ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Редакционный совет
Садовничий В.А., акад. РАН (*председатель*)
Бетелин В.Б., акад. РАН
Васильев В.Н., чл.-корр. РАН
Жижченко А.Б., акад. РАН
Макаров В.Л., акад. РАН
Панченко В.Я., акад. РАН
Стемпковский А.Л., акад. РАН
Ухлинов Л.М., д.т.н.
Федоров И.Б., акад. РАН
Четверушкин Б.Н., акад. РАН

Главный редактор
Васенин В.А., д.ф.-м.н., проф.

Редколлегия
Антонов Б.И.
Афонин С.А., к.ф.-м.н.
Бурдонов И.Б., д.ф.-м.н., проф.
Борзовс Ю., проф. (Латвия)
Гаврилов А.В., к.т.н.
Галатенко А.В., к.ф.-м.н.
Корнеев В.В., д.т.н., проф.
Костюхин К.А., к.ф.-м.н.
Махортов С.Д., д.ф.-м.н., доц.
Манцивода А.В., д.ф.-м.н., доц.
Назиров Р.Р., д.т.н., проф.
Нечаев В.В., д.т.н., проф.
Новиков Б.А., д.ф.-м.н., проф.
Павлов В.Л. (США)
Пальчунов Д.Е., д.ф.-м.н., доц.
Петренко А.К., д.ф.-м.н., проф.
Позднеев Б.М., д.т.н., проф.
Позин Б.А., д.т.н., проф.
Серебряков В.А., д.ф.-м.н., проф.
Сорокин А.В., к.т.н., доц.
Терехов А.Н., д.ф.-м.н., проф.
Филимонов Н.Б., д.т.н., проф.
Шапченко К.А., к.ф.-м.н.
Шундеев А.С., к.ф.-м.н.
Щур Л.Н., д.ф.-м.н., проф.
Язов Ю.К., д.т.н., проф.
Якобсон И., проф. (Швейцария)

Редакция
Лысенко А.В., Чугунова А.В.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Годунов А. Н., Солдатов В. А. Конфигурирование многопроцессорных систем в операционной системе реального времени Багет 243

Иванов И. Ю. Расширенная модель LP-вывода на булевой решетке . . . 252

Костенко К. И. Правила оператора вывода для формализма абстрактного пространства знаний 258

Большаков А. А., Макарук Р. В. Программный комплекс для анализа характеристик и оценки уровня информационной безопасности вычислительной сети на основе нечетких моделей 268

Полевая О. М. Архитектура корпоративной информационной системы стратегического управления 283

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

© Издательство "Новые технологии", "Программная инженерия", 2016

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 7
N 6
2016

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

Editorial Council:

SADOVNICHY V. A., Dr. Sci. (Phys.-Math.), Acad. RAS (*Head*)
BETELIN V. B., Dr. Sci. (Phys.-Math.), Acad. RAS
VASIL'EV V. N., Dr. Sci. (Tech.), Cor.-Mem. RAS
ZHIZHCHENKO A. B., Dr. Sci. (Phys.-Math.), Acad. RAS
MAKAROV V. L., Dr. Sci. (Phys.-Math.), Acad. RAS
PANCHENKO V. YA., Dr. Sci. (Phys.-Math.), Acad. RAS
STEMPKOVSKY A. L., Dr. Sci. (Tech.), Acad. RAS
UKHLINOV L. M., Dr. Sci. (Tech.)
FEDOROV I. B., Dr. Sci. (Tech.), Acad. RAS
CHETVERTUSHKIN B. N., Dr. Sci. (Phys.-Math.), Acad. RAS

Editor-in-Chief:

VASENIN V. A., Dr. Sci. (Phys.-Math.)

Editorial Board:

ANTONOV B.I.
AFONIN S.A., Cand. Sci. (Phys.-Math)
BURDONOV I.B., Dr. Sci. (Phys.-Math)
BORZOVS JURIS, Dr. Sci. (Comp. Sci), Latvia
GALATENKO A.V., Cand. Sci. (Phys.-Math)
GAVRILOV A.V., Cand. Sci. (Tech)
JACOBSON IVAR, Dr. Sci. (Philos., Comp. Sci.), Switzerland
KORNEEV V.V., Dr. Sci. (Tech)
KOSTYUKHIN K.A., Cand. Sci. (Phys.-Math)
MAKHORTOV S.D., Dr. Sci. (Phys.-Math)
MANCIVODA A.V., Dr. Sci. (Phys.-Math)
NAZIROV R.R. , Dr. Sci. (Tech)
NECHAEV V.V., Cand. Sci. (Tech)
NOVIKOV B.A., Dr. Sci. (Phys.-Math)
PAVLOV V.L., USA
PAL'CHUNOV D.E., Dr. Sci. (Phys.-Math)
PETRENKO A.K., Dr. Sci. (Phys.-Math)
POZDNEEV B.M., Dr. Sci. (Tech)
POZIN B.A., Dr. Sci. (Tech)
SEREBRJAKOV V.A., Dr. Sci. (Phys.-Math)
SOROKIN A.V., Cand. Sci. (Tech)
TEREKHOV A.N., Dr. Sci. (Phys.-Math)
FILIMONOV N.B., Dr. Sci. (Tech)
SHAPCHENKO K.A., Cand. Sci. (Phys.-Math)
SHUNDEEV A.S., Cand. Sci. (Phys.-Math)
SHCHUR L.N., Dr. Sci. (Phys.-Math)
YAZOV Yu. K., Dr. Sci. (Tech)

Editors: LYSENKO A.V., CHUGUNOVA A.V.

CONTENTS

Godunov A. N., Soldatov V. A. Configuring Multiprocessor Systems in RTOS Baget 243

Ivanov I. Yu. Extended Model of LP-inference on Boolean Lattice 252

Kostenko K. I. The Rules for Abstract Knowledge Spaces Inference Operator 258

Bolshakov A. A., Makaruk R. V. The Software Solution for the Analysis of the Characteristics and Evaluation of Information Security Level of a Computer Network based on Fuzzy Models 268

Polevaya O. M. Architecture of Information System for Strategic Management 283

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.436.4 DOI: 10.17587/prin.7.243-251

А. Н. Годунов, канд. физ.-мат. наук, зав. отделом, e-mail: nkag@niisi.ras.ru,
В. А. Солдатов, канд. техн. наук, ст. науч. сотр., e-mail: nkvalera@niisi.ras.ru,
Научно-исследовательский институт системных исследований РАН (НИИСИ РАН),
г. Москва

Конфигурирование многопроцессорных систем в операционной системе реального времени Багет

Для многопроцессорных вычислительных систем представлено описание взаимосвязей устройств, функционирующих в коммуникационной среде RapidIO. Рассмотрен язык высокоуровневого описания системы (ВОС), разработанный для конфигурирования многопроцессорных систем, работающих под управлением операционной системы реального времени (ОСРВ) Багет и средства трансляции ВОС. Предложена информационная поддержка для решения задач инициализации, резервирования и разрыва/восстановления соединения на основе информации, содержащейся в описании системы.

Ключевые слова: *ОСРВ Багет, конфигурирование, RapidIO, многопроцессорные системы, маршрутизация*

Введение

Параллельные вычисления, выполняемые на различных процессорах, являются одним из способов повышения производительности вычислительных систем. Различные устройства, входящие в распределенную вычислительную систему, могут соединяться между собой посредством шин с общим доступом. Специально для соединения микросхем в рамках одной печатной платы, а также для соединения между собой нескольких плат была разработана коммуникационная среда RapidIO [1]. По таким характеристикам, как пропускная способность, задержки при передаче пакетов в сети, надежность и масштабируемость RapidIO не только не уступает, но и превосходит другие коммуникационные средства [2].

Среди комплексов программ, которые хорошо подходят для выполнения на многопроцессорных установках, можно выделить программы цифровой обработки сигналов. Так, например, на отечественной линейке процессоров [3] разработана технология программирования [4], позволяющая разбивать обработку сигналов на отдельные стадии, выделять каждой стадии необходимое количество вычислительных ресурсов и организовывать передачу данных между стадиями.

При проектировании подобных многопроцессорных систем особое значение приобретают такие задачи, как:

— определение топологии сети, которая связывает узлы системы между собой;

— начальная инициализация коммуникационной среды;

— согласованная загрузка всех процессоров системы;

— конфигурирование программ, выполняющихся на процессорах системы;

— запись в перепрограммируемое постоянное запоминающее устройство (ППЗУ) каждого процессора требуемого программного обеспечения (ПО);

— резервирование некоторой части системы для повышения надежности функционирования;

— возможность выполнения требуемых вычислений на некотором подмножестве оборудования системы;

— сохранения/восстановления работоспособности после выключения/включения части оборудования системы.

Алгоритмы, выполняющие перечисленные выше задачи, для своей работы требуют детального описания состава оборудования системы и его свойств, которое определяется при конфигурировании системы.

1. Коммуникационная среда

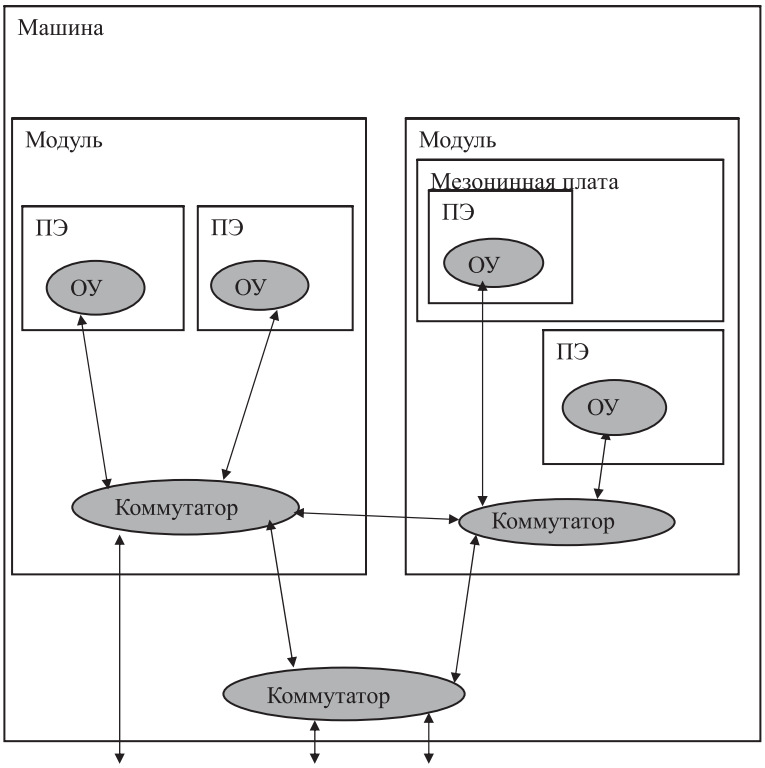
Коммуникационная среда RapidIO представляет собой сеть с коммутацией пакетов, которая состоит из узлов, соединенных физическими каналами связи [5, 6]. Каждый узел имеет один или несколько портов, являющихся конечными точками физических каналов. При описании возможных типов узлов будем ссылаться на линейку реального оборудования, разработанного во ВНИИСИ РАН [2, 3]. Узел коммуникационной среды RapidIO может принадлежать к одному из описанных ниже типов.

Оконечное устройство (ОУ) характеризуется уникальным в пределах вычислительной системы RIO-идентификатором — числом, лежащим в определенном диапазоне, который зависит от аппаратной реализации RapidIO. Как правило, ОУ имеет только один порт. В реальной линейке оборудования ОУ представлено микросхемой ВМ7, которая используется в микропроцессоре Комдив128-РИО.

Коммутатор всегда имеет несколько портов и предназначен для маршрутизации, т. е. сообщение, полученное из одного порта, в соответствии с таблицей маршрутизации перенаправляется в другой порт. Таблицы маршрутизации записываются при начальной инициализации коммуникационной среды RapidIO. Коммутаторы бывают двух видов: с одной таблицей маршрутизации для всего устройства и с отдельной таблицей маршрутизации для каждого порта. Коммутатор не имеет RIO-идентификатора. Микросхема КП3Я с восемью параллельными портами из линейки реального оборудования является примером узла такого типа.

Гибридное устройство является одновременно и коммутатором, и ОУ. Гибридному устройству, в зависимости от его использования в системе, может быть как назначен RIO-идентификатор, так и нет. В линейке реального оборудования этот тип устройства представлен микросхемой с двумя портами ВМ6, которая используется в качестве ОУ в микропроцессоре Комдив64-РИО, и микросхемой также с двумя портами ВГ18Я, которая используется в качестве коммутатора.

На рисунке коммуникационная среда RapidIO, которая является составной частью распределенной вычислительной системы, выделена серым цветом. Каждый узел описываемой системы будем называть **компонентом**. Тогда всю распределенную вычислительную систему можно представить в виде иерархии типов компонентов, представленных на рисунке.



Иерархия компонентов системы

Процессорный элемент (ПЭ) представляет собой микропроцессор с подключенным к нему ОУ, а также различными запоминающими устройствами, наплатными контроллерами, внешними интерфейсами и прочими устройствами, доступ к которым возможен только из этого ПЭ. В линейке реального оборудования этот компонент представлен микропроцессорами Комдив128-РИО и Комдив64-РИО. В их состав входит ОУ и сетевой интерфейс TCP/IP.

Мезонинная плата (мезонин) может включать ПЭ. В линейке реального оборудования этот компонент представлен микропроцессором на мезонине Комдив128-РИО с ОУ.

Модуль может включать ПЭ, ОУ, мезонины, коммутаторы и преобразователи "витая пара — оптика". В линейке реального оборудования представлены шесть различных видов модулей с процессорами, коммутаторами и преобразователем "витая пара — оптика".

Машина (прибор, крейт) состоит из модулей. В линейке реального оборудования содержатся бортовые цифровые вычислительные машины, которые в своем составе имеют от четырех до двадцати различных модулей.

Относительное положение ПЭ в иерархии компонентов системы определяется географическим адресом — числом, полученным на основе физических номеров машины, модуля в машине и ОУ на модуле. Процессорный элемент может самостоятельно определить свой географический адрес, прочитав содержимое определенного регистра.

2. Конфигурирование распределенной системы

Конфигурирование ПЭ, который является узлом распределенной вычислительной системы, условно можно разделить на две части:

- описание ПЭ как отдельного вычислительного узла (локальная конфигурация);
- описание ПЭ как части распределенной системы (глобальная конфигурация).

К локальной конфигурации можно отнести параметры операционной системы (ОС), не относящиеся к взаимодействию с другими узлами системы, локальная конфигурация узла не зависит от конфигурации других узлов.

В глобальной конфигурации описываются параметры, определяющие взаимосвязь между узлами систе-

мы, а именно: топология (перечисление узлов системы и соединения их между собой), каналы передачи данных, адреса сетевых интерфейсов, указания на части узлов системы, которые используются как резервные по отношению к основным, и т. д. Параметры глобальной конфигурации, относящиеся к различным узлам, должны быть взаимосвязаны. Несогласованное изменение параметров одного узла может привести к неработоспособности всей системы. Поэтому следует рассматривать глобальную конфигурацию как единую для всех узлов системы, даже если физически ее фрагменты, относящиеся к различным узлам, хранятся отдельно.

До определенного момента в ОСРВ Багет [7, 8] конфигурация узла состояла из:

— конфигурации ядра ОС;

— ARINC-конфигурирования.

Результаты конфигурации запоминались в соответствующих файлах. При этом каждый такой файл относился к пакету поддержки модуля (ППМ) и содержал исключительно локальную конфигурационную информацию. Остальные конфигурационные файлы включают в себя данные, относящиеся как к локальной, так и к глобальной конфигурации. Согласование отдельных частей глобальной конфигурации, разбросанной по конфигурационным файлам различных узлов, представляет собой непростую задачу, которая существенно усложняется с появлением коммуникационной среды RapidIO. Уже сейчас число ПЭ в некоторых вычислительных системах достигает нескольких сотен.

Специально для решения этой задачи были разработаны язык высокоуровневого описания системы (ВОС) и утилита xmltool.

На языке ВОС описываются данные о взаимосвязях узлов системы, о местах хранения загрузочных образов ПЭ, определяются сетевые адреса узлов, сведения, необходимые для записи в ППЗУ процессоров требуемого ПО, и другие данные, описывающие многопроцессорную систему в целом. Язык ВОС ориентирован на описание систем, в которых основной коммуникационной средой является RapidIO, а во вспомогательных целях используется Ethernet. Высокоуровневое описание системы хранится в одном или в нескольких взаимосвязанных файлах в формате XML. Исходное описание обрабатывается компилятором ВОС (hldc), который проверяет корректность заданной информации и преобразует ее в низкоуровневое описание системы (файл в формате XML). Далее проводится его преобразование в целевое описание системы (ЦОС) — объектный файл в формате ELF. В ЦОС содержится информация, заданная в ВОС, а также дополнительные данные, рассчитанные в ходе работы компилятора, например, таблицы маршрутизации для коммутаторов системы. Доступ к ЦОС выполняется с помощью библиотеки поддержки коммуникационной среды RapidIO, которая поставляется вместе с ОСРВ Багет.

Утилита xmltool, выполняемая на инструментальной машине, позволяет проверять непротиворечивость конфигураций отдельных узлов и автоматически формировать согласованные файлы ARINC-конфигурации.

3. Язык ВОС

Вычислительная система может состоять из большого числа компонентов, при этом разновидностей этих компонентов немного. Чтобы не описывать внутреннее устройство каждого компонента на языке ВОС, предлагается описывать **тип** компонента. Каждый тип соответствует множеству физически идентичных компонентов. В качестве примеров типа можно назвать "процессорный элемент Комдив128-РИО" или "модуль ЦП-РИО-64А". Если описанный тип встречается в системе только один раз, тогда можно не описывать компонент, соответствующий этому типу, так как описание типа компонента является также и описанием компонента. Вся описываемая система представляется в виде одного высокоуровневого компонента, содержащего все остальные компоненты. Для этой цели описывается особый тип — "система" (System). Таким образом, для описания аппаратного устройства системы надо описать ее тип, а также типы всех компонентов, которые входят в эту систему. Описание типа состоит из описания аппаратных особенностей компонентов этого типа и не содержит никакой информации об использовании компонентов данного типа, поэтому описание типов может создаваться и поставляться производителем компонентов этого типа. Например, если в системе используются только типовые машины, тогда системному интегратору при описании аппаратной конфигурации достаточно описать тип системы, перечислив в нем используемые машины, указав их типы и взаимосвязи.

Программная конфигурация системы описывается отдельно от аппаратной и состоит в задании значений атрибутов для компонентов системы. Таким образом, можно сказать, что ВОС состоит из двух логических частей:

— описания типов (структуры);

— описания атрибутов (свойств).

3.1. Типы и компоненты

Каждый тип компонента, описываемый в ВОС, должен принадлежать одному из перечисленных далее **классов**:

- RioEndpoint — ОУ коммуникационной среды RapidIO;
- RioSwitch — коммутатор RapidIO;
- RioSwitchingEndpoint — гибридное устройство, является одновременно и коммутатором, и ОУ;
- ProcessingElement — ПЭ;
- Mezzonine — мезонинная плата;
- Module — модуль;
- Machine — машина (крейт);
- MachineGroup — группа машин;
- System — система.

Классу System может принадлежать только один тип — система. Компонент данного типа называется компонентом-системой.

Аппаратные компоненты системы вложены друг в друга и образуют иерархию. Оконечные устрой-

ства, коммутаторы и гибридные устройства не имеют дочерних компонентов и соответствующие им компоненты являются элементарными устройствами RapidIO. Хотя на практике ОУ и ПЭ физически представляют собой одну микросхему, при описании типов считается, что ОУ является дочерним компонентом ПЭ. Описание типа включает в себя имя типа, класс типа, описания дочерних компонентов, внутренних связей, внешних портов и аппаратных особенностей типа.

Дочерним компонентам типа назначаются уникальные в пределах этого уровня иерархии имена. Такой подход позволяет идентифицировать любой компонент системы его полным именем, составленным из имен всех содержащих его компонентов в порядке вложенности, начиная с самого верхнего, исключая имя компонента типа System, и разделенных символом "/". Например, полное имя ПЭ состоит из имени машины, имени модуля и имени самого процессорного элемента — "/M01-00/B64-03/K64-1".

Для каждого дочернего компонента должен быть указан его тип, описанный в ВОС. Все компоненты, кроме компонента, принадлежащего к типу System, должны иметь хотя бы один внешний порт. Для компонента элементарного типа внешний порт соответствует физическому порту RapidIO. Для компонентов всех остальных типов каждый внешний порт соответствует одному из внешних портов дочернего элемента. Например, внешние порты компонента типа Module представляют собой выведенные наружу внешние порты расположенных на модуле дочерних компонентов, таких как ПЭ, мезонин, коммутатор, или гибридное устройство.

Внутренние связи компонента — это связи между внешними портами его дочерних компонентов. Описание внутренних связей определяет топологию физических связей между компонентами системы.

Описание типа также может включать в себя описание различных специфических аппаратных особенностей, зависящих от класса. Например, идентификатор модели устройства RapidIO (для типов элементарных классов), подключенные интерфейсы ethternet (для типов ПЭ).

3.2. Маршрутизация и режимы использования ЦОС

Компилятор ВОС самостоятельно рассчитывает маршруты между всеми ОУ, присутствующими в системе. Маршруты прокладываются по кратчайшим путям. Если требуется, чтобы маршрут между какими-то конкретными ОУ отличался от рассчитываемого автоматически, тогда этот маршрут можно задать явно с помощью XML-элемента Route. Все остальные маршруты по-прежнему будут автоматически рассчитываться компилятором ВОС.

Информация, содержащаяся в ЦОС, может использоваться в одном из нескольких возможных режимов, каждый из которых определяет свои значения атрибутов. В языке ВОС определены: режим загрузки; режим записи в ППЗУ; режим штатной работы ОС. Первые два режима предназначены для загрузчика, третий используется ОС. В каждом режиме для загрузчика системным интегратором может быть задан произвольный подрежим. В зависимости от режима и подрежима атрибуты компонентов могут иметь различные значения, что позволяет выполнять загрузку или запись в ППЗУ с различными значениями параметров, например, файлы, используемые загрузчиком, могут располагаться на различных устройствах. При загрузке исполняемые файлы, содержащие требуемое ПО, помещаются в оперативную память ПЭ системы и им передается управление. При режиме записи в ППЗУ проводится запись указанных файлов в ППЗУ ПЭ системы.

Режим загрузки задается XML-элементом LMode, режим записи в ППЗУ — XML-элементом RMode, а режим штатной работы ОС задается XML-элементом OSMode. Значения атрибутов, заданные вне перечисленных выше XML-элементов, используются во всех подрежимах всех режимов.

3.3. Атрибуты

Атрибут — это данные, заданные для некоторого компонента системы и описывающие определенный аспект его работы. С помощью атрибутов в ВОС задаются конфигурационные данные, которые определяют:

- особенности идентификации и маршрутизации в среде RapidIO;
- способы взаимодействия с инструментальной ЭВМ;
- настройки сетевых интерфейсов;
- параметры загрузки ПЭ;
- параметры начального старта системы;
- резервирование части системы.

Данные, назначенные атрибуту для некоторого аппаратного компонента, будем называть набором значений атрибута для этого компонента. Набор значений, в зависимости от атрибута, может включать в себя одно, несколько или ни одного значения. Наборы значений одного атрибута для различных компонентов одного типа могут не совпадать. Каждое значение атрибута представляет собой текстовую строку или набор значений его податрибутов.

Имя атрибута совпадает с именем соответствующего XML-элемента, используемого для его описания.

Атрибут может принимать во многих компонентах одинаковые значения или значения, имеющие схожую структуру. В языке ВОС для того чтобы избежать многократного указания одинаковых или однотипных значений предусмотрены переменные, фильтры и условные конструкции.

Переменные играют вспомогательную роль и используются для параметризации значений атрибутов и других переменных, а также в качестве селекторов в условных конструкциях. Задание значения переменной выполняется с помощью определения переменной. Каждое определение включает в себя имя переменной и ее значение. Значение, заданное

в определении переменной, может меняться от одного компонента к другому в силу использования простой или арифметической подстановки.

Простая подстановка — это замена конструкции вида $(var), где var — имя некоторой переменной, на значение указанной переменной.

Арифметическая подстановка, которая также может использоваться для определения значений атрибутов или переменных, задается в виде $[expr], где expr — арифметическое выражение, которое может содержать целые числа, имена переменных, круглые скобки и арифметические операторы "+", "–", "*", "/".

Приведем пример задания значений переменных:

```
<!-- Имя вычислительной системы-->
<Var Name="PRJ_NAME" Value="Stand-001"/>
<!--Имена файлов ЦОС-->
<Var Name="PRJ_COS" Value = "StandCOS/>
<Var Name="PRJ_COS_OBJ" Value = $(PRJ_COS).o/>
```

Имена, смысл значений и способ использования переменных определяются системным интегратором при описании системы на языке ВОС. Однако следующие специальные переменные считаются заданными неявно для всех компонентов и имеют особый смысл:

- ClassName — имя класса компонента;
- TypeName — имя типа компонента;
- MachinePhysNum — физический номер машины, содержащей компонент;
- ModPhysNum — физический номер модуля, содержащего компонент;
- ModPosition — номер позиции модуля в крейте;
- ProcPhysNum — физический номер ПЭ, содержащего компонент;
- ShortName — имя компонента, заданное в определении его родительского типа;
- FullName — имя компонента и имена всех содержащих его компонентов, расположенных в порядке вложенности, начиная с самого крупного, и разделенные символами "/";
- MachineGroupName — имя группы машин, в которой расположен компонент.
- MachineName — имя машины, в которой расположен компонент;
- ModuleName — имя модуля, на котором расположен компонент;
- FullMachineName — полное имя машины, в которой расположен компонент;
- RioId — RIO-идентификатор компонента, который назначается компилятором ВОС только ПЭ (и соответствующим им ОУ) и ОУ, которые связаны с высокоскоростными каналами (ВСК).

Определения переменных и атрибутов могут находиться внутри **фильтров** (Filter), которые позволяют указать, к каким именно компонентам относятся описанные внутри него определения переменных и атрибутов. Условие выбора имен компонентов может быть простым или составным. Простое условие описывается XML-элементом Match. Условию удовлетворяют компоненты, полные имена которых содержат указанный шаблон (Pattern). На основе простых условий можно описывать составные условия с помощью XML-элементов Not, Or, And. Если один фильтр вложен во второй, то компонент соответствует вложенному фильтру только тогда, когда он соответствует и внешнему фильтру, например:

```
<Filter>
   <!-- ... для всех ПЭ на модулях B64 ... -->
   <Match Pattern = "B64-"/>
    <Filter>
     <!-- ... кроме ПЭ на модуле B64-05-->
      <Not>
       <Match Pattern="B64-05"/>
      </Not>
    </Filter>
</Filter>
```

Условная конструкция (Cond) позволяет в зависимости от выполнения условия включать или отбрасывать определения переменных и атрибутов, описанных внутри конструкции. Отличие условной конструкции от фильтра заключается в том, что в фильтре условие накладывается на имя компонента, а в условной конструкции — на значение указанной переменной. Так же, как и для фильтров, условие может быть простым и составным. Простое условие описывается XML-элементом Match. Условию удовлетворяют все компоненты, для которых значение указанной в нем переменной (Var) содержит в себе заданный шаблон (Pattern). Составное условие (как и для фильтров) описывается на основе простых с помощью XML-элементов Not, Or, And. В шаблоне переменной, для которой проверяется условие, может выполняться подстановка переменных, т. е. шаблон может быть задан в виде значения некоторой переменной.

В качестве примера можно рассмотреть атрибут EthernetInterface, значения которого описывают настройки сетевого интерфейса. Так как к ПЭ может быть подключено несколько сетевых интерфейсов, этот атрибут может иметь несколько значений, каждое из которых будет описывать параметры одного интерфейса. Наборы значений этого атрибута для разных ПЭ имеют схожую структуру и содержат значения как совпадающие между собой, так и отличающиеся друг от друга. Описание атрибута EthernetInterface для всех ПЭ элементов, содержащих в полном имени "K64-1", может иметь следующий вид:

```
<Filter>
  <Match Pattern="K64-1" />
<Cond>
  <Match Var = "ModPosition" Pattern="2"/>
   <EthernetInterface>
     <Name>de0</Name>
     <IpAddr>192.168.0.$(RioId)</IpAddr>
     <NetMask>255.255.255.0</NetMask>
   </EthernetInterface>
```

```
</Cond>
  <EthernetInterface>
    <Name>rio0</Name>
    <IpAddr>192.168.1.$(RioId)</IpAddr>
    <NetMask>255.255.255.0</NetMask>
    <Type>rio</Type>
  </EthernetInterface>
</Filter>
```

Как видно из приведенного примера, значения атрибута можно сразу задать для набора компонентов, выбранных из всего множества компонентов с помощью фильтрации по именам. Тем компонентам, которые располагаются на втором модуле, назначается два сетевых интерфейса.

Значение IP-адреса (атрибут IpAddr) определяется в зависимости от значения переменной RioId (RIO-идентификатора компонента).

