ISSN 2220-3397

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн
6
2020
Том 11

Рисунок к статье **О. А. Змеева, Д. О. Змеева, А. Н. Даниленко**

«ПЕРЕНОС ПРАКТИК ESSENCE В СРЕДУ AZURE DEVOPS SERVER»

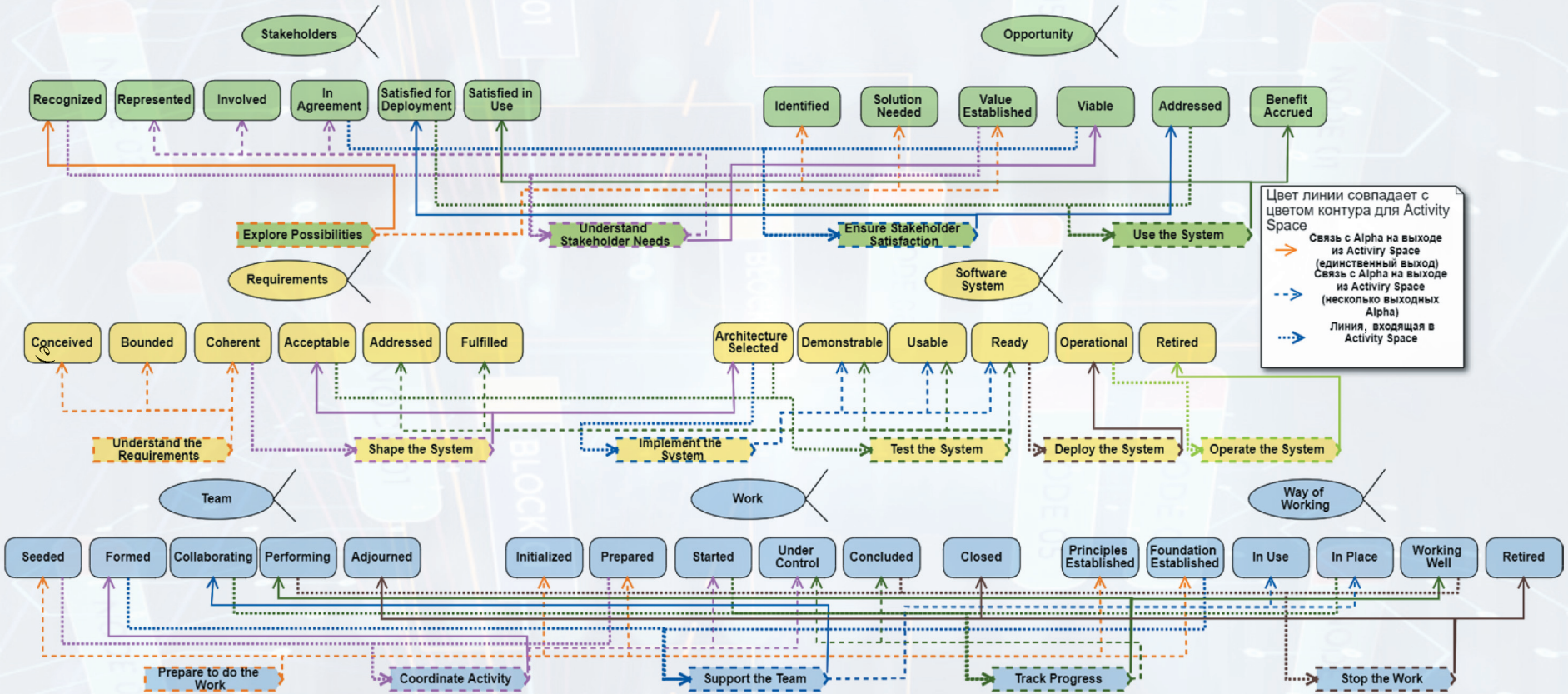

Рис. 3. Теоретический контур Essence

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

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

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

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 (индекс по Объединенному каталогу "Пресса России" — **22765**) 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 и базу данных RSCI на платформе Web of Science.
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.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 11
N 6
2020

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

CONTENTS

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

УДК 004.04 DOI: 10.17587/prin.11.311-321

О. А. Змеев, д-р физ.-мат. наук, проф., ozmeyev@gmail.com,
Д. О. Змеев, аспирант, denis.zmeev@accounts.tsu.ru,
А. Н. Даниленко, магистр, danilenko.andrey.n@gmail.com,
Национальный исследовательский Томский государственный университет

Перенос практик Essence в среду Azure DevOps Server

Одним из ключевых вопросов программной инженерии является вопрос построения эффективных с точки зрения практики методологий разработки информационных систем. В качестве теоретического подхода для сравнения и описания методологий в последнее время активно развивается язык Essence как часть инициативы SEMAT. Однако текущие результаты пока, в основном, применяют для теоретического и методического описания методологий разработки информационных систем. Практическое использование Essence для решения прикладных задач только начинает развиваться. В данной работе предложена методика трансформации методологии, описанной в терминах языка Essence на абстрактном уровне, в среду управления проектами Azure DevOps Server. Данную методику можно использовать для проверки и практического использования описанной методологии.

Ключевые слова: *SEMAT, Essence, среда управления проектами, Azure DevOps Server, процесс разработки, методология разработки, практика разработки*

Введение

Если проанализировать исторический ход развития технических наук, можно заметить, что катализатором этого развития был поиск решений практических проблем и задач [1]. С течением времени поиск и накопление решений создавали объем наблюдений, которые позволяли создать новый уровень теоретических результатов и связать его с прочими сферами знаний [2]. Программная инженерия — признанная в мире научно-техническая дисциплина, которая находится на достаточно раннем этапе своего развития. Процесс построения общепризнанных научных основ на данном этапе только запущен, происходит обобщение накопленного опыта, успехов и неудач [3]. С учетом бурного роста технологических возможностей и скорости изменения подходов к разработке информационных систем текущее состояние программной инженерии приводит к конфликтам, возникающим при взаимодействии науки, практики и образования. В результате работающие в каждой из этих сфер специалисты испытывают трудности при обсуждении общих вопросов, даже если это представители одной и той же сферы. Так, в программной инженерии с момента ее появления существуют большие сложности в понимании процессов разработки программного обеспечения и их влияния друг на друга [4]. Причина в том, что в настоящее время в программной инженерии отсутствует устоявшаяся научная база, которая в технических науках является фундаментом обсуждения различных тем и связующим звеном при изменениях научной парадигмы [2].

Для решения амбициозной задачи "воссоздание программной инженерии как дисциплины с четкой методологией" [5] в 2009 г. была принята инициатива SEMAT (*Software Engineering Method and Theory*). В 2010 г. в рамках этой инициативы появился первый фундаментальный результат, который был сформулирован в виде документа-концепции и стал своеобразным техническим заданием для решения заявленной задачи. Представленная на публичное обсуждение концепция предусматривала "создание ядра для описания существующих и будущих так называемых практик (методик работы и способов их реализации), а также методов (методологий проектирования и разработки)" [6]. В рамках работ инициативы SEMAT такое ядро Essence было сформировано и представлено в 2013 г. в работе [7], а в 2014 г. — в виде стандарта OMG (*Object Management Group*) [8].

С одной стороны, разработанный стандарт фиксирует базовые сущности в разработке программного обеспечения и определяет связи между ними, с другой стороны, предлагает набор визуальных элементов для описания практик программной инженерии в терминах стандартного ядра. После выхода стандарта появилось достаточно большое число работ, в которых разбирались основные понятия новой модели: Alpha, Alpha State, Activity и т. д. [5, 9].

На период написания настоящей статьи еще сложно оценить влияние нового стандарта на программную инженерию. С одной стороны, в момент его подготовки многие известные исследователи и практики, использующие Agile-методологии (гибкие методологии), высказывались об инициативе SEMAT достаточно критически [10—12]. С другой стороны, появилось достаточно много работ, показывающих применение стандарта [8] для решения различных задач в области программной инженерии. Например, в работе [13] в идеологии SEMAT

представлена практика применения вариантов использования. В работах [14, 15] в рамках нового стандарта представлена одна из самых известных Agile-методологий — SCRUM. На базе образовательного трека инициативы SEMAT предпринимаются попытки разработки специальных курсов, которые будут знакомить будущих программных инженеров с новым стандартом [16, 17]. Появляются работы, в которых стандарт используется для решения практических задач [18]. Разрабатываются программные средства, использующие нотацию этого стандарта [19—21].

Таким образом можно отметить, что сообществом SEMAT ведется методическая и планомерная работа по внедрению нового стандарта в практическую деятельность. Однако, по мнению авторов, существует еще одна задача, которую необходимо решить, чтобы расширить область практического применения Essence. Она заключается в необходимости разработать инструментальные средства и подходы к их использованию, которые позволят реализовывать практики, представленные в нотации Essence, в различного рода системах управления проектами по разработке программного обеспечения.

В настоящей работе представлена методика переноса практик, описанных с помощью Essence, в веб-приложение для управления проектами и задачами. С одной стороны, данный подход предоставляет возможность менеджеру проекта управлять задачами в рамках привычной системы управления проектом. С другой стороны, такая система предоставляет функциональные возможности для реализации всех действий, направленных на отслеживание изменения состояний сущностей, описанных в соответствующей практике в нотации Essence. Таким образом, автоматически поддерживается гарантируемая теоретическим описанием Essence целостность применяемой практики. При этом соблюдение всех предположений и ограничений используемой практики перестает полностью зависеть от компетенции менеджера проекта. В качестве примера переноса метода в представленной работе рассмотрен метод Agile Essentials [14] (состоящий из набора связанных практик), а в качестве приложения для управления проектами — Azure DevOps Server компании Microsoft [22].

Настоящая статья состоит из трех последовательных логических частей (разделов). В разд. 1 подробно рассмотрены связи между теоретическим и практическим контурами, определенными в стандарте Essence. Основное внимание сакцентировано на тех особенностях, которые необходимо учитывать при переносе практики, описанной в терминах Essence, в среду управления проектами. В разд. 2 рассмотрены механизмы расширения Azure DevOps Server, установлено взаимно однозначное соответствие между элементами модели практики стандарта и классами, предназначенными для расширения функциональности системы управления проектами. В разд. 3 продемонстрирована практика, реализованная в среде управления проектами. В Заключении приведен анализ направлений для дальнейших исследований.

1. Теоретический и практический контуры в стандарте Essence

Учитывая тот факт, что Essence является стандартом OMG, его формальное описание выполнено на языке моделирования Meta-Object Facility (MOF) [23]. В рамках стандарта явно выделены все основные сущности (такие как Alpha, Work Product, Activity и Activity Space) и связи между ними. Концептуальные отношения между сущностями представлены на рис. 1. Поскольку Essence еще не стал стандартом де-факто, кратко опишем основные рассматриваемые элементы.

Сущности Alpha (*Abstract Level Project Health*, Альфа), Alpha State (Состояние Альфы), Sub-Alpha (Суб-Альфа) связаны между собой и введены в языке Essence как способ измерения и отслеживания прогресса проекта в различных семантически значимых зонах. Alpha проходит через набор Alpha State, которые содержат набор утверждений Checkbox (например, для Alpha "программная система" есть состояние "выбрана архитектура", одно из утверждений которого "технологии выбраны"), и эти утверждения либо истинны для проекта, либо ложны. Родительская сущность Alpha считается достигшей определенного Alpha State-уровня, если все утверждения этого Alpha State истинны. Ядро Essence фиксирует для любого проекта семь основных Alpha. Сущность Sub-Alpha (она не введена на рис. 1, но активно используется) — это дополнительная сущность, которая вводится как вспомогательный инструментальный механизм описания определенных практик. Сущность Sub-Alpha, во-первых, связана с родительской Alpha, во-вторых, повторяет логику Alpha (обладает собственным набором Alpha State, связана с Work Product, также может включать в себя дополнительные Sub-Alpha).

Сущность Work Product (рабочий продукт) семантически наиболее близка к термину "артефакт", достаточно широко применяемому в программной

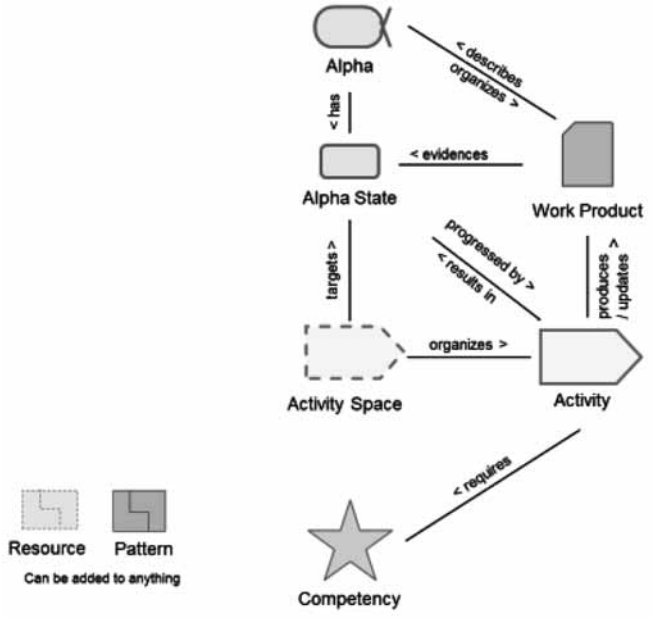

Рис. 1. Концептуальный обзор языка [8]

инженерии. Она позволяет описывать создаваемые, модифицируемые и используемые результаты выполненных работ в тех или иных практиках. Необходимо отметить, что у Work Product существуют Level of Details (уровень детализации), которые выполняют роль, аналогичную Alpha State, однако с существенным отличием: для Work Product достаточно, чтобы хотя бы одно утверждение было истинным.

