ru

Volchihin Vladimir Ivanovich
Doctor of engineering sciences, professor, president of 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: vt@pnzgu.ru

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

Карамышева Надежда Сергеевна
кандидат технических наук, доцент, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: vt@pnzgu.ru

Karamysheva Nadezhda Sergeevna
Candidate of engineering sciences, associate professor, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Образец цитирования:

Волчихин, В. И. Концептуальные модели функциональной архитектуры мобильных реконфигурируемых агентно-ориентированных распределенных вычислительных систем / В. И. Волчихин, С. А. Зинкин, Н. С. Карамышева // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2020. – № 4 (56). – С. 65–91. – DOI 10.21685/2072-3059-2020-4-6.