4. Синтаксис ВОС

Высокоуровневое описание системы представляет собой XML-документ, соответствующий определенной схеме. Этот документ для удобства составления и сопровождения может быть распределен по разным файлам. Отдельный файл может содержать описание определений некоторой части системы, например, в таком файле могут быть описаны типы стандартных аппаратных компонентов. Для определения набора файлов ВОС пользователем выделяется корневой файл, который указывается в качестве исходного компилятору. В этом файле должен быть описан тип класса System. Остальные типы, атрибуты и переменные могут указываться в различных файлах ВОС в произвольном порядке. Рассмотрим основные элементы ВОС.

Элементы SystemDescription, Include и TryInclude используются для того, чтобы описание системы можно было распределить по различным файлам.

SystemDescription — это корневой элемент каждого файла, содержащего исходный текст ВОС. Ниже перечислены атрибуты этого элемента, каждый из которых может быть опущен:

- HldName — имя описания, учитывается только в случае, когда файл, содержащий элемент, является корневым, т. е. это тот файл, имя которого подается на вход транслятору ВОС;
- HldVersion — номер версии ВОС;
- FileVersion — номер версии файла, содержащего элемент.

Имя и номера версий не проверяются и не интерпретируются, их смысл определяется пользователем. При обработке ВОС в ЦОС сохраняется весь список файлов с их версиями.

Ниже приведен список дочерних элементов SystemDescription, каждый из них может быть задан произвольное число раз:

- Include — обязательное включение файла ВОС;
- TryInclude — необязательное включение файла ВОС;
- Type — описание типов аппаратных компонентов;
- Var — определение переменной;
- Cond — условная конструкция;
- Filter — фильтр по именам компонентов;
- LMode — режим загрузки;
- RMode — режим записи в ППЗУ;
- OSMode — режим штатной работы;
- Reconf — параметры режима записи в ППЗУ;
- Mount — монтирование файловой системы;
- Master — определение ведущего ПЭ;
- LoaderFlag — флаги загрузки ПЭ;
- Load — параметры загрузки ПЭ;
- ReplFile — имя файла, который будет записан в ППЗУ;
- EthernetInterface — параметры сетевого интерфейса;
- DefaultIpGateway — шлюз по умолчанию в IP-сети;
- IpRoute — маршрут в IP-сети;
- SymbolicId — символическое имя ПЭ;
- TargetPath — путь к каталогу с образом ОС для ПЭ;
- AConfigPath — путь к файлу с ARINC-конфигурацией ПЭ;
- Route — маршрут между двумя ОУ.

Элементы **Include** и **TryInclude** являются дочерними по отношению к SystemDescription и предназначены для включения указанного файла в текущий файл. Различие между этими элементами состоит в том, что в случае отсутствия указанного файла соответственно будет зафиксирована ошибка или выдано предупреждение.

Элемент **Type** предназначен для описания типа аппаратных компонентов системы. Родительским элементом для него является корневой элемент файла ВОС SystemDescription. Атрибут Class определяет имя класса, которому принадлежит описываемый тип. Возможные значения были перечислены выше в подразд. 3.1. Атрибут Name задает имя описываемого типа. Имена типов должны быть непустыми и уникальными среди всех типов, описанных в данном ВОС.

У элемента Type определен следующий список дочерних элементов:

- Component — описывает вложенный аппаратный компонент, этот элемент может быть повторен произвольное число раз;
- Description — задает текстовый комментарий к типу; этот элемент, как и все последующие из этого списка, может быть задан не более одного раза;
- NetworkDevices — описывает сетевые интерфейсы;
- ExternalPorts — определяет внешние порты типа;
- InternalLinks — описывает связи между дочерними компонентами;
- RioIdentity — идентификатор модели устройства;
- RioRouting — параметры маршрутизации RapidIO.

Допустимость использования дочерних элементов в описании типа зависит от класса, указанного в атрибуте Class.

Элемент **Component** предназначен для описания вложенного (дочернего) аппаратного компонента. У элемента Component есть обязательные атрибуты Type (задает имя типа компонента) и Name (имя компонента). Атрибут PhysAddr употребляется только для типов, поддерживающих нумерацию на аппаратном уровне, т. е. принадлежащих классам ProcessingElement, Mezzonine, Module, Machine. Значение атрибута ReserveFor задает имя компонента, для которого описываемый компонент является резервным. Этот атрибут применим для типов, принадлежащих классам Machine или MachineGroup. Типы основного и резервного компонентов должны совпадать, а физические адреса — различаться. Необязательный атрибут Comment позволяет задать пользовательский комментарий описываемого аппаратного узла.

С помощью элемента Component строится вся иерархия аппаратных компонентов системы, корнем которой является компонент типа System.

Элемент **Description** задает текстовый комментарий к описываемому типу аппаратного компонента. Этот комментарий никак не интерпретируется и не влияет на формирование ЦОС.

Для описания сетевых интерфейсов ПЭ используется элемент **NetworkDevices**, который имеет дочерний элемент Ethernet.

Элемент Ethernet описывает сетевой интерфейс, подключенный к ПЭ. У этого элемента присутствует атрибут Name, который задает имя интерфейса. В ВОС не делается различий между интерфейсами, встроенными в процессорный элемент, внешними и эмулируемыми.

Приведем пример описания сетевых интерфейсов.

```
<Type Class = "ProcessingElement" Name = "K64-RIO">
  . . .
  <NetworkDevices>
    <Ethernet Name = "ed9"/>
    <Ethernet Name = "rio0"/>
  </NetworkDevices>
  . . .
</Type>
```

ExternelPorts является корневым элементом для описания внешних портов типа. У этого элемента присутствует дочерний элемент Port, описывающий внешние порты типа.

Пример:

```
<ExternalPorts>
  <Port .../>
  <Port .../>
  . . .
</ExternalPorts>
```

Элемент Port предназначен для описания внешних портов типа. Внешний порт — это оконечная точка для указания связи с внешними портами других компонентов. Атрибут Name задает имя внешнего порта. Атрибут ConnectedTo задает имя внешнего порта одного из дочерних компонентов, используемого в качестве внешнего порта текущего типа. Если описываемый порт является аппаратным портом, тогда атрибут PhysNum задает номер порта, а атрибут Type определяет тип порта. Тип порта может принимать значения "s" (последовательный), "p" (параллельный), "v" (порт ВСК). Атрибуты ConnectedTo и PhysNum не могут быть заданы одновременно.

Приведем пример описания внешних типов.

```
<Port Name = "p4" PhysNum = "4" Type = "p"/>
<Port Name = "s0" ConnectedTo = "VG18:s0"/>
```

Для описания внутренних связей типа предназначен элемент **InternalLinks**, у которого присутствует дочерний элемент Link, описывающий связи внутри типа. Пример:

```
< InternalLinks >
  < Link ... />
  < Link ... />
  . . .
</ InternalLinks >
```

Элемент Link определяет внутреннюю связь между внешними портами дочерних компонентов типа. Атрибуты Port1 и Port2 описывают оконечные порты связи. Каждый из оконечных портов связи задается в виде "component:port", где component — это имя дочернего компонента текущего типа, а port — это имя внешнего порта этого компонента. Тем самым задается связь между портами дочерних компонентов описываемого типа.

Связь считается двунаправленной, поэтому не имеет значения, какой порт указан в атрибуте Port1, а какой в атрибуте Port2:

```
<Link Port1 = "KP3:p7" Port2 = "K128-0:p0" />
```

Элемент **RioIdentity** задает идентификатор модели устройства RapidIO. Родительским элементом является элемент Type. Идентификатор модели одинаков для всех устройств RapidIO одной модели. Его значение хранится в регистре Device Identity конфигурационного пространства RapidIO. В текущей версии язык ВОС поддерживает следующие идентификаторы:

- 0x01280074 — Комдив128-РИО;
- 0x87170074 — Комдив64-РИО;
- 0x87100074 — 1890ВГ18Я;
- 0x00010074 — 1890КП3Я;
- 0x0500000d — Tundra TSI500;
- 0x87110074 — МПОН.

Элемент **RioRouting** определяет особенности работы коммутатора RapidIO. У этого элемента есть дочерние элементы TablesType и MaxId.

Элемент MaxId задает максимальное значение идентификатора RapidIO, маршрутизацию к которому способен выполнять коммутатор. Для элемента MaxId может быть задано число 255 или 65535. Значение этого элемента влияет на построение схемы маршрутизации. Коммутатор, поддерживающий только идентификаторы до 255, не может участвовать в маршруте к ОУ с идентификатором, превышающим значение 255.

Элемент TablesType может принимать одно из двух предопределенных значений: perport, если для каждого входного порта коммутатора имеется своя таблица маршрутизации, или single, если для всех входных портов используется одна таблица.

Пример:

```
<RioRouting>
  <TablesType>perport</ TablesType>
  <MaxId>255</ MaxId>
</RioRouting>
```

5. Библиотека поддержки коммуникационной среды RapidIO

Как уже было отмечено выше, в результате компиляции описания системы, составленного на языке ВОС, получается файл в формате ELF, содержащий ЦОС; этот файл с помощью редактора связей помещается в целевой образ системы. Для доступа к информации, содержащейся в ЦОС, предназначена библиотека поддержки коммуникационной среды RapidIO, которая поставляется совместно с ОСРВ Багет. Перечислим назначение основных функций этой библиотеки.

Функция начальной инициализации коммуникационной среды RapidIO на основании информации, содержащейся в ЦОС, осуществляет настройку всех коммутаторов в пределах одной машины. Вся информация, необходимая для прокладки маршрутов, содержится в ЦОС. Один ПЭ каждой машины является ведущим (*master*), все остальные — ведомые (*slave*). Граничные коммутаторы настраиваются таким образом, что позволяют восстанавливать соединения после разрыва и, в случае необходимости, перестраивать маршруты с основной машины на резервную и наоборот. Каждому ПЭ присваивается RIO-идентификатор.

Функция инициализации ПЭ, на основании его описания в ЦОС и параметров конфигурации ПЭ, осуществляет настройку сетевых интерфейсов, монтирование файловых систем, чтение заданной ARINC-конфигурации.

Служебные функции библиотеки предназначены для диагностики состояния коммуникационной среды RapidIO. Функция трассировки маршрута позволяет узнать маршрут от текущего узла к узлу назначения. Функция выводит все промежуточные коммутаторы и порты, ведущие к узлу назначения. Функция определения доступных устройств формирует список ОУ, доступных из текущего узла. Список упорядочен по возрастанию значения RIO-идентификатора. Функция получения топологии системы выводит список всех узлов системы и их соединений между собой, список ограничивается узлами одного прибора, но для граничных коммутаторов текущего прибора указываются граничные коммутаторы соседних приборов, с которыми есть соединение. Функции вывода статистики передачи данных позволяют получить информацию о каналах, по которым передаются данные в коммуникационной среде RapidIO, интенсивность обмена и число ошибок.

Драйвер шины RapidIO должен учитывать возможность разрывов соединений с последующим их восстановлением, а именно — в зависимости от значения параметра, задающего максимальное время ожидания, пользовательских вызовов функций обмена данными, он должен уметь настраиваться на режим работы, позволяющий восстанавливать передачу данных после восстановления ранее разорванного соединения.

Заключение

Рассмотрены средства конфигурирования для многопроцессорных вычислительных систем, функционирующих в коммуникационной среде RapidIO под управлением ОСРВ Багет. Конфигурирование выполняется системным интегратором, который составляет на языке ВОС описание системы в целом, используя библиотеку описания оборудования, поставляемую совместно с компилятором ВОС. Применение описанных средств позволяет в рамках предложенной технологии существенно упростить сложный процесс решения задачи взаимодействия между отдельными процессорами системы, а также сводит к описательному уровню решения таких задач, как холодное резервирование части оборудования, загрузки многих процессоров однотипными или различными образами ОС, запись в ППЗУ ПЭ требуемого ПО.

Разработанные средства конфигурирования нашли практическое применение в системах обработки гидроакустических сигналов.

Список литературы

1. **Fuller S.** RapidIO®: The Embedded Systems Interconnect. John Wiley & Sons, Ltd, 2005. 360 p.
2. **Бобков С. Г., Задябин С. О.** Перспективные высокопроизводительные вычислительные системы промышленного применения на базе стандарта RapidIO // Электроника, микро- и наноэлектроника: сб. науч. тр. М.: МИФИ, 2009. С. 114—121.
3. **Бобков С. Г., Аряшев С. И., Барских М. Е., Зубковский П. С., Ивасюк Е. В.** Высокопроизводительные расширения архитектуры универсальных микропроцессоров для ускорения инженерных расчетов//Информационные технологии. 2014. № 6. С. 27—37.
4. **Райко Г. О., Павловский Ю. А., Мельканович В. С.** Технология программирования многопроцессорной обработки гидроакустических сигналов на вычислительных устройствах семейства "КОМДИВ" // Гидроакустика. 2014. № 20 (2). С. 85—92.
5. **RapidIO** Interconnect Specification. Part1: Input/Output Logical Specification. Revision 3.1 — RapidIO Trade Association, 2014.
6. **RapidIO** Interconnect Specification. Part 2: Message Passing Logical Specification. Revision 3.1 — RapidIO Trade Association, 2014.
7. **Годунов А. Н.** Операционная система реального времени Багет 3.0 // Программные продукты и системы. 2010. № 4. С. 15—19.
8. **Годунов А.Н., Солдатов В. А.** Операционные системы семейства Багет (сходства, отличия и перспективы) // Программирование. 2014. № 5. С. 68—76.

Configuring Multiprocessor Systems in RTOS Baget

A. N. Godunov, e-mail: nkag@niisi.ras.ru, **V. A. Soldatov,** e-mail: nkvalera@niisi.ras.ru,
Federal State Institution "Scientific Research Institute of System Analysis of the Russian Academy of Science", Moscow, 117218, Russian Federation

Corresponding author:

Soldatov Valeriy A., Senior Scientific Researcher, Federal State Institution "Scientific Research Institute of System Analysis of the Russian Academy of Science", Moscow, 117218, Russian Federation, e-mail: nkvalera@niisi.ras.ru

Received on March 21, 2015
Accepted on April 01, 2015

Parallel computations executed on multiprocessor systems is an important way to increase computer systems performance. Devices in multiprocessor computer system can be interconnected by a shared bus. RapidIO was developed to interconnect both chips on the board and several boards. RapidIO is a network composed of processing elements (such as processor, switch, etc) intercconnected by physical communication channels. High-level system description language (HLD) was developed to describe multiprocessor systems in RapidIO communication environment that operates under RTOS Baget. Data for multiprocessor systems specified in HLD allows to solve the following problems:

— network topology specification;
— communication environment initializing;
— synchronous load and start of all system CPU;
— configuring software running on system processors;
— saving software image to each system CPU's ROM;
— reserving hardware subsystems for increasing system reliability;
— performing required calculations using specified hardware subsystems;
— saving/recovering operability after some hardware subsystem restart;

The described tools were applied in hydroacoustic signal processing systems.

Keywords: *RTOS Baget, configure, RapidIO, multiprocessor systems, routing*

For citation:

Godunov A. N., Soldatov V. A. Configuring Multiprocessor Systems in RTOS Baget, *Programmnaya Ingeneria*, 2016, vol. 7, no. 6, pp. 243—251.

DOI: 10.17587/prin.7.243-251

References

1. **Fuller S.** *RapidIO®: The Embedded Systems Interconnect.* John Wiley & Sons, Ltd, 2005. 360 p.

2. **Bobkov S. G., Zadyabin S. O.** Perspektivnye vysokoproizvoditel'nye vychislitel'nye syytemy promyshlennogo primeneniay na baze standarta RapidIO (Promising high performance computing industrial applications based on the RapidIO standard), *Elektronika, Mikro- i Nanoelektronika: sb. nauchn. tr.*, 2009, pp. 114—121 (in Russian).

3. **Bobkov S. G., Aryashev S. I., Barskyh M. E., Zubkovskiy P. S., Ivasyuk E. V.** Vysokoproizvoditel'nye raschireniay architektury universal'nych mikroprocessorov dlay uskorenia ingenernych raschetov (High-Performance Extensions of Microprocessor Architecture for Speeding-Up of Scientific and Engineering Calculations), *Informacionnye Technologii*, 2014, no. 6, pp. 27—37 (in Russian).

4. **Raiko G. O., Pavlovsky Yu. A., Mel'kanovich V. S.** Technologiay programmirovaniay mnogoprocessornoy obrabotki gidroakusticheskich signalov na vychislitel'nych ustroistvach semeistva "KOMDIV" (Technology of programming of multiprocessor processing of hydroacoustic signals on "комdiv" computer set), *Gidroakustika,* 2014, no. 20 (2), pp. 85—92 (in Russian).

5. **RapidIO** Interconnect Specification. Part 1: Input/Output Logical Specification. Revision 3.1 — RapidIO Trade Association, 2014.

6. **RapidIO** Interconnect Specification. Part 2: Message Passing Logical Specification. Revision 3.1 — RapidIO Trade Association, 2014.

7. **Godunov A. N.** Operatsionnaya sistema real'nogo vremeni Baget 3.0 (Real-time operating system baget 3.0**)**, *Programmnye Producty i Systemy*, 2010, no. 4, pp. 15—19 (in Russian).

8. **Godunov A. N., Soldatov V. A.** Operatsionnaye sistemy semeistva baget (skhodstvo, otlichiya i perspektivy) (Operating Systems of the Baget Family (Likeness, Differences and Perspectives), *Programmirovanie*, 2014, no. 5, pp. 68—76 (in Russian).

УДК 512.563.6 DOI: 10.17587/prin.7.252-257

И. Ю. Иванов, аспирант, e-mail: hour1scorp@gmail.com,
Воронежский государственный университет

Расширенная модель LP-вывода на булевой решетке

При разработке и исследовании интеллектуальных систем продукционного типа оказываются эффективными алгебраические методы на основе математических решеток. К таковым, в частности, относится метод LP-вывода, эффективность которого продемонстрирована в последнее десятилетие рядом авторов. В настоящей статье рассмотрена расширенная модель LP-вывода, которая может быть применена в подобного рода системах с семантикой полного набора логических связок пропозиционального языка.

Ключевые слова: *продукционно-логические системы, математические решетки, пропозициональная логика, обратный вывод, LP-структуры, LP-вывод, продукционно-логические уравнения, общее решение*

Введение

В информатике широко распространены системы продукционного типа, основывающиеся на правилах, или продукциях, вида "A порождает B", где A и B — элементы некоторой иерархии. Примером такой системы можно считать компьютерную программу, состояние которой в каждый момент времени определяется набором значений переменных, а правила перехода из одного состояния в другое (продукции) задаются исполняемым программным кодом. Сюда же относятся экспертные продукционные системы, правила которых имеют импликативную семантику: "если имеет место A, то справедливо B". В этом случае A называют предпосылкой, а B — заключением правила. Возможны и другие интерпретации подобных правил [1, 2].

В рамках экспертных продукционных систем решают задачи двух типов, связанные с логическим выводом: прямой вывод и обратный вывод. В задаче прямого вывода по имеющемуся множеству начальных фактов, истинность которых известна, необходимо путем применения правил в прямом направлении (от предпосылки к заключению) получить множество новых фактов, истинность которых не была установлена ранее. Этот процесс может иметь различные критерии останова: установление истинности некоторого множества целевых фактов, невозможность установления истинности для большего числа новых фактов и т. д. В задаче обратного вывода фиксируется факт, истинность которого неизвестна (гипотеза). Далее путем анализа правил в обратном направлении (от заключения к предпосылке) подтверждается либо опровергается истинность выдвинутой гипотезы. Для каждой из указанных задач, как минимум, характерны две особенности. Во-первых, в общем случае это высокая вычислительная сложность, связанная с экспоненциальной сложностью алгоритмов пропозициональной логики, применяемых для обработки предпосылок и заключений правил. Во-вторых, это необходимость выполнения запросов к внешним источникам информации (например, базам знаний, экспертам и т. д.) для установления истинности фактов на определенных шагах вывода. При этом очевидно, что скорость обработки внешних запросов (например, операций чтения с жесткого диска) значительно ниже скорости работы вычислительного устройства (например, микропроцессора), применяемого для решения задачи. Поэтому возникает актуальная задача минимизации числа подобного рода "медленных" запросов.

Известно, что естественную и эффективную форму представления знаний дают алгебраические методы, базирующиеся на математических решетках [3]. На этом подходе основывается теория LP-структур, представленная в работе [4]. LP-структура представляет собой математическую решетку с заданным на ней дополнительным бинарным отношением, моделирующим множество продукций. Тип решетки, а также набор свойств, которыми должно обладать дополнительное отношение, выбирается в каждом конкретном случае в зависимости от предметной области. В работе [4] на атомно-порожденной решетке введен и исследован класс продукционно-логических уравнений, а также предложен метод обратного логического вывода, основанный на решении таких уравнений (LP-вывод). Он соответствует продукционной системе, правила которой формируются с использованием единственной логической связки — конъюнкции. В процессе LP-вывода [5] осуществляется оптимизация числа запросов к внешним источникам информации, что практически продемонстрировано в реализованной программной системе [6].

В статье [7] введен расширенный класс продукционно-логических уравнений в LP-структуре нулевого порядка — алгебраической системе, в основе кото-

рой лежит булева решетка. Такие уравнения могут применяться для оптимизации обратного вывода в продукционно-логических системах, использующих полный набор связок пропозициональной логики в предпосылках и заключениях правил. В работах [8, 9] исследованы вопросы разрешимости, а также предложен метод поиска частных решений уравнений указанного класса.

В настоящей работе рассмотрены вопросы, связанные с поиском общего решения продукционно-логических уравнений в LP-структуре нулевого порядка. Полученные результаты завершают формирование расширенной модели LP-вывода на булевой решетке.

1. LP-структуры нулевого порядка и их связь с продукционными системами

Рассмотрим основные понятия из теории решеток [10] и теории LP-структур [4], необходимые для дальнейшего изложения.

Решеткой называется множество $\mathbb{F}$, на котором задано отношение частичного порядка $\leqslant$, для любых двух элементов $x, y \in \mathbb{F}$ которого в $\mathbb{F}$ существуют точная нижняя и точная верхняя грани, называемые пересечением и объединением элементов x, y и обозначаемые $x \wedge y$, $x \vee y$ соответственно.

Решетка $\mathbb{F}$ называется дистрибутивной, если для любых элементов $x, y, z \in \mathbb{F}$ выполняются равенства $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ и $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.

Решетка $\mathbb{F}$ называется ограниченной, если существуют элементы $O, I \in \mathbb{F}$, называемые точной нижней и точной верхней гранью соответственно, такие, что для любого $x \in \mathbb{F}$ выполнено $O \leqslant x \leqslant I$.

Ограниченная решетка $\mathbb{F}$ называется решеткой с дополнениями, если для любого $x \in \mathbb{F}$ существует элемент $x' \in \mathbb{F}$, называемый дополнением, такой, что $x \wedge x' = O$, $x \vee x' = I$.

Булевой решеткой называется дистрибутивная решетка с дополнениями.

Пусть далее $\mathbb{F}$ — булева решетка и пусть R — бинарное отношение на $\mathbb{F}$.

Отношение R называется $\wedge$-дистрибутивным, если для любых пар $(x, y_1), (x, y_2) \in R$ справедливо $(x_1, y_1 \wedge y_2) \in R$, и $\vee$-дистрибутивным, если для любых пар $(x_1, y)(x_2, y) \in R$ справедливо $(x_1 \vee x_2, y) \in R$.

Отношение R называется дистрибутивным, если оно одновременно $\wedge$-дистрибутивно и $\vee$-дистрибутивно.

Отношение R называется продукционно-логическим, если оно содержит отношение частичного порядка, определенное на решетке $\mathbb{F}$, а также обладает свойствами рефлексивности, транзитивности и дистрибутивности.

Продукционно-логическим замыканием отношения R называется наименьшее продукционно-логическое отношение, содержащее R.

Два отношения R_1, R_2 на $\mathbb{F}$ называют логически эквивалентными, если они имеют общее продукционно-логическое замыкание.

Пара элементов $(x, y) \in \mathbb{F} \times \mathbb{F}$ (не обязательно принадлежащая R) называется логически связанной отношением R (этот факт обозначается $x \xrightarrow{R} y$), если выполнено одно из следующих условий:

$$(x, y) \in R; \quad x \leqslant y; \tag{1}$$

существуют $y_1, y_2 \in \mathbb{F}$ такие, что

$$x \xrightarrow{R} y_1, \quad x \xrightarrow{R} y_2 \text{ и } y_1 \wedge y_2 = y; \tag{2}$$

существуют $x_1, x_2 \in \mathbb{F}$ такие, что

$$x_1 \xrightarrow{R} y, \quad x_2 \xrightarrow{R} y \text{ и } x_1 \vee x_2 = x; \tag{3}$$

существует $z \in \mathbb{F}$ такой, что

$$x \xrightarrow{R} z \text{ и } z \xrightarrow{R} y. \tag{4}$$

При этом y называется образом x, а x — прообразом y при отношении $x \xrightarrow{R} y$.

В работе [4] доказано, что для произвольного отношения R на $\mathbb{F}$ его продукционно-логическое замыкание существует и совпадает со множеством всех пар, логически связанных этим отношением (т. е. с $\xrightarrow{R}$).

Отметим, что правила (2), (3) эквивалентны их обобщенным аналогам:

существуют $a_1, ..., a_l, b_1, ..., b_l \in \mathbb{F}$ такие, что

$$a_i \xrightarrow{R} b_i, \ i = 1, ..., l \text{ и } a_1 \wedge ... \wedge a_l = x, \ b_1 \wedge ... \wedge b_l = y;$$

существуют $c_1, ..., c_m, d_1, ..., d_m \in \mathbb{F}$ такие, что

$$c_j \xrightarrow{R} d_j, \ j = 1, ..., m \text{ и}$$
$$c_1 \vee ... \vee c_m = x, \ d_1 \vee ... \vee d_m = d. \tag{5}$$

С помощью LP-структуры нулевого порядка можно моделировать экспертную продукционную систему следующим образом. Пусть имеется множество элементарных высказываний $A = \{\alpha_1, ..., \alpha_p\}$. Образуем из высказываний множества A с помощью логических операций конъюнкция, дизъюнкция и отрицание всевозможные составные высказывания. Обозначим полученное множество $\mathbb{A}$. На множестве $\mathbb{A}$ рассмотрим отношение $\Leftrightarrow$ эквивалентности высказываний. Тогда фактор-множество $A/\Leftrightarrow$ есть булева решетка $\mathbb{F}$, называемая алгеброй Линденбаума-Тарского [3]. При этом множеству правил системы можно поставить в соответствие отношение R на $\mathbb{F}$, а его логическое замыкание (совпадающее с $\xrightarrow{R}$) содержит множество всех логических связей, порождаемых правилами.

В дальнейшем будем рассматривать только конечные булевы решетки. Поскольку конечная решетка является булевой тогда и только тогда, когда она изоморфна булеану некоторого конечного множества [11], то без ограничения общности можно считать, что $\mathbb{F}$ — булеан конечного множества F целых чисел от 0 до $n - 1$. В связи с этим будем использовать

обозначения, принятые в теории множеств: элементы $\mathbb{F}$ обозначать большими латинскими буквами (если не оговорено иное), операции пересечения, объединения и дополнения через $\cap, \cup,$ и $\bar{\cdot}$ соответственно, отношение частичного порядка через $\subseteq$.

Атомами решетки $\mathbb{F}$ называют одноэлементные подмножества F, коатомами — $(n-1)$-элементные подмножества F. Будем обозначать атомы и коатомы маленькими латинскими буквами. Заметим, что всякий элемент $X \in \mathbb{F}$, $X \neq F$ имеет единственное представление в виде пересечения коатомов $X = x_1 \cap ... \cap x_q$, называемое коразложением. Через $Coat(X) = \{x_1, ..., x_q\}$ обозначим множество коатомов, составляющих коразложение элемента X. При этом по определению положим $Coat(F) = \varnothing$. Множество всех коатомов решетки $\mathbb{F}$ обозначим $\mathcal{F}$.

2. Продукционно-логические уравнения в LP-структуре нулевого порядка

Определения и результаты, которые представлены далее, были введены и опубликованы в работах [7—9].

Определение 1. Продукционно-логическим уравнением в LP-структуре нулевого порядка называют уравнение вида

$$R^L(X) = B, \tag{6}$$

где B — фиксированный в $\mathbb{F}$ элемент; X — искомый в $\mathbb{F}$ элемент.

Определение 2. Коатом $x \in \mathbb{F}$ называют начальным при отношении R, если для любой пары $A \xrightarrow{R} B$ из того, что $x \in Coat(B)$, следует $x \in Coat(A)$.

Множество начальных при отношении R коатомов обозначим $\mathcal{F}_0(R)$.

Определение 3. Начальным множеством решетки $\mathbb{F}$ при отношении R называется нижняя полурешетка $\mathbb{F}_0(R)$, образованная замыканием множества $\mathcal{F}_0(R)$ относительно операции пересечения $\cap$.