Элементы Essence Activity Space (пространство активностей) и Activity (активность) описывают динамическую составляющую используемых практик, т. е. непосредственно то, что будут выполнять члены команды разработчиков. Элемент Activity Space представляет собой абстрактное описание некоторого вида действий, а Activity описывает детализованные задачи, которые необходимо выполнять в тех или иных практиках.

Кроме представленных выше сущностей, стандарт определяет структуру данных, которая строго и формально фиксирует все связи и сущности, необходимые для описания практик и методов Essence. Пример из стандарта, описывающего часть структуры данных, необходимой для описания Alpha, Alpha State, Work Product, представлен на рис. 2. На момент написания данной работы существующие программные инструменты, поддерживающие Essence и позволяющие описывать новые практики, имеют структуру данных, схожую с представленной в стандарте.

Необходимо отметить, что в этой части стандарта не вводится Sub-Alpha как отдельный вид сущностей. Для них используется тот же класс Alpha, но с использованием класса AlphaContainment.

При всех своих достоинствах формат MOF очень сложен для применения при решении практических задач. Нетрудно заметить, что даже концептуальные диаграммы Essence (вводимые в том же стандарте) выполнены в демонстрационном, а не формальном стиле взаимодействия сущностей MOF. Необходимо отметить, что стандарт Essence сам по себе не содержит структурных связей (а только семантические) с принятыми в профессиональной деятельности сущностями, даже такими общими, как проект, задачи, исполнители. Более того, из представленных в стандарте описаний языка очевидно не следует, как именно применять практику, описанную в терминах стандарта, в проекте по разработке программного обеспечения. Для понимания этой ситуации необходимо разобраться с разными контурами, которые негласно вводятся языком Essence.

Во-первых, вводится концептуальное (теоретическое) описание метода программной инженерии — теоретический контур Essence. Для этого формально определяются связи элементов Alpha, Alpha State и Activity Space. Стандарт вводит три основные зоны для взаимодействия этих элементов — Customer, Solution и Endeavor, определяет для каждой из них конкретный набор сущностей элементов Alpha и состояний и выстраивает множество связей между этими элементами, которое обеспечивает теоретическую полноту этого набора. Схематически эти отношения представлены на рис. 3, см. вторую сторону обложки.

Нетрудно заметить, что в теоретическом контуре основной поток взаимодействия сущностей, опреде-

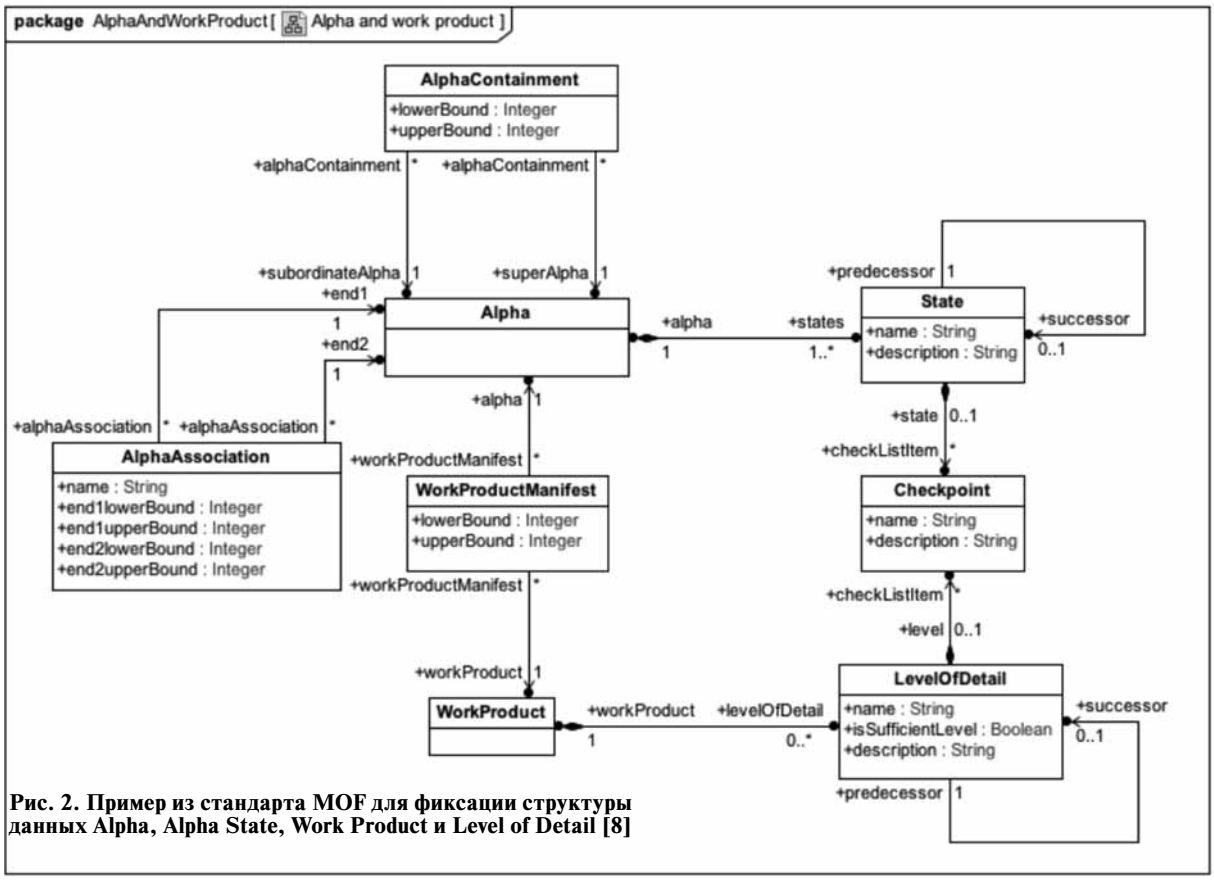

Рис. 2. Пример из стандарта MOF для фиксации структуры данных Alpha, Alpha State, Work Product и Level of Detail [8]

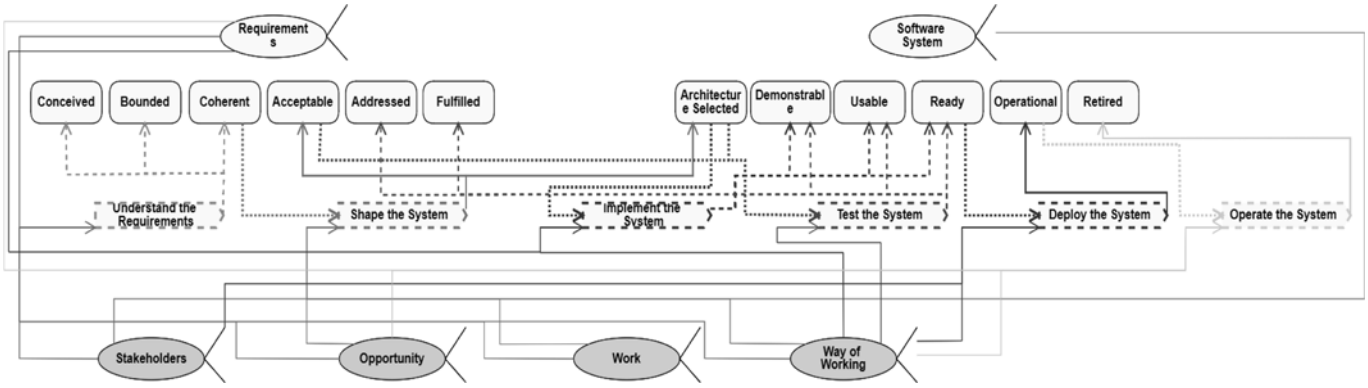

Рис. 4. Детализация теоретического контура для зоны Solution

ленных стандартом Essence, осуществляется между элементами Activity Space, которые принадлежат конкретной зоне. Связь с Alpha обеспечивается на уровне параметров входа для каждого элемента Activity Space двумя разными вариантами (рис. 4).

Первый вариант — это ассоциация следующего содержания: выполняя эту Activity Space, необходимо учитывать текущий прогресс связанных Alpha (например, Alpha "Opportunity" — входная Alpha для Activity Space "Shape the System"). Второй вариант — это ограничение, суть которого в следующем: чтобы эту Activity Space можно было реализовать, определенная Alpha должна достигнуть конкретное Alpha State (например, для Activity Space "Shape the System" Alpha "Requirements" должна быть в Alpha State "Coherent").

Во-вторых, при переходе в плоскость реального использования в стандарте предусматривается построение практического контура через элементы Activity и Work Product, а также детализация на уровне Sub-Alpha. Выстраивание этого контура происходит в процессе описания конкретной практики в терминах стандарта (рис. 5).

Выполняя эту работу, разработчик практики вводит конкретные объекты Activity (Refine Product Backlog, Prepare a Product Backlog Item, Agree Definition of Done) и связывает их с одним или несколькими элементами Activity Space. Наличие этой связи переносит ограничения теоретического контура в плоскость возможности их применения в практической деятельности. Далее определяются связи между Activity и Work Product (Product Backlog, Test Case, Definition of Done), а для контроля над прогрессом выполнения реальных работ вводятся элементы Sub-Alpha. Таким образом, именно процесс погружения практики в язык стандарта обеспечивает связь теоретического и практического контуров. Заметим, что именно наличие такого описания делает набор устоявшихся действий практикой в терминах стандарта. Примеры такого рода описаний можно найти в работах [13, 14, 17, 26].

Необходимо отметить, что ни концептуальные схемы Essence, ни детальные схемы практик не отмечают формальной логики достижения состояния Alpha. Согласно стандарту каждое состояние, описываемое с использованием этого формализма, характеризуется набором Checkbox. При этом только единовременное выполнение принятых условий позволяет определить, что Alpha достигла этого состояния (пример листа Checkbox дан на рис. 6).

Однако ни в описании теоретического контура Essence с указанием общих Alpha и Activity Space, ни при описании практик, для Activity не указывается на выполнение каких именно Checkbox эта Activity направлена. Например, если для практик, которые детализируют Sub-Alpha, фиксируется, что Activity на выходе имеет Sub-Alpha в нужном состоянии, то можно только предположить, что в таком случае в процессе выполнения Activity все условия для выполнения всех Checkbox указанного состояния Sub-Alpha должны быть выполнены. Но при этом для общих Alpha, которые описывают общее состояние проекта, вопрос об отметке Checkbox и, следовательно, об определении

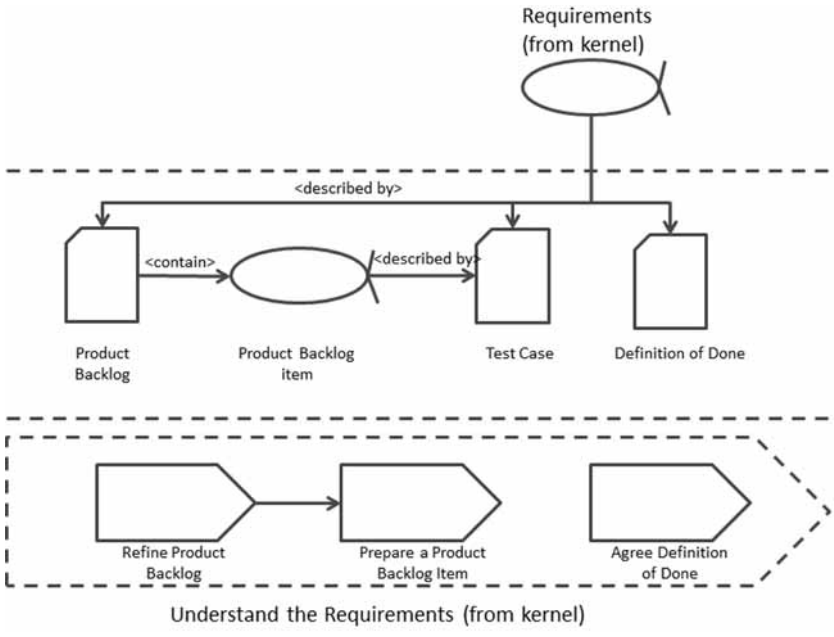

Рис. 5. Практика Product Backlog Essentials [14]

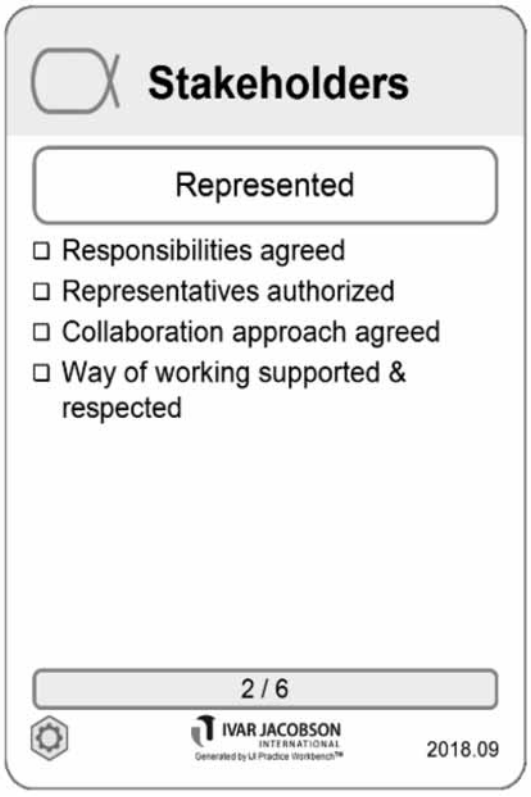

Рис. 6. Карточка состояния Alpha

текущего достигнутого состояния, не определяется используемой практикой (рис. 7).