С точки зрения продукционной системы множество начальных элементов — это множество тех высказываний, истинность которых не выводима на основании имеющегося множества правил.

Определение 4. Частным решением уравнения (6) называют максимальный прообраз B из начального множества $\mathbb{F}_0(R)$. Общим решением уравнения (6) называют множество всех его частных решений. Приближенным решением уравнения (6) называют любой прообраз B из $\mathbb{F}_0(R)$.

Введем следующие предположения относительно вида уравнения (6):

$$B \neq F; \tag{7}$$

$$R \text{ не содержит пар с } F \text{ в левой части.} \tag{8}$$

Предположим, что $B = B_1 \cap ... \cap B_k$, $B_i \in \mathbb{F}$, $i = 1, ..., k$. Справедлива теорема.

Теорема 1. Пусть χ_i частное решение уравнения $R^L(X) = B_i$, $i = 1, ..., k$. Тогда множество χ, образованное из максимальных элементов множества $M = \{X_{11} \cap ... \cap X_k | X_i \in \chi_i,\ i, ..., k\}$, есть общее решение уравнения (6).

С помощью теоремы 1 решение уравнения (6) в работе [8] было сведено к решению множества уравнений типа

$$R^L(X) = b, \tag{9}$$

где правая часть в виде коатома b "пробегает" множество $Coat(B)$.

Для разработки метода решения уравнения типа (9) вводились понятия канонического отношения на булевой решетке и расслоения канонического отношения.

Определение 5. Отношение R называется каноническим, если оно не пересекается с отношением $\leqslant$ и правые части всех его пар являются коатомами.

Было показано, что для произвольного отношения R существует логически эквивалентное ему каноническое отношение.

Обозначим R^x подмножество R, образованное парами с коатомом x в правой части, а $\mathcal{R}$ — множество коатомов из правых частей пар отношения R.

Определение 6. Слоем R_t в отношении R называется подмножество его пар, взятых по одной из каждого множества R^x, $x \in \mathcal{R}$.

Всякий слой $R_t \subseteq R$ представим в виде ориентированного графа $G_{R_t} = (V, U)$ следующим образом. Множество вершин V совпадает со множеством $\mathcal{F}$ коатомов $\mathbb{F}$. Дуга (x, y) принадлежит множеству связей U, если в R_t существует пара (X, y) такая, что $x \in Coat(X)$.

Пусть R_t — фиксированный слой в R. В работе [9] была доказана теорема, с помощью которой можно отыскивать частные решения уравнения

$$R_t^L(X) = b \tag{10}$$

на графе G_{R_t}.

Теорема 2. Предположим, что выполнены следующие условия:

1) вершина x имеет нулевую полустепень захода [12] в графе G_{R_t};

2) существует путь $(y_0, y_1, ..., y_f, y_{f+1})$ в G_{R_t} такой, что $y_0 = x$, $y_{f+1} = b$ и $y_i \neq y_j$ для любых $i \neq j$, $i, j \in \{0, ..., f+1\}$;

3) в G_{R_t} не существует дуг (b, y_i), $i = 1, ..., f$.

Тогда x — частное решение уравнения (10).

3. Общее решение продукционно-логического уравнения в LP-структуре нулевого порядка

Теорема 2 позволяет находить частные решения уравнения (9) с помощью обхода вершин графов слоев отношения R. Возникает вопрос: каким образом можно найти все частные решения уравнения (9) (т. е. его общее решение)? Ответ на этот вопрос, с учетом теоремы 1, даст возможность находить об-

щее решение исходного уравнения общего вида (6) (безусловно, с учетом предположений (7), (8)).

Замечание 1. Если $b \in \mathbb{F}_0(R)$, то уравнение (9) имеет тривиальное частное решение $\chi = \{b\}$. Действительно, ввиду правила (1) имеем $b \xrightarrow{R} b$, а предположение о наличии частного решения $x \neq b$ оказывается несостоятельным, поскольку в таком случае была бы выполнена связь $x \xrightarrow{R} b$, что противоречило бы $b \in \mathbb{F}_0(R)$.

Далее предполагаем, что $b \notin \mathbb{F}_0(R)$. Поскольку в правой части уравнения (9) записан коатом, поиск ответа на поставленный вопрос необходимо начинать с исследования логических связей с коатомом в правой части.

Итак, рассмотрим произвольную логическую связь $X \xrightarrow{R} b$, где b — коатом.

Лемма 1. Если $X \not\subseteq b$, то для любого коатома $x \in Coat(X)$ справедливо $x \xrightarrow{R} b$.

Доказательство. Пусть $x \in Coat(X)$. Тогда справедливо представление $X = X_1 \cap x$ при некотором $X_1 \in \mathbb{F}$, откуда

$$X_1 \cap x \xrightarrow{R} b. \tag{11}$$

Ввиду $b \cap x \subseteq x$ по правилу (1) получаем

$$b \cap x \xrightarrow{R} b. \tag{12}$$

Из связей (11), (12) по правилу (5) получаем

$$(X_1 \cap x) \cup (b \cap x) \xrightarrow{R} b,$$

или

$$(X_1 \cup b) \cap x \xrightarrow{R} b.$$

По условию теоремы $X \not\subseteq b$. Это означает, что $X_1 \cup b = F$. Тогда из последней связи получаем $x \xrightarrow{R} b$.

Следствие 1. Частным решением уравнения (9) может быть только коатом. Обратно, всякий коатом $x \in \mathbb{F}_0(R)$, для которого справедливо $x \xrightarrow{R} b$, является частным решением уравнения (9).

Доказательство. Докажем первую часть следствия. Пусть X — частное решение уравнения (9). Предположим противное: X не является коатомом. Рассмотрим $x \in Coat(X)$. Ввиду предположения, естественное неравенство $X \subseteq x$ оказывается строгим:

$$X \subset x. \tag{13}$$

Поскольку $X \xrightarrow{R} b$ и $b \notin \mathbb{F}_0(R)$, то по лемме 1 имеем $x \xrightarrow{R} b$, т. е. x — частное решение уравнения (9). С учетом неравенства (13) получаем противоречие с тем, что X — частное решение уравнения (9).

Докажем вторую часть следствия. Пусть $x \in \mathbb{F}_0(R)$ — коатом, для которого справедливо $x \xrightarrow{R} b$. Поскольку x есть максимальный элемент $\mathbb{F}_0(R)$ [9], то x — частное решение уравнения (9).

Следствие 2. Всякое частное решение уравнения (10) является также и частным решением уравнения (9).

Доказательство. Пусть x — частное решение уравнения (10). Тогда $x \in \mathbb{F}_0(R_t)$ и $x \xrightarrow{R_t} b$. Ввиду $\mathbb{F}_0(R_t) = \mathbb{F}_0(R)$ [8] и $\xrightarrow{R_t} \subseteq \xrightarrow{R}$ [4], получаем $x \in \mathbb{F}_0(R)$ и $x \xrightarrow{R} b$. Тогда по следствию 1 имеем: x — частное решение уравнения (9).

Следствие 2 позволяет осуществлять поиск частных решений уравнения (9), сужая отношение R до его слоев R_t и применяя поисковые алгоритмы на соответствующих графах G_{R_t}.

В свою очередь лемма 1 позволяет доказать теорему об общем решении уравнения (9).

Теорема 3. Для общего решения χ уравнения (9) справедливо следующее утверждение: если $b \in \mathbb{F}_0(R)$, то $\chi = \{b\}$, иначе, если существует по крайней мере одно частное решение χ уравнения (9), то $x = \mathcal{F}_0(R)$, иначе $\chi = \varnothing$.

Доказательство. Если $b \in \mathbb{F}_0(R)$, то по замечанию 1 сразу получаем $x = \{b\}$. Далее предполагаем, что $b \notin \mathbb{F}_0(R)$.

Пусть существует частное решение x уравнения (9). Тогда выполнена связь

$$x \xrightarrow{R} b. \tag{14}$$

Рассмотрим произвольный коатом $y \in \mathbb{F}_0(R)$. Ввиду неравенства $x \cap y \subseteq x$, по правилу (1) получаем

$$x \cap y \xrightarrow{R} x. \tag{15}$$

Применяя к связям (14), (15) правило (4), получим $x \cap y \xrightarrow{R} b$. Тогда по лемме 1 получаем $y \xrightarrow{R} b$, откуда делаем вывод, что y — частное решение уравнения (9). В силу произвольности выбора коатома $y \in \mathbb{F}_0(R)$, окончательно получаем $\chi \in \mathcal{F}_0(R)$.

Если же уравнение (9) не имеет частных решений, то, очевидно, $\chi = \varnothing$.

Наконец, теорема 3 позволяет доказать теорему об общем решении уравнения (6). Чтобы это сделать, введем некоторые обозначения.

Пусть $B = b_1 \cap \ldots \cap b_g$ — коразложение правой части B. Рассмотрим уравнения

$$R^L(X) = b_i,\ i = 1, \ldots, g. \tag{16}$$

Предположим, что для некоторого числа $0 \leqslant s \leqslant g$ выполнено

$$b_j \in \mathbb{F}_0(R),\ j = 1, \ldots, s; \tag{17}$$

уравнение $R^L(X) = b_j$ имеет частное решение

$$x_j \neq b_j,\ j = s + 1, \ldots, g. \tag{18}$$

Отметим, что при наличии частного решения у каждого уравнения из набора (16) с правой частью, не принадлежащей $\mathcal{F}_0(R)$, условия (17), (18) всегда

могут быть удовлетворены путем соответствующей нумерации уравнений в указанном наборе.

Теорема 4. Для общего решения χ уравнения (6) справедливо утверждение: если $s = 0$, то $\chi = \mathcal{F}_0(R)$, иначе $\chi = \{b_1 \cap ... \cap b_s\}$.

Доказательство. Пусть $s = 0$. Тогда по теореме 3 каждое уравнение из (16) имеет общее решение $\mathcal{F}_0(R)$. По теореме 1 общее решение уравнения (6) состоит из максимальных элементов множества

$$M_1 = \{X_1 \cap ... \cap X_g | X_j \in \mathcal{F}_0(R),\ j = 1\ ...,\ g\}.$$

Очевидно, что максимумы во множестве M_1 достигаются в том случае, когда $X_1 = ... = X_g$, т. е. множество максимумов M_1 совпадает с $\mathcal{F}_0(R)$, откуда $\chi = \mathcal{F}_0(R)$.

Предположим теперь, что $s > 0$. Вновь по теореме 1 получаем, что общее решение уравнения (6) состоит из максимальных элементов множества

$$M_2 = \{b_1 \cap ... \cap b_s \cap X_{s+1} \cap ... \cap X_g | X_j \in \mathcal{F}_0(R),\ j = s+1,\ ...,\ g\}.$$

Поскольку при любых $X_j \in \mathcal{F}_0(R)$, $j = s + 1$, ..., g справедливо неравенство

$$b_1 \cap ... \cap b_s \cap X_{s+1} \cap ... \cap X_g \subseteq b_1 \cap ... \cap b_s,$$

причем $b_1 \cap ... \cap b_s \in M_2$, то множество M_2 имеет наибольший элемент $b_1 \cap ... \cap b_s$. Отсюда сразу следует, что $\chi = \{b_1 \cap ... \cap b_s\}$.

С учетом теоремы 4, а также результатов, полученных ранее в работах [7—9], можно сформулировать следующий алгоритм поиска общего решения уравнения (6).

Шаг 1. Построить коразложение правой части $B = b_1 \cap ... \cap b_g$.

Шаг 2. Построить начальное множество $\mathcal{F}_0(R) = \mathcal{F} \backslash \mathcal{R}$.

Шаг 3. Перенумеровать уравнения из набора (16) таким образом, чтобы для некоторого $0 \leqslant s \leqslant g$ было выполнено b_1, ..., $b_s \in \mathcal{F}_0(R)$, причем s был бы максимален среди всех возможных подобных нумераций.

Шаг 4. Для каждого уравнения $R^L(X) = b_i$, $s + 1 \leqslant i \leqslant g$ найти по крайней мере одно частное решение в любом из слоев отношения R с помощью соответствующих графов.

Шаг 5. Если на шаге 4 для некоторого i_0 не удается найти ни одного частного решения, то общее решение уравнения (6) найти невозможно (что отнюдь не означает его отсутствие), и алгоритм завершает работу.

Шаг 6. Если $s = 0$, то $\mathcal{F}_0(R)$ — общее решение уравнения (6).

Шаг 7. Если $s > 0$, то $\{b_1 \cap ... \cap b_s\}$ — общее решение уравнения (6).

Заключение

Исследование логических связей с коатомом в правой части, порожденных отношением на булевой решетке, позволило доказать теорему об общем решении продукционно-логического уравнения в LP-структуре нулевого порядка. Полученные результаты дали возможность сформулировать алгоритм поиска общего решения.

Результаты, представленные в настоящей работе, завершают построение расширенной модели LP-вывода на булевой решетке. Такой подход может применяться для оптимизации продукционных систем с семантикой полного набора связок пропозициональной логики.

Работа поддержана грантом РФФИ № 15-07-05341.

Список литературы

1. **Чечкин А. В.** Математическая информатика. М.: Наука, гл. ред. физ.-мат. лит., 1991. 416 с.
2. **Махортов С. Д.** Основанный на решетках подход к исследованию и оптимизации множества правил условной системы переписывания термов // Интеллектуальные системы. Теория и приложения. 2009. Т. 13. С. 51—68.
3. **Расёва Е., Сикорски Р.** Математика метаматематики: пер. с англ. М.: Наука, гл. ред. физ.-мат. лит., 1972. 591 с.
4. **Махортов С. Д.** Математические основы искусственного интеллекта: теория LP-структур для построения и исследования моделей знаний продукционного типа / под ред. В. А. Васенина. М.: МЦНМО, 2009. 304 с.
5. **Болотова С. Ю., Махортов С. Д.** Алгоритмы релевантного обратного вывода, основанные на решении продукционно-логических уравнений // Искусственный интеллект и принятие решений. 2011. № 2. С. 40—50.
6. **Махортов С. Д.** Интегрированная среда логического программирования LPExpert // Информационные технологии. 2009. № 12. С. 65—66.
7. **Иванов И. Ю.** Продукционно-логические уравнения на булевых решетках // Вестник Воронежского государственного университета. Сер. Физика. Математика. 2013. № 1. С. 170—177.
8. **Иванов И. Ю.** Приближенное решение продукционно-логического уравнения на булевой решетке // Нейрокомпьютеры: разработка, применение. 2014. № 10. С. 53—63.
9. **Ivanov I. Yu.** About particular solutions of production-logical equations on LP-structure of zero order // Modern Informatization Problems in the Technological and Telecommunication Systems Analysis and Synthesis: Proceedings of the XXI-th International Open Science Conference. Yelm, WA, USA, January, 2016 / Editor in Chief Dr. Sci., Prof. O. Ja. Kravets. Yelm: Science Book Publishing House LLC, 2016. P. 321—327.
10. **Гретцер Г.** Общая теория решеток: пер. с англ. / Под ред. Д. М. Смирнова. М.: Мир, 1981. 456 с.
11. **Биркгоф Г.** Теория решеток: пер. с англ. М.: Наука, гл. ред. физ.-мат. лит., 1984. 568 с.
12. **Кристофидес Н.** Теория графов. Алгоритмический подход. М.: Мир, 1978. 432 с.

Extended Model of LP-inference on Boolean Lattice

I. Yu. Ivanov, hour1scorp@gmail.com, Voronezh State University, 394006, Voronezh, Russian Federation

Corresponding author:

Ivanov Ilya Yu., Postgraduate Student, Voronezh State University, 394006, Voronezh, Russian Federation, e-mail: hour1scorp@gmail.com

Received on March 18, 2016
Accepted on March 30, 2016

Production systems are widespread in informatics. They are based on productions, or rules, of the from "A produces B", where A and B are the elements of some hierarchy. Software, the state of which is defined by a set of variables' values and rules of transition from one state to another (productions) determined by executing code, is an example of aforementioned system. Another example is expert production systems, the rules of which have implicative semantics: "if A holds, then B is true". In this case A is called premise and B is called conclusion of a rule.

There are two types of problems to be solved within expert production systems: forward inference and backward inference. Each of these problems has at least two special features. Firstly, the high computational complexity which is caused by exponential complexity of propositional logic algorithms, which are applied to process premises and conclusions of rules. Secondly, the need to obtain truth of facts on particular stages of inference for external sources of data querying (e.g., knowledge bases, experts etc.). Obviously, speed of external queries processing is extremely lower than speed of computational device (e.g., CPU). Therefore, a problem of "slow" queries quantity minimizing arises.

Previously S. D. Makhortov introduced the concept of production-logical equations on atom-derived lattice. These equations can be applied to perform LP-inference — algebraic method of backward inference that minimizes a number of "slow" queries in a system that uses only one logical connection, conjunction, in its rules. Later, a new class of production logical equations on boolean lattice has been introduced by the author of the paper. With these equations, it is possible to implement LP-inference that performs similar optimization in systems that use a full set of zero-order propositional language's operations in its rules: conjunctions, disjunctions and negations. In addition, the solvability of these equations was investigated and the method of particular solutions finding was proposed.

In this paper, the questions related to general solution of production-logical equation on LP-structure of zero order are examined. Obtained results complete building of extended model of LP-inference on boolean lattice.

Keywords: *productions, production-logical systems, mathematical lattices, propositional logic, backward inference, LP-structures, LP-inference, production-logical equations, particular solutions, general solution*

Acknowledgements: *This work was supported by the Russian Foundation for Basic Research, project nos. 15-07-05341.*

For citation:

Ivanov I. Yu. Extended Model of LP-inference on Boolean Lattice, *Programmnaya Ingeneria,* 2016, vol. 7, no. 6, pp. 252—257

DOI: 10.17587/prin.7.252-257.

References

1. **Chechkin A. V.** *Matematicheskaja informatika* (Mathematical Informatics), Moscow, Nauka, gl. red. fiz.-mat. lit., 1991, 416 p. (in Russian).

2. **Mahortov S. D.** Osnovannyj na reshetkah podhod k issledovaniju i optimizacii mnozhestva pravil uslovnoj sistemy perepisyvanija termov (Lattice-based Approach to Study and Optimize a Set of Conditional Terms Rewriting System's Rules), *Intellektual'nye Sistemy. Teorija i Prilozhenija*, 2009, vol. 13, pp. 51—68. (in Russian).

3. **Rasjova E., Sikorski R.** *Matematika metamatematiki:* per. s angl. (The Mathematics of Metamathematics), Moscow, Nauka, gl. red. fiz.-mat. lit., 1972, 591 pp. (in Russian).

4. **Mahortov S. D.** *Matematicheskie osnovy iskusstvennogo intellekta: teorija LP-struktur dlja postroenija i issledovanija modelej znanij produkcionnogo tipa* (Mathematical Foundations of Artificial Intelligence: Theory of LP-structures for the Construction and Studying of Knowledge Models of Production Type) / Ed by. V. A. Vasenin, Moscow, MCNMO, 2009. 304 pp. (in Russian).

5. **Bolotova S. Ju., Mahortov S. D.** Algoritmy relevantnogo obratnogo vyvoda, osnovannye na reshenii produkcionno-logicheskih uravnenij (The Algorithms of Relevant Backward Inference Based on Production-logical Equation Solution), *Iskusstvennyj Intellekt i Prinjatie Reshenij,* 2011, no. 2, pp. 40—50 (in Russian).

6. **Mahortov S. D.** Integrirovannaja sreda logicheskogo programmirovanija LPExpert (Integrated Development Environment for Logical Programming LPExpert), *Informacionnye Tehnologii*, 2009, no. 12, pp. 65—66 (in Russian).

7. **Ivanov I. Yu.** Produkcionno-logicheskie uravnenija na bulevyh reshjotkah (Production-logical Equations on Boolean Lattices), *Vestnik Voronezhskogo gosudarstvennogo universiteta. Ser. Fizika. Matematika*, 2013, no. 1, pp. 170—177 (in Russian).

8. **Ivanov I. Yu.** Priblizhennoe reshenie produkcionno-logicheskogo uravnenija na bulevoj reshetke (Particular Solution of Production-logical Equation on Boolean Lattice), *Nejrokomp'jutery: Razrabotka, Primenenie,* 2014, no. 10, pp. 53—63 (in Russian).

9. **Ivanov I. Yu.** About particular solutions of production-logical equations on LP-structure of zero order, *Modern Informatization Problems in the Technological and Telecommunication Systems Analysis and Synthesis: Proceedings of the XXI-th International Open Science Conference*, Yelm, WA, USA, January, 2016/Editor in Chief Dr. Sci., Prof. O. Ja. Kravets. Yelm, Science Book Publishing House LLC, 2016, pp. 321—327.

10. **Gratzer G.** *Obshhaja teorija reshjotok*: per. s angl. (General Lattice Theory) / Ed. by. D. M. Smirnov, Moscow, Mir, 1981, 456 p. (in Russian).

11. **Birkgof G.** *Teorija reshjotok:* per. s angl. (Lattice Theory), Moscow, Nauka, gl. red. fiz.-mat. lit., 1984, 568 p. (in Russian).

12. **Christofides N.** *Teorija grafov. Algoritmicheskij podhod* (Graph Theory. An Algorithmic Approach), Moscow, Mir, 1978. 432 p. (in Russian).

УДК 004.82 DOI: 10.17587/prin.7.258-267

К. И. Костенко, канд. физ.-мат. наук, зав. каф., e-mail kostenko@kubsu.ru,
Кубанский государственный университет, г. Краснодар

Правила оператора вывода для формализма абстрактного пространства знаний*

Определены порождающие принципы и инварианты правил управления оператором вывода для моделей областей знаний, создаваемых с использованием формализма абстрактного пространства знаний. Оператор вывода реализует процессы нахождения целей и синтеза фрагментов сложных знаний, структура которых согласована со структурой решаемых задач. Для этого применяется комбинирование схем прямого и обратного вывода.

Ключевые слова: *область знаний, пространство знаний, логико-математическая модель, структура задачи, элементарное знание, синтез знаний, обратный вывод*

Введение

Концепция формализма (формального представления) знаний, которая рассматривается в настоящей статье, предназначена для исследования и разработки унифицированных универсальных инвариантов программных систем представления и обработки знаний. Согласно этой концепции, формализм знаний — это математическая система, применяемая для моделирования содержания и процессов решения профессиональных задач из разных областей знаний [1]. Такой формализм определяется с помощью перечислимого множества объектов, интерпретируемого как многообразие фрагментов представления отдельных знаний вместе с вычислимой операцией композиции и разрешимым отношением вложения фрагментов.

Представления знаний образуют разрешимое подмножество семейства объектов формализма. Композиция и вложение определяют алгебраическую и семантическую структуры знаний. Использование понятия вложения формализмов позволяет сравнивать структурную и семантическую сложности разных подходов к моделированию знаний [2]. Функциональный компонент формализмов знаний определяют содержательно полные системы операций, задаваемые для разных этапов существования интеллектуальных систем. Они представляются унифицированными абстрактными классами, включающими классы морфизмов декомпозиции, структуризации, классификации и трансформации. Принципиальным элементом многообразия содержательных операций над знаниями является механизм вывода. Он определяет возможности представления и решения профессиональных задач средствами применяемого формализма.

Операции интеллектуальных процессов

Интеллектуальная составляющая информационных систем, программно реализующих модель представления знаний, обеспечивается применением операций, моделирующих процессы мышления. Содержательно полная система типов операций над формализованными знаниями реализует представления о функциональной структуре указанных процессов с помощью унифицированных универсальных порождающих принципов и инвариантов процессов. К ним относятся анализ (представленный методами декомпозиции сложных знаний на части) и синтез (структурное конструирование сложного знания из отдельных его фрагментов).

Детализация отмеченных инвариантов операций дополняет их: сравнением (установлением различия и сходства между отдельными знаниями); абстрагированием (распознаванием существенного и удалением несущественного); обобщением (расширением соответствующих знаниям классов сущностей); унификацией (согласованием знаний разной общности); классификацией и систематизацией (распознаванием специальных свойств знаний). Использование формализованных уточнений представленного разнообразия операций путем применения интегрированного механизма вывода создает возможности для реализации взаимодействующих процессов распознавания, трансформации, композиции и извлечения фрагментов знаний, сценариев обработки знаний. Следует отметить, что такой подход позволяет достаточно полно моделировать систематизированные представления об операциях мышления.

Абстрактные пространства знаний

Абстрактные пространства знаний относятся к классу иерархических формализмов знаний. Каждое такое пространство определяется как $\Im = (M, D, \circ, \prec)$, где M — множество конфигураций (представлений знаний); D — перечислимое множество фрагментов конфигураций, $\circ$ — композиция, а $\prec$ отношение вложения фрагментов. Множество M составляют два бесконечных, дизъюнктных, разрешимых множества

* Работа выполнена при поддержке РФФИ грант № 16-01-00214.

элементарных M_0 и неэлементарных M_1 конфигураций. На множестве элементарных конфигураций определено отношение порядка ρ_0 с наименьшим элементом Λ. Такие конфигурации представляются одновершинными структурами.

Унифицированную схему представления неэлементарных конфигураций определяют тройки, близкие к структурам языка *RDF* [3]. Каждая такая конфигурация z состоит из конфигураций z_1 и z_2, связанных конкретным отношением r, и представляется тройкой (z_1, r, z_2). Многообразие отношений между конфигурациями является перечислимым и обозначается R. Элементами этого семейства являются разрешимые отношения, для которых дополнительно разрешимо отношение теоретико-множественного вложения, обозначаемое ρ_1. Отличные от конфигураций фрагменты представляются тройками $(\_, z, r)$ и $(z, \_, r)$, где $z \in M$ и $r \in R$, а также (r_1, r_2), где $r_1, r_2 \in R$. Множество R содержит специальные классы $\perp = \varnothing$ и $T = D \times D$.

Структура конфигураций абстрактного пространства знаний может быть определена с помощью вычислимых отображений разложения и связывания конфигураций ε: $M \to M \times M$ $\psi : M_1 \to R$ [2]. Здесь ε определяет разложение всякой конфигурации z на две компоненты, а ψ — отношение между элементами $\varepsilon(z)$. Структурное представление всякой конфигурации получается последовательным разложением конфигурации вплоть до элементарных конфигураций, разложениями которых является пара (Λ, Λ). Полное структурное представление (ПСП) конфигурации имеет вид нагруженного бинарного дерева. Висячие вершины этого дерева размечены элементарными конфигурациями, а остальные — отношениями из R, которые выполняются между конфигурациями, представленными правым и левым поддеревьями таких вершин. Вложение конфигураций связано с существованием трассирований их ПСП, которые задаются изотонными отображениями множеств вершин. Разметки связываемых такими трассированиями внутренних и висячих вершин ПСП конфигураций дополнительно должны быть сравнимыми в ρ_1 и ρ_0 [2].

Для исследования алгебраической структуры абстрактных пространств знаний удобной является схема операции композиции, позволяющая конструировать представления конфигураций абстрактных пространств знаний из простейших фрагментов:

$$z_1 \circ z_2 = \begin{cases} (z_1, z_2, \perp), \text{ если } z_1, z_2 \in M \\ (z_1, \_, r_2), \text{ если } z_1 \in M \;\&\; z_2 = (r_1, r_2) \;\&\; [z_1]_\lambda = r_1 \\ (\_, z_1, r_1), \text{ если } z_1 \in M \;\&\; z_2 = (r_1, r_2) \;\&\; [z_1]_\lambda = r_2 \;\&\; [z_1]_\lambda \neq r_1 \\ z, \text{ если } z_1 = (z_3, \_, r) \;\&\; \varepsilon(z) = (z_3, z_2) \;\&\; (z_3, z_2) \in r \\ z, \text{ если } z_2 = (\_, z_3, r) \;\&\; \varepsilon(z) = (z_1, z_3) \;\&\; (z_1, z_3) \in r \\ z, \text{ если } z_1 = (z_3, \_, r) \;\&\; z_2 = (\_, z_4, r) \;\&\; \varepsilon(z) = (z_3, z_4) \;\&\; (z_3, z_4) \in r \\ \Lambda, \text{ иначе.} \end{cases}$$

В приведенной формуле выражение $[z_1]_\lambda$ означает отношение, приписанное корню структурного представления z_1.

Практическое применение конфигураций абстрактных пространств знаний предполагает уточнение форматов представления задач и алгоритмов их решения. Рассматриваемый далее общий механизм вывода для абстрактных пространств выполняет анализ и синтез знаний, представленных в модели соответствующей области знаний, с помощью фрагментов конфигураций, которые, в свою очередь, соответствуют структурам подлежащих решению задач. Адаптация универсальных структурных и семантических инвариантов ПСП конфигураций к особенностям конкретных областей знаний реализуется построением множеств M и R. Для этого определяется множество элементарных конфигураций, представляемых неделимыми объектами, а также семейство отношений на этом множестве, определяющих многообразие знаний простой структуры. Обязательным атрибутом элементарных конфигураций является назначение или роль представляемого знания. Остальные конфигурации модели области знаний в формализме абстрактного пространства знаний конструируются из простых знаний и отношений с помощью специальной системы правил. Схемы правил определяют варианты использования отношений при конструировании сложных знаний, что позволяет расширить определения отношений на множество D.

Логико-математические модели области знаний

Базовым элементом формализованного описания всякой области знаний является специальная логико-математическая модель. Ее гомоморфное расширение порождает единое, связное, семантическое представление этой области, допускающее использование для решения различных профессиональных задач [4]. Указанную модель образуют системы классов данных, морфизмов, предикатов и процессов. Описание класса включает области имен, форматов, свойств и алгоритмов, относящихся к этому классу. Здесь имена классов определяют способы указания на классы в представлениях отдельных знаний, форматы — схемы обозначения элементов классов, свойства — соотношения, справедливые для области знаний, алгоритмы — различные процедуры обработки элементов описаний классов. Алгоритмы обеспечивают возможность разных вычислений, основанных на представленных в знаниях закономерностях. Алгоритмы являются объектами, которые расширяют возможности эффективного решения задач управления и профессионального использования модели области знаний. С элементами области свойств классов могут связываться алгоритмы проверки

истинности, нахождения значений всех или части параметров свойств, при которых оказывается возможным использование или трансформация свойств. Нахождение таких значений может быть реализовано алгоритмами разных типов. В частности, допустим поиск необходимых значений во внешних информационных ресурсах с помощью запросов к базам эмпирических данных или поисковым системам.

Зависимости между классами представляются отношениями вложения, агрегирования, использования, эквивалентности и гомоморфного расширения. Модель области знаний составляют отношения между классами, именами, форматами, свойствами и алгоритмами, которые задаются явно или являются алгоритмически распознаваемыми по описаниям классов. При этом достигается возможность точных определений, основанных на конструкциях языка логических и математических выражений, составляющих язык описаний моделей областей знаний. Последнее также верно и для слабо формализуемых областей знаний, в которых преобладают эмпирические закономерности и факты, которые моделируются с использованием ограниченного языка описания моделей. Основой таких моделей являются теоретико-множественные конструкции и методы. Трансформация логико-математической модели конкретной области знаний в форматы абстрактного пространства знаний реализуется с помощью построения базы простых знаний. Такие базы могут быть конечными, что делает аналоги абстрактных операций незамкнутыми на множествах конфигураций. Форматом простых знаний являются тройки *arb*, где *a* и *b* — элементарные конфигурации, а *r* — отношение между ними. Обязательным атрибутом конфигураций моделей конкретных областей знаний является роль знания, представленного этой конфигурацией. Она классифицирует знание иерархической структуры относительно содержания области знаний.

Концепция элементарного знания

Свойство элементарности знания отражает структурную характеристику его представления, которое объявляется неделимым. Содержание конкретной элементарной конфигурации может быть сложным и, как следствие, оно представляется с помощью конфигураций, связанных с заданной конфигурацией определенными отношениями. Конструкция элементарного знания является способом представления отдельной сущности в модели области знаний. Такой способ позволяет использовать подобное задание в описаниях свойств других знаний, постановках задач, а также в схемах работы со знаниями. Содержание элементарного знания может отражаться в содержании другого такого знания. Последнее отношение между знаниями обозначается как ρ_0. Основой процесса составления конфигураций механизмом вывода для абстрактных пространств знаний является база простых знаний, извлекаемых из логико-математической модели области знаний и представляемых тройками *arb*.

Унифицированная система отношений для конфигураций

Уточним адаптированный к обозначенным ранее операциям мышления фрагмент системы общих отношений, применяемых для составления сложных конфигураций, которые конструируются с помощью механизма вывода. Рассматриваемое далее семейство отношений обладает содержательной достаточностью для решения разнообразных задач. Это семейство допускает последующее дополнение и уточнение своих элементов. Представим указанное семейство как разбиение на классы основных и вспомогательных отношений. Отношения первой группы моделируют существенные предметные зависимости между отдельными знаниями. Вспомогательные отношения применяются для группирования фрагментов знаний с близкими свойствами в структуру бинарного дерева. Рассматриваемое далее семейство основных (предметных) отношений общезначимое и допускает использование для произвольных областей знаний. Такие отношения применяются в структурных представлениях отдельных знаний, постановок задач, а также правил, используемых для управления механизмом вывода, моделирующего процесс конструирования конфигураций по запросам. Конструкции используемых и генерируемых правилами конфигураций оперируют порождающими знаниями и отношениями из рассматриваемых семейств. Применяемая далее система ролей для элементарных знаний включает такие слабо формализуемые категории, как: *понятие*, *свойство*, *функция*, *значение*, *условие*, *логическая операция*. Рассмотрим пример семейства основных отношений.

- *Равно*, обозначается как = и используется для связывания понятий с принимаемыми ими значениями, оно выполняется для любой пары (*a*, *b*), где *a* — элемент множества понятий, а *b* — множество значений *a*. Примерами пар, принадлежащих данному отношению, являются: *скорость ветра* = 10; *ширина окна* = 1,2; *толщина стены* = 0,5; *глубина промерзания почвы* = 0,8.

- *Равно при условии* $\binom{C}{=}$ является расширением отношения *равно*, составленным парами (*a*, *b*), где *b* — фрагмент конфигурации, содержащий элемент множества возможных значений понятия *a* вместе с некоторым условием, которое должно выполняться для того, чтобы указанное значение имело место. Данному отношению принадлежат пары: *показатель риска* = 1,2, если *скорость ветра* $\geqslant$ 20; *начальная ставка* = 100, если *среднее значение ставок* = 500; или *столовый прибор* = *палочки*, если *клиент — китаец*; *размер штрафа* = 5, *если оплата проведена в течение 14 дней*.

- *Является* ($\subseteq$), это отношение, связывающее пары понятий, для которых имеет место вложение

множеств соответствующих им объектов и отражающее сравнение, для которого выражение $a \subseteq b$ означает, что b это более общее понятие, чем a. Для составления сложных конфигураций, конструируемых в целях решения конкретных задач, потребуется следующий фрагмент отношения $\subseteq$, который удобно представить частью диаграммы рассматриваемого отношения, приведенной на рис. 1.

Здесь вложения понятий *буря* и *засуха* в качественно разные классы *страховой случай*, *сопутствующий фактор* и *метеорологическая ЧС* позволяют интегрировать отличающиеся профессиональные представления об одном и том же понятии. Сокращение *ЧС* применяется для понятия *чрезвычайная ситуация*.

- *Имеет частью* ($\triangleleft$), выполняется для таких понятий, что всякий элемент множества, соответствующего первому понятию, содержит элемент множества объектов, представленного вторым понятием. Это универсальное отношение агрегирования, с помощью которого моделируются представления о структуре объектов в произвольных областях знаний. Указанному отношению принадлежат следующие пары: *определенный интеграл* $\triangleleft$ *дифференциал*; *определенный интеграл* $\triangleleft$ *границы*; *определенный интеграл* $\triangleleft$ $\triangleleft$ *подынтегральное выражение*; *границы* $\triangleleft$ *верхняя граница*; *границы* $\triangleleft$ *нижняя граница*; *ценная бумага* $\triangleleft$ *эмитент*. Рассматриваемое общее отношение агрегирования при необходимости может представляться более точными отношениями, обеспечивающими дополнительные возможности работы со знаниями. Например, структурные элементы рассмотренных выше природных явлений могут дополнительно классифицироваться как *пространственные* и *временные*, связанные с соответственно с пространственной структурой явления и этапами его существования.
- *Зависит от* ($\prec$), связывает фрагмент, роль которого это *свойство* или *функция* с фрагментом, роль которого — *понятие*. Примерами пар, для которых выполняется данное отношение, являются: *цена предложения* $\prec$ *объем предложения*; *ожидаемая загруженность предприятия* $\prec$ *время года*; *скорость распространения пожара* $\prec$ *скорость ветра*. Данным отношением отражаются факторы, влияющие на реализацию или проявление некоторого понятия.
- *Обладает свойством* ($\triangleq$), представляет зависимость между понятием и фрагментом с ролью *свойство*, относящимся к этому понятию. Это отношение позволяет классифицировать понятия, задавая их свойства. Примерами элементов данного отношения являются: *навес* $\triangleq$ *защищает от дождя*; *пожар нефти* $\triangleq$ *не тушится водой*; *золото* $\triangleq$ *устойчиво к окислению*; *ценная бумага* $\triangleq$ *действительна только при условии официальной регистрации*.

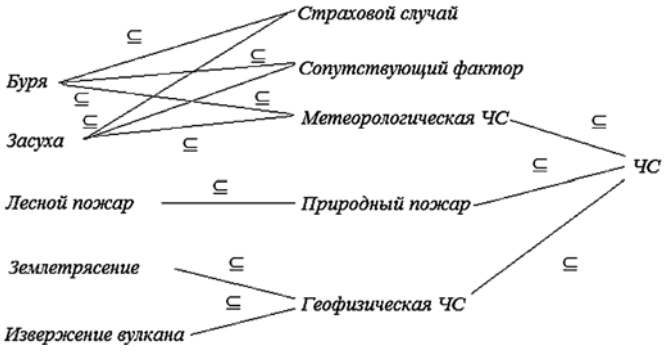


Рис. 1. Фрагмент отношения *являться* для понятий *ЧС*

- *Равносильно* ($\equiv$), выполняется между фрагментами конфигураций, имеющими эквивалентное содержание. В частности, этим отношением моделируется синонимия или точные определения понятий, представленные тройками: *Вальтер Скот* $\equiv$ $\equiv$ *автор романа Айвенго*; *килограмм* $\equiv$ *масса одного кубического дециметра воды*. Отношение равносильности используется для связывания эквивалентных утверждений, задаваемых сложными конструкциями. Например, *неориентированный связный граф без петель имеет цикл Эйлера* $\equiv$ *неориентированный связный граф без петель является четным графом*.
- *Вычисляется* ($:=$), это отношение связывает понятие с функцией или методом, определяющим значение этого понятия по значениям других понятий. Элементы данного отношения представляются следующими выражениями: *напряжение* $:=$ *закон Ома*; *площадь треугольника* $:=$ *формула Герона*; *корень уравнения* $:=$ *метод последовательных приближений*; *стоимость заказа* $:=$ *суммирование отобранных позиций*. Важность указанного отношения связана с возможностью применения указания на функцию или метод вычисления значения, который может быть выполнен, если известны значения понятий, от которых зависит функция или метод.
- *Следует* ($\Rightarrow$), это отношение, связывающее свойства, для которых истинность одного свойства влечет истинность другого свойства, обозначающее причинно-следственную связь между двумя фрагментами представления знаний. Такое отношение обычно связывает логические выражения, представленные комбинациями предметных предикатов. Справедливы следующие случаи следования: *написал*$(x, y) \Rightarrow$ *автор*(x, y); *продал*$(z, y, z) \Rightarrow$ $\Rightarrow$ *купил*(x, z); *были_одновременно*$(z, y, z) \Rightarrow$ $\Rightarrow$ *могли_встретиться*(x, y). В последней паре символы переменных x, y, z соответствуют понятиям *человек*, *человек* и *происшествие*. Рассматриваемое отношение может выполняться для таких отдельных фактов, представляющих причинно-следственные связи, как *прошел дождь* $\Rightarrow$ *загрязнение воздуха уменьшилось*; *прошел дождь и ударил мороз* $\Rightarrow$ *началось оледенение*.
- *Принадлежит* ($\in$), означает, что выбранный объект является элементом класса объектов, представленного заданным понятием. Это отношение используется для представления фактов и общих утверждений. При условии неотличимости элементов и составленных из них одноэлементных множеств данное отношение оказывается частью отношения *является*.

Элементами семейства вспомогательных отношений являются перечисленные далее отношения:

- *Логическая связь* ($\div$), это отношение, позволяющее конструировать логические комбинации свойств, представляемых фрагментами конфигураций с использованием элемента понятия *логическая операция*, со-

ставляющего специальный класс элементарных знаний, содержащий множество $\{\&, \vee, \neg, \sqsupset, \Diamond, \exists, \forall\}$. Потребность в рассматриваемом отношении связана с необходимостью моделирования произвольных иерархических структур нагруженными бинарными деревьями. Первый элемент пары, связанной отношением *логическая связь*, это обозначение логической операции, а второй элемент образует конструкцию, интегрирующую операнды такой операции. Рассмотрим утверждение *воздух может содержать диоксид серы*. Формализованное представление этого утверждения допускает формализованное описание ◊÷(воздух ◁ *диоксид серы*), в котором используется отношение *логическая связь*.

- *Последовательная серия* (...), это отношение, используемое для построения структур упорядоченных последовательностей объектов, связанных между собой одним и тем же основным или вспомогательным отношением.
- *Параллельная серия* (⁝), это отношение, связанное с конструированием последовательностей объектов, каждый из которых связан с первым объектом последовательности заданным отношением.
- *Пара* (◁▷) обозначает отношение, связывающее пары фрагментов знаний, являющихся частями сложного фрагмента знаний, использующего другие элементы.
- *Значение — условие* (▷◁), это отношение, применяемое при моделировании конфигураций, составленных фрагментами с ролями *условие* и *значение*, между которыми установлено отношение *равно при условии*. В структуре конфигурации вспомогательное отношение *значение — условие* применяется аналогично отношению *пара*.

Базу простых знаний отдельной области знаний образует множество троек *arb*, которые конструируются с использованием рассмотренных ранее отношений, извлекаемое из описаний классов логико-математической модели области знаний. Приведенные определения отношений являются частичными. Они могут быть расширены на множество неэлементарных конфигураций, конструируемых с помощью правил синтеза новых конфигураций, представляющих простые и сложные знания. Бинарная структура фрагментов конфигураций делает необходимой унификацию форматов фрагментов, использующих отношения, для которых число связываемых объектов не равно двум. Семейство таких форматов составляют образцы фрагментов конфигураций, составленные с использованием основных и вспомогательных отношений. Рассмотрим примеры таких фрагментов для представления логических выражений, составленных с использованием унарных и бинарных логических операций. В указанном случае логическое выражение вида $A * B$ $(*A)$, где * — бинарная (унарная) логическая операция, а A и B (A) — это логические выражения (выражение), представляется конфигурацией, изображенной на рис. 2 слева (справа).

Структурное представление задач

Отдельная задача задается с помощью фрагментов конфигураций, в которых уточняется семейство начальных данных и целей. Если таких фрагментов

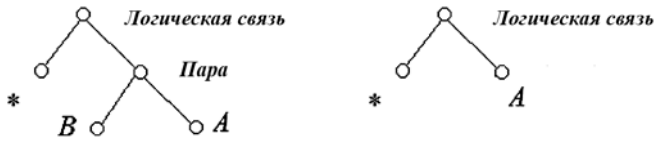


Рис. 2. Фрагменты конфигураций для логических выражений

несколько, то они составляют логическую структуру задачи, составленную фрагментами перечисленных типов. Поэтому всякую задачу можно представить нагруженным деревом, вершинам которого приписаны объекты разных типов. Элементы комбинации задач, связанных логическими операциями, составляют подзадачи этой задачи. Эта комбинация отражается с помощью элементов верхних ярусов структурного представления задачи. Логическая структура может быть одноэлементной. Она относится к отношению между двумя сущностями и соответствует элементарной задаче. Например, задачи *буря* ◁ X и *буря* ≜ X, иерархическая структура которых представлена на рис. 3, *а*, *б,* связаны соответственно с получением списков этапов и свойств заданного метеорологического процесса. Задачи, представления которых содержат не более одного фрагмента для предметного отношения, содержащего основное отношение, связанное с целью задачи, также являются элементарными задачами.

Простой является задача, составленная тремя условиями, выраженными с использованием предметных отношений: *скорость ветра* = X, *местоположение* = (a, b) и *время* = ($y.m.d.h$), в котором X — цель, (a, b) — координаты места, а ($y.m.d.h$) — время, для которого требуется найти значение скорости ветра. Структурное представление этой задачи также приведено на рис. 3, *в*. Удаление из постановки последней задачи элементов, задающих начальные данные, не препятствует процессу ее решения, поскольку используемая модель области знаний может содержать ссылку на процедуру поиска скорости ветра во внешнем ресурсе, использующую значения места и времени. Решение задачи применения указанной процедуры связано с нахождением семейства параметров этой процедуры и активацией задач нахож-

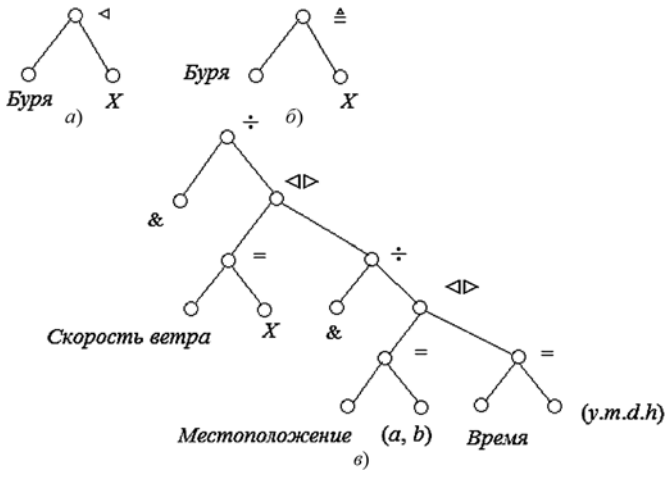


Рис. 3. Примеры структурных представлений простых задач

дения значений параметров. Последние реализуются запросом на представление недостающих начальных данных задачи, для которого можно предусмотреть разные алгоритмы реализации.

Конструкция логического фрагмента структуры сложной задачи использует отношение *логическая связь*, приписанное корню фрагмента, содержащего также обозначение конкретной логической операции и операнды операции, соответствующие подзадачам. Внутренние вершины, лежащие ниже логической структуры, размечены основными и вспомогательными отношениями. Висячие вершины структурного представления задачи — это элементарные знания или цели задачи. Для обозначения целей используются символы переменных. Вершины целей размечены символами переменных. Одинаковые символы переменных в структуре задачи соответствуют равным фрагментам конфигураций, конструируемых механизмом вывода. Рассмотрим пример сложной задачи, оперирующей несколькими целями, которая связана с нахождением свойств заданного природного явления, выявлением факторов, влияющих на их проявление, и может быть определена с помощью выражения (наводнение $\triangleq X$) & ($X \prec Y$). Структурное представление указанной задачи приведено на рис. 4.

Цели X и Y, рассматриваемые совместно, определяют общий формат нахождения решения, который составляет множество представления свойств. Каждому свойству соответствует множество объектов, представляющих понятия, значения которых влияют на проявление свойств. Решение задачи извлекается из фрагмента конфигурации, построенного из простых знаний по специальным правилам, согласованным с постановкой задачи. Для этого применяется подходящее трассирование в структуру фрагмента, позволяющее удалить его несущественные и вспомогательные элементы.

Сокращение множества целей приведенной задачи до одной, например Y, трансформирует содержание отыскиваемого решения во множество различных параметров, которые влияют на свойства рассматриваемого природного процесса. Выбор разных множеств символов переменных в качестве целей в общем случае приводит к разным задачам. Содержание задачи, постановка которой не содержит целей, связано с проверкой возможности синтеза фрагмента конфигурации в формализме абстрактного пространства знаний, в которое трассируется представление этой задачи.

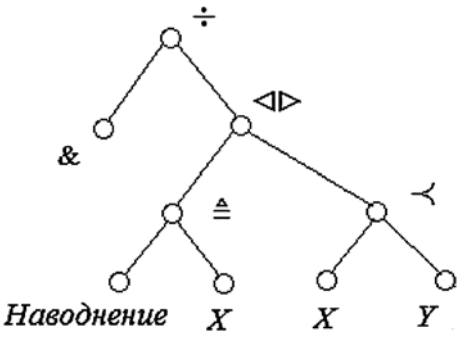


Рис. 4. Структурное представление задачи выявления факторов природного явления

Управление механизмом вывода для абстрактных пространств знаний

Рассмотрим схему практического использования инварианта для структуры конфигурации абстрактного пространства знаний. Она основана на правилах отбора и конструирования конфигураций, интегрирующих разнообразные знания, связанные с решением конкретных задач. Такие конфигурации составляются из элементарных конфигураций и семантических отношений между ними.

Представленный далее механизм решения задач основан на конструктах абстрактного пространства знаний и является аналогом синтеза в процессе мышления. Синтез конфигураций реализуется построением конфигураций, соответствующих решаемым задачам, из простых знаний с использованием специальных правил. Такие конфигурации интегрируют результаты обработки простых знаний. Они содержат решения задач, извлекаемые с помощью подходящих трассирований представлений задач в конфигурации. Правила конструирования фрагментов конфигураций оперируют условиями применения и образцами фрагментов конфигураций, из которых составляются посылки и заключения правил. Всякий образец посылки или заключения правила задается с помощью структуры бинарного дерева, вершины которого размечены отношениями, элементарными знаниями, а также именованными фрагментами, содержание которых произвольное.

Соответствие конкретного фрагмента конфигурации образцу связано с проверкой совпадения структуры и разметок вершин с точностью до анонимных фрагментов. Механизм вывода реализует регулярный процесс нахождения и применения правил. Получаемые при этом результаты используются механизмом вывода при поиске и применении других правил. В правилах конкретизируются схемы конструирования конфигураций, связанных с фрагментами постановки задач, оперирующими разными видами отношений. Многообразия правил, применяемых для разных интеллектуальных систем, могут отличаться. Это приводит к несовпадению возможностей решения профессиональных задач для интеллектуальных систем с совпадающими механизмом вывода и базой знаний, но разными системами правил синтеза фрагментов конфигураций.

Рассмотрим примеры универсальных правил, группируемых в классы по значениям отношений между фрагментами конфигураций, приписанными корневым вершинам структурных представлений элементарных задач. Для представления правил применяют схемы, составленные с использованием объектов трех типов: структурированных описаний образцов, условий, проверяемых для фрагментов образцов, а также вызовов процедур, вычисляющих требуемые значения по значениям параметров, которые содержатся в образцах. Отдельное правило представляется с помощью конструкции вида $\Sigma_1, ..., \Sigma_k \Rightarrow \Psi$,

где Σ_1, ..., Σ_k — конъюнкция посылок, а Ψ — образец, составляющий заключение правила.

Приводимые ниже схемы подобны схемам конструирования для методов формального анализа понятий и правдоподобного вывода [3, 4] и могут быть полностью формализованы. При этом проверка соответствия фрагментов конфигураций образцам, проверка условий и выполнение процедур допускают алгоритмизацию.

• *Правила композиции для иерархии в отношении являться* ($\subseteq$). Данную группу составляют правила моделирования структуры отношения вложения на заданном семействе множеств объектов, представленных понятиями области знаний. Такое моделирование осуществляется конструированием фрагментов конфигураций, представляющих конечные фрагменты диаграммы отношения *являться* на множестве понятий. Схематические представления правил построения конфигурации приведены на рис. 5. Фрагменты конфигураций, конструируемые с помощью приведенных правил, обеспечивают решение задачи $a \subseteq X$, где a — конкретное понятие области знаний, а X — символ неизвестной, обозначающий цель задачи. Правила конструирования структур нагруженных бинарных деревьев, реализуемые представленными на рис. 5 схемами, обеспечивают последовательное построение таких деревьев для фиксированных понятий.

Для отношений *являться* и *иметь частью* предполагается, что множество троек, представляющих эти отношения в базе простых знаний, неизбыточное и составлено для всех таких пар $(a, b) \in \rho$, для которых если $(a, b) \in \rho$, то не существует такого c, для которого $(a, c) \in \rho$ и $(c, b) \in \rho$.

Приведенные схемы относятся к произвольным фрагментам конфигураций, структуры которых представлены образцами соответствующей структуры, содержащимся в базе знаний или уже построенным с помощью процедуры механизма прямого вывода. В схеме, представленной на рис. 5, *а,* реализуется правило выполнения одного шага процесса формирования последовательной серии в отношении *являться*. Схема, представленная на рис. 5, *б*, описывает расширение параллельной серии фрагмента формируемой иерархической структуры, продолжающей последовательную серию вложенных или расширяемую систему непосредственных потомков *a* из этого фрагмента. Символ *D* обозначает произвольный

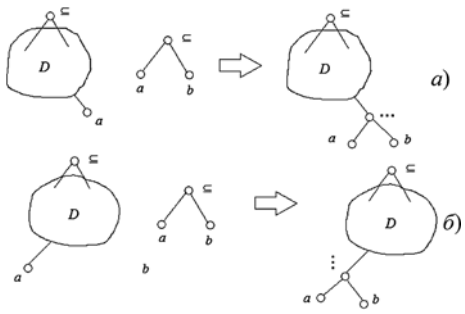


Рис. 5. Правила построения фрагментов иерархии в отношении *являться*

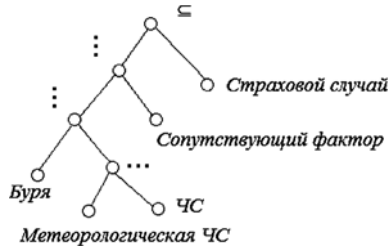


Рис. 6. Фрагмент отношения *являться* для системы понятий ЧС

фрагмент конфигурации. Приведенные правила являются недетерминированными, а их применение (в произвольном порядке до тех пор, пока это возможно) формирует полную конструкцию диаграммы отношения *являться*, выполняющегося для заданного понятия по базе простых знаний модели области знаний. Висячая вершина первых образцов в приведенных схемах распознается по условию связывания с предшествующей вершиной иерархии в качестве первого или второго потомка конструируемого дерева. На рис. 6 приведен возможный вид иерархии для фрагмента отношения *являться* рис. 1, формируемый с точностью до порядка элементов параллельных серий.

Аналогичные правила можно определить для задач $X \subseteq a$ и $X \subseteq Y$. В последнем случае формируется семейство деревьев, представляющих фрагменты отношения $\subseteq$, обрабатываемых дополнительными правилами сцепления фрагментов. Недетерминированность рассмотренных правил позволяет отделить задачу распознавания возможных действий при конструировании фрагментов конфигураций от вопроса организации решения задачи с наименьшей вычислительной сложностью и решать эти задачи раздельно.

• *Правила композиции для иерархии в отношении иметь частью* ($\lhd$). Правила композиции рассматриваемой группы относятся к решению задачи вида $a \lhd x$, где a — конкретное понятие, а x — понятие, составляющее цель задачи. Семейство правил представляют схемы, аналогичные схемам для отношения *являться*. Указанные схемы получаются заменой разметок вершин $\subseteq$ в схемах рис. 2 на $\lhd$. Схема дополнительного правила изображена на рис. 7.

Последняя схема имеет пять посылок. В ней учтена возможность извлечения знаний о частях *a* из частей более общих понятий, чем *a*. В таком случае тройка $a \lhd c$ по предположению не включается в базу простейших знаний, поскольку является избыточной.

Первая посылка обозначает произвольный фрагмент конфигурации, составленный с использованием рассмотренных правил, корню которого приписано отношение агрегирования. Вторая посылка обозначает фрагмент конфигурации, составленный с помощью правил для отношения *являться*, который определен полностью, на что указывает символ *, относящийся ко всему образцу второй посылки, заключенному в вертикальные скобки. Посылка $b \lhd c$ обозначает простое знание, встраиваемое в синтезируемую конфигурацию. Посылки $a \in Q$ и $b \notin D$ обозначают дополнительные требования к понятиям *a* и *b*, представленные условиями на вхождение в области *Q* и *D*.

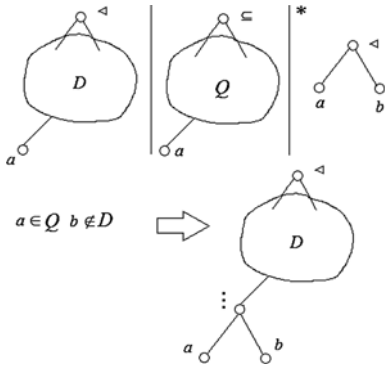


Рис. 7. Схема дополнительного правила для иерархии в отношении *иметь частью*

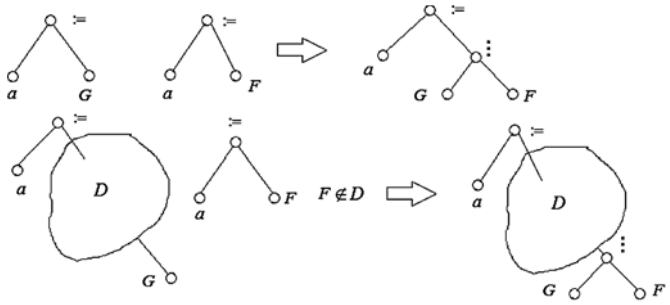


Рис. 8. Правила конструирования серии функциональных объектов, вычисляющих значение заданного понятия

- *Правила группирования функций вычисления значения понятия.* Нахождение значений понятий с помощью знаний об алгоритмах и формулах, выполняющихся в некоторой области знаний, является одним из распространенных видов простых задач, решаемых на базе формализованной модели этой области. Каждая такая задача представляется выражением $a := X$. Здесь a — понятие, а X — неизвестное значение этого понятия, составляющее цель задачи. Процесс решения рассматриваемой задачи составляют несколько видов этапов. Схема, приведенная на рис. 8, относится к первому этапу, которым реализуется интеграция знаний о функциональных объектах (алгоритмах, формулах и методах) нахождения значений произвольных заданных понятий с помощью конструкции параллельной серии.