Причина в том, что при описании практики для Activity обычно используют служебную отметку "contributes to" (пример на рис. 7), что пока наблюдается в большинстве официально опубликованных описаний практик [24].

В результате вопрос о выполнении Checkbox и достижении Alpha нужных Alpha State остается на уровне принятия экспертного решения командой и менеджментом проекта. При этом прикладное применение практики на представленных диаграммах ограничивается элементами Work Product и Activity. Как следствие, с точки зрения использования Essence в рамках реализации проекта оно нуждается в более детальном описании. Например, на рис. 8 детализируется представленная выше практика Product Backlog Essentials (см. рис. 5).

Легко заметить, что ассоциации детализируются изменениями состояний Sub-Alpha и уровней детализации Work Product. Даже для такой достаточно несложной практики диаграмма не дает хорошего наглядного представления о взаимосвязях элементов.

В контексте среды управления проектом реализация практики определяется как последовательность следующих шагов.

Шаг 1. Запускается экземпляр Activity в виде назначения конкретных задач конкретным исполнителям, играющим соответствующие Team Role.

Шаг 2. Реализация этих задач изменяет состояние экземпляра класса Work Product. Заметим, что непосредственное содержание этого объекта может находиться в другом средстве автоматизации процесса разработки, как, например, исходный код разрабатываемой системы находится в системе контроля версий.

Шаг 3. Экземпляр Activity в форме назначенной задачи признается завершенным. В зависимости от используемой среды управления проектом и используемых в компании практик основанием для такого признания может быть как личное решение исполнителя, так и автоматическая простановка статуса, который изменяется в результате прохождения автоматического тестирования.

Шаг 4. На основании текущего состояния экземпляра класса Work Product принимается решение о полноте реализации соответствующих элементов экземпляра класса Sub-Alpha (Checkbox, State).

Шаг 5. Достигнутый прогресс на уровне экземпляра класса Sub-Alpha позволяет принять решение о текущем состоянии экземпляра родительской Alpha (Checkbox, State).

Описанную таким способом процедуру можно реализовать, и благодаря ей перенести любую описанную практику в среду управления проектами. Однако при этом нужно учитывать, что на момент написания данной статьи развитие Essence и описанных этим стандартом методов приводит к тому, что сами по себе практики и их элементы самостоятельны. Кроме как семантически, они практически не взаимосвязаны с другими практиками и их элементами на уровне связей самого Essence. Таким образом, для решения задачи переноса метода Essence необходимо и достаточно получить метод переноса

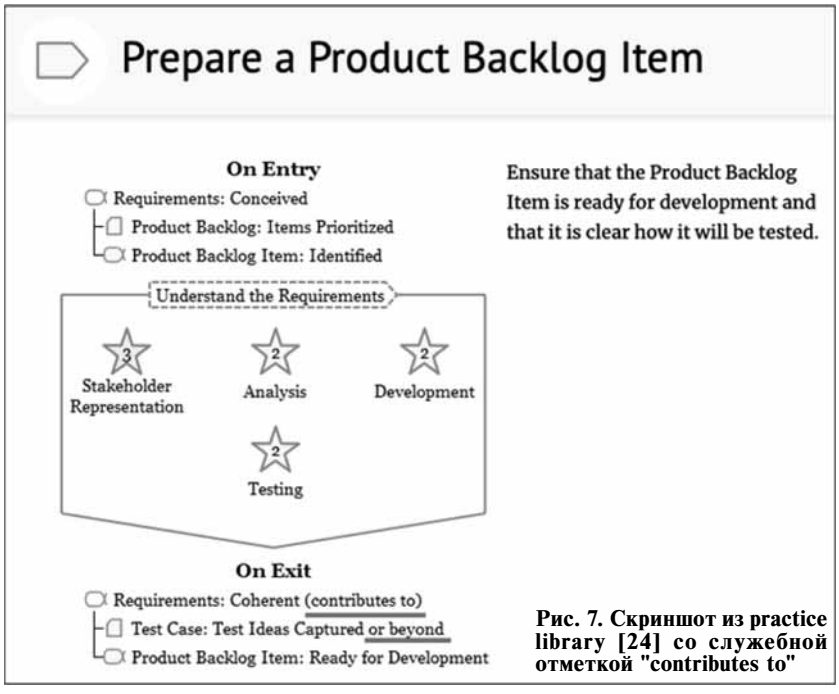

Рис. 7. Скриншот из practice library [24] со служебной отметкой "contributes to"

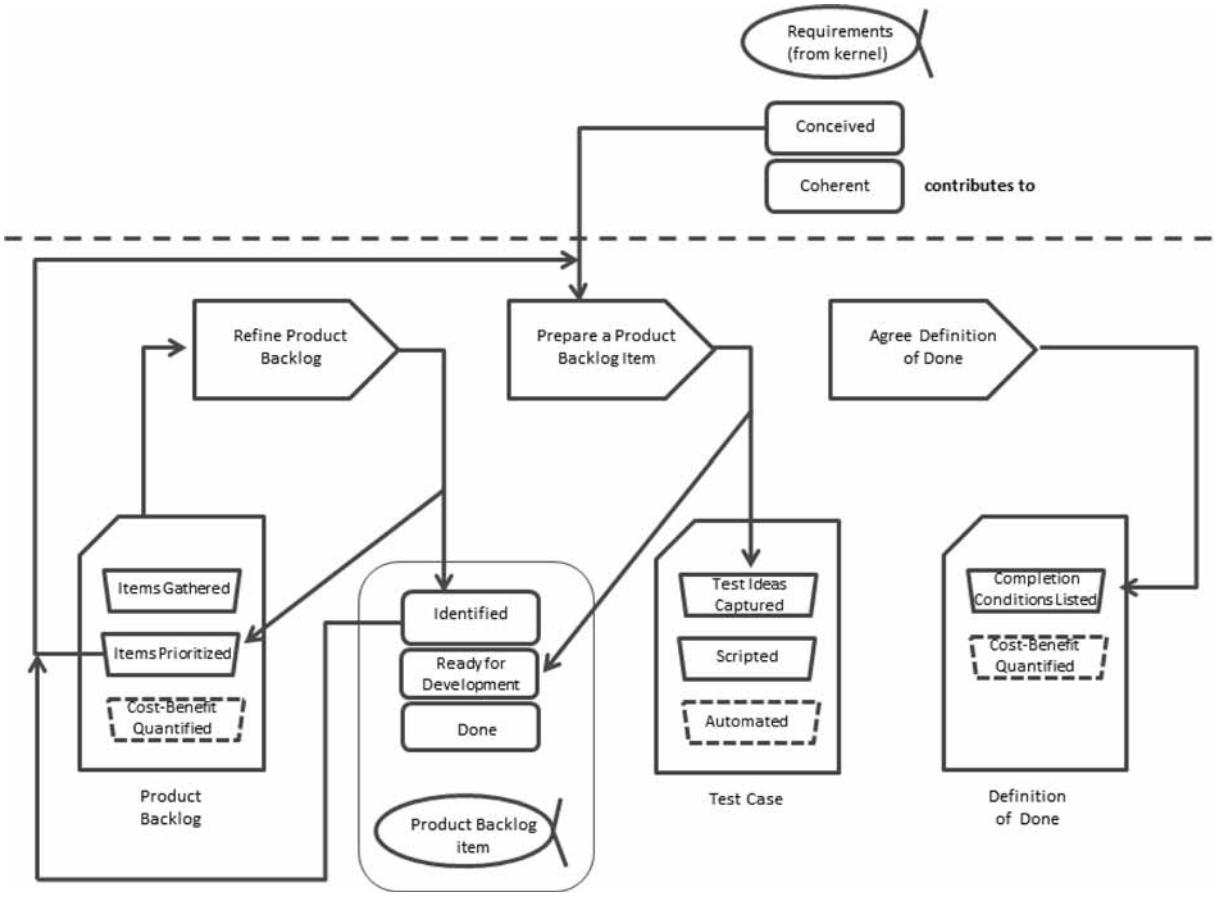

Рис. 8. Детализация практики Product Backlog Essentials

отдельной практики и при необходимости повторить этот метод, учитывая общие ограничения языка (например, уникальность основных Alpha).

2. Перенос практики в Azure DevOps Server

Чтобы реализовать работу с практиками и методами через Azure DevOps Server, был проведен анализ механизмов расширения данной системы.

Главным механизмом расширения в Azure DevOps Server является возможность создания шаблонов процессов (Process Templates). Каждый проект в Azure DevOps Server обязательно базируется на одном из шаблонов процессов, внутри которого можно определить компоненты для работы с проектами. Чтобы иметь возможность переносить методы и практики Essence в Azure DevOps Server, был создан шаблон процесса под названием Essence. Этот шаблон было решено настроить таким образом, чтобы он содержал в себе компоненты, по свойствам и поведению идентичные элементам языка Essence, с помощью которых можно на практике осуществлять работу с любыми методами и практиками. Такие компоненты в Azure DevOps Server называются рабочими элементами (Work Item).

Рабочий элемент (Work Item) — это компонент, который используется для отслеживания заданий, работ или задач в проектах Azure DevOps Server. Каждый рабочий элемент должен относиться к какому-то определенному типу (Work Item type), от которого будут зависеть его поля (fields) и другие параметры. Если проводить аналогию с объектно-ориентированным подходом, каждый рабочий элемент является экземпляром какого-то класса, фиксирующего тип рабочего элемента. От типа рабочего элемента зависит набор полей (field), который он будет иметь, и набор статусов (state), в которые можно переводить рабочий элемент. Рабочий элемент может быть также связан с другими рабочими элементами. На этапе проектирования было решено реализовать сущности языка Essence как отдельные типы рабочих элементов.

Помимо элементов язык Essence описывает логику взаимодействия между ними. Для реализации логики в Azure DevOps Server используются правила (rules). Правило — это сценарий, исполнение которого автоматически вызывается в тот момент, когда изменяется и сохраняется рабочий элемент того типа, к которому привязано конкретное правило. Правила позволяют автоматизировать систему управления проектом. Встроенная система создания правил Azure DevOps Server не имеет достаточных функциональных возможностей чтобы реализовать всю необходимую логику языка Essence. По этой причине было решено использовать плагин TFS Aggregator — расширение для Azure DevOps Server, позволяющее создавать правила путем написания скриптов.

В Azure DevOps Server имеется также расширяемый встроенный механизм импорта рабочих элементов в проект. Этот стандартный механизм позволяет

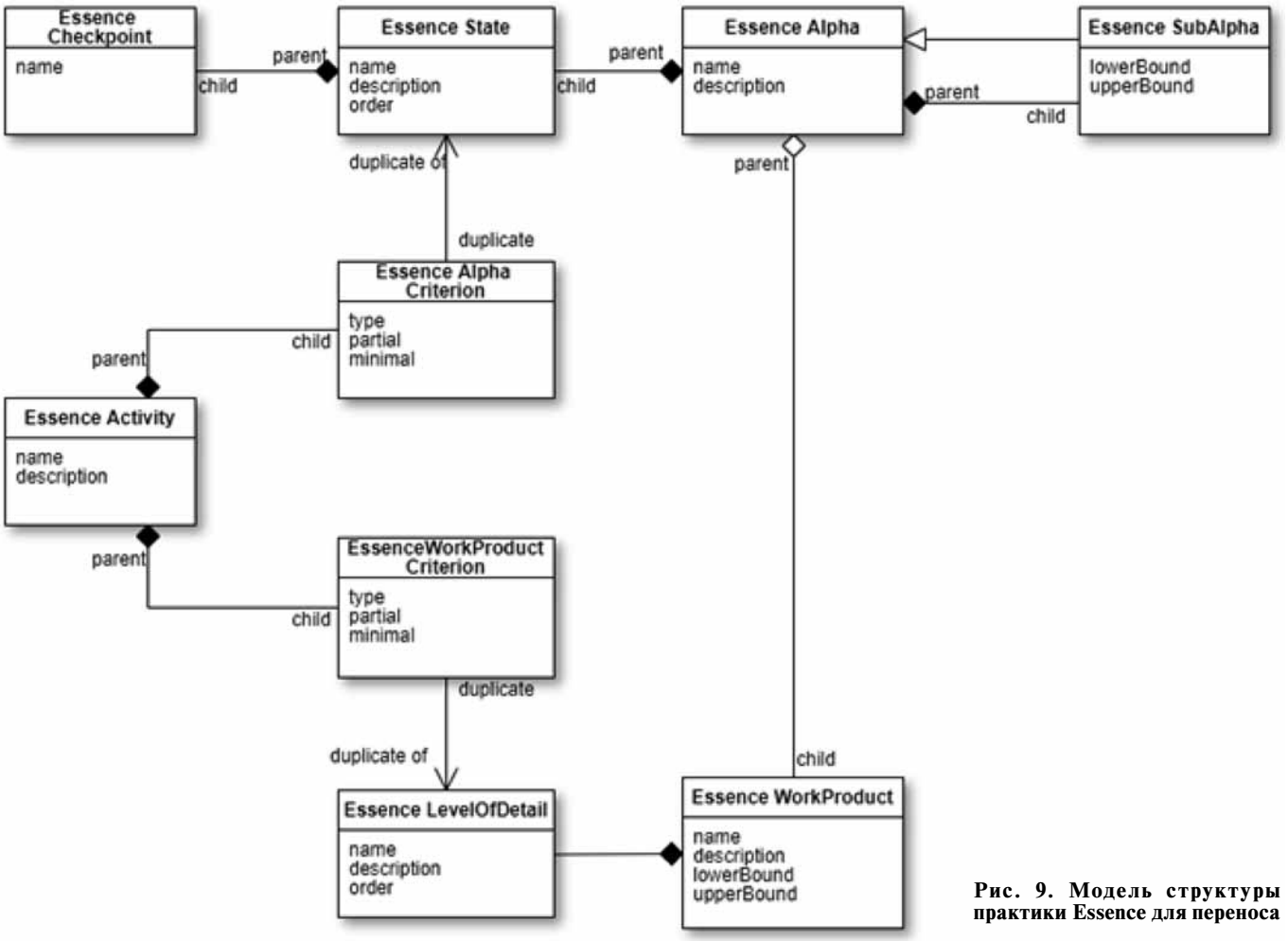

Рис. 9. Модель структуры практики Essence для переноса

импортировать рабочие элементы через программу Microsoft Excel. Данный механизм позволяет не просто импортировать элементы со всеми необходимыми полями, но и обозначить при этом связи между ними, используя так называемые запросы (Queries).

В связи с тем, что процесс разработки или метод, полностью описанный в языке Essence, представляет собой полное и формальное методологическое описание, для решения вопросов практического использования практик, описанных в Essence, было необходимо редуцировать модель так, чтобы она позволяла хранить информацию об основных сущностях процесса и связи между ними. Для этого была разработана модель данных, выраженная в форме диаграммы классов UML, которая представлена на рис. 9.

Данная модель позволяет учитывать процесс реализации практики с точки зрения системы управления проектами и является достаточно полной, чтобы учитывать основные сущности и связи между ними на языке Essence. С точки зрения механизмов Azure DevOps Server все представленные на рис. 9 сущности преобразуются с использованием механизма расширения в однозначно соответствующие типы рабочих элементов. Необходимо также отметить, что при переносе конкретных практик прагматичным решением стало не переносить сущность Activity Space. Основная причина этого решения в том, что с точки зрения прикладного применения переносимые практики Activity Space уже реализованы с помощью соответствующих Activity. При этом следует отметить, что поскольку практики направлены на описание повторяемых и зафиксированных подходов для достижения определенного результата, то такой перенос не имеет особого смысла и будет избыточным. К тому же если сопоставить используемые сущности с введенными на рис. 1, то можно заметить отсутствие сущности Competency, что также является следствием принятых решений. В подавляющем большинстве случаев среды управления проектами не содержат структурированных "личных дел" или профилей исполнителей, которые можно было бы сопоставить с уровнями компетенций из Essence. С учетом отмеченных соображений, чтобы не создавать для менеджмента дополнительных ограничений и необходимости решать задачу нечеткого сравнения, Competency также не переносятся в среду управления проектами.

3. Реализация

Помимо переноса структур данных из элементов Essence необходимо также, чтобы рабочий процесс в проекте, созданном на основе разработанного шаблона, проходил по той логике, которая предполагается при использовании языка Essence. Необходимо было реализовать логику для типов рабочих элементов, пред-

ставляющих элементы языка Essence, в качестве правил в Azure DevOps Server. Для этого на начальном этапе были определены основные функциональные особенности языка Essence. Далее представлен список этих основных функциональных особенностей языка Essence.

1. При закрытии всех Checkbox Alpha State — Alpha переводится в соответствующее состояние.

2. При закрытии одного Checkbox уровня детализации Work Product — Work Product переводится в соответствующий уровень детализации.

3. При переводе Alpha в новое Alpha State — все Activity, у которых Alpha в следующем Alpha State является входным критерием, становятся доступны для создания.

4. После завершения действия должны быть соблюдены выходные критерии для завершаемого Activity: Alpha должны быть в соответствующих состояниях, а Work Products — в соответствующих Level of Detail.

Представим пример наиболее сложной реализации данной логики (п. 3) за счет использования возможностей TFS Aggregator (рис. 10).

При вызове метода SetStatus() объекта EssenceState в случае, если статус Closed (т. е. у какой-либо Alpha определенного Alpha State все Checkbox были отмечены), происходит вызов у объекта EssenceAlpha, инициирующий EssenceState-цикл, который для всех связанных через AlphaCriterion Activity разрешает метод копирования. Одна из особенностей реализации практик Essence состоит в том, чтобы вместо разработки сложных конструкторов, обеспечивающих все необходимые связи, выполнять процедуру копирования, аналогичную паттерну "Прототип" [25], т. е. при импорте практики Essence в Azure DevOps Server создается виртуальная копия всех необходимых сущностей, а в процессе использования практики все новые объекты копируются из прототипа.

Таким образом, решение для интеграции методов, описанных на языке Essence, в систему управления проектами Azure DevOps Server было реализовано в виде шаблона процессов в Azure DevOps Server под названием Essence. В разработанном шаблоне для реализации элементов языка Essence были использованы типы рабочих элементов. При этом рабочий элемент представляет собой элемент конкретной практики с типом, соответствующим элементу языка

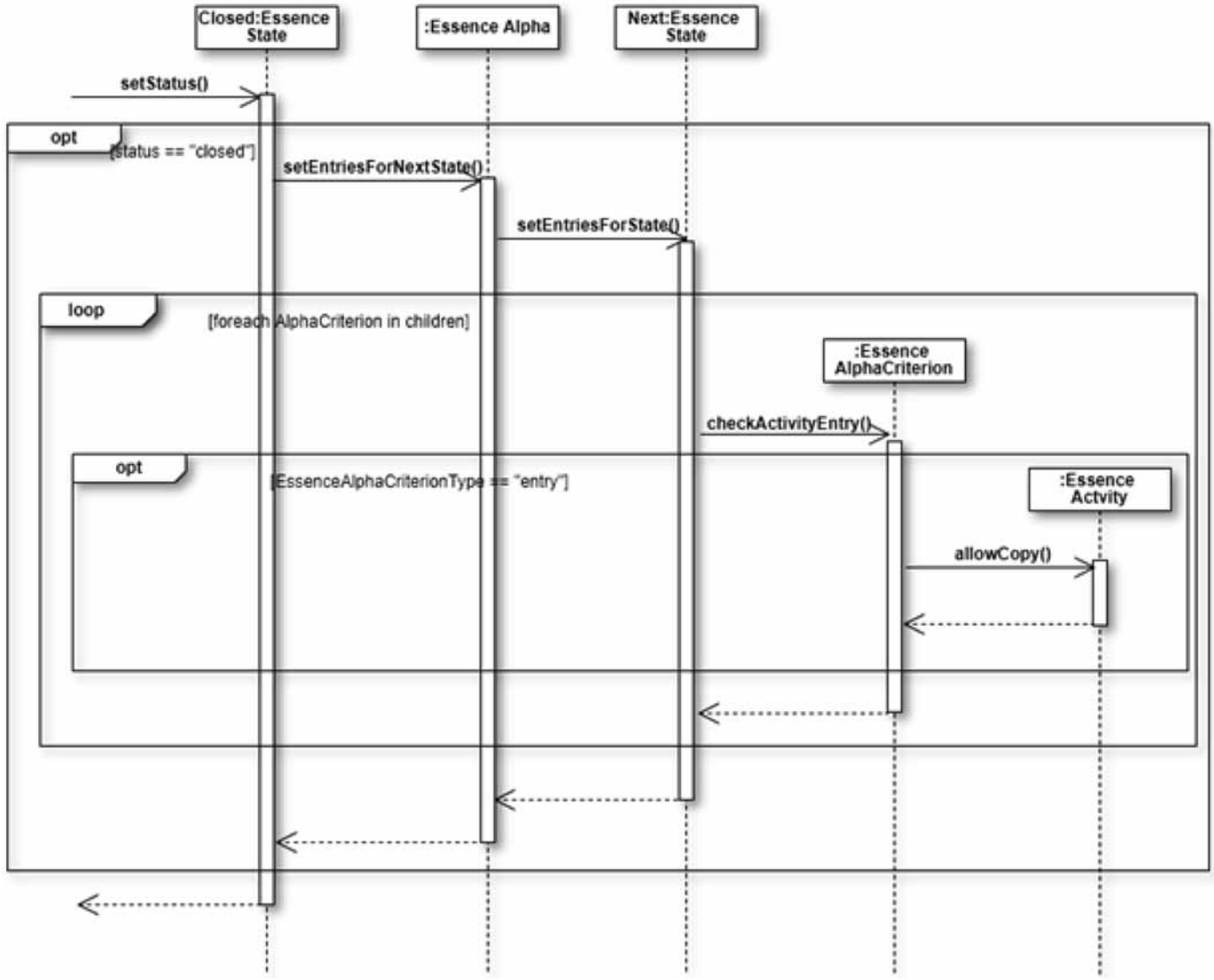

Рис. 10. Диаграмма последовательности для правила реализации предоставления доступа к созданию новых Activity в случае, если меняется Alpha State

Essence. Импорт практик и методов в проект Azure DevOps Server, построенный на шаблоне Essence, реализован через преобразование файлов методов и практик из приложения Practice Workbench в структуру, необходимую для использования стандартного механизма импорта рабочих элементов Azure DevOps Server через Microsoft Excel. Функциональные возможности, необходимые для ведения проекта в рамках методов, описанных на языке Essence, реализованы в виде правил для плагина TFS Aggregator.

В качестве примера на основе разработанного шаблона был создан проект, в который из Practice Workbench был импортирован метод Agile Essentials [24], состоящий из набора различных практик.

Для удобства работы с рабочими элементами были реализованы представления, позволяющие легко отслеживать и менять статусы элементов практик и методов в проекте. В качестве примера приведен скриншот страницы (рис. 11), на которой отображаются состояния Alpha-требований (Requirements) с Checkboxes.

Кроме того, рабочие элементы можно изменять с помощью страницы редактирования. На рис. 12 приведен скриншот страницы редактирования рабочего элемента типа Essence Activity на примере действия Prepare Product Backlog.

Разработанное решение позволяет создавать привычные для проектных команд задачи, которые за счет дополнительных связей и типов рабочих элементов получают теоретическую целостность с точки зрения практик Essence. Как видно на рис. 12, "115 Prepare Product Backlog" непосредственно связан с Work Product "137 Test Case" в текущем его уровне детализации.