Первая (вторая) схема на рис. 8 инициирует начало (продолжение) процесса построения параллельной серии, составленной всеми различными объектами, представляющими знания о нахождении значений понятия *a*.

- *Правила интеграции параметров функциональных зависимостей.* Следующая группа правил моделирует реализации вторых этапов процесса решения рассматриваемой задачи. Ее составляют недетерминированные правила формирования списков параметров, от которых зависят конкретные функциональные зависимости. Представления таких правил приведены на рис. 9. Они аналогичны схемам на рис. 8.

Результатом выполнения приведенных правил является параллельная серия, которую составляют без повторений все параметры, значения которых используются процедурой нахождения значения *F*. При этом порядок следования параметров функциональной зависимости *F* в конструируемом фрагменте конфигурации не детерминирован. Это влечет необходимость изменения схемы передачи параметров процедурам, которые предоставляются как множество значений вместе с наименованием параметров, к которым относятся такие значения.

- *Правило определения значения понятия по функциональным зависимостям.* Приводимое на рис. 10 правило относится в третьему этапу процесса решения задачи вычисления значения, принимаемого заданным параметром.

Семейство посылок приведенного правила составляют шесть объектов разных типов. Если посылки выполнены, то выполняется действие по формированию заключения, представляющего простое знание, задающее значение понятия *a*. Правило предполагает полное или частичное построение последовательности объектов, позволяющих вычислять значение этого понятия. Такая последовательность должна содержать объект, представляющий отображение *F*, для которого сформирована серия всех понятий, от которых зависит значение этого отображения. Четвертая посылка утверждает, что $\{a_1, ..., a_k\}$ — это множество всех параметров *F*. При этом множество Q вершин образца, соответствующего второй посылке, формализуется как все висячие вершины без вершины, размеченной объектом *F*. Пятая посылка связана с нахождением или проверкой, условия, что значения всех параметров *F* известны. Последняя посылка $A_F(a_1 = c_1, ..., a_k = c_k) = c$ указывает на необходимость применения процедуры вычисления значения *F* на наборе начальных данных $a_1 = c_1, ..., a_k = c_k$. Для этого применяется алгоритм логико-математической модели области знаний, который поставлен в соответствие *F*.

Правила рассмотренных трех групп позволяют моделировать процессы вычислений в формализме функциональных сетей. Они обеспечивают нахождение значений понятий, которые могут быть вычислены по начальным данным с помощью семейств функциональных зависимостей между понятиями, представленными различными методиками, применяемыми в моделируемой области знаний. Семейства таких зависимостей могут оказаться сложными для профессионального применения специалистами, содержать обширное разнообразие параметров разной природы и схем их использования. Рассмотрим, например, такой параметр, как продолжительность пожара в заданном

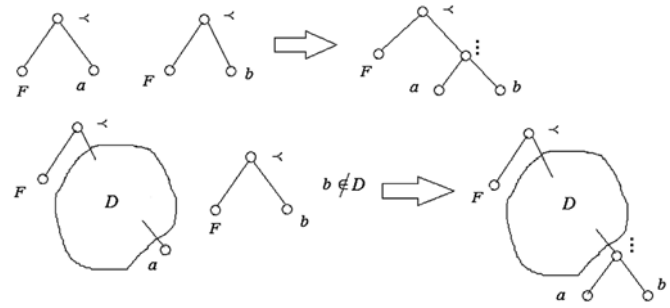


Рис. 9. Правила формирования списков параметров функций

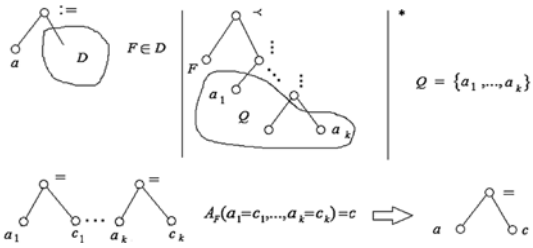


Рис. 10. Правило завершения нахождения значения понятия

помещении. Значение этого параметра определяется из соотношения $q * t * F_{\text{горения}} = QW_{\varepsilon}\gamma$, где q — удельное значение теплоты пожара; t — продолжительность пожара; $F_{\text{горения}}$ — площадь поверхности горения; Q — количество теплоты сгорания; W_{ε} — объем горючих веществ; γ — плотность горючих веществ. Кроме того, значение q вычисляется по формуле $q = z * Q * n * \beta$, где z — коэффициент химического недожега, зависящий от вида горючего материала; n — весовая плотность сгорания; β — коэффициент, значение которого определяется из таблицы или по эмпирическим формулам, одна из которых имеет вид $\beta = \frac{6F_{\text{ок}}}{F_{\text{п}}}$. В последней формуле $F_{\text{п}}$ и $F_{\text{ок}}$ — это площадь пола и проемов помещения, которые могут быть получены по эмпирическим данным зданий и помещений, размещенным в специальной базе данных, и вычисляются по значению адреса и номера помещения. Вычисления, проводимые на основе приведенных и подобных формул, несложные, а их применение алгоритмизируется на основе рассмотренных ранее схем.

- *Моделирование элементов логического вывода.* В заключение обсудим кратко некоторые возможности моделирования разных элементов процессов логического вывода. Они дополняют возможности осуществления алгебраических вычислений анализом и логическим обоснованием свойств конкретных понятий, составляющих содержание области знаний для конкретных начальных данных решаемых задач. К ним относятся схемы конструирования фрагментов конфигураций, представляющих утверждения о равносильности или следовании некоторых свойств из заданных свойств. Специальный интерес представляет следование при определенных условиях. Например, это может выражаться с помощью значений понятий, от которых зависит выполнимость рассматриваемых свойств. Простые правила манипулирования логическими связями представлены схемами рис. 11.

Здесь символы A и B обозначают фрагменты конфигураций с ролью *свойство*, составленные в виде логической комбинации используемых предметных предикатов. Схема на рис. 11, *а* представляет правило логического следования *modus ponens*, схема на рис. 11, *б* — правило субординации при выводе в системах с модальностями $(\Box A \ \& \ A \Rightarrow B) \Rightarrow \Diamond B$ [5]. Схема на рис. 11, *в* задает правило составления параллельной серии, составленной следствиями заданного свойства A. Приведенные правила связаны с решением задачи вида $A \Rightarrow X$, где A обозначает фрагмент конфигурации с ролью *свойство*, а X — цель задачи, значение роли которой равно *свойство*, которое следует из A.

- *Конкретизация в схемах логического вывода.* Применение правила логического следования связано с использованием значений параметров свойств, при которых посылка правила оказывается истинной. Поэтому истинность заключения устанавливается для тех значений понятий, когда последнее имеет место. Приводимая на рис. 12 схема получения логического следования учитывает необходимость использования и распространения на заключение значений понятий, обеспечивающих выполнимость посылок, при которых объявляется истинным заключение правила.

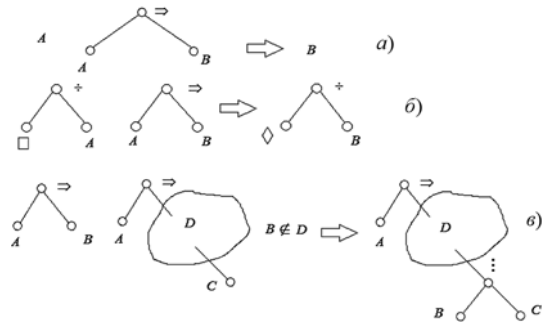


Рис. 11. Примеры правил моделирования логического вывода

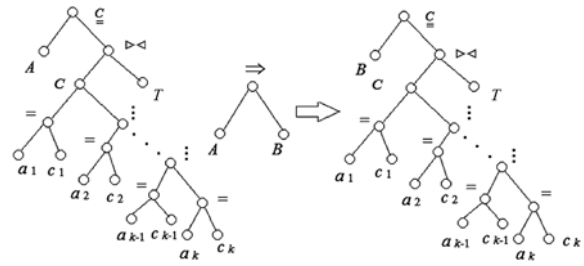


Рис. 12. Правила следования с конкретизацией

Последняя схема моделирует правило для логического следования, когда посылка следования истинна при конкретных значениях параметров для конструкции A, если для области знаний истинно соотношение $A \Rightarrow B$. Истинность A выражается конфигурацией, содержащей утверждение об истинностном значении A, равном T при некотором условии. В схеме условие представлено конъюнкцией равенств $(a_1 = c_1)$ & ... & $(a_k = c_k)$. Если это так, то объявляется истинным значение заключения B при условии $(a_k = c_k)$ & ... & $(a_k = c_k)$. Использованные в рассмотренной схеме отношения между конфигурациями включают отношения *равно при условии* $\binom{C}{=}$, *условие* $(\overset{C}{=})$ и *условие-значение* $(\triangleright\triangleleft)$. Здесь отношение $a \overset{C}{=} b$ связывает элементарную конфигурацию логического условия a с конфигурацией b, содержащей логическое значение и условие, при котором это значение достигается для a. Схема на рис. 12 соответствует случаю, когда условие, при котором принимается заданное значение, является конъюнкцией равенств.

Элементы управления механизмом вывода

Специальные атрибуты и форматы посылок правил позволяют управлять реализацией правил механизма вывода. В частности, атрибут * означает необходимость полного построения заданной структуры. Посылки ус-

ловия, представленные для фрагментов конструкций других посылок правила, моделируются проверяющими эти условия компонентами механизма вывода. Примерами таких условий являются $B \notin D$ или $F \in D$, где D — семейство вершин конфигурации, формализуемое с помощью теоретико-множественного выражения с использованием стандартных обозначений вершин такого дерева с помощью двоичных наборов, а также множеств висячих и внутренних вершин дерева, соответствующего фрагменту конфигурации. Процедурные посылки правил связаны с выполнением алгоритмов, для которых известны значения необходимых параметров.

Заключение

Приведенные в работе примеры правил построения фрагментов конфигураций абстрактного пространства знаний вместе с форматом представления задач допускают автоматизацию процесса решения таких задач. Ее реализует оператор вывода, основанный на конструкции механизма обратного вывода [6]. Вывод управляет процессом применения правил на основе системы целей, извлекаемых из логической структуры решаемой задачи, которая затем пополняется целями, связанными с посылками таких правил. При этом обеспечиваются как возможность моделирования процессов анализа и синтеза мышления, так и сочетание логических и алгебраических инструментов для реализации указанных процессов.

Список литературы

1. **Костенко К. И.** Формализмы представления знаний и модели интеллектуальных систем. Краснодар: Кубанский гос. ун-т, 2015. 300 с.
2. **Костенко К. И., Лебедева А. П.** О формализованных описаниях пространств знаний // Программная инженерия. 2013. № 8. С. 25—34.
3. **Вагин В. Н., Поспелов Д. А.** Достоверный и правдоподобный вывод в интеллектуальных системах. М.: Физматлит, 2008. 704 с.
4. **Kuznetsov S. O.** Machine learning and formal concept analysi//LNCS. 2004. Vol. 2961. P. 287—312.
5. **Фейс Р.** Модальная логика. М.: Наука, 1974. 520 с.
6. **Джексон П.** Введение в экспертные системы. М.: Вильямс, 2001. 623 с.

The Rules for Abstract Knowledge Spaces Inference Operator

K. I. Kostenko, kostenko@kubsu.ru, Kuban State University, Krasnodar, 350040, Russian Federation

Corresponding author:

Kostenko Konstantin I., kostenko@kubsu.ru, Head of Chair, Kuban State University, Krasnodar, 350040, Russian Federation
e-mail: kostenko@kubsu.ru

Received on March 21, 2016
Accepted on March 31, 2016

The initial concept of knowledge representation formalism is based on unified invariants for algebraic and semantic structures of knowledge. These invariants are proposed to be the implementation of independent knowledge properties specifications for notions of knowledge construction and knowledge transformation. Abstract knowledge spaces are defined as special class of formalisms with marked binary trees as unified knowledge structure format. These formalisms offer more sophisticated opportunities for knowledge representation and transformation in comparison with set-theoretic formalisms, learning spaces and descriptive logics. Tasks setting format and inference operator, as foundation for tasks solution procedure, are investigated in the submitted paper as additional invariants and considered to be the elements for subsequent development of the concept of knowledge representation formalisms. These invariants are specified for extensive class of abstract knowledge spaces application areas and submitted by thoroughly analyzed specification for simple and complex knowledge and also for tasks, constructed with the sets of elementary knowledge and subject semantic relations. Knowledge formalism inference operator is defined as a goals-controlled system operated by rules of goal knowledge synthesis that applies forward and backwards chaining inference schemes. The rule antecedents belong to classes named as templates, conditions and procedures calls. Rules consequents are elements of the templates class and define subject areas knowledge structures as composition performed over template antecedent's implementations. Task solution is extracted from goal knowledge structure synthetized by inference operator with one of abstract knowledge space tracing operations.

Keywords: *knowledge area, knowledge space, task structure, elementary knowledge, knowledge synthesis, inference operator, backward inference, forward inference*

Acknowledgements: *This work was supported by the Russian Foundation for Basic Research, project nos. 16-01-00214*

For citation:

Kostenko K. I. The Rules for Abstract Knowledge Spaces Inference Operator, *Programmnaya Ingeneria*, 2016, vol. 7, no 6, pp. 258—267.

DOI: 10.17587/prin.7.258-267

References

1. **Kostenko K. I.** *Formalizmy predstavlenija znanij I modeli intellektualnyh sistem* (Knowledge representation formalisms and intelligent systems models), Krasnodar, Kuban state university, 2015, 300 p. (in Russian).
2. **Kostenko K. I., Lebedeva A. P.** O formalizovannyh opisanijah prostranstv znanij (On formalized knowledge spaces descriptions), *Programmnaya Ingeneria*, 2013, no. 8, pp. 25—34 (in Russian).
3. **Vagin V. N., Pospelov D. A.** *Dostovernyj i pravdopodobnyj vyvod v intellectual'nyh sistemah* (Reliable and plausible inference in intelligent systems), Moscow, Fizmatlit. 2008. 704 p. (in Russian).
4. **Kuznetsov S. O.** Machine learning and formal concept analysis, *LNCS*, 2004, vol. 2961, pp. 287—312.
5. **Feys R.** *Modalnaja logika* (Modal logic), Moscow, Nauka, 1974, 520 p. (in Russian).
6. **Jackson P.** *Vvedenie v Jekspertnye sistemy* (Introduction to expert systems), Moscow, Viljams, 2001, 623 p. (in Russian).

УДК 004.415.2; 004.67 DOI: 10.17587/prin.7.268-282

А. А. Большаков, д-р техн. наук, проф., e-mail: aabolshakov57@gmail.com,
Р. В. Макарук, канд. техн. наук, ст. препод., e-mail: n.diablo.n.f@list.ru,
Санкт-Петербургский государственный технологический институт
(технический университет)

Программный комплекс для анализа характеристик и оценки уровня информационной безопасности вычислительной сети на основе нечетких моделей

Предложены нечеткие модели, включая структуру и параметры функций принадлежности, а также алгоритмы для анализа характеристик и оценки уровня информационной безопасности вычислительных сетей. Их отличительной особенностью является возможность учета объективной, а также субъективной информации в виде суждений, знания и опыта экспертов. Кроме этого, предложенный подход позволяет осуществлять анализ характеристик и оценку уровня информационной безопасности в условиях неполноты информации. Также рассмотрен разработанный программно-алгоритмический комплекс, реализующий созданные нечеткие модели и алгоритмы.

Ключевые слова: *нечеткие модели, производительность, надежность, безопасность, вычислительные сети, программно-алгоритмический комплекс*

Введение

Вычислительные сети (ВС) являются сложными системами, предназначенными для решения широкого круга задач. В настоящее время все большее внимание уделяется таким сетям. Среди задач, решаемых с помощью ВС, наиболее важными являются обеспечение производительности, надежности и безопасности работы пользователей и, собственно, сетей [1].

Технологии ВС оказывают существенное влияние на уровень экономической конкурентоспособности и национальной безопасности любого государства в современном мире. Сетевые инфраструктуры государственных учреждений и частных предприятий обычно подключены к глобальной сети Интернет. Однако эти технологии имеют ряд проблем, связанных с основными характеристиками ВС [2].

На различных предприятиях ВС представляют совокупность различных сегментов, в которые входит большое число рабочих станций, серверов и сетевого оборудования. Таким образом, перед администраторами ВС возникает ряд сложных задач по обеспечению качества работы сети, в том числе информационной защиты. Для обеспечения безопасности требуется разработка не частных механизмов защиты, а реализация системного подхода, включающего комплекс взаимосвязанных мер, таких как использование технических и программных средств, нормативно-правовых актов, организационных мероприятий, морально-этических мер противодействия и т. п. [3, 4]. Комплексный характер работ обусловливает необходимость значительных затрат ресурсов, которые требуются от администраторов по киберзащите для оценки информационной безопасности ВС и реализации мер по защите ресурсов сетей от различных угроз. Проблема информационной защиты становится еще более серьезной в связи с развитием и распространением ВС, территориально распределенных и с удаленным доступом к совместно используемым ресурсам. Построенные комплексы состоят из многих компонентов, мониторинг качества каждого из них вручную весьма сложен. Возможное уголовное преследование за утерю конфиденциальной информации делает работу администратора кропотливой и ответственной [2, 5].

По данным корпорации Cisco, опубликованным в исследовании 2015 г. "Visual Networking Index: Forecast and Methodology", совокупные темпы годового роста глобального IP-трафика (*Compound Annual Growth Rate*) превысят 23 % от 2014 к 2019 г., что в 64 раза превысит показатели 2005 г. [6]. Количество распределенных сетевых атак, их разновидности и максимальный размер ботнета в 2015 г. выросли на 21 % по сравнению с прошлым годом [7]. В 2014 г. центром стратегических и международных исследований CSIS (*Center for Strategic and International Studies*)

и компанией McAfee опубликован доклад, в котором была приведена оценка мирового ущерба от преступлений в сфере информационных технологий — свыше 400 млрд долл. США в год [8].

Необходимо обеспечить качественную работу ВС, причем понятие качества обслуживания включает такие характеристики, как производительность, надежность и информационная безопасность. Каждая из перечисленных характеристик оценивается различными показателями:

- производительность оценивается скоростью передачи информации и задержкой передачи пакетов;
- надежность оценивается долей потери пакетов, доступностью, отказоустойчивостью;
- безопасность оценивается наличием взаимосвязанных мер [1].

Обычно анализ ВС выполняется на основе вероятностных подходов, в последнее время — с использованием математического аппарата теории нечетких множеств [4, 9, 10]. Проводя анализ характеристик ВС и оценку уровня информационной безопасности, необходимо учитывать не только объективные характеристики, но и субъективные факторы: суждение, знания и опыт экспертов. Кроме этого, процессы, протекающие в ВС, характеризуются существенной долей неопределенности, нестабильности и т. п. Все эти факторы являются существенным препятствием для создания моделей на основе классических математических теорий и методов. Поэтому для решения задач анализа характеристик и оценки информационной безопасности перспективными являются модели и системы с нечеткой логикой [1, 2, 5, 11, 12].

Вопросам анализа и оценки характеристик ВС посвящены работы Д. Феррари, В. Г. Олифера, Н. А. Олифера, Э. Танненбаума [9], использовавших классические математические теории и методы, и работы Т. Тэрано, Е. А. Басыни, А. Г. Корченко, R. P. Lippmann, Ma Ruhui [2, 10, 13, 14], использовавших математический аппарат теории нечетких множеств. В работах, базирующихся на классических подходах, основное внимание уделяется оценке производительности и надежности. Аппарат теории нечетких множеств использовался в основном для оценки информационной безопасности ВС. Так, в работе [15] автор описал систему управления трафиком ВС, основанную на генетических алгоритмах и нечеткой логике, для противодействия сетевым угрозам и повышения общего уровня информационной безопасности.

Анализ показал, что работы, связанные с созданием методов и алгоритмов комплексного анализа и оценки характеристик ВС в условиях неопределенности и неполноты исходной информации, которыми характеризуются ВС, практически отсутствуют. В связи с этим синтез методов и алгоритмов анализа характеристик и оценки информационной безопасности ВС в условиях неопределенности и неполноты данных является актуальным.

Постановка задачи

Целью работы, результаты которой представлены в статье, является создание программно-алгоритмического комплекса (ПАК), нечетких моделей, методов и алгоритмов анализа характеристик и оценки информационной безопасности ВС, позволяющих проводить анализ и оценку в условиях неопределенности и неполноты исходных данных, а также повысить эффективность функционирования ВС.

Для достижения поставленной цели необходимо решить перечисленные далее задачи.

1. Анализ характеристик ВС и критериев их оценки.
2. Создание структуры ПАК. Разработка требований к составу функциональных характеристик.
3. Разработка нечетких моделей, методов описания характеристик ВС в условиях неопределенности и неполноты исходных данных.
4. Разработка алгоритмов анализа и оценки характеристик ВС в условиях неопределенности и неполноты исходных данных.
5. Построение программного модуля поддержки принятия решения, формирующего рекомендации по улучшению оценки безопасности.
6. Анализ результатов тестирования ПАК анализа и оценки характеристик ВС.

Анализ характеристик ВС и критериев их оценки

Задачу анализа характеристик и оценки информационной безопасности ВС поставим следующим образом: для заданной ВС сформулировать способы анализа и оценки производительности E_i, надежности R_i и безопасности F_i получения необходимых характеристик для их оценки $\mathbf{V}e_i$, $\mathbf{V}r_i$, $\mathbf{V}_i$, а также сохранения информации о полученных оценках в базу данных с последующей возможностью повторного доступа к ним для анализа.

Оценка производительности ВС зависит от анализа значений вектора параметров $\mathbf{V}e_i$, полученных активным измерением, либо введенных пользователем системы вручную. Учитывая полученные данные, необходимо выполнить оценку производительности ВС. Характеристика $\mathbf{V}e_i$ определяется следующим образом:

$$\mathbf{V}e_i = \{M_{OWD}, M_{SIR}, M_{TR}\}, \quad (1)$$

где M_{OWD}, M_{SIR}, M_{TR} — метрики односторонней задержки пакета, средней скорости работы сети и времени реакции сети, соответственно.

Таким образом, задачу оценки производительности ВС можно сформулировать следующим образом: на основе анализа данных $\mathbf{V}e_i$ (1), учитывая возможность их отсутствия по ряду параметров, получить оценку производительности E_i.

Оценка надежности ВС зависит от анализа составляющих характеристику $\mathbf{V}r_i$, полученных активным измерением и вводом пользователем системы

данных по доступности сетевых узлов (при долговременной работе разрабатываемого ПАК возможно автоматизированное отслеживание) и возможности их отказоустойчивости. По полученным данным требуется осуществить оценку надежности ВС. Вектор параметров $\mathbf{V}r_i$ определяется следующим образом:

$$\mathbf{V}r_i = \{M_L, M_A, M_{FT}\}, \quad (2)$$

где M_L, M_A, M_{FT} — метрики доли потерянных пакетов, доступности узла и отказоустойчивости, соответственно.

Задача оценки надежности ВС формулируется следующим образом: на основе анализа данных $\mathbf{V}r_i$ (2), полученных активным измерением и вводом пользователем системы, учитывая возможность отсутствия данных по ряду параметров из $\mathbf{V}r_i$, сформировать оценку надежности R_i.

Для формирования оценки информационной безопасности, анализируемой ВС, целесообразно вычислить промежуточные оценки по данным сканирования Fs_i и опроса Fa_i. Затем на основе полученных промежуточных оценок сформируется итоговая оценка информационной безопасности ВС F_i.

Оценка информационной безопасности ВС по данным сканирования зависит от значений параметров ВС: число работающих сервисов каждого типа, общее число рабочих станций и серверов, типы используемых операционных систем и т. п. На этапе анализа характеристик ВС необходимо автоматизированное сканирование характеристик сети модулем сканирования, либо их ввод вручную пользователем системы. Ручной ввод характеристик предусматривается для возможности работы с системой на этапе проектирования ВС, либо для моделирования определенных ситуаций. С помощью анализа полученных характеристик формируются результирующие данные по необходимому вектору параметров $\mathbf{V}_i$ для оценки информационной безопасности ВС по результату сканирования:

$$\mathbf{V}_i = \begin{Bmatrix} P_{80}, P_{443}, P_{21}, P_{S22}, P_{W22}, P_{23}, \\ P_{3306}, P_{389}, P_{445}, O_W, O_U, O_O \end{Bmatrix}, \quad (3)$$

где P_{80}, P_{443}, P_{21}, P_{S22}, P_{W22}, P_{23}, P_{3306}, P_{389}, P_{445}, O_W, O_U, O_O — относительное количество: веб-сервисов; веб-сервисов, работающих по протоколу HTTPS; FTP-сервисов; SSH-сервисов на серверах; SSH-сервисов на рабочих станциях; *telnet*-сервисов; MySQL-серверов; LDAP-серверов; SAMBA-серверов; устройств, работающих под управлением ОС MS Windows; устройств, работающих под управлением ОС UNIX-like; устройств, работающих под управлением ОС, отличных от MS Windows и UNIX-like. По вектору параметров $\mathbf{V}_i$ (3) осуществляется оценка информационной безопасности ВС по результату сканирования Fs_i. Необходимо отметить, что для оценивания допускается отсутствие некоторых из значений в параметрах вектора $\mathbf{V}_i$ [5].

Оценка информационной безопасности по данным опроса формируется в зависимости от ответов пользователя системы. Опросник составлен в результате интервьюирования экспертов в области информационной безопасности [11]. На настоящем этапе осуществляется опрос пользователя системы, данные ответов формируют значения вектора параметров $\mathbf{A}_i$, необходимые для оценки ВС по результату опроса. В опрос включено множество вопросов, в числе которых: "доступ к серверам ограничен?", "на рабочих станциях сотрудников установлены безопасные пароли?", "уровень доверия к пользователям", "используется технология Honeypot?" и т. д. Пользователю допускается не отвечать на некоторые из вопросов. С использованием ответов на вопросы опросника проводится оценка информационной безопасности ВС по результату опроса пользователя системы Fa_i [5, 11].

Система использует нечеткие модули, в состав которых входят нечеткие продукционные базы правил. Базы правил сформированы в результате интервьюирования экспертов в области ВС и информационной безопасности [11, 12].

Создание структуры ПАК. Функциональные характеристики

Анализ требований к характеристикам ВС, критериям комплексной оценки их функционирования позволил разработать структуру ПАК анализа характеристик и оценки информационной безопасности ВС.

Предлагаемый комплекс состоит из нескольких подсистем. Общая структурная схема ПАК представлена на рис. 1 [5].

Дадим краткую характеристику модулей и интерфейсов.

- *Интерфейс администратора базы данных.* Позволяет следить за актуальностью базы данных (ПО) для построения защиты. В базе содержатся списки ПО для построения комплекса защиты. Кроме того, интерфейс предназначен для поддержания в актуальном состоянии списков разрешенного и запрещенного к использованию на рабочих местах ПО.
- *Интерфейс инженера знаний.* Предназначен для дополнения и изменения базы знаний. Инженер знаний работает с экспертами, вместе они формируют базу знаний, на основе которой проводятся анализ и оценка характеристик ВС.
- *Модуль оценки характеристик ВС.* На основании данных о сети, полученных на основе автоматизированного сканирования, или ввода пользователем системы, формирует оценку характеристик ВС.
- *Модуль поиска необходимого ПО для защиты локальной вычислительной сети (ЛВС).* На основе оценки безопасности ВС и желаемого уровня безопасности проводит поиск в базе данных ПО для построения защиты. Сформировав список необходимого ПО, этот модуль передает его в модуль визуализации.
- *Модуль поиска рекомендаций.* Формирует рекомендации по увеличению уровня безопасности сети.
- *Модуль мониторинга текущего состояния сети.* Связан с программами-мониторами и работает не на стадии проектирования ВС, а в процессе ее эксплуатации.