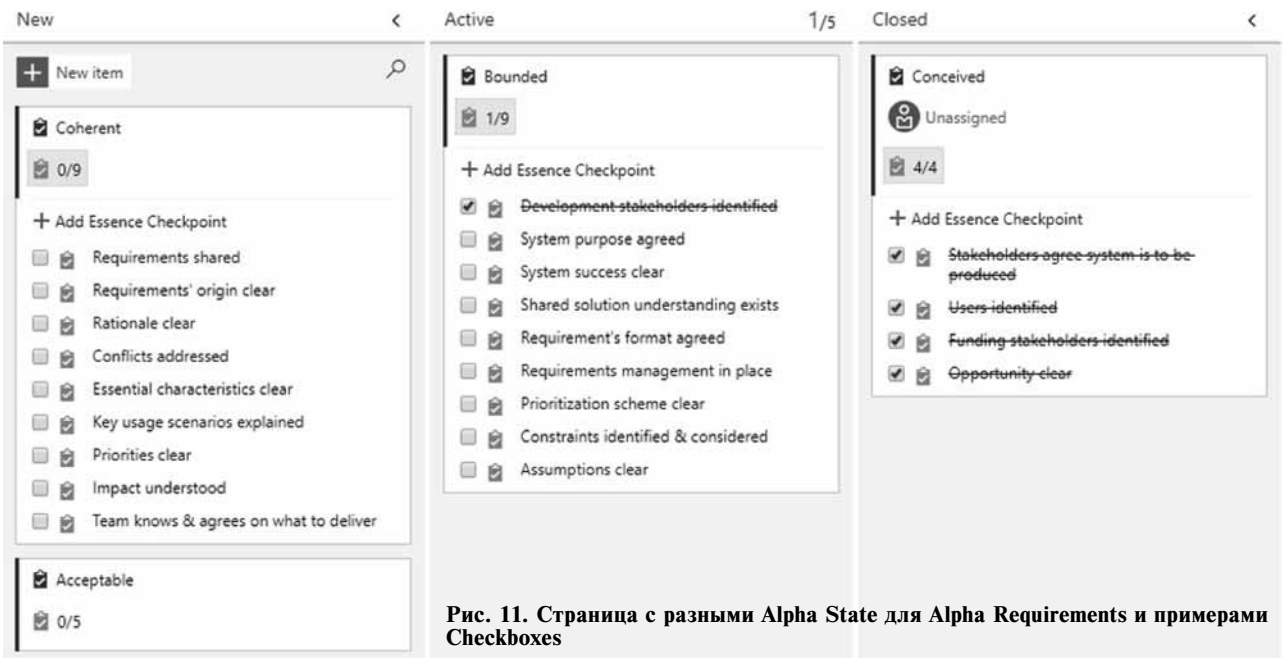

Рис. 11. Страница с разными Alpha State для Alpha Requirements и примерами Checkboxes

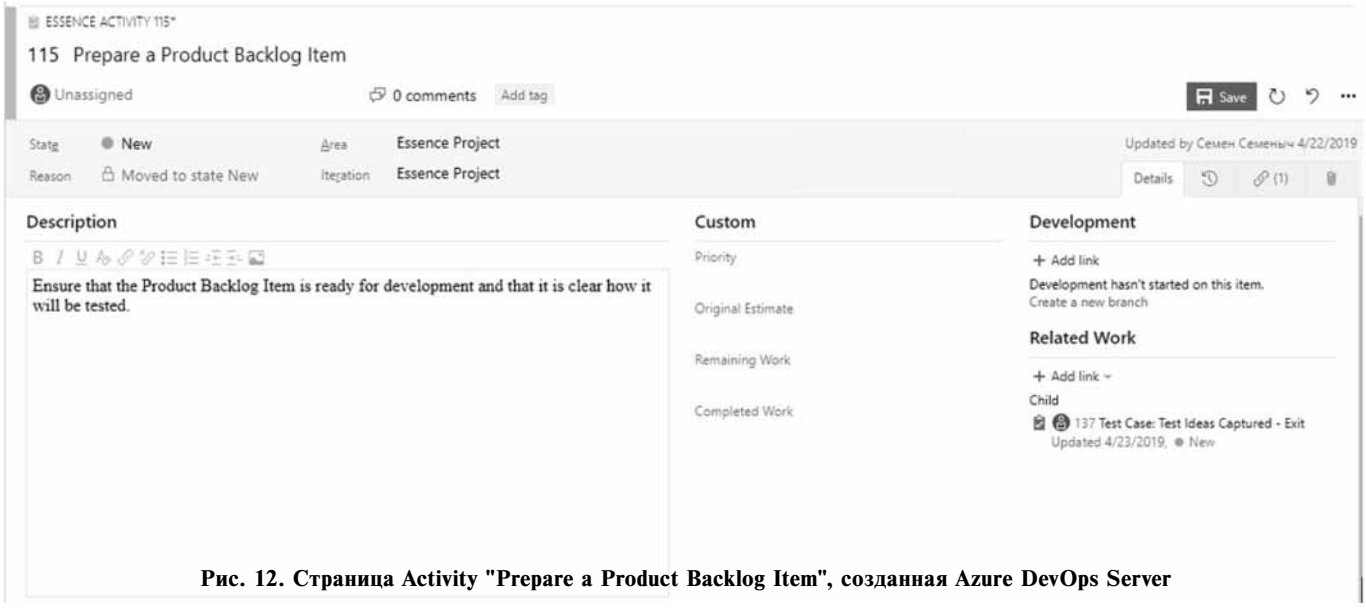

Рис. 12. Страница Activity "Prepare a Product Backlog Item", созданная Azure DevOps Server

Кроме структурных и семантических связей, разработанное решение также поддерживает динамическую логику использования практик Essence, например, автоматический перевод Work Product в нужные Level of Details.

Заключение

Представлено решение задачи переноса практик, описанных в терминах SEMAT Essence, в среду управления проектами Azure DevOps Server. На примере показано, как за счет механизмов расширения системы можно реализовать ограничения всех теоретических установок практик Essence, выполняя прикладные задачи в рамках процесса разработки проекта по реализации программного обеспечения.

Представленные в работе подходы могут быть использованы в двух направлениях. С одной стороны, предложенный подход применим при переносе практик в другие среды управления проектами [26, 27]. С другой стороны, реализация формального теоретического контура в реальных системах управления проектами позволяет собирать, анализировать и сравнивать данные вне зависимости от того, в какой среде управления они были получены. Более того, эти данные изначально имеют теоретическую разметку в терминах Essence. Наличие таких массивов данных позволяет решать задачи по оптимизации работ в IT-компаниях на основе методов статистической обработки или с помощью интеллектуального анализа этих данных.

Список литературы

1. **Иванов Б. И., Чешев В. В.** Становление и развитие технических наук. — Л.: Наука. Ленингр. отд-ние, 1977. — 264 с.
2. **Кун Т.** Структура научных революций. — М.: АСТ, 2009. — 310 с.
3. **Johnson P., Ekstedt M., Jacobson I.** 2012 Where's the Theory for Software Engineering? // IEEE Software. — 2012. — Vol. 29, No. 5. — P. 96. URL: http://dx.doi.org/10.1109/MS.2012.127.
4. **Kajko-Mattsson M.** Software engineering suffers from the beehive syndrome // Information Science and Digital Content Technology (ICIDT) 2012. 8th International Conference on Computing Technology and Information Management. — 2012. — Vol. 1. — P. 49—52.
5. **Позин Б. А.** SEMAT — Software Engineering Method and Theory. О чем, зачем и кому это нужно? // Программная инженерия. — 2014. — № 11. — С. 3—5.
6. **Jacobson I., Meyer B., Soley R.** Software engineering method and theory — a vision statement. URL: https://www.ympu.com/en/document/read/11408476/semat-vision-statement
7. **Jacobson I., Ng P.-W., McMahon P. E., Spence I., Lidman S.** The Essence of Software Engineering: Applying the SEMAT Kernel. — Addison Wesley, 2013. — 224 p.
8. **Essence** — Kernel and Language for Software Engineering Methods Version 1.2. URL: http://semat.org/documents/20181/57862/formal-18-10-02.pdf/866c80c0-cdc8-488b-bcf8-0c67cb60b5d7
9. **Левенчук А. И.** Основа — сущности и язык для методов программной инженерии (OMG Essence). URL: https://ailev.livejournal.com/1051048.html
10. **Fowler M.** SEMAT. 2010. URL: http://martinfowler.com/bliki/Semat.html
11. **Some** critiques of the Semat initiative. URL: http://semat.org/blog/-/blogs/some-critiques-of-the-semat-initiative
12. **Aranda J.** Against SEMAT. URL: http://catenary.wordpress.com/2009/11/29/against-semat/
13. **USE-CASE** 2.0 ESSENTIALS | SCALABLE AGILE PRACTICE. URL: https://www.ivarjacobson.com/software-use-case-essentials
14. **Park J., McMahon P., Myburgh B.** Scrum Powered by Essence. // ACM SIGSOFT Software Engineering Notes. — 2016. — Vol. 41, No. 1. — P. 1—8. DOI: 10.1145/2853073.2853088.
15. **Jacobson I., Lawson H., Ng P.-W.** et al. Running with Scrum. — ACM Books, 2019. — 401 p.
16. **Zapata C., Jacobson I.** A First course in software engineering methods and theory// Dyna rev.fac.nac.minas [online]. — 2014. — Vol. 81, No. 183. — P. 231—241. URL: http://dx.doi.org/10.15446/dyna.v81n183.42293.
17. **Jacobson I., Lawson H. B., Ng P.-W.** et al. The Essentials of Modern Software Engineering. — ACM Books, 2019. — 371 p.
18. **Позин Б. А., Горбунова Е.** Развитие базовой модели SEMAT для жизненного цикла заказных ответственных программных систем // Материалы четвертой Научно-практической конференции "Актуальные проблемы системной и программной инженерии". Сб. трудов. — М.: НИУ ВШЭ, 2015. — С. 170—171.
19. **ESSENCE** ENTERPRISE 365. URL: https://www.ivarjacobson.com/essence-enterprise
20. **SEMAT** Essence Kernel Tool. URL: https://semat.herokuapp.com/
21. **Graziotin D., Abrahamsson P.** A Web-based modeling tool for the SEMAT Essence theory of software engineering // Journal of Open Research Software. — 2013. — Vol. 1, No. 1. — P. e4. URL: http://dx.doi.org/10.5334/jors.ad
22. **Azure** DevOps Services. URL: https://azure.microsoft.com/ru-ru/services/devops
23. **Meta-Object** Facility. URL: https://www.omg.org/mof/
24. **Agile** Essentials | Practice Library. URL: https://practicelibrary.ivarjacobson.com/
25. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д.** Приемы объектно ориентированного проектирования. Паттерны проектирования: справочник. — М.: ДМК Пресс, 2007. — 368 с.
26. **Даниленко А. Н., Змеев Д. О., Змеев О. А., Тамазлыкарь Д. В.** Импорт модели SEMAT Essence Practice Workbench в среду управления проектами и задачами Redmine // Информационные технологии и математическое моделирование (ИТММ-2019): материалы XVIII Междунар. конф. им. А. Ф. Терпугова, 26—30 июня 2019 г. Ч. 1. — Томск: Изд-во НТЛ, 2019. — С. 27—30.
27. **Zmeev D., Zmeev O., Tamazlykar D.** Implementation of Essence Practice into project management system Redmine // Actual Problems of Systems and Software Engineering (Invited Papers). APSSE 2019, 12—14 November 2019, Moscow, proceedings. [S. l.]: IEEE Computer Society, 2019. — P. 116—125.

Implementation of Essence Practice into Azure DevOps Server System

O. A. Zmeev, ozmeyev@gmail.com, **D. O. Zmeev,** denis.zmeev@accounts.tsu.ru,
A. N. Danilenko, danilenko.andrey.n@gmail.com, National Research Tomsk State University, 634050, Russian Federation

Corresponding author:

Zmeev Denis O., PhD Student, National Research Tomsk State University, 634050, Russian Federation
E-mail: denis.zmeev@accounts.tsu.ru

Received on April 03, 2020
Accepted on September 16, 2020

Nowadays one of the important issues in software engineering is the question of creating business effective software development methods. The Essence language which was created by the SEMAT initiative makes it possible to have a common foundation and comparation tool for very different software development processes. However, the use of Essence is mostly focused on describing software development methods in their general form. In this work we propose a way to use Essence and its supporting tools to transform abstract practice into the Azure DevOps Server system entities and rules. To do this we firstly consider how relations from abstract level of Essence language transform into a specific practice by describing entities and relations between them. Then we demonstrate how a practice described in the Essence language should work using typical software project management system features. Then we look thorough different Azure DevOps Server features which allow to modify and upgrade systems. Finally, we describe how to modify an Azure DevOps Server system to support import of Essence practices and methods. To conclude, we consider in this paper a general description of Essence, what differs the abstract level of Essence from an Essence practice, what must be done to implement an Essence practice to any project management system, and describe our modification of the Azure DevOps Server.

Keywords: *SEMAT, Essence, software project management system, Azure DevOps Server, software development process (development method), development practice*

For citation:

Zmeev O. A., Zmeev D. O., Danilenko A. N. Implementation of Essence Practice into Azure DevOps Server System, *Programmnaya Ingeneria*, 2020, vol. 11, no. 6, pp. 311—321

DOI: 10.17587/prin.11.311-321

References

1. **Ivanov B. I., Cheshev V. V.** *Formation and development of technical sciences*, Leningrad, Nauka, Leningrad department, 1977, 264 p. (in Russian).

2. **Kuhn T. S.** *The Structure of Scientific Revolutions*, Chicago, University of Chicago Press, 1962, 210 p.

3. **Johnson P., Ekstedt M., Jacobson I.** Where's the Theory for Software Engineering? *IEEE Software*, 2012, vol. 29, no. 5, pp. 96, available at: http://dx.doi.org/10.1109/MS.2012.127.

4. **Kajko-Mattsson M.** Software engineering suffers from the beehive syndrome, *Information Science and Digital Content Technology (ICIDT), 8th International Conference on Computing Technology and Information Management*, 2013, vol. 1, pp. 49—52.

5. **Pozin B. A.** SEMAT — Software Engineering Method and Theory. About what, why, and who need it?, *Programmnaya Ingeneria*, 2014, no. 11. pp. 3—5 (in Russian).

6. **Jacobson I., Meyer B., Soley R.** Software engineering method and theory — a vision statement, available at: https://www.yumpu.com/en/document/read/11408476/semat-vision-statement

7. **Jacobson I., Ng P.-W., McMahon P. E., Spence I., Lidman S.** *The Essence of Software Engineering: Applying the SEMAT Kernel*, Addison Wesley, 2013. 224 p.

8. **Essence** — Kernel and Language for Software Engineering Methods Version 1.2, available at: http://semat.org/documents/20181/57862/formal-18-10-02.pdf/866c80c0-cdc8-488b-bcf8-0c67cb60b5d7

9. **Levenchuk A. I.** Foundation — entity and language for methods of software engineering (OMG Essence), available at: https://ailev.livejournal.com/1051048.html (in Russian).

10. **Fowler M.** SEMAT. 2010, available at: http://martinfowler.com/bliki/Semat.html

11. **Some** critiques of the Semat initiative available at: http://semat.org/blog/-/blogs/some-critiques-of-the-semat-initiative

12. **Aranda J.** Against SEMAT. 2009, available at: http://catenary.wordpress.com/2009/11/29/against-semat/

13. **USE**-CASE 2.0 ESSENTIALS | SCALABLE AGILE PRACTICE, available at: https://www.ivarjacobson.com/software-use-case-essentials

14. **Park J., McMahon P., Myburgh B.** Scrum Powered by Essence, *ACM SIGSOFT Software Engineering Notes*, 2016, vol. 41, no. 1, pp. 1—8. DOI: 10.1145/2853073.2853088.

15. **Jacobson I., Lawson H., Ng P.-W., McMahon P., Goedicke M.** *Running with Scrum.*, ACM Books, 2019, 401 p.

16. **Zapata C., Jacobson I.** A First course in software engineering methods and theory// Dyna rev.fac.nac.minas [online]. 2014, vol.81, no. 183, pp. 231—241, available at: http://dx.doi.org/10.15446/dyna.v81n183.42293I.

17. **Jacobson I., Lawson H. B., Ng P.-W., McMahon P. E., Goedicke M.** *The Essentials of Modern Software Engineering,* ACM books, 2019, 371 p.

18. **Pozin B. A., Gorbunova E.** Evolution main model SEMAT for lifecycle of development responsible software system, *Materialy chetvertoj Nauchno-prakticheskoj konferencii "Aktual'nye problemy sistemnoj i programmnoj inzhenerii"*, Moscow, NRU HSE, 2015, pp. 170—171 (in Russian).

19. **ESSENCE** ENTERPRISE 365, available at: https://www.ivarjacobson.com/essence-enterprise

20. **SEMAT** Essence Kernel Tool, available at: https://semat.herokuapp.com/

21. **Graziotin D., Abrahamsson P.** A Web-based modeling tool for the SEMAT Essence theory of software engineering, *Journal* of *Open Research Software*, 2013, vol. 1, no. 1, pp. e4, available at: http://dx.doi.org/10.5334/jors.ad.

22. **Azure** DevOps Services, available at: https://azure.microsoft.com/ru-ru/services/devops/

23. **Meta-Object** Facility, available at: https://www.omg.org/mof/

24. **Agile** Essentials | Practice Library, available at: https://practicelibrary.ivarjacobson.com/

25. **Gamma E., Helm R., Johnson R., Vlissides J.** *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994, 395 p.

26. **Danilenko A. N., Zmeev D. O., Zmeev O. A., Tamazlykar D. V.** Import of model of SEMAT Esscence practice workbench into software project management system Redmine, *Informacionnye tekhnologii i matematicheskoe modelirovanie (ITMM-2019): materialy XVIII Mezhdunar. konf. im. A. F. Terpugova*, 26—30 June 2019, Part. 1, Tomsk, NTL, 2019. pp. 27—30 (in Russian).

27. **Zmeev D., Zmeev O., Tamazlykar D.** Implementation of Essence Practice into project management system Redmine, *Actual Problems of Systems and Software Engineering (Invited Papers),* APSSE 2019, 12—14 November 2019, Moscow, proceedings. [S. l.]: IEEE Computer Society, 2019, pp. 116—125.

УДК 519.724.2 DOI: 10.17587/prin.11.322-329

И. Б. Казаков, аспирант, i_b_kazakov@mail.ru,
Московский государственный университет им. М. В. Ломоносова

Передача информации в каналах, задаваемых структурами частичного стирания. Часть 2*

Настоящая работа является частью 2 статьи автора, опубликованной ранее. Основным понятием, представленным в части 1 статьи, является структура частичного стирания. Настоящая работа посвящена построению протокола передачи информации в канале, задаваемом указанным понятием. Для данного протокола представлена математическая модель. Также построен конкретный пример, для которого оценена пропускная способность.

Ключевые слова: *скрытые каналы, блуждания по плоскости, структура частичного стирания, равномерное кодирование, пропускная способность*

Введение[1]

Настоящая работа является продолжением опубликованной ранее работы автора [1], посвященной скрытому каналу блужданий по плоскости. Блуждания по плоскости изучают в связи с задачей построения скрытого канала через многопользовательские online-игры. Имеется передающий информацию посредством движений по плоскости субъект, традиционно называемый Алисой. Считается, что Алиса движется с постоянной скоростью, причем она может двигаться в заранее зафиксированном числе направлений, отстоящих друг от друга на равные углы. За ее движениями наблюдает принимающий информацию субъект, традиционно называемый Бобом. Часть информации теряется: Боб видит местоположения Алисы не во все моменты времени, а только в некоторые.

В рамках изложенного в работе [1] материала указанные блуждания по плоскости были сведены к блужданиям в пространстве $\mathbb{Z}^k$. Рассматривается множество всех траекторий зафиксированной длины N. На данном множестве, соответственно выбору моментов, в которые Боб видит местоположение Алисы, определено 2^N разбиений. Каждому разбиению приписана вероятность, сумма всех приписанных вероятностей равна 1. Таким образом, на данном множестве траекторий определено то, что в работе [1] было названо структурой частичного стирания.

В настоящей работе представлен именуемый протоколом "равномерного кодирования" способ передачи информации по каналу, задаваемому указанной структурой, и построена его математическая модель. Также построен конкретный пример, для которого представлены оценки пропускной способности.

Представим структуру дальнейшего изложения. В разд. 1 напомнены введенные в работе [1] определения. Способ надстройки канала полного стирания описан в разд. 2. В разд. 3 изложена его математическая модель и доказаны теоремы об оценках средней пропускной способности, после чего в разд. 4 представлен конкретный пример. В Заключении подведены итоги и поставлены задачи для будущих исследований.

1. Основные определения

В данном разделе напомним определения из работы [1].

Определение 1.1. Структурой частичного стирания называется тройка $\left(A, \mathfrak{T}, \{p_T\}_{T\in\mathfrak{T}}\right)$, где A — это некий алфавит; $\mathfrak{T}$ — множество, состоящее из разбиений (или, что то же самое, отношений эквивалентности) алфавита A; $\{p_T\}_{T\in\mathfrak{T}}$ — веса-вероятности, приписанные указанным разбиениям. Общая сумма данных весов, поскольку они также названы вероятностями, принимается равной 1.

Сделаем необходимые пояснения. Периодически Алиса отправляет Бобу некий символ $a \in A$. В свою очередь, Боб получает лишь часть информации: сведения о том, какое именно разбиение T было выбрано, а также в каком классе $\pi_T(a)$ находился исходный символ a.

Множество пар вида $(\pi_T(a), T)$ для всех символов $a \in A$ и всех разбиений T обозначается как алфавит Боба B.

Определение 1.2. Справедливо отношение $a \mapsto b$, $a \in A$, $b \in B$, если $b = (\pi_T(a'), T)$ и $a \in \pi_T(a')$.

Определение 1.3. Справедливо отношение $\alpha \mapsto \beta$, $\alpha \in A^*$, $\beta \in B^*$, если $|\alpha| = |\beta| = k$ и $\forall i{=}1...k\ \ \alpha(i) \mapsto \beta(i)$.

2. Равномерное кодирование

Данный раздел посвящен описанию протокола, т. е. схемы поведения Алисы и Боба, называемого "равномерным кодированием". Дадим понятие равномерного кода в подразд. 2.1. Далее, в подразд. 2.2 надстроим канал полного стирания поверх канала частичного стирания, используя рассматриваемый протокол.

---

* Часть 1 статьи опубликована в журнале "Программная инженерия" № 5, 2020, с. 277—284.

2.1. Понятие

Прежде всего, будем предполагать, что у Алисы имеется некая входная лента, а у Боба — выходная. На входной ленте напечатаны символы из некоего алфавита *S*. Цель действий Алисы и Боба — отпечатать на выходной ленте копию того, что находится на входной, может быть, заменяя при этом некоторые символы дополнительным символом стирания *.

Простейшим способом организовать протокол с требуемыми свойствами является так называемое "равномерное кодирование". Его суть в следующем: поставим в соответствие каждому символу $s \in S$ некое слово α_s, которое Алиса будет передавать Бобу, прочитав *s* с входной ленты. Данное слово передается потактово, т. е. на протяжении $|\alpha_s|$ тактов Алиса последовательно высылает Бобу символы $\alpha_s(i)$ из алфавита *A*. В свою очередь, если на текущем такте произошло событие X_T, то Боб принимает символ $\beta(i) = (\pi_T(\alpha_s(i)), T)$.

Здесь принимается, что все слова α_s имеют одинаковую длину *n*, поэтому Боб всегда знает, в какой момент Алиса заканчивает передавать слово, соответствующее очередному символу на входной ленте.

Теперь рассмотрим действия Боба именно в один из моментов, когда Алиса заканчивает передачу. Начиная с предшествующего ему момента аналогичного характера (или с первого момента, если таковых не имеется) Боб последовательно получал символы $b_1, b_2, ..., b_i$ Значит, можно считать, что Боб принял слово $\beta = b_1 b_2 ... b_n \in B^*$. Согласно ранее представленным определениям выполнено $\alpha_s \mapsto \beta$.

В описанный выше момент Боб также должен что-то напечатать на выходной ленте. Для осуществления этого действия ему следует по принятому слову β определить изначально прочитанный Алисой с входной ленты символ *s*. Очевидно, что с точки зрения информации, имеющейся у Боба, Алиса могла прочитать любой из символов $s \in S$ такой, что $\alpha_s \mapsto \beta$. Существование хотя бы одного такого символа гарантированно самим определением равномерного кодирования. Если таковой символ единственен, то именно его Бобу и следует отпечатать. В противном случае следует отпечатать *, так как Алиса могла прочитать любой из данных символов, и Боб не располагает информацией о том, какой из них был в действительности ею прочитан.

Представим далее формализованное выражение сформулированных выше действий.

Определение 2.1. Равномерный код (соотнесенный со структурой частичного стирания) — это инъективное отображение $K : S \to A^n$.

Определение 2.2. Пусть дан некий равномерный код *K*. Множество $K(S) = \{\beta \in B^* \mid \exists s \in S\, K(s) \mapsto \beta\}$ назовем множеством принимаемых слов Боба.

Определение 2.3. Пусть дано некое $\beta \in K(S)$. Будем говорить, что β приписан символ *s*, если это единственный символ из *S*, для которого $K(s) \mapsto \beta$. Если таковых символов два и более, то считаем что β приписан символ стирания *.

Изучим теперь вопрос о вероятности печати символа *. Предположим опять, что Алиса прочитала с ленты символ *s*. Как отмечалось выше, это означает, что она отправила Бобу слово $K(s)=\alpha_s$. Боб, в свою очередь, может получить любое слово $\beta \in B^*$ такое, что $\alpha \mapsto \beta$.

Какое именно слово β было получено, зависит от того, какие отношения эквивалентности T_1, ..., T_n были выбраны из структуры частичного стирания. Очевидно, что вероятность данного получения равна произведению $p_\beta = p_{T_1} p_{T_2} ... p_{T_n}$.

Теперь отметим, что такие слова β, $\alpha_s \mapsto \beta$ делятся на два класса. Первому классу принадлежат те, которым приписан символ *s*. Второму классу — те, которым приписан символ стирания *. Следовательно, суммируя p_β по обоим классам, получаем вероятности p_0^s и p_1^s, $p_0^s + p_s^1 = 1$. Данные вероятности представляют собой вероятности отпечатывания на выходной ленте символов * и *s* соответственно.

Для практических целей порядок суммирования может быть изменен. Действительно, зафиксируем вновь разбиения T_1, ..., T_n. Выбор таковых разбиений, в свою очередь, задает отношения эквивалентности на множестве слов из A^n: $a_1 a_2 ... a_n \sim a'_1 a'_2 ... a'_n$ тогда и только тогда, когда $a_1 \overset{T_1}{\sim} a'_1$, $a_2 \overset{T_2}{\sim} a'_2$,..., $a_n \overset{T_n}{\sim} a'_n$. Заметим, что если положить $b_i = (\pi_{T_i}(a_i), T_i)$, то $a_1 ... a_n$, $a'_1 ... a'_n \mapsto b_1 ... b_n$.

Из представленных выше отношений следует, что данное отношение эквивалентности задает разбиение и на подмножестве $K(S) \subset A^n$, , причем два слова α_{s_1}, α_{s_2} отображаются в одно и то же слово β тогда и только тогда, когда принадлежат одному и тому же классу заданного разбиения.

Пусть Алиса отправила слово α, принадлежащее одноэлементному классу. Тогда по однозначно определяемому β, которое получил Боб, он, во-первых, сможет восстановить исходные разбиения T_1, ..., T_n, а во-вторых, само слово α. В противном случае, т. е. если α не принадлежало одноэлементному классу, Боб не может восстановить данное слово, поскольку не имеет информации для принятия решения, какое именно слово из класса было отправлено Алисой.

Исходя из представленных выше соображений, опишем алгоритм расчета вероятностей p_s. Прежде всего, объявим массив такой, что каждой его ячейке взаимно-однозначно соответствует некоторое слово $\alpha_s \in K(S)$. Далее, в цикле по всем возможным выборам разбиений $T_1 ... T_n$ следует добавлять вероятность $p_{T_1} p_{T_2} ... p_{T_n}$ к значениям ячеек, которым соответствуют члены одноэлементных классов разбиения, определеннего по T_1, ..., T_n описанным выше способом.

2.2. Структуры полного стирания

Подытожим изложенное выше. Алиса считывает с входной ленты символ *s* и отправляет Бобу слово α_s. Боб печатает на выходной ленте или * с вероятностью $1 - p_s$, или *s* с вероятностью p_s, причем отправка и прием занимают *n* тактов. Теперь можно осуществить очередную абстракцию и перейти к рассмотрению некого канала, определенного "поверх" исходного.

Определение 2.4. Структура полного стирания — это алфавит *S*, приписанные каждому символу $s \in S$ вероятности p_s, а также "цена отправки" *n*.

Интерпретация следующая: Алиса пересылает по каналу очередной символ $s \in S$. Боб, в свою очередь, с вероятностью $1 - p_s$ его успешно распознает (т. е. получает само s), а с вероятностью p_s терпит неудачу (т. е. получает символ стирания *).

Равномерное кодирование, таким образом, есть не что иное, как способ построения канала полного стирания поверх канала частичного стирания.

3. Среднее число циклов передачи

Рассмотрим вопрос о пропускной способности введенного выше канала полного стирания. Определим в подразд. 3.1 протокол, называемый циклической передачей, в подразд. 3.2 представим его математическую модель. Оценки среднего числа циклов передачи, которое является обратно пропорциональной пропускной способности величиной, представлены в подразд. 3.3 и доказаны в подразд. 3.4.

3.1. Циклическая передача символов

Будем полагать теперь, что задачей Алисы является отправка Бобу некого заранее определенного слова $\hat{s}=s_1...s_m \in S^*$ с использованием канала полного стирания. Осуществлять требуемое она будет, передавая составляющие символы в циклическом порядке: сначала s_1, потом s_2 и так далее до s_m, после чего вновь начиная с s_1.

Боб, в свою очередь, принимает отправленные Алисой символы. Предполагается, что он знает как длину исходного слова $\hat{s}$, так и то, какая по счету позиция данного слова передается на текущем такте.

Действия Боба, связанные с обработкой получаемой информации, далее опишем следующим образом. У него есть массив из m ячеек, в который могут быть записаны символы алфавита S. Предположим, что Алиса выслала i-ю позицию слова $\hat{s}$, т. е. символ s_i. В соответствии с определением структуры полного стирания Боб примет или само s_i, или *.

Если было принято s_i, то принятый символ записывается в i-ю ячейку массива. В противном случае, т. е. в случае принятия символа стирания *, Боб не совершает никаких действий.

Боб считает передачу информации завершенной, когда массив оказывается полностью заполненным. Действительно, после данного момента он уже знает, какое изначальное слово $\hat{s}$ передает Алиса.

3.2. Математическая модель

В предшествующей работе автора [2] была построена математическая модель, описывающая передачу информации посредством пересылки блоков. Имеется в виду, что блок не всегда пересылается удачно. В работе [2] было принято допущение, что вероятность удачной передачи блока неизменна, а сами события удачной передачи независимы.

В настоящем исследовании аналогичная ситуация, только вместо блоков речь идет о символах, а вероятность успешной передачи зависит от конкретного символа. С учетом полученных результатов далее представим учитывающую их математическую модель рассматриваемого процесса.

Обозначим как l^i случайную величину, значение которой равно номеру цикла, на котором Боб "примет i-ю позицию", т. е. запишет символ s_i в i-ю ячейку массива.

Утверждение 3.1.

$$P(l^i = k) = (1 - p_{s_i})^{k-1} p_{s_i}, \quad P(l^i \leqslant k) = 1 - (1 - p_{s_i})^k$$

Доказательство.

1. Действительно, передать символ s_i именно на k-м проходе цикла — передавать его неуспешно $k - 1$ начальных проходов, а успешно — именно на k-м.

2. $P(l^i \leqslant k) = P(l^i = 1) + ... + P(l^i = k) =$

$$= p_{s_i}\left(1 - p_{s_i}\right)^0 + p_{s_i}\left(1 - p_{s_i}\right)^1 + ... + p_{s_i}\left(1 - p_{s_i}\right)^{k-1} =$$

$$= p_{s_i}\left(1 + ... + (1 - p_{s_i})^{k-1}\right) = p_{s_i}\frac{1 - \left(1 - p_{s_i}\right)^k}{1 - \left(1 - p_{s_i}\right)} = 1 - \left(1 - p_{s_i}\right)^k.$$

Что и требовалось доказать.

Также введем обозначение для $L = \max(l^1, ..., l^m)$ — случайной величины, имеющей значение числа циклов передачи.

Интерес представляет ее математическое ожидание EL. В силу того, что у разных символов S разные вероятности успешной передачи, данное математическое ожидание зависит от состава слова $\hat{s}$.

Примем, что алфавит S состоит из d символов. Для удобства будем обозначать $S = \{s'_1, ..., s'_d\}$. Вероятности удачной пересылки переобозначим как $p_1, ..., p_d$, т. е. $p_i = p_{s'_i}$. Также для вероятностей неудачной пересылки будем использовать обозначение $q_i = 1 - p_i$. Полагаем далее, что слово $\hat{s}$ состоит из m_1 символов s'_1; m_2 символов s'_2, ..., m_d символов s'_d. Соответственно, $m_1 + m_2 + ... + m_d = m$. Обозначим соответствующий биноминальный коэффициент как $C_m^k = \frac{m!}{k!(m-k)!}$.

3.3. Представление результатов

Полагаем: $H_m = 1 + \frac{1}{2} + ... + \frac{1}{m}$ — гармоническая сумма.

Утверждение 3.2 (об оценке интегралом). Для среднего числа циклов передачи EL справедлива оценка $EL = I + \gamma$, где

$$I = \int_0^\infty \left(1 - \left(1 - q_1^k\right)\left(1 - q_2^k\right)...\left(1 - q_d^k\right)\right)dk, \quad \gamma \in [0, 1].$$

Теорема 3.1 (точное значение). Выполнено тождество

$$I = \sum_{0 \leqslant k_i \leqslant m_i, (k_1,...,k_d) \neq (0,...,0)} \frac{(-1)^{k_1+...+k_d} C_{m_1}^{k_1}...C_{m_d}^{k_d}}{k_1 \ln q_1 + ... + k_d \ln q_d}.$$

Теорема 3.2 (оценка снизу и сверху). Для представленного выше интеграла I также справедливы следующие оценки снизу и сверху:

$$I_1 \leqslant I \leqslant I_2,$$

где $I_1 = \int_0^\infty \left(1 - \left(1 - q_{\min}^k\right)^m\right)dk = -\frac{1}{\ln q_{\min}} H_m,$

$$I_2 = \int_0^{\infty}\left(1-\left(1-q_{\max}^k\right)^m\right)dk = -\frac{1}{\ln q_{\max}} H_m.$$

Замечание. Данная оценка в точности соответствует тому огрублению, как если бы среднее время передачи оценивалось посредством обращения внимания лишь на максимальную и минимальную вероятности успешной передачи, определенные в структуре полного стирания.

В заключение повторим вывод, сделанный в работе [2]. Как верхняя, так и нижняя оценки числа циклов передачи возрастают от m логарифмически. Отсюда следует, что пропускная способность, обратно пропорциональная среднему числу циклов передачи, с увеличением длины слова $\hat{s}$ уменьшается достаточно медленно, а именно как $\frac{1}{\ln m}$, что является приемлемым для практических целей.

3.4. Доказательства теорем

Предварительно выпишем некоторые факты, уже доказанные в работе [2].

Факт.

$$EL = \sum_{k=1}^{\infty} P(L \geqslant k) = \sum_{k=1}^{\infty}(1 - P(L \leqslant k-1)) =$$

$$= \sum_{k=0}^{\infty}(1 - P(L \leqslant k)).$$

Факт. Пусть имеется неотрицательная монотонно убывающая непрерывная функция f на интервале $[0, \infty)$. Тогда, если сходится ряд $\sum_{k=0}^{\infty} f(k)$, то также сходится и несобственный интеграл $\int_0^{\infty} f(k)dk$, и выполнена оценка ряда интегралом:

$$0 \leqslant \sum_{k=0}^{\infty} f(k) - \int_0^{\infty} f(k)\,dk \leqslant f(0).$$

Факт. Функция $f(k) = 1-\left(1-q^k\right)^m$, $0 < q < 1$, является непрерывной, неотрицательной, а также монотонно убывающей на интервале $[0, \infty)$. Также сходится ряд $\sum_{k=0}^{\infty} f(k)$.

Факт. $I(q, m) = \int_0^{\infty}\left(1-\left(1-q^k\right)^m\right)dk = -\frac{1}{\ln q} H_m.$

Принимаются следующие обозначения:

$$p_{\max} = \max(p_1, ..., p_d), \quad p_{\min} = \min(p_1, ..., p_d),$$

$$q_{\max} = 1 - p_{\min}, \quad q_{\min} = 1 - p_{\max}.$$

Подсчет математического ожидания EL, таким образом, следует начать с подсчета вероятностей $P(L \leqslant k)$.

Утверждение 3.3. Выполнено тождество

$$P(L \leqslant k) = \left(1-q_1^k\right)^{m_1}\left(1-q_2^k\right)^{m_2}...\left(1-q_d^k\right)^{m_d}.$$

Доказательство.

1. Действительно,

$$P(L \leqslant k) = P(\max(l^1, ..., l^m) \leqslant k) =$$

$$= P(l^1 \leqslant k)...P(l^m \leqslant k).$$

2. Среди значений $P(l^1 \leqslant k)$, $P(l^2 \leqslant k)$, ..., $P(l^m \leqslant k)$ имеются m_1 значений $\left(1-(1-p_1)^k\right)$, m_2 значений $\left(1-(1-p_2)^k\right)$, ..., m_d значений $\left(1-(1-p_d)^k\right)$.

3. Откуда, с учетом $q_i = 1 - p_i$, и следует требуемое.

Что и требовалось доказать.

Положим теперь $f(k) = 1 - P(L \leqslant k) = 1-\left(1-q_1^k\right)^{m_1}\left(1-q_2^k\right)^{m_2}...\left(1-q_d^k\right)^{m_d}$ и проверим условия оценки суммы ряда $\sum_{k=0}^{\infty} f(k)$ несобственным интегралом.

Лемма 3.1. Справедлива оценка $EL = I + \gamma$, где

$$I = \int_0^{\infty}\left(1-\left(1-q_1^k\right)^{m_1}\left(1-q_2^k\right)^{m_2}...\left(1-q_d^k\right)^{m_d}\right)dk, \quad \gamma \in [0, 1].$$

Доказательство.

1. Введем вспомогательное обозначение: $f_q(k) = 1 - q^k$. Тогда $f = 1 - f_{q_1}^{m_1}...f_{q_d}^{m_d}$.

2. Достаточно легко заметить, что функции f_q являются возрастающими и непрерывными. Следовательно, сама f убывает и также является непрерывной функцией. Очевидно также, что $f \geqslant 0$ на интервале $[0, \infty)$.

3. Поскольку из $q_1 \leqslant q_2$ следует $f_{q_1} \geqslant f_{q_2}$, то $0 \leqslant f \leqslant 1 - f_{q_{\max}}^m = 1-\left(1-q_{\max}^k\right)^m$.

4. Ряд $\sum_{k=0}^{\infty}\left(1-\left(1-q_{\max}^k\right)^m\right)$ сходится. Следовательно, сходится и ряд $\sum_{k=0}^{\infty} f(k)$.

5. То есть выполнены все условия для оценки ряда интегралом. Применяя данную оценку, с учетом $EL = \sum_{k=0}^{\infty}(1 - P(L \leqslant k)) = \sum_{k=0}^{\infty} f(k)$, получаем требуемое.

Утверждение 3.2 доказано.

Подсчитаем теперь введенный выше интеграл. Представим сначала его точное значение.

Утверждение 3.4. Выполнено тождество

$$I = I(q_1, m_1, q_2, m_2, ... q_d, m_d) =$$

$$= \int_0^{\infty}\left(1-\left(1-q_1^k\right)\left(1-q_2^k\right)...\left(1-q_d^k\right)\right)dk =$$

$$= \sum_{0 \leqslant k_i \leqslant m_i,(k_1,...,k_d)\neq(0,...,0)} \frac{(-1)^{k_1+...+k_d} C_{m_1}^{k_1}...C_{m_d}^{k_d}}{k_1 \ln q_1 + ... + k_d \ln q_d}.$$

Доказательство.

1. В первую очередь осуществим замену переменных: $k = \ln y$. k меняется в пределах от 0 до ∞, следовательно, $y = \mathbf{e}^k$ меняется от 1 до ∞. Также

заменим параметры: $r_i = \ln q_i$. Так как предполагается, что $0 < q_i < 1$, то $r_i < 0$. Таким образом, запишем:

$$I = \int_1^{\infty} \left(1 - \left(1 - y^{\ln q_1}\right)^{m_1} \left(1 - y^{\ln q_2}\right)^{m_2} \ldots \left(1 - y^{\ln q_d}\right)^{m_d}\right) \frac{dy}{y}.$$

2. Теперь раскроем скобки в произведении под интегралом:

$$I = \int_1^{\infty} \left(1 - \sum_{k_1=0}^{m_1} \sum_{k_2=0}^{m_2} \ldots \sum_{k_n=0}^{m_n} C_{m_1}^{k_1} \ldots C_{m_n}^{k_n} (-1)^{k_1+\ldots+k_n} y^{k_1 r_1 + \ldots + k_n r_n}\right) \frac{dy}{y} =$$

$$= \int_1^{\infty} \sum_{0 \leqslant k_i \leqslant m_i, (k_1,\ldots,k_n) \neq (0,\ldots,0)} C_{m_1}^{k_1} \ldots C_{m_n}^{k_n} (-1)^{k_1+\ldots+k_n+1} y^{k_1 r_1 + \ldots + k_n r_n - 1} dy =$$

$$= \sum_{0 \leqslant k_i \leqslant m_i, (k_1,\ldots,k_n) \neq (0,\ldots,0)} C_{m_1}^{k_1} \ldots C_{m_n}^{k_n} (-1)^{k_1+\ldots+k_n+1} \int_1^{\infty} y^{k_1 r_1 + \ldots + k_n r_n - 1} dy.$$

3. Легко проверить, что при всех $r < 0$ выполняется $\int_1^{\infty} y^{r-1} dy = -\frac{1}{r}$.

4. Как следствие,

$$I = \sum_{0 \leqslant k_i \leqslant m_i, (k_1,\ldots,k_n) \neq (0,\ldots,0)} \frac{(-1)^{k_1+\ldots+k_n} C_{m_1}^{k_1} \ldots C_{m_n}^{k_n}}{k_1 r_1 + \ldots + k_n r_n}.$$

Подставляя обратно $r_i = \ln q_i$, получаем требуемое. Что и требовалось доказать.

Доказана теорема 3.1. Оценка представленной выше многомерной знакопеременной суммы является нетривиальной задачей. Для практических целей возможно использовать простые оценки сверху и снизу, рассчитываемые по значениям $q_{\min}$ и $q_{\max}$.

Сделаем необходимые пояснения. Так как выполнено

$$\left(1 - \left(1 - q_{\min}^k\right)\right) \leqslant \left(1 - \left(1 - q_1^k\right)^{m_1} \ldots \left(1 - q_d^k\right)^{m_d}\right) \leqslant$$
$$\leqslant \left(1 - \left(1 - q_{\max}^k\right)\right),$$

то и для интеграла $I_1 \leqslant I \leqslant I_2$, где

$$I_1 = \int_0^{\infty} \left(1 - \left(1 - q^k\right)^m\right) dk = -\frac{1}{\ln q_{\min}} H_m,$$

$$I_2 = \int_0^{\infty} \left(1 - \left(1 - q^k\right)^m\right) dk = -\frac{1}{\ln q_{\max}} H_m.$$

Что и составляет утверждение теоремы 3.2.

4. Пример модельной реализации

Построим конкретный пример канала передачи информации через блуждания по плоскости и оценим вероятность ошибки, а также его пропускную способность. В подразд. 4.1 представим параметры данного примера, а также сформулируем и докажем утверждение о средней пропускной способности. Таблицы значений вероятности ошибки и пропускной способности представим в подразд. 4.2, 4.3 соответственно.

4.1. Параметры

Напомним, что в части 1 работы в качестве структуры частичного стирания рассматривалось множество траекторий блужданий по плоскости фиксированной длины *N*. При этом предполагалось, что движения идут в фиксированном числе направлений *j*, отстоящих друг от друга на равные углы. Выберем параметры: $j = 6$ и $N = 4$, т. е. движения фактически происходят по шестиугольному паркету, а траектории имеют четыре шага. Таким образом, всего в наличии $6^4 = 1296$ различных траекторий. Соответственно, выбор одной из имеющихся содержит $\log_2(1296) \approx 10,3399$ бит информации.

Сама траектория передается за четыре такта. Однако для гарантированного разделения Бобом последовательно передаваемых траекторий между отдельными передачами должен быть интервал выжидания. Поясним, что под "разделением траекторий" имеется в виду отделение конечного момента передачи предыдущей траектории от начального момента последующей. Между этими моментами, таким образом, должен быть интервал времени, на котором считается, что передачи информации не происходит. Примем, что длительность тактового интервала составляет также четыре такта. Таким образом, передача одного символа из соответствующей структуры частичного стирания занимает восемь тактов.

Далее необходимо указать вероятности, приписанные всем возможным разбиениям множества траекторий. Таких имеется $2^4 = 16$. Для указанной цели примем следующее. События вида "на данном такте Боб получил сведения о местоположении Алисы" независимы, а все их вероятности принимаются равными некой заранее определенной постоянной величине p_{base}. С учетом этого обстоятельства, отношению эквивалентности T будет приписана вероятность $p_{base}^v \left(1 - p_{base}\right)^{4-v}$, где v — это число тактов, в которые Боб видит Алису. Будут рассматриваться ее значения $p_{base} = 0,1,\ 0,2,\ \ldots,\ 0,9$.

Опишем равномерное кодирование, надстраивающее канал полного стирания под каналом частичного стирания. Вообще говоря, так как равномерное кодирование является обобщением стратегии "передавать один и тот же символ несколько раз подряд", то в данном случае именно таковую стратегию и будем использовать. Это означает, что $S = A$, $K(a) = aa...a$. Число повторений, далее обозначаемое как *repeat*, будет варьироваться от 1 до 4.

В соответствии с тем, как было определено ранее в теоретической части, для каждой траектории $a \in A = S$ определены вероятности успешной передачи p_a.

Следует также определиться с длиной слова $\hat{s}$, циклически передаваемого Алисой. Примем $|\hat{s}| = 1024$. Так как на практике требуется отправлять неограниченный объем информации, то фактически это означает, что иногда (а именно по окончании получения очередного блока из 1024 символов) Бобу все-таки дозволена обратная связь, т. е. дозволено подавать Алисе сигнал вида "передача завершена".

Для оценок пропускной способности понадобится гармоническая сумма:

$$H_{1024} = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{1024} \approx 7,5092.$$

Утверждение 4.1. Средняя пропускная способность канала, имеющего определенные выше параметры, равна $\log_2(1296)\frac{FPS/8}{repeat * EL}$ бит/с, где FPS — число тактов в секунду; EL — среднее число циклов передачи.

Доказательство.

1. Действительно, блок из 1024 символов из алфавита A содержит $1024\log_2(1296)$ бит информации, так как $|A| = 1296$.

2. Каждый цикл передачи, очевидно, состоит из 1024*repeat* актов передачи траекторий из A. Всего в среднем осуществляется $EL1024repeat$ актов передачи.

3. Каждый акт передачи, как было отмечено ранее, занимает 8 тактов. Таким образом, в секунду их происходит $\frac{FPS}{8}$.

4. Сопоставив выводы п. 1—3, получаем требуемое.

Что и требовалось доказать.

Далее для оценки пропускной способности зафиксируем значение $FPS = 72$.

В представленном выше утверждении упомянуто среднее число циклов EL. Однако вместо точного его значения будем использовать оценки сверху и снизу, полученные в теоретической части настоящей работы. Это означает, что $EL = I + \gamma$, $I_{\min} \leqslant I \leqslant I_{\max}$, где $I_{\min} = \frac{-1}{\ln(1 - p_{\max})}$, $I_{\max} = \frac{-1}{\ln(1 - p_{\min})}$, $\gamma \in [0, 1]$.

Будем пользоваться представленной в утверждении формулой, заменяя EL оценками $I_{\min}$, $I_{\max}$, тем самым получая оценки пропускной способности снизу и сверху соответственно. Таким образом, среди всех вероятностей p_a следует найти минимальную $p_{\min}$ и максимальную $p_{\max}$. Соответственно будут найдены верхние и нижние оценки пропускной способности.

Программная реализация расчетов состоит из трех последовательных этапов: перечисления траекторий, подсчета вероятностей p_a и вычисления пропускной способности. Поясним подробнее каждый из перечисленных этапов.

Для перечисления траекторий прежде всего необходим круговой многочлен $\Phi_6(x) = x^2 - x + 1$, а также остатки деления на него одночленов 1, x, x^2, x^3, x^4, x^5. Перечислим соответственно данные остатки: 1, x, $-1 + x$, -1, $-x$, $1 - x$. Таким образом, Алиса блуждает по $\mathbb{Z}^2$, за один такт передвигаясь по одному из шести направлений, которым соответствуют следующие векторы: (1, 0), (1, 1), (−1, 1), (−1, 0), (0, −1), (1, −1). Рассматривая последовательности из представленных векторов, получаем требуемые траектории.

Подсчет вероятностей успешного приема будет осуществлен способом, уже указанным в теоретической части настоящей работы: для каждого возможного выбора отношений эквивалентности T_1, ..., T_{repeat} нужно найти одноэлементные классы соответственно задаваемого им разбиения. Относительно деталей отметим, что фактически речь идет о задаче поиска уникальных строк в таблице. Пропускная способность оценивается согласно формуле из представленного выше утверждения.

Исходный код программной реализации размещен в github-репозитории https://github.com/ibkazakov/article5

Представим далее таблицы промежуточных и конечных результатов. Под промежуточными результатами понимаются значения $p_{\min}$, $p_{\max}$, под конечными — верхние и нижние оценки пропускной способности канала.

В целях сравнения приведем также две особых оценки пропускной способности, не относящиеся непосредственно к представленной выше модели.

Предполагается, что Боб видит местоположение Алисы на всех тактах. В первом случае предполагается, что передача ведется посредством использования алфавита траекторий длины 4, и что на каждые 4 такта передачи приходятся 4 такта паузы. Во втором случае предполагается, что никаких пауз нет, т. е. информация передается непосредственно на каждом такте.

Соответственно, пропускная способность в первом случае составляет $\frac{FPS}{8}\log_2(1296) = 36\log_2(6) \approx$ $\approx 93,05865$ бит/с. Во втором случае она равна $FPS \log_2(6) = 72\log_2(6) \approx 186,1173$ бит/с.

4.2. Таблицы вероятностей

Представим в табличном виде значения вероятностей стирания для различных значений вероятности фиксирования Бобом местоположения Алисы. Значения минимальной и максимальной вероятностей представлены в табл. 1, 2 соответственно.

4.3. Таблицы оценок пропускной способности

На основании формулы, представленной выше в утверждении 4.1, рассчитаем также нижние и верх-

Таблица 1

Значения $p_{\min}$

p_{base}	Число повторений *repeat*			
	1	2	3	4
0,1	0,0001	0,0013	0,0054	0,0140
0,2	0,0016	0,0168	0,0567	0,1215
0,3	0,0081	0,0677	0,1863	0,3334
0,4	0,0256	0,1678	0,3778	0,5740
0,5	0,0625	0,3164	0,5862	0,7725
0,6	0,1296	0,4979	0,7675	0,9015
0,7	0,2401	0,6857	0,8963	0,9680
0,8	0,4096	0,8493	0,9684	0,9936
0,9	0,6561	0,9606	0,9960	0,9996

Таблица 2

Значения p_{max}

p_{base}	Число повторений *repeat*			
	1	2	3	4
0,1	0,1000	0,1900	0,2710	0,3439
0,2	0,2000	0,3600	0,4880	0,5904
0,3	0,3000	0,5100	0,6570	0,7600
0,4	0,4000	0,6370	0,7840	0,8704
0,5	0,5000	0,7500	0,8750	0,9375
0,6	0,6000	0,8400	0,9360	0,9744
0,7	0,7000	0,9100	0,9730	0,9919
0,8	0,8000	0,9600	0,9920	0,9984
0,9	0,9000	0,9900	0,9990	0,9999

Таблица 3

Оценка пропускной