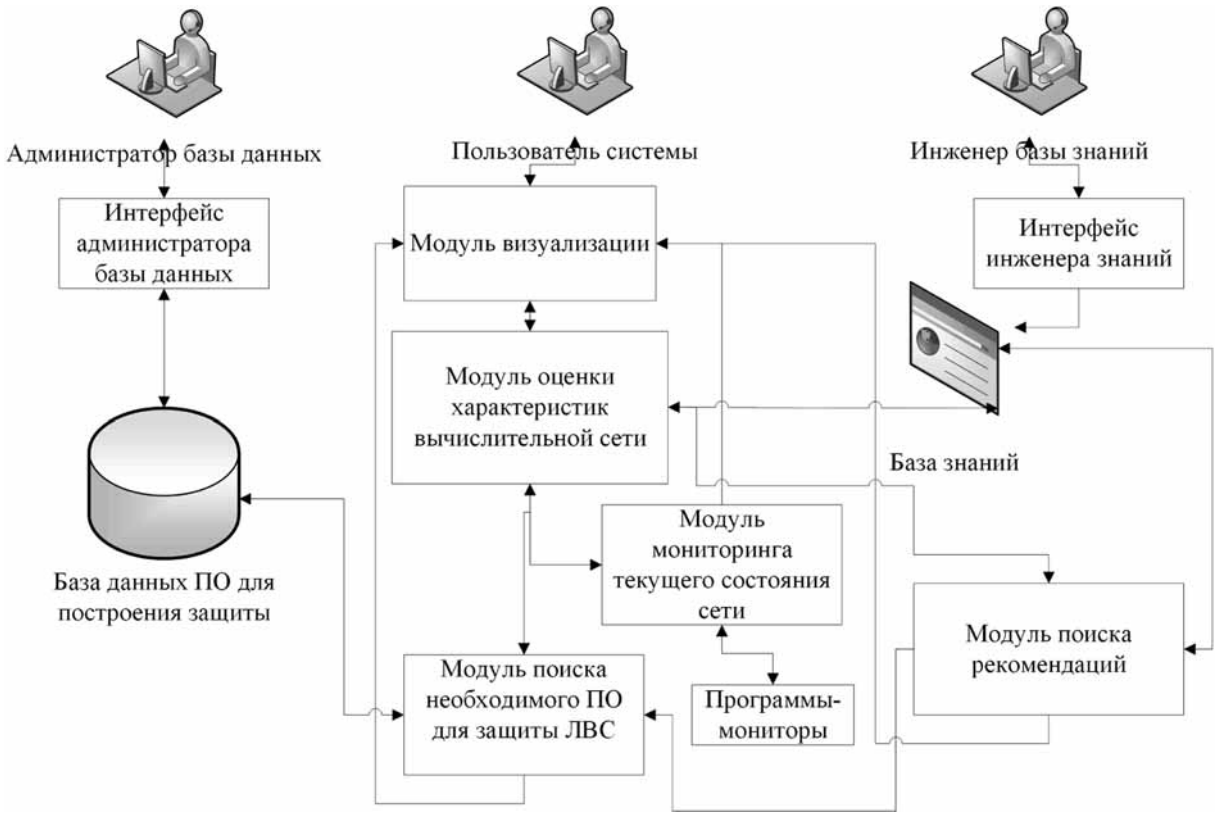


Рис. 1. Общая структура ПАК

• *База данных ПО для построения защиты.* Содержит компоненты и их характеристики для формирования списка необходимого ПО для повышения уровня безопасности ВС. Кроме того, база содержит данные о предыдущих исследованиях сети и о предложенных мерах, если это необходимо [5, 16].

На первом этапе комплекс анализирует текущее состояние ВС и на основе нечеткой логики формирует оценки производительности, надежности и информационной безопасности, а также предлагает набор решений для улучшения безопасности, если это необходимо. Данные для анализа могут быть получены автоматизированным способом — сканированием характеристик ВС, и/или вводом в ручном режиме (используется для оценки характеристик проектируемой сети). Для анализа сети в реальном масштабе времени используются специальные программы-мониторы, установленные на каждом компьютере сети. Отличие предлагаемого ПАК от аналогичных продуктов состоит в следующем: оценка характеристик ВС основывается на методах нечеткой математики; серверная часть работает только под управлением ОС UNIX, которая по данным исследований [17—20] является наиболее надежной и безопасной по сравнению с ОС от корпорации Microsoft; программы-мониторы функционируют под следующими ОС: Linux (минимальная версия ядра 2.6.16), FreeBSD (минимальная версия 7.0), Microsoft Windows (минимальная версия XP); ПАК работает с пользователем от стадии проектирования ВС до стадии ее работы [1, 5, 16].

Анализ характеристик и оценка информационной безопасности ВС предусматривает две стадии: получение значений параметров ВС и ответов пользователя системы на опросник. На основе полученных данных выполняется оценка производительности, надежности и информационной безопасности ВС [1, 5].

На первом этапе пользователь системы выбирает метод получения характеристик сети: автоматизированное сканирование, ручной ввод пользователем системы, ручной ввод системным администратором анализируемой сети, загрузка протокола сканирования программой nmap. При выборе автоматизированного метода получения характеристик запускается модуль сканирования сети, по работе которого формируется протокол. Далее протокол сканирования и/или введенные вручную данные анализируются для формирования значений характеристик $\mathbf{V}e_i$ (1), $\mathbf{V}r_i$ (2) и $\mathbf{V}_i$ (3).

На втором этапе пользователь системы отвечает на вопросы опросника. Ответы формируют значения вектора параметров $\mathbf{A}_i$, необходимые для оценки уровня безопасности по данным опроса Fa_i.

Последний этап заключается в формировании оценок производительности E_i, надежности R_i и безопасности F_i ВС. Для реализации соответствующих вычислительных процедур необходим переход к единой математической платформе, позволяющий

описать информацию в условиях неопределенности и неполноты данных. Для этого используется математический аппарат нечетких моделей.

Разработка нечетких моделей, методов описания характеристик ВС в условиях неопределенности и неполноты исходных данных

Проблема принятия решений — одна из наиболее распространенных классов задач с практическим приложением. В большинстве таких случаев решения принимаются в условиях, когда исходные данные точно не известны. Для работы с неточно известными величинами часто применяют методы классической математики. При этом понятие нечеткости, или расплывчатости, отождествляют со случайностью. Однако различие между случайностью и нечеткостью состоит в том, что случайность связана с неопределенностью принадлежности (непринадлежности) объекта ко множеству, а состояние нечеткости связано с классами, в которых используются градации степени принадлежности, задающие промежуточные значения между полной и неполной принадлежностью объектов к этим классам [10].

В системах защиты информации большую роль играют не полностью определенные факторы, тогда как в классической теории множеств каждый элемент, который входит в определенный список, обязательно принадлежит множеству. Однако задачи, которые не поддаются строгой формализации, человек решает с использованием собственных субъективных и расплывчатых представлений. Устранить разногласие между классической математикой и неопределенностями реального мира [10] призвана нечеткая логика.

В работах американского ученого Лотфи Заде положено начало развитию теории, которая обеспечивает описание моделей принятия решений в условиях нечеткой информации. Так, нечеткие методы решения поставленных задач характеризуются тем, что используют лингвистические переменные вместо традиционных числовых, простые отношения описываются с использованием нечетких высказываний, а более сложные — нечеткими алгоритмами. Базы правил позволяют достаточно быстро обрабатывать сложные соединения. Таким образом, важным преимуществом моделей, построенных на нечеткой логике, является большая гибкость и адекватность реальному миру, а также более быстрое по сравнению с традиционными моделями получение результата [10].

Решая задачу оценки характеристик ВС, необходимо определить, какие из параметров производительности, надежности и безопасности должны быть обеспечены в оцениваемой ВС. Практика показывает, что в ВС не для всей информации необходимо обеспечение заданных значений всех параметров. Например, при рассмотрении безопасности известно, что для общедоступной информации не требуется обеспечивать ее секретность. При оценке

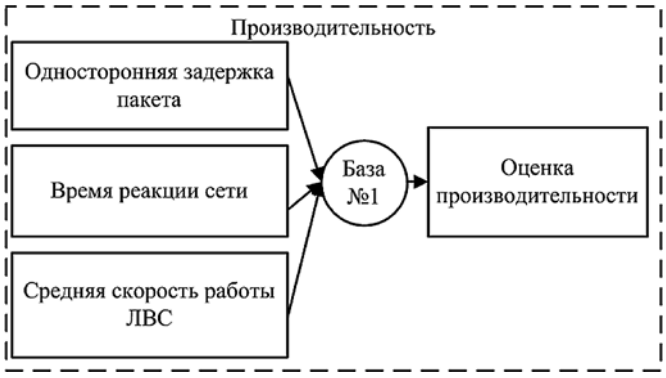


Рис. 2. Модель оценки производительности ВС

безопасности необходимо осуществить анализ возможных угроз и их влияния на характеристики безопасности. Далее формируются экспертные запросы и осуществляется их ранжирование. После проведения интервьюирования специалистов в области ВС строятся нечеткие лингвистические переменные и базы правил [10—12].

Для решения задачи оценки производительности ВС на основе серии стандартов RFC 2679, 2330, 2681, 3393, рекомендаций [9] и интервьюирования экспертов разработана модель, представленная на рис. 2 [1]. Здесь элементы в виде прямоугольника — нечеткие лингвистические переменные; элементы в виде круга — нечеткие операционные модули, основанные на правилах.

Для решения задачи оценки надежности ВС на основе рекомендаций [9] и интервьюирования экспертов разработана модель, представленная на рис. 3 [1].

Для решения задачи оценки безопасности ВС на основе серии стандартов ГОСТ Р ИСО/МЭК 15408 и ГОСТ Р ИСО/МЭК 27001 и интервьюирования экспертов в области информационной безопасности разработана модель, представленная на рис. 4 [1, 5, 11, 12, 15, 17, 18].

Для формирования нечетких моделей оценки характеристик ВС использовалось специализированное ПО Fuzzy ToolBox [21]. Также для реализации методов нечеткой математики в представленном ПАК ис-

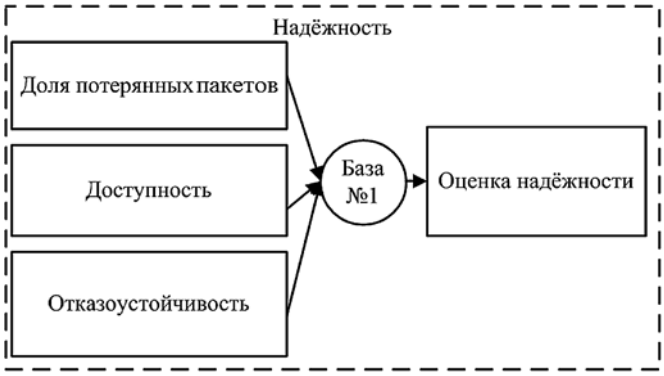


Рис. 3. Модель оценки надежности ВС

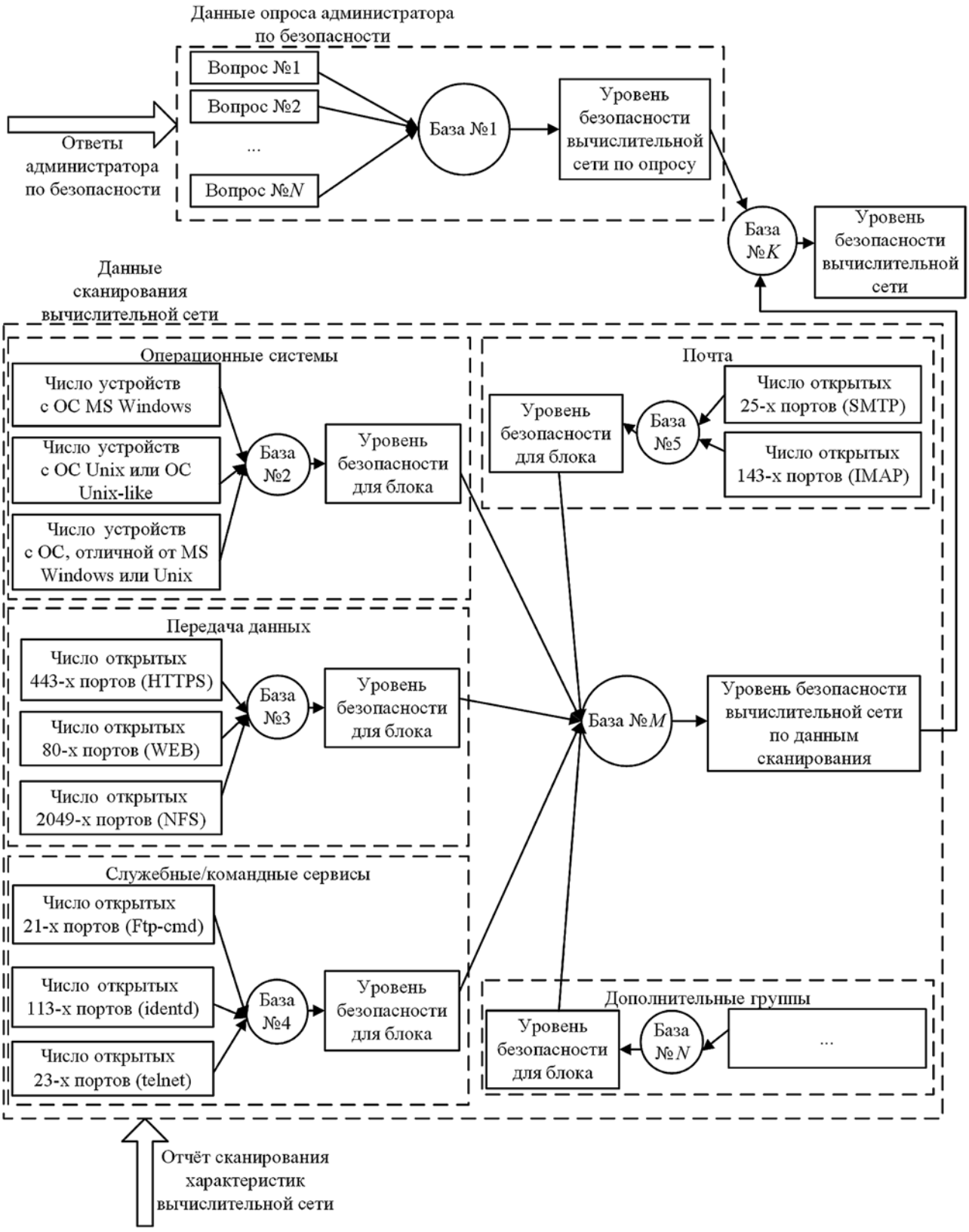


Рис. 4. Модель оценки информационной безопасности ВС

пользовался математический модуль из пакета Fuzzy ToolBox, построенный с использованием языка C. При этом написана программа, а также отдельная функция, которая по результатам работы основной формирует функции принадлежности с использованием выбранной в качестве аппроксимирующей Pi-функции для построения нечеткой математической модели оценки безопасности ВС:

$$\mu(x) = \begin{cases} 0 & x < a_1 \\ sfunc_1(a_1, b_1, x) & a_1 \leqslant x \leqslant b_1 \\ 1 & b_1 < x < b_2 \\ sfunc_2(a_2, b_2, x) & b_2 \leqslant x \leqslant a_2 \\ 0 & x > a_2 \end{cases},$$

где $sfunc_1(a_1, b_1, x) = \begin{cases} \dfrac{(x-a)^2}{2(c-a)^2}, & x \leqslant c \\ 1 - \dfrac{(b-x)^2}{2(c-a)^2}, & x > c \end{cases}$,

$$sfunc_2(a_2, b_2, x) = \begin{cases} 1 - \dfrac{(b-x)^2}{2(c-a)^2}, & x \leqslant c \\ \dfrac{(x-a)^2}{2(c-a)^2}, & x > c \end{cases},$$

где $c = \dfrac{a+b}{2}$.

Пример работы программы в среде MATLAB для оценки информационной безопасности представлен на рис. 5. На рис. 5, *а* представлен результат работы алгоритма нечетких *c*-средних, а на рис. 5, *б* — результат перевода к виду Pi-функции.

Созданные алгоритмы используют разработанные нечеткие модели, ниже приведено описание алгоритмов определения уровня информационной безопасности и производительности ВС.

Разработка алгоритмов анализа и оценки характеристик ВС в условиях неопределенности и неполноты исходных данных

Алгоритм оценки информационной безопасности представлен на рис. 6 [1]. Он позволяет получить оценку информационной безопасности ВС по данным анализа параметров сети. На вход алгоритма подаются значения ряда переменных, необходимых для начала процесса анализа характеристик сети: диапазон IP-адресов и метод сканирования. Результатом анализа являются значения характеристики Fs_i. Далее проводится опрос пользователя системы по опроснику, ответы формируют значения характеристики Fa_i. На следующем этапе с использованием полученных данных формируется оценка информационной безопасности ВС F_i.

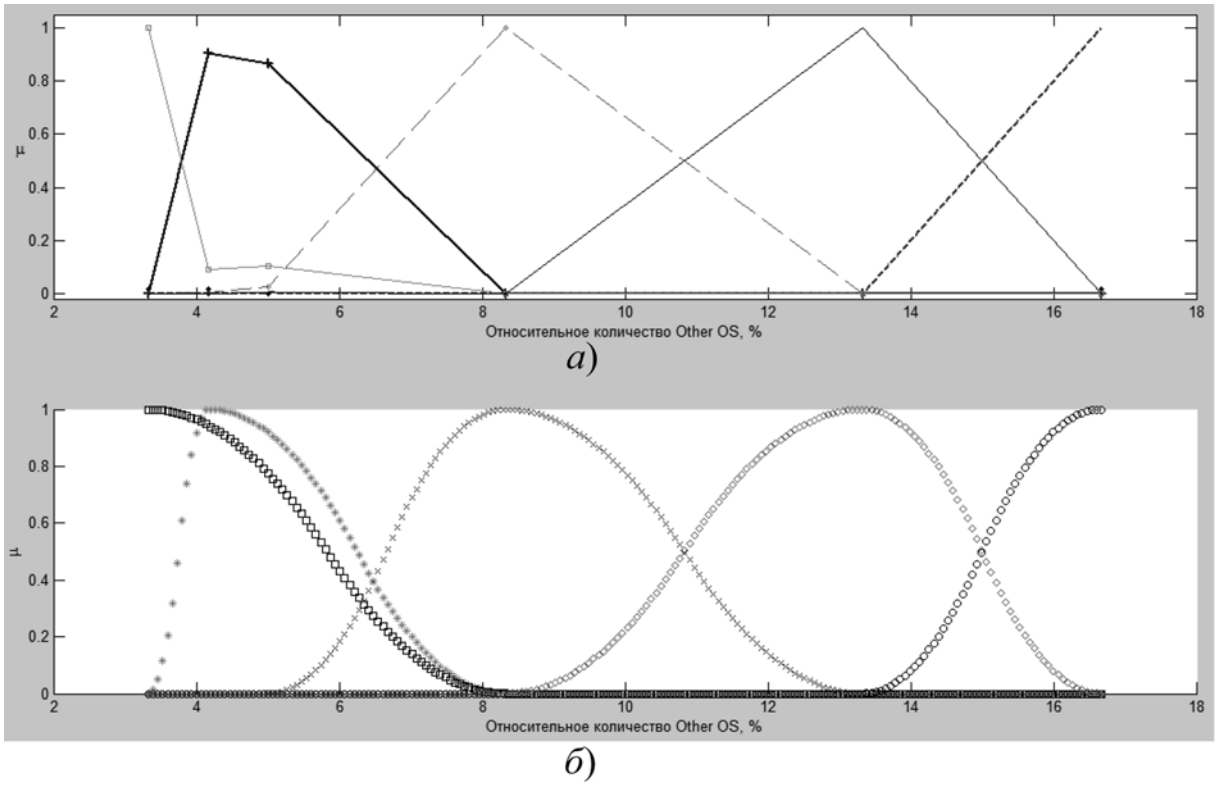


Рис. 5. Пример функции принадлежности

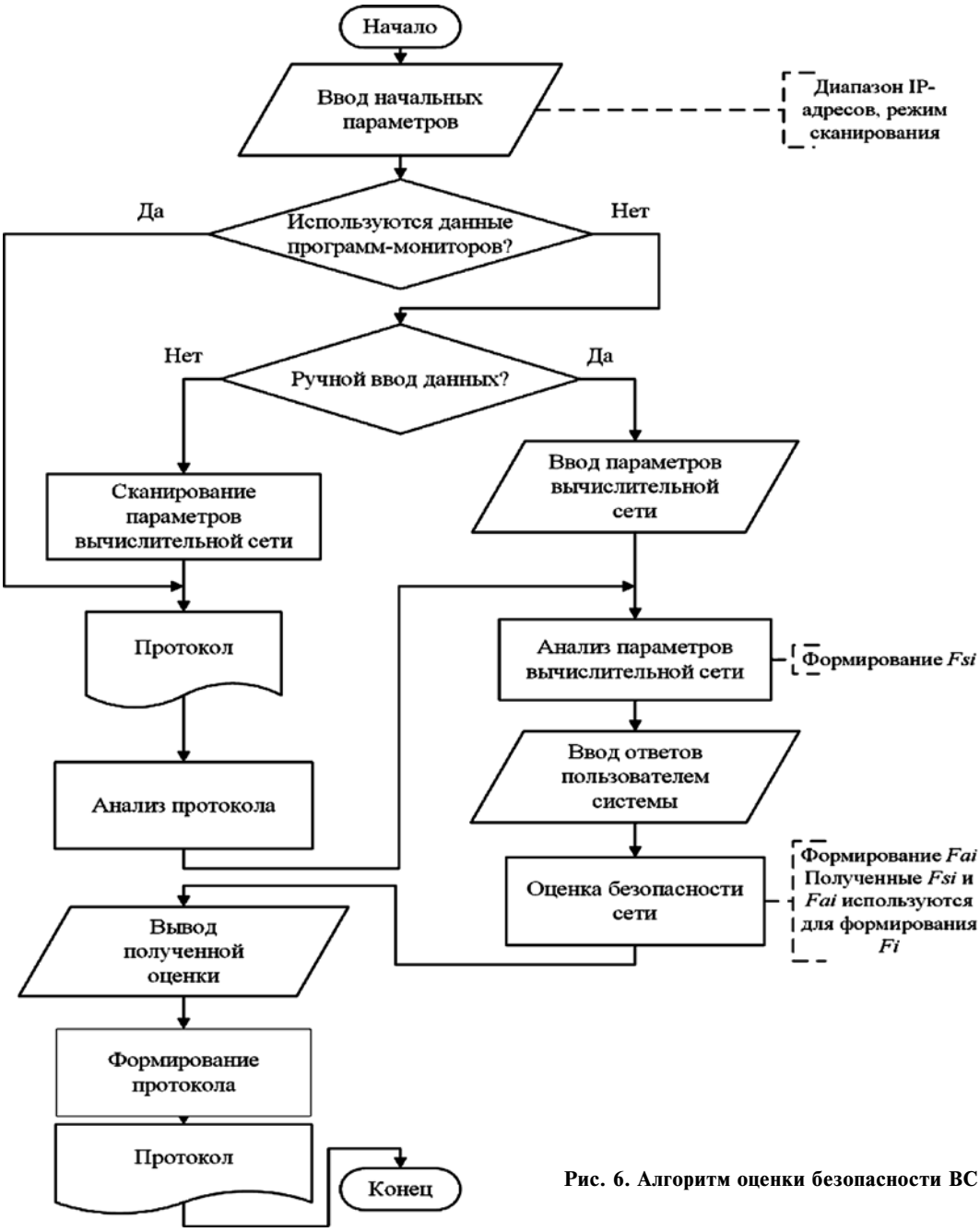


Рис. 6. Алгоритм оценки безопасности ВС

После установки начальных параметров для анализа сети, в случае выбора автоматизированного метода сбора характеристик ВС без использования программ-мониторов, начинает работать модуль сканирования, также возможно использование имеющегося протокола изучения сети с использованием программы *nmap*, вместо запуска программного модуля. Для модуля сканирования возможно задание следующих режимов: *paranoid, sneaky, polite, normal, aggressive, insane*. Первые два режима предназначены для получения данных в сетях с установленными системами обнаружения вторжений. Сканирование в таком режиме требует много времени. Так, *polite*-режим снижает интенсивность сканирования для меньшего потребления пропускной способности и машинных ресурсов. *Aggressive*-режим повышает интенсивность сканирования, предполагая использование довольно быстрой и надежной сети. *Insane*-режим предполагает использование быстрой сети, а также снижение точности для повышения скорости [1, 16].

При использовании программ-мониторов нагрузка на ВС значительно снижается, а качество и точность результатов повышаются. Связано это с тем, что запущенные модули ПАК на клиентских ма-

шинах имеют возможность собрать более детальную информацию по используемым сервисам, в том числе по защищенным межсетевыми экранами, или получить данные по установленному и используемому ПО локально, без применения ВС.

Модуль сканирования и программы-мониторы формируют отчеты, в результате анализа которых формируются значения вектора параметров $\mathbf{V}_i$ (3). В случае выбора ручного ввода характеристик ВС значения параметров вводятся пользователем системы или системным администратором. Сформированные значения характеристики $\mathbf{V}_i$ (3) анализируются с применением разработанных нечетких моделей. По полученным в каждой модели промежуточным ответам на основе нечеткой продукционной базы правил формируется оценка информационной безопасности ВС по данным сканирования Fs_i. Далее пользователю системы необходимо ответить на вопросы опросника. По ответам формируются значения вектора параметров $\mathbf{A}_i$. Полученные значения $\mathbf{A}_i$ анализируются с использованием разработанной нечеткой подмодели "Опросник". Результатом анализа является оценка информационной безопасности ВС по данным опроса Fa_i. На последнем этапе с применением базы правил и полученных оценок Fs_i и Fa_i формируется итоговая оценка информационной безопасности ВС F_i.

На рис. 7 представлен алгоритм оценки производительности, который подробно описан в работе [1],

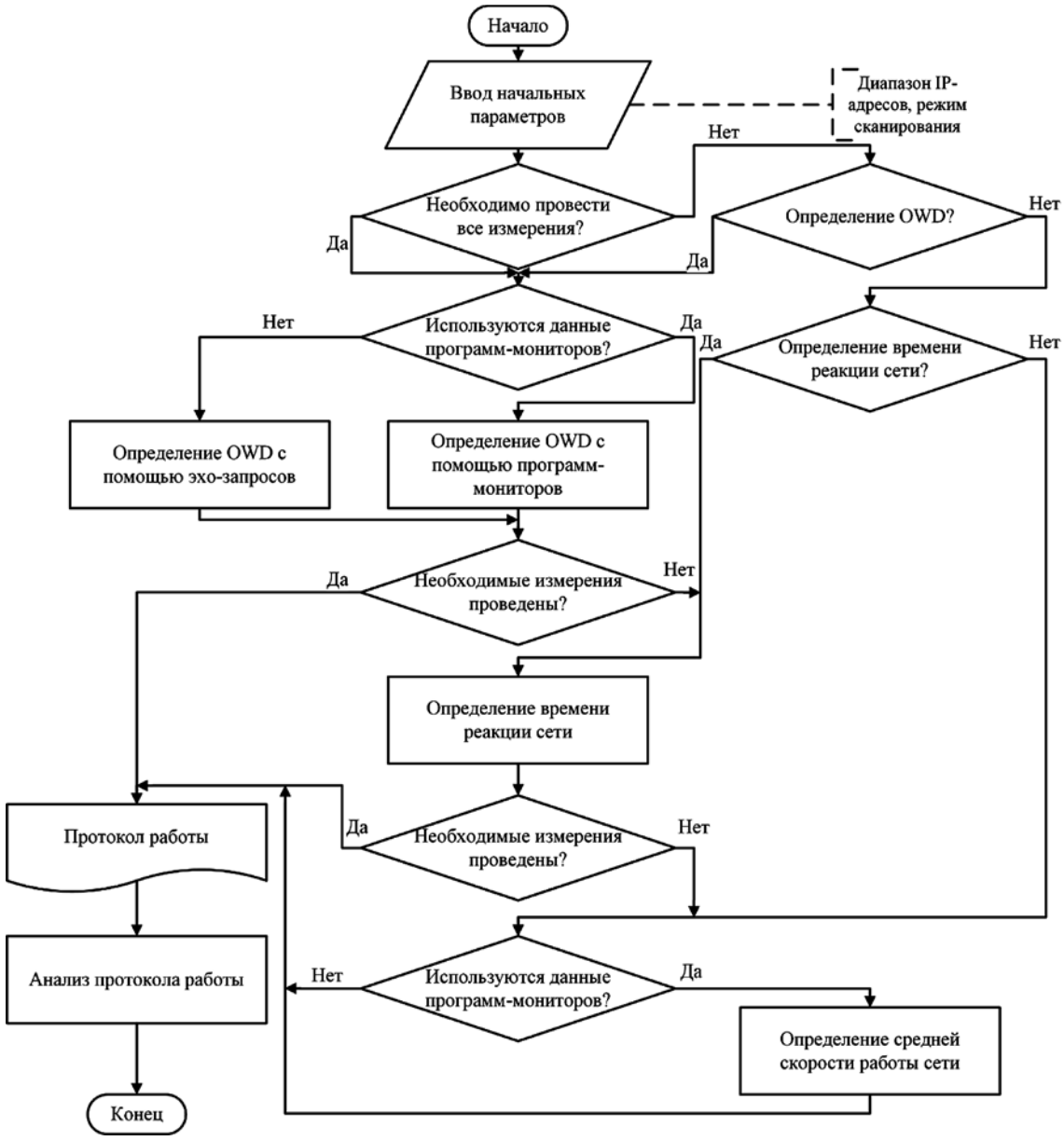


Рис. 7. Алгоритм оценки производительности ВС

где также предложен метод сбора и получения оценки надежности ВС.

Производительность и надежность ВС измеряются в соответствии со стандартами IPPM. В этих стандартах описываются каждая характеристика как случайная величина, правила получения выборок, приводятся рекомендуемые статистические оценки, которыми следует пользоваться при обработке полученной выборки значений [1].

Оценка характеристик ВС проводится по пятибалльной шкале: "низкая", "ниже среднего", "средняя", "выше среднего", "высокая".

Используемые в ПАК нечеткие модели для выдачи результата используют машину нечеткого вывода интерпретационного типа (Тонга), которая обеспечивает отображение текущего вектора входных переменных в вектор выходных переменных по следующим шагам:

- фаззификация;
- нечеткая инференция:
  - оценка ситуаций (агрегация);
  - взвешивание;
  - обобщение (аккумуляция);
- дефаззификация (используется метод MaxDot).

Использование нечетких моделей позволило также реализовать встроенный в систему программный модуль, который формирует рекомендации по улучшению оценки информационной безопасности ВС. При этом в работе предполагается, что ВС, как сложная система, может содержать ряд сегментов.

Построение программного модуля поддержки принятия решения, формирующего рекомендации по улучшению оценки безопасности

Структура ВС может включать следующие сегменты:

- административный сегмент — мониторинг и управление сетью и политикой безопасности;
- сегмент локальных рабочих мест — индивидуальные компьютеры пользователей, стационарные рабочие станции;
- мобильный сегмент рабочих мест — мобильные рабочие станции;
- коммуникационный сегмент — устройства коммуникации, коммутации и шифрования: криптошлюзы, шифрующие маршрутизаторы, коммутаторы с функциями создания виртуальных сетей (VLAN), точки беспроводного доступа, межсетевые экраны;
- внутренний сегмент серверов — серверы, доступ к которым возможен только для клиентов, находящихся внутри корпоративной сети: серверы внутреннего документооборота, серверы БД, серверы приложений, серверы печати, DHCP-серверы;
- сегмент демилитаризованной зоны — серверы, доступ к которым организован по виртуально-независимому каналу VPN для сотрудников корпорации, находящихся за ее пределами и компаний-партнеров: серверы интерактивного общения, ftp-серверы, серверы документооборота;
- открытый сегмент серверов — серверы, доступ к которым возможен с любого компьютера сети Интернет: web-серверы, mail-серверы, серверы DNS, proxy-серверы.

Для каждого из сегментов необходимо осуществить выбор и, по возможности, настройку средств информационной безопасности, необходимых для заданной категории аппаратно-программных средств. Критериями для выбора средств безопасности являются оценка безопасности ВС F_i и требования, предъявляемые к уровню защиты информации. Правила для проектирования системы информационной безопасности, ее состава и присвоения ей текущего уровня защиты описаны в базе знаний. На начальном этапе пользователь системы оценивает безопасности ВС с помощью программного комплекса и получает оценку F_i [5].

Обеспечение информационной безопасности при передаче данных между программами-мониторами и серверной частью

Обеспечение безопасности данных является одним из ключевых моментов при разработке и использовании ПАК. Комплекс имеет клиент-серверную архитектуру и централизованное хранилище данных. Для обеспечения защиты передаваемых данных между программами-мониторами и серверной частью разработан типовой протокол передачи данных, использующий ГОСТ Р 34.10—2001 и ГОСТ 28147—89. Во время первого установления сеанса связи с сервером программа-монитор запрашивает открытый ключ сервера и отправляет ему собственный открытый ключ. Сервер запрашивает разрешение на добавление новой программы-монитора у пользователя системы, в случае положительного ответа сервер высылает собственный открытый ключ. Дальнейший обмен сообщениями между сервером и программами-мониторами происходит по симметричному алгоритму, ключ для которого генерируется для сеанса случайным способом и отсылается принимающей стороне, зашифрованный открытым ключом принимающей стороны. Последовательность обмена данными отображена на рис. 8.

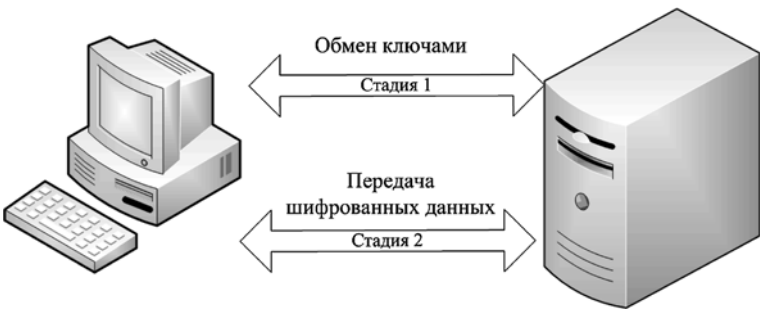


Рис. 8. Последовательность обмена данными

Здесь на стадии 1 осуществляется обмен ключами между клиентом и сервером, на стадии 2 — собственно передача шифрованных данных.

При разработке ПАК рассматривались также известные протоколы передачи данных в ВС, например, TLS. Однако их использование влечет установку на сервер и клиент большого числа прикладного ПО, необходимого для их работы, например, пакета OpenSSL (LibreSSL). В связи с этим было принято решение использования части исходного кода криптографической библиотеки Botan для интеграции в ПАК, что позволило не устанавливать дополнительные пакеты на целевой ОС. Также при дальнейшем развитии ПАК и использовании его на крупных предприятиях необходимо получение сертификатов ФСТЭК и ФСБ России, однако при использовании криптографических ключей более 56 бит для симметричных и 128 бит для асимметричных не отечественных алгоритмов, могут возникнуть трудности при сертификации ПО.

Анализ результатов тестирования ПАК анализа и оценки характеристик ВС

Тестирование ПАК оценки характеристик ВС проводилось на кафедре систем автоматизированного проектирования и управления Санкт-Петербургского государственного технологического института (технического университета) (САПРиУ СПбГТИ(ТУ)). По результатам [5] на кафедре были предприняты меры по повышению уровня информационной безопасности. Также за прошедшее время с момента написания работы [5] существенно доработан описываемый ПАК. Для нового анализа результатов тестирования было проведено сканирование. Частично данные характеристик ВС кафедры САПРиУ СПбГТИ(ТУ) и опроса пользователя системы представлены в табл. 1—4.

Для сбора данных в автоматизированном режиме в ПАК из меню "Анализ характеристик сети" выбран пункт "С автоматизированным сканированием сети". Далее в настройках сканера для минимизации нагрузки на ВС и с учетом наличия ПО для предупреждения о вторжении, был выбран режим сканирования *Sneaky* (рис. 9, см. вторую сторону обложки).

Результат сканирования ВС заносится в протокол и анализируется для получения значения векторов параметров $\mathbf{V}e_i$ (1), $\mathbf{V}r_i$ (2), $\mathbf{V}_i$ (3). Результат анализа характеристик ВС для (3) представлен на рис. 10, см. вторую сторону обложки.

На следующем этапе работы происходит формирование значения вектора параметров $\mathbf{A}_i$ из ответов пользователя системы на опросник (рис. 11).

На основе полученных значений характеристик $\mathbf{V}_i$ (3) и $\mathbf{A}_i$ формируются соответствующие оценки по данным сканирования сети Fs_i, по данным опроса пользователя системы — Fa_i. Далее по сформированным характеристикам Fs_i и Fa_i осуществляется оценка уровня информационной безопасности ВС (рис. 12).

Таблица 1

Результат сканирования ВС кафедры САПРиУ СПбГТИ(ТУ) (безопасность)

Характеристика сети	Результат сканирования
Всего открытых портов	135
Среднее число открытых портов на машину	3
Число открытых 80-х портов	7
Число открытых 443-х портов	0
Число открытых 2049-х портов	1
Число открытых 23-х портов	0
Число открытых 113-х портов	0
Число открытых 21-х портов	1
Число открытых 143-х портов	0
Число открытых 25-х портов	0
Число машин под управлением ОС MS Windows	39
Число машин под управлением ОС UNIX или UNIX-like	9
Число машин под управлением ОС, отличной от MS Windows или ОС UNIX, или UNIX-like	7
Число компьютеров в сети	55

Таблица 2

Ответы пользователя системы на опросник

Вопрос	Ответ
Имеется возможность воспользоваться постороннему человеку компьютером, установленным в офисе?	Да
Имеется в ВС демилитаризованная зона?	Нет
Используются в ВС технологии Honeypot?	Нет
На компьютерах, обрабатывающих секретные данные, имеется выход в сеть Интернет?	Да
Протокол SSH используется только на серверах?	Да
На компьютерах ВС используется антивирусное ПО?	Да
Используются межсетевые экраны?	Да
На серверах ВС используется ПО предупреждения о вторжении?	Да
Предусмотрено четкое наказание за нарушение политики безопасности?	Нет
Используется система резервного копирования?	Да

Таблица 3

Результат сканирования ВС кафедры САПРиУ СПбГТИ(ТУ) (производительность)

Характеристика сети	Результат сканирования
Односторонняя задержка пакетов M_{OWD}, мс	10
Средняя скорость работы сети M_{SIR}	Нет данных
Время реакции сети M_{TR}, мс	5

Таблица 4

Результат сканирования ВС кафедры САПРиУ СПбГТИ(ТУ) (надежность)

Характеристика сети	Результат сканирования
Доля потерянных пакетов M_L	0,2
Доступность M_A	Нет данных
Отказоустойчивость M_{FT}	Да

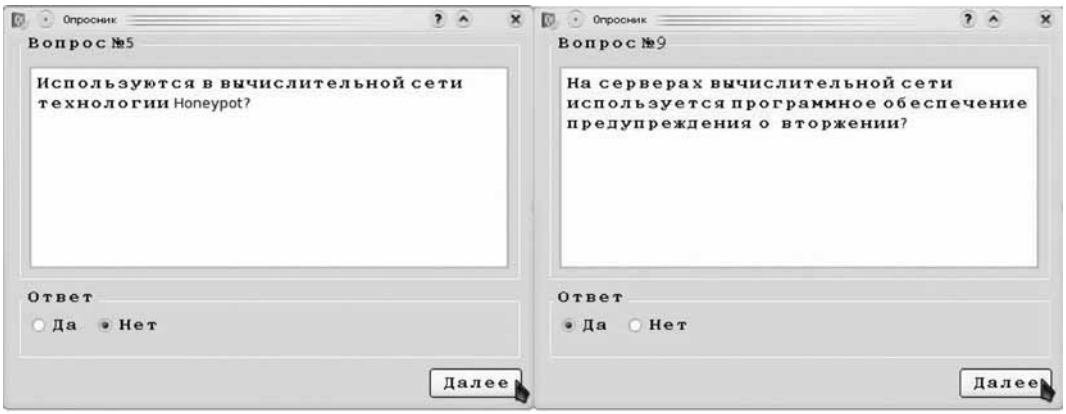


Рис. 11. Опрос пользователя системы

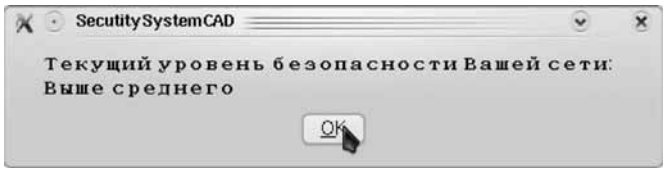


Рис. 12. Оценка уровня информационной безопасности ВС

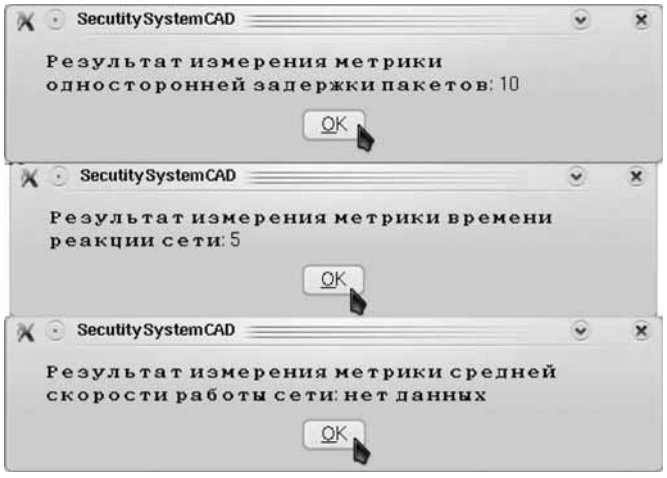


Рис. 13. Результат измерения метрик для получения оценки производительности

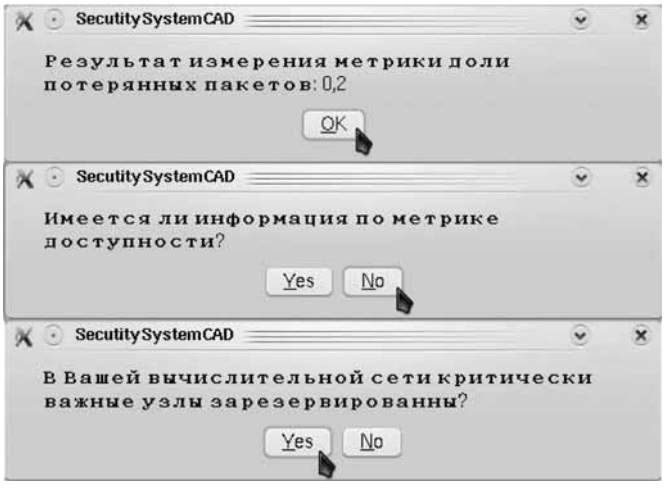


Рис. 14. Результат измерения метрик для получения оценки надежности

Далее по полученным векторам параметров $\mathbf{V}e_i$ (1) (рис. 13) и $\mathbf{V}r_i$ (2) (рис. 14) формируются оценки производительности E_i и надежности R_i (рис. 15).

На последнем этапе работы пользователь системы, если это нужно, задает уровень информационной безопасности сети, который ему необходим. На основе полученной ранее оценки уровня информа-

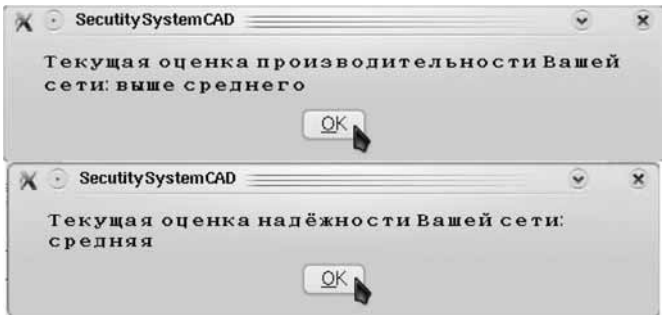


Рис. 15. Оценки производительности и надежности ВС

ционной безопасности ВС и необходимого уровня ПАК формирует рекомендации по выбору компонентов комплекса защиты и их настройке, а также различные организационные меры (рис. 16).

Кроме получения описанных выше характеристик ПАК позволяет получить комплексную оценку ВС. Она необходима для принятия решения на высшем уровне, мониторинга ситуации в целом, выявления тренда ВС. В случае отклонения значения комплексной характеристики за пределы установленного интервала (штатная ситуация) осуществляется анализ причин и мест возникновения неудовлетворительного состояния ВС. Для этого рассматриваются значения оценок нижнего уровня.

Заключение

В настоящее время множество различных по конфигурации и назначению ВС применяется для решения обширного спектра задач. Это инициирует поиск новых способов повышения эффективности их работы, причем необходимым условием является постоянное проведение процедур оценивания характеристик ВС.

При обзоре рынка программных средств анализа и оценки характеристик, ВС выявлен общий недостаток большинства известных программных средств — отсутствие комплексного решения задачи оценки таких характеристик, как производительность, надежность и информационная безопасность, а также сложность внедрения подобных систем на практике. Представленные на рынке комплексы не учитывают такую информацию, как опыт и суждение экспертов. Эти данные субъективны, однако они являются весьма существенными для оценки характеристик ВС. Кроме этого, представленные на рынке программные средства предназначены, в основном, для работы в сформированных ВС и не используются на этапе их проектирования.

Таким образом, разработанный ПАК, реализующий синтезированные нечеткие модели и алгоритмы, позволяет решить поставленную задачу. Кроме этого, предложенную программно-аппаратную систему целесообразно применять на стадии проектирования ВС, снижая экономическую нагрузку на предприятие, связанную с покупкой и использованием компонентов, не соответствующих требуемому уровню.

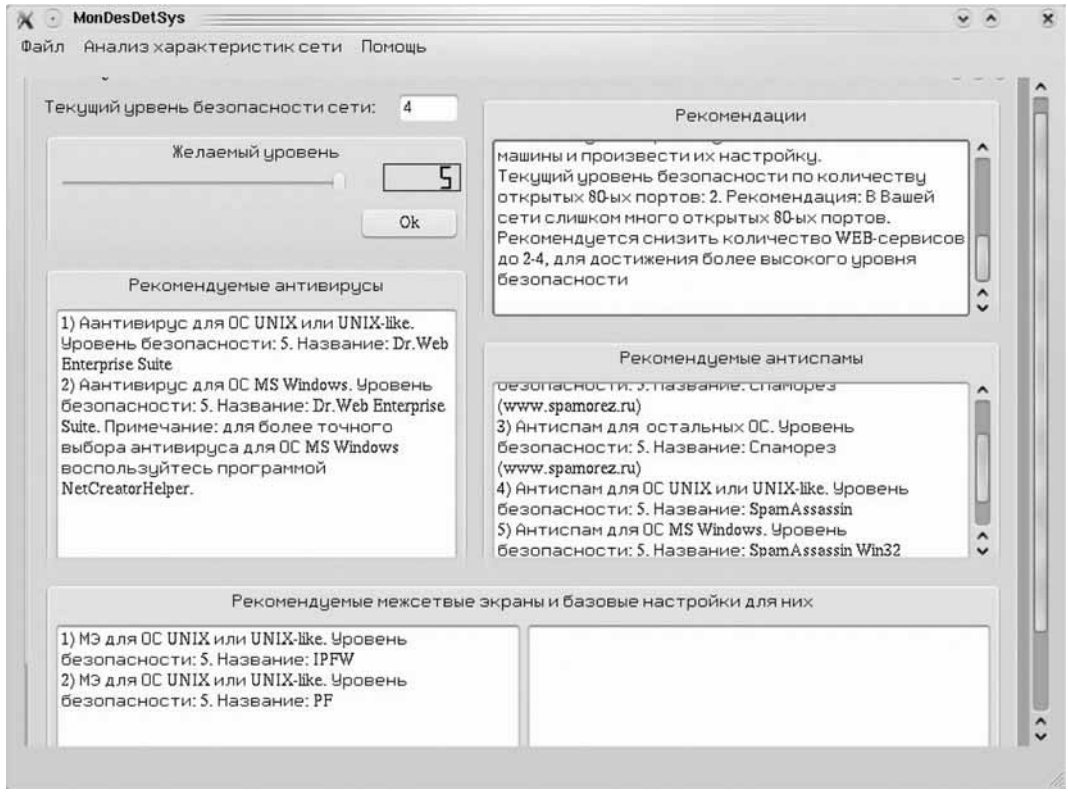


Рис. 16. Рекомендации по повышению уровня информационной безопасности

Список литературы

1. **Макарук Р. В., Гиляров В. Н.** Нечеткие модели и программный комплекс для анализа характеристик вычислительной сети // Научные ведомости БелГУ. 2013. № 22 (165), Вып. 28/1. С. 161—166.
2. **Басыня Е. А.** Разработка и исследование системы интеллектуально-адаптивного управления трафиком вычислительной сети: автореф. дис. ... канд. техн. наук (05.13.01). "Новосибирский государственный технический университет". Новосибирск, 2014. 23 с.
3. **Большаков А. А., Макарук Р. В.** Параметрический синтез информационной защиты локальных вычислительных сетей на основе нечетких моделей // Атояновские чтения: мат. круг. стола, проведенного в рамках междунар. науч.-практ. конф. "Проблемы и перспективы инновационного развития экономики". Симферополь-Алушта-Севастополь, Ноябрь 2015. Саратов: КУБиК, 2016. С. 325—332.
4. **Ажмухамедов И. М.** Динамическая нечеткая когнитивная модель влияния угроз на информационную безопасность системы // Безопасность информационных технологий. 2010. № 2. С. 68—72.
5. **Макарук Р. В., Гиляров В. Н., Новикова О. Г.** Поддержка при выборе компонент информационной защиты локальных вычислительных сетей // Известия Санкт-Петербургского государственного технологического института (технического университета). 2012. № 16 (42). С. 52—56.
6. **Cisco** visual networking index: forecast and methodology, 2013—2018. Cisco VNI. — San Jose, 2014. URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html.
7. **Digital** attack map: top daily DDoS attack worldwide. Arbor Networks. — Burlington, 2016. URL: http://www.digitalattackmap.com.
8. **2014 McAfee** report on the global cost of cybercrime. CSIS-Center for Strategic and International Studies. Washington, 2014. URL: http://csis.org/event/2014-mcafee-report-global-cost-cybercrime.
9. **Олифер В. Г., Олифер Н. А.** Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. СПб.: Питер, 2010. 943 с.
10. **Корченко А. Г.** Построение систем защиты информации на нечетких множествах. Киев: МК-пресс, 2006. 316 с.
11. **Макарук Р. В., Гиляров В. Н.** Построение функций принадлежности для определения уровня безопасности сети по результатам опроса экспертов // ММТТ-26: сб. тр. XXVI междунар. науч. конф.: в 2-х ч. Ч. 2. Ангарск, 2013. С. 42—44.
12. **Макарук Р. В., Трактанов Д. А.** Оценка текущего уровня безопасности вычислительной сети с использованием мягких вычислений // ММТТ-25, сб. тр. XXV междунар. науч. конф., г. Волгоград, 29—31 мая 2012: в 10-ти томах. Саратов, 2012. Т. 4. С. 15—18.
13. **Lippmann R. P., Cunningham R. K.** Improving intrusion detection performance using keyword selection and neural networks // Computer Networks. 2000. Vol. 34, Issue 4. P. 597—603.
14. **Ruhui Ma, Yuan Liu, Xing Lin.** Network anomaly detection based on wavelet fuzzy neural network with modified QPSO // International Journal of Distributed Sensor Networks. 2009. Vol. 5, Issue 1. P. 49.
15. **Французова Г. А., Гунько А. В., Басыня Е. А.** Самоорганизующаяся система управления трафиком вычислительной сети: метод противодействия сетевым угрозам // Программная инженерия. 2014. № 3. С. 16—20.
16. **Макарук Р. В., Большаков А. А.** Нечеткие модели и программный комплекс оценки характеристик вычислительной сети// ММТТ-28: сб. тр. XXVIII Междунар. науч. конф.: в 12-ти томах. Т. 7 / под общ. ред. А. А. Большакова. Саратов: Саратов. гос. техн. ун-т, 2015; Ярославль: Ярослав. гос. техн. ун-т; Рязань: Рязанск. гос. радиотехн. ун-т, 2015. С. 3—8.
17. **Vacca J. R.** Network and System Security: Second Edition. Elsevier Inc., 2014. 429 p.
18. **Iglesias J. A., Ledezma A., Sanchis A.** Evolving classification of UNIX users' behaviors // Evolving Systems. 2014. Vol. 5, Issue 4. P. 231—238.
19. **Reshetova E., Karhunen J., Nyman T., Asokan N.** Security of OS-level virtualization technologies // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2014. Vol. 8788. P. 77—93.
20. **National** Vulnerability Database. National Institute of Standards and Techology. USA, 2016. URL: https://nvd.nist.gov
21. **Гиляров В. Н., Токмаков А. Н.** Формализация знаний в нечетких экспертных системах// Приборы и системы. Управление, контроль, диагностика. 2001. № 9. С. 58—61.

The Software Solution for the Analysis of the Characteristics and Evaluation of Information Security Level of a Computer Network based on Fuzzy Models

A. A. Bolshakov, aabolshakov57@gmail.com, **R. V. Makaruk**, e-mail: n.diablo.n.f@list.ru, Saint-Petersburg State Technological Institute (Technical University), 190013, Saint Petersburg, Russian Federation

Corresponding author:

Bolshakov Aleksandr A., Professor, Saint-Petersburg State Technological Institute (Technical University), 190013, Saint Petersburg, Russian Federation, e-mail: aabolshakov57@gmail.com

Received on February 29, 2016
Accepted on March 28, 2016

Fuzzy models with structure and coefficients of membership functions and algorithms for analysis of characteristics and estimating the information security level of computer networks are proposed. Their specific characteristic is possibility to include objective and subjective data in form of experts' judgements, knowledge and experience. Suggested method also allows providing an analysis of characteristics and information security level estimation in the conditions of incomplete data. The developed algorithmical and software solution implementing mentioned fuzzy models and algorithms are also reviewed.

Keywords: *fuzzy model, performance, reliability, security, computer networks, software and algorithmic complex*

For citation:

Bolshakov A. A., Makaruk R. V. The Software Solution for the Analysis of the Characteristics and Evaluation of Information Security Level of a Computer Network based on Fuzzy Models, *Programmnaya Ingeneria*, 2016, vol. 7, no 6, pp. 268—282.

DOI: 10.17587/prin.7.268-282

References

1. **Makaruk R. V., Giljarov V. N.** Nechjotkie modeli i programmnyj kompleks dlja analiza harakteristik vychislitel'noj seti (Fuzzy models and software package for analysis characterictics of computer network). *Nauchnye Vedomosti BelGU,* 2013, no. 22 (165), issue 28/1, pp. 161—166 (in Russian).

2. **Basinya E. A.** Razrabotka i issledovanie sistemy intellektual'no-adaptivnogo upravlenija trafikom vychislitel'noj seti, Extended abstract of PhD dissertation (Engineering). Novosibirsk State Technical University, Novosibirsk, 2014 (in Russian).

3. **Bolshakov A. A., Makaruk R. V.** Parametricheskij sintez informacionnoj zashhity lokal'nyh vychislitel'nyh setej na osnove nechjotkih modelej (Parametric Synthesis of Information Security of Local Area Networks Based on Fuzzy Models), *Atojanovskie chtenija: mat. krug. stola, provedjonnogo v ramkah mezhdunar. nauchno-prakticheskoj konf. "Problemy i perspektivy innovacionnogo razvitija jekonomiki"*, Simferopol'-Alushta-Sevastopol', November 2015, Saratov, Izd-vo KUBiK, 2016, pp. 325—332 (in Russian).

4. **Azhmukhamedov I. M.** Dinamicheskaya nechetkaya kognitivnaya model' vliyaniya ugroz na informatsionnuyu bezopasnost' sistemy (Dynamic Model of the Impact of Threats to Information Security System), *Bezopasnost' Informatsionnykh Tekhnologiy*, 2010, no. 2, pp. 68—72 (in Russian).

5. **Makaruk R. V., Giljarov V. N., Novikova O. G.** Podderzhka pri vybore komponent informacionnoj zashhity lokal'nyh vychislitel'nyh setej (Support at the choice the component of information protection of LAN), *Izvestija Sankt-Peterburgskogo Gosudarstvennogo Tehnologicheskogo Instituta (Tehnicheskogo Universiteta),* 2012, no. 16 (42), pp. 52—56 (in Russian).

6. **Cisco** visual networking index: forecast and methodology, 2013—2018. Cisco VNI. San Jose, 2014, available at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

7. **Digital** attack map: top daily DDoS attack worldwide. Arbor Networks. Burlington, 2016, available at: http://www.digitalattackmap.com

8. **2014** McAfee report on the global cost of cybercrime. CSIS-Center for Strategic and International Studies. Washington, 2014, available at: http://csis.org/event/2014-mcafee-report-global-cost-cybercrime

9. **Olifer V. G., Olifer N. A.** *Komp'juternye seti. Principy, tehnologii, protokoly* (Computer network. Principles, technologies, protocols), 4th edition. Saint Petersburg, Piter, 2010, 943 p. (in Russian).

10. **Korchenko A. G.** *Postroenie sistem zashhity informacii na nechjotkih mnozhestvah* (Construction of information protection systems on fuzzy sets), Kiev, MK-press, 2006, 316 p. (in Russian).

11. **Makaruk R. V., Giljarov V. N.** Postroenie funkcij prinadlezhnosti dlja opredelenija urovnja bezopasnosti seti po rezul'tatam oprosa jekspertov (The construction of membership functions for determining network security level according to the results of a survey of experts), *MMTT-26: sb. tr. XXVI mezhdunar. nauch. konf.*, part 2, Angarsk, 2013, pp. 42—44 (in Russian).

12. **Makaruk R. V., Traktanov D. A.** Ocenka tekushhego urovnja bezopasnosti vychislitel'noj seti s ispol'zovaniem mjagkih vychislenij (Assessment of the current security level of computer networks using soft computing), *MMTT-25: sb. tr. XXV mezhdunar. nauch. konf.,* Volgograd, 29—31 May 2012, Saratov, 2012, vol. 4, pp. 15—18 (in Russian).

13. **Lippmann R. P., Cunningham R. K.** Improving intrusion detection performance using keyword selection and neural networks, *Computer Networks,* 2000, vol. 34, iss. 4, pp. 597—603.

14. **Ruhui Ma, Yuan Liu, Xing Lin.** Network anomaly detection based on wavelet fuzzy neural network with modified QPSO, *International Journal of Distributed Sensor Networks*, 2009, vol. 5, issue 1, pp. 49.

15. **Frantsuzova G. A., Gunko A. V., Basinya E. A.** Samoorganizujushhajasja sistema upravlenija trafikom vychislitel'noj seti: metod protivodejstvija setevym ugrozam (Self-Organizing Control System of Computer Network Traffic: Method of Counteraction from Network Threats), *Programmnaya Ingeneria*, 2014, no. 3, pp. 16—20 (in Russian).

16. **Makaruk R. V., Bolshakov A. A.** Nechjotkie modeli i programmnyj kompleks ocenki harakteristik vychislitel'noj seti (Fuzzy models and software solution for an assessment of a computer network), *MMTT-28: sb. tr. XXVIII Mezhdunar. nauch. konf.* Vol. 7. / Ed. by A. A. Bolshakov. Saratov, Saratov. gos. tehn. un-t, 2015, Jaroslavl', Jaroslav. gos. tehn. un-t; Rjazan', Rjazansk. gos. radiotehn. un-t, 2015, pp. 3—8 (in Russian).

17. **Vacca J. R.** *Network and System Security*: Second Edition. Elsevier Inc., 2014. 429 p.

18. **Iglesias J. A., Ledezma A., Sanchis A.** Evolving classification of UNIX users' behaviors, *Evolving Systems,* 2014, vol. 5, issue 4, pp. 231—238.

19. **Reshetova E., Karhunen J., Nyman T., Asokan N.** Security of OS-level virtualization technologies, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8788, pp. 77—93.

20. **National** Vulnerability Database. National Institute of Standards and Techology. USA, 2016, available at: https://nvd.nist.gov

21. **Giljarov V. N., Tokmakov A. N.** Formalizacija znanij v nechjotkih jekspertnyh sistemah (Formalization of fuzzy knowledge in expert systems), *Pribory i Sistemy. Upravlenie, Kontrol', Diagnostika*, 2001, no. 9, pp. 58—61 (in Russian).

УДК 65.011.56 DOI: 10.17587/prin.7.283-288

О. М. Полевая, аспирант, e-mail: o.m.balakhonova@gmail.com, Российский университет дружбы народов, г. Москва

Архитектура корпоративной информационной системы стратегического управления

Представлены результаты анализа архитектуры системы управления эффективностью бизнеса и на их основе предложена архитектура корпоративной информационной системы стратегического управления. Предложены концептуальные решения по построению информационных систем, интегрирующих данные из различных источников, а также систем управления и систем поддержки принятия решений.

Ключевые слова: *системы управления эффективностью бизнеса, CPM-системы, архитектура информационных систем, стратегическое управление*

Введение

Основной целью корпоративной информационной системы стратегического управления (КИС СУ) является поддержка процессов разработки и реализация решений, касающихся развития компании в целом в долгосрочной перспективе. К ним относятся долгосрочные цели компании и стратегические решения и действия по их достижению. Такая система должна поддерживать полный цикл стратегического управления [1], обеспечивающий решение задач на перечисленных далее этапах его реализации.

- Поддержка процесса анализа внутренней и внешней сред компании.
- Помощь пользователю при определении единой цели развития компании, при формировании образа и системы ценностей компании, видения и миссии компании.
- Поддержка процесса оценки вариантов стратегических целей и стратегий их достижения.
- Поддержка процессов прогнозирования последствий при реализации стратегии и оценки рисков, возникающих при реализации стратегии.
- Поддержка процесса принятия решения о выборе стратегии.
- Поддержка процесса планирования реализации стратегии.
- Поддержка процесса контроля исполнения стратегии.

Описанные выше этапы реализации процесса стратегического управления являются довольно сложными. Они состоят из множества других процессов, каждый из которых подразумевает анализ большого объема информации. Результаты анализа предыдущего процесса могут быть при этом использованы в задачах текущего процесса.

Обязательным требованием к КИС СУ является возможность ее интеграции с системами поддержки оперативного контура управления, так как они являются основным источником для анализа текущей обстановки. Под оперативным контуром управления подразумевается текущая деятельность компании, обеспечивающая выполнение уже существующих бизнес-процессов и не затрагивающая перспектив развития компании в будущем.

Системы стратегического управления должны уметь анализировать данные, поступающие из различных открытых источников, в частности, из сети Интернет. К таким данным относятся информация о внешнем окружении компании (информация о конкурентах, существующих и потенциальных клиентах, о поставщиках и акционерах, о законодательстве, нормативных актах и т. д.). Эта информация оказывает существенное влияние на принятие решения при выборе стратегии [2].

Корпоративную информационную систему стратегического управления можно отнести к классу CPM-систем (*corporate performance management*) [3, 4] — систем управления эффективностью бизнеса. С точки зрения системной архитектуры, эти системы являются корпоративными информационно-аналитическими системами, обладающими специфической функциональностью поддержки ключевых процессов управления. К числу таких процессов относятся финансовые процессы, процессы оперативного планирования, консолидации и отчетности, моделирования, анализа и мониторинга ключевых показателей эффективности бизнеса. Особенностью CPM-систем, как и большинства корпоративных систем, является присутствие механизмов интеграции различных компонентов (модулей) системы. Это обеспечивает поддержку единого управленческого цикла "стратегия — ее воплощение — контроль — корректирующие изменения".

Архитектура CPM-систем

Как класс программного обеспечения CPM-системы появились в 1999—2000 гг. (согласно данным международной организации The Data Warehousing Institute), однако на настоящее время отсутствуют стандарты построения подобных систем. Проведенный анализ [2] различных CPM-систем позволил выявить типовые особенности их архитектуры, основанные на концепции хранилища данных. В общем виде архитектура системы управления эффективно-

стью бизнеса описывается схемой со следующими выделенными слоями (рис. 1):

1) извлечение, преобразование, консолидация и загрузка данных;
2) хранение данных;
3) обработка данных;
4) представление данных.

Технологические процессы функционирования CPM-систем состоят в следующем. Данные извлекают из различных внешних и внутренних источников данных в соответствии с заранее определенными формами и регламентами. Затем данные проверяют на соответствие форматов, согласованность и полноту, трансформируют по определенным правилам (например, может выполняться расчет показателей) и помещают в хранилище или витрины данных. После этого пользователи имеют возможность:

- ♦ запустить процессы обработки данных (например, процесс анализа по ключевым показателям) на основе модели, принятой в CPM-системе;
- ♦ получить необходимую им информацию для построения различных табличных и графических ее представлений, прогнозирования, моделирования и выполнения других аналитических задач.

Если рассматривать КИС СУ, то подобная архитектура имеет ряд перечисленных далее существенных недостатков.

- Источником данных могут выступать неструктурированные данные из различных сайтов, содержащие информацию в различных форматах, включая текст, аудио, видео, геоинформационные данные и др. В качестве хранилищ данных в настоящее время широко используют реляционные базы данных (БД), неэффективно работающие с нетиповыми для них видами данных. Как следствие, появляется необходимость в решении задач по извлечению только необходимой в системе информации и ее преобразованию в типовые для реляционной БД форматы (числовые и символьные).
- Источником данных могут выступать облачные решения. Как правило, обновление информации в об-

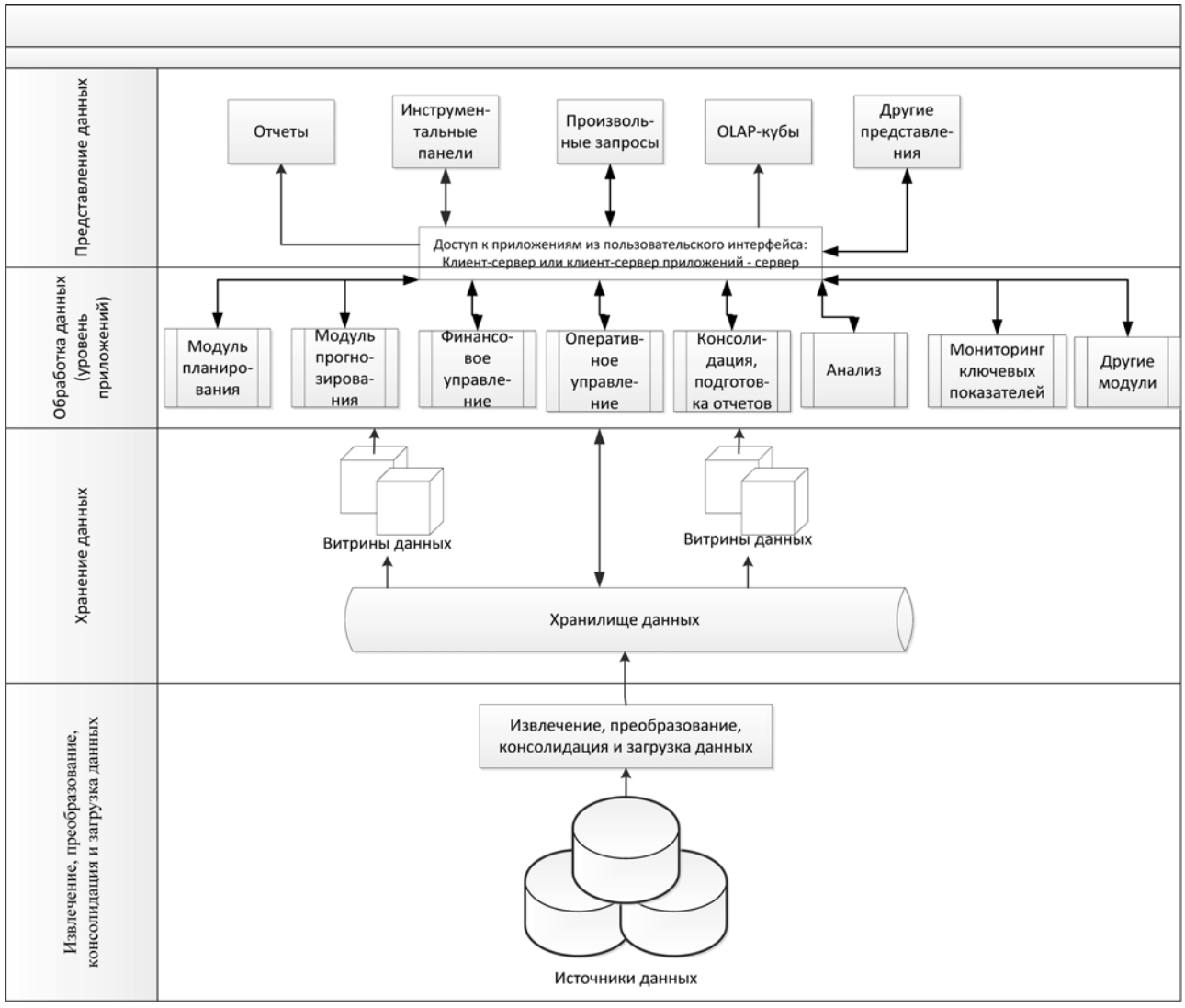


Рис. 1. Архитектура CPM-систем

лачных системах происходит относительно часто, что приводит к изменениям в метаинформации, которая используется на этапе преобразования и загрузки данных в хранилище. Несвоевременное обновление метаинформации в системах интеграции приводит к тому, что такая информация не загружается в систему.

- После процедур извлечения и преобразования в хранилище данных загружают проверенные, согласованные и хронологически целостные данные. При этом на осуществление этих процедур тратится некоторое время, что приводит к задержке во времени между загруженной информацией и реальной. Для осуществления стратегического мониторинга в условиях высокодинамичной среды (например, в периоды кризиса) необходимо сократить до минимума этот временной промежуток или уметь предугадывать изменение состояния среды.
- В архитектуре отсутствует обратная связь с уровня анализа данных к источникам данных, т. е. система является "пассивным" наблюдателем. Стратегическое управление предполагает не только сбор и обработку информации для проведения анализа (входящий поток информации), но и обратное воздействие (исходящий поток информации, управленческие воздействия на внутреннюю среду компании).
- В основе КИС СУ лежит описание предприятия в виде различных его моделей: процессной, организационной, ресурсной, целевой и др. В архитектуре CPM-систем нет места для хранения и использования этих моделей. Однако системы управления эффективностью бизнеса предусматривают настройку своей модели под нужды конкретного предприятия.
- Для эффективного управления предприятием необходимо не только обрабатывать явные данные, полученные из источников, но и уметь извлекать знания из загруженной в хранилище информации.
- Работа с КИС СУ предполагает привлечение экспертов. Знания, полученные от них, также должны сохраняться в системе.
- Процесс стратегического управления предполагает принятие решений. В представленной на рис. 1 архитектуре отсутствуют классические элементы системы поддержки принятия решений, включающей блок анализа данных по проблемным вопросам и блок принятия решений.
- Функциональные модули CPM-систем могут существовать для решения отдельных задач стратегического управления. Как правило, в компаниях присутствует несколько решений от различных поставщиков. Для того чтобы обеспечить единый процесс стратегического управления, необходимо иметь двусторонний доступ к этим компонентам. Для реализации такой связи можно использовать единый интерфейс для обмена данными (шина данных).

Архитектура корпоративной информационной системы стратегического управления

С учетом изложенного выше, автором предлагается архитектура КИС СУ, представленная на рис. 2, см. третью сторону обложки.

В предложенной архитектуре выделяют перечисленные далее слои (уровни).

- Источники данных: внешние и внутренние источники данных. К внешним источникам относятся геослужбы, системы экологического мониторинга в регионе, биржевые службы, законодательная база, Интернет-ресурсы и др. К внутренним источникам относятся системы классов ERP (*Enterprise Resource Planning*, управление ресурсами предприятия), CRM (*Customer Relationship Management*, управление отношениями с клиентами), МТО (материально-техническое обеспечение), системы документооборота, системы взаимодействия со стейкхолдерами, системы вовлечения [5] и др.
- Поиск и извлечение данных из источников данных. Преобразование, интеграция и загрузка извлеченных данных. Данный уровень представлен ETL-средствами (Extract, Transform, Load), ECM-средствами (*Enterprise Content Management*, управление корпоративным контентом), CIS-средствами (*Continuous Integration System*, системы непрерывной интеграции), шлюзами B2B, инструментальными средствами облачной интеграции и другими средствами, объединенными на основе платформы гибридной интеграции.
- Хранение данных, загруженных из источников данных. Данный уровень содержит хранилище и витрины данных.
- Управление знаниями, извлеченными из хранилища данных или полученными в результате работы пользователя с системой. Данный уровень содержит средства для извлечения знаний, а именно — систему глубинного анализа данных (*Data Mining*), а также — хранения знаний в виде базы знаний. Система глубинного анализа данных служит для превращения информации в знание о тенденциях и закономерностях. Она позволяет анализировать скрытые закономерности на основе гипотез или с помощью статистических процедур.
- Управление моделями. Данный уровень содержит средства для создания и модификации моделей предприятия, а также базу моделей для их хранения.
- Обработка данных, знаний (уровень приложений). Содержит различные функциональные приложения, интегрированные между собой с помощью ESB-шины (*Enterprise Service Bus*, сервисная шина предприятия).
- Поддержка пользователей. Данный уровень представлен средствами генерации и управления диалогом, средствами безопасности, блоком анализа проблем и блоком поддержки принятия решений.
- Представление информации: отчеты, панели, запросы, диалоги, пользовательские представления и т. д.
- Взаимодействие с другими системами на основе ESB-шины.

Опишем технологические процессы функционирования КИС СУ. Источники данных для системы можно разделить на два типа: внешние, содержащие информацию об окружающей микро- и макросреде компании, и внутренние, содержащие информацию

о внутренней среде компании. Информация о внешней среде, как правило, является слабоструктурированной, разноформатной и нерегулярной, основным источником информации являются открытые внешние системы (например, как было отмечено выше, геослужбы, системы экологического мониторинга в регионе, биржевые службы, законодательная база и т. п.) и, в частности, сеть Интернет. Для поиска информации в сети Интернет необходим специальный программный модуль — система мониторинга сети. Он является неотъемлемой надстройкой над источниками данных, относящимся к подлежащим анализу интернет-ресурсам.

Информация о внутренней среде компании может поступать из различных внутренних информационных систем — локальных, облачных, мобильных (количественная информация), а также в виде структурированных отчетов, подготовленных экспертами (неколичественная информация) или документов. Для такого рода информации заранее известны форматы и методологии ее передачи для использования в КИС СУ.

Информация, необходимая для функционирования КИС СУ, поступает из различных источников, поэтому она может содержать дублирующие, некорректные или несогласованные данные. Кроме того, в источниках содержится первичная информация, а при стратегическом управлении, как правило, используют уже агрегированные данные. Поэтому важно загружать в систему интегрированную и агрегированную информацию.

Поскольку ИТ-ландшафт компании характеризуется как гибридный, то необходимо иметь возможность объединять как локальные, так и облачные и мобильные системы. Сложность решения такой задачи состоит в том, что скорость обновления метаданных в облачных системах не зависит от ИТ-служб компании и происходит достаточно часто. В 2013 г. аналитики Forrester ввели понятие гибридной интеграции. Ключевыми возможностями платформы для такой интеграции являются управление жизненным циклом метаданных и интероперабельность в реальном времени [5]. Платформа гибридной интеграции поддерживает разнообразные средства интеграции: ESB-шины; ETL-средства; ECM-средства; CIS-средства; шлюзы B2B; механизмы облачной интеграции, такие как CBI (*Cloud-Based Integration*, сервисы интеграции на базе облака) и iPaaS-средства (*Integration-Centric Platform-as-a-Service*).

Для интеграции данных предлагается использовать технологию "Хаб и спицы" [6, 7]. Она состоит в организации маршрутов взаимодействия интегрируемых систем через центральный узел — хаб. В качестве хаба в корпоративной системе стратегического управления будет выступать платформа гибридной интеграции.

Технология интеграции данных "Хаб и спицы" представляет ряд дополнительных возможностей для КИС СУ. Во-первых, топология не зависит от физической архитектуры информационной системы, а определяет логические маршруты взаимодействия и передачи данных между интегрированными системами. Во-вторых, благодаря введению промежуточного звена, с применением такой технологии уменьшается число связей между приложениями, устраняются прямые связи, а система интеграции становится более гибкой и дешевой в эксплуатации. Если меняется одно из интегрированных приложений, то нужно модифицировать только одну связь, а именно — между данным приложением и хабом. В-третьих, хаб позволяет обрабатывать большие объемы данных, что является ключевым требованием к КИС СУ.

Интегрированные данные загружаются в хранилище или витрины данных КИС СУ. При стратегическом управлении важно оперировать не только полученной и агрегированной информацией, но и выявлять скрытые закономерности и тенденции. Для этого служит блок извлечения знаний, представленный системами глубинного анализа данных (*Data Mining*) [8]. Он позволяет анализировать скрытые закономерности в хранилище данных на основе гипотез или с помощью статистических процедур. Выявленные закономерности являются знанием об управляемой системе и сохраняются в базу знаний.

Модели предприятия (организационная, функциональная, процессная, ролевая, целевая и т. д.) хранятся в базе моделей. Модели создаются и изменяются в самой системе.

Хранилище данных, база знаний и база моделей являются основными объектами КИС СУ. Обработка данных, знаний и моделей происходит на следующем уровне — уровне приложений. На данном уровне содержатся приложения для поддержки процессов стратегического управления: стратегического мониторинга; стратегического анализа; принятия стратегических решений; стратегического планирования; реализации стратегии и контроля ее исполнения. Для обеспечения непрерывного процесса стратегического управления необходимо интегрировать эти приложения между собой. В качестве технологии интеграции приложений предлагается использовать ESB-шину [7], поскольку число бизнес-процессов для взаимодействия между приложениями ограничено, а объем передаваемой информации невелик.

Для того чтобы система могла представлять данные пользователю в удобном виде, создается пользовательский интерфейс, обеспечивающий всевозможные представления данных: сводные отчеты; витрины; графики; инструментальные панели руководителя. Для подготовки представлений данных используются дополнительные средства (уровень поддержки пользователей). Средства генерации и управления диалогом позволяют управлять разнообразными стилями ведения диалога для различных пользователей, обеспечивая гибкую поддержку пользователя. Средства безопасности позволяют настраивать доступ к набору функций и данных для различных пользователей. Блок анализа проблем и блок поддержки принятия решений позволяют организовать взаимодействие пользователя и системы в процессе принятия стратегических решений. Эти блоки включают в себя процедуры и методы, позволяющие сформулировать поставленный вопрос, с помощью баз знаний, моделей и данных проанализировать возможности его решения и получить результат. При этом знания экспертов, полученные

в процессе принятия решений, сохраняются в базу знаний.

В результате стратегического управления появляются стратегические артефакты, тактические артефакты и отчетность. К стратегическим артефактам относятся долгосрочные цели компании и стратегические решения и действия по их достижению. К тактическим артефактам относятся: дерево целей; дорожные карты; система показателей; информационные модели уровня компании (процессные, ролевые, организационные и др.); программы; инициативы; календарные планы; бюджеты; внутренние политики и методики и т. п. [1]. Публичная и внутренняя отчетности являются косвенным продуктом корпоративной системы стратегического управления и используются во внутренних процессах стратегического управления. Такие виды отчетности также могут быть использованы для других целей, например, при составлении публичной отчетности по международным стандартам финансовой отчетности (МСФО).

Заключение

Стратегическое управление является верхним уровнем в процессной модели предприятия. Оно управляет всеми другими процессами компании, используя информацию из различных внешних и внутренних источников. Корпоративная информационная система стратегического управления призвана поддержать этот сложный процесс. Требования к данной системе высоки. Она должна обрабатывать информацию из различных разнородных источников; уметь находить скрытые закономерности, выявляя, тем самым, скрытые возможности и угрозы; поддерживать процессы стратегического мониторинга, анализа, принятия решений, формализации стратегии, планирования, прогнозирования, бюджетирования, реализации стратегии и контроля ее исполнения, а также предоставлять гибкий интерфейс для эффективной работы различных пользователей.

Предлагаемая автором архитектура информационной системы предполагает устранение недостатков, присущих архитектуре существующих CPM-систем. Она позволяет обрабатывать нетипичные данные из различных источников, в том числе облачных; использовать знания, полученные при обработке источников данных и при работе экспертов; поддерживать принятие решений на основе моделей, в том числе моделей принятия решений конкретным пользователем. Компонентная архитектура уровня приложений, интегрированная на основе ESB-шины, позволяет использовать различные функциональные модули от различных поставщиков, а средства управления диалогом позволяют эффективно работать с системой различным пользователям. Кроме того, КИС СУ является активной системой, а не пассивным наблюдателем, что позволяет ей управлять деятельностью компании.

Список литературы

1. **Ансофф И.** Стратегическое управление. СПб.: Питер, 2009. 344 с.
2. **Балахонова О. М.** Обзор информационных систем для решения задач стратегического менеджмента // Экономика, статистика, информатика. Вестник УМО. 2015. № 5. С. 154—159.
3. **Iervolino C., Van Decker J. E., Chandler N.** Magic Quadrant for Corporate Performance Management Suites. Gartner [Official website]. Art. No.:G00238924, published 14.02.2013. URL: https://www.gartner.com/doc/2686317/magic-quadrant-corporate-performance-management/ (дата обращения 25.02.2015).
4. **Bange C., Marr B., Bange A.** Performance Management — Current Challenges and Future Directions. Advanced Performance Institute [Official website]. A global survey of the maturity of Performance Management processes. By BARC, 2009. URL: http://www.ap-institute.com/media/14133/pm_processes.pdf (дата обращения 25.02.2015).
5. **Дубова Н.** Гибридная интеграция // Открытые системы. СУБД. 2014. № 8. URL http://www.osp.ru/os/2014/08/13043484/ (дата обращения 25.02.2015).
6. **Russom P.** Data Integration Architecture: What It Does, Where It's Going, and Why You Should Care. TDWI. Source for In-Depth Education and Research on All Things Data. URL https://tdwi.org/articles/2008/05/27/data-integration-architecture-what-it-does-where-its-going-and-why-you-should-care.aspx?m=1 (дата обращения 25.02.2015).
7. **Добровольский А.** Интеграция приложений: методы взаимодействия, топология, инструменты // Открытые системы. СУБД. 2006. № 9. URL: http://www.osp.ru/os/2006/09/3776464/ (дата обращения 25.02.2015).
8. **Шмаков А.** Глубинный анализ данных в режиме реального времени: Oracle Real Time Decisions. URL: http://citforum.ru/database/oracle/ortd/ (дата обращения 25.02.2015).

Architecture of Information System for Strategic Management

O. M. Polevaya, o.m.balakhonova@gmail.com, Russian Peoples Friendship University, Moscow, 117198, Russian Federation

Corresponding author:

Polevaya Olga M., Postgraduate Student, Russian Peoples Friendship University, Moscow, 117198, Russian Federation

Received on March 17, 2016
Accepted on March 29, 2016

Strategic management is one of the main business processes in a company. It consists of numerous sub-processes: strategic analysis, strategic planning, deciding on strategic aims and ways for their fulfillment, key performance indicator monitoring, etc. These processes need information that may be extracted from internal and external information systems. Therefore, when we speak about corporate information system for strategic enterprise management (CIS SEM) we should

think about it as of integrated information system with specific functionality. The closest to CIS SEM class of information systems is a corporate performance management system (CPM-system). The article describes results of corporate performance management system's architecture analysis. It describes disadvantages of typical CPM-system's architecture concerning information system for strategic management. First, there is no component for operating with non-typical data such as audio, video streams, geoinformation, etc. Second, there is no component for data integration supporting different data sources, including cloud sources. Third, CPM systems don't use knowledge, but strategic management is considered to exploit latent tendencies and expert knowledge. Forth, information system for strategic management should support users during their decision-making, but CPM systems provide only nonspecific user interface.

The article describes the architecture of information system for strategic management developed according to the analysis results and eliminated CPM-system architecture disadvantages. Proposed architecture consist of some layers. Data sources integrated using "hab and spokes" technology, where the hab is a hybrid integration platform. Information from Internet-sources extracts with a special component — Internet monitoring system (a sort of a crawler). Integrated data is stored in a data warehouse. Data mining tools extract knowledge from stored data and collect it in a knowledge warehouse. Models that are used in strategic management processes store in model warehouse. These three warehouses are the core of the system.

Application layer consists of different functional modules integrated with enterprise service bus (ESB). Component architecture of this layer allows one to use functional modules developed by different distributors. Control actions on internal information systems realizes by service bus.

There are other conceptual proposals for developing IT systems that integrate data from various sources as well as management systems and decision-making systems in the article.

Keywords: *corporate performance management system, CPM system, information system architecture; strategic management*

For citation:

Polevaya O. M. Architecture of information system for strategic management, *Programmnaya Ingeneria*, 2016, vol. 7, no. 6, pp. 284—288.

DOI: 10.17587/prin.7.283-288

References

1. **Ansoff I.** *Strategicheskoe upravlenie* (Strategic management). Saint Petersburg, Piter, 2009, 344 p. (in Russian).

2. **Balakhonova O. M.** Obzor informacionnyh sistem dlja reshenija zadach strategicheskogo menedzhmenta (Review of information systems for strategic management), *Economic, Statistic and Informatics. Vestnik UMO*, 2015, no. 5, pp. 154—159 (in Russian).

3. **Iervolino C., Van Decker J. E., Chandler N**. *Magic Quadrant for Corporate Performance Management Suites.* Gartner [Official website]. Art. No.:G00238924, published 14.02.2013, available at: https://www.gartner.com/doc/2686317/magic-quadrant-corporate-performance-management/

4. **Bange C., Marr B., Bange A.** *Performance Management — Current Challenges and Future Directions.* Advanced Performance Institute. A global survey of the maturity of Performance Management processes. By BARC, 2009, available at: http://www.ap-institute.com/media/14133/pm_processes.pdf

5. **Dubova N.** Hybrid integration (Gibridnaja integracija), *Otkrytye sistemy. SUBD,* 2014, no. 8, available at: http://www.osp.ru/os/2014/08/13043484/ (in Russian).

6. **Russom P.** *Data Integration Architecture: What It Does, Where It's Going, and Why You Should Care*, TDWI, Source for In-Depth Education and Research on All Things Data, available at: https://tdwi.org/articles/2008/05/27/data-integration-architecture-what-it-does-where-its-going-and-why-you-should-care.aspx?m=1

7. **Dobrovolskiy A.** Integracija prilozhenij: metody vzaimodejstvija, topologija, instrumenty (Application integration: collaboration methods, topology, tools), *Otkrytye sistemy. SUBD*, 2006, no. 9, available at: http://www.osp.ru/os/2006/09/3776464/ (in Russian).

8. **Shmakov A.** *Glubinnyj analiz dannyh v rezhime real'nogo vremeni: Oracle Real Time Decisions* (Deep real time data mining: Oracle Real Time Decisions), available at: http://citforum.ru/database/oracle/ortd/ (in Russian).

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева.* Корректор *Е. В. Комиссарова*

Сдано в набор 05.04.2016 г. Подписано в печать 18.05.2016 г. Формат 60×88 1/8. Заказ PI616
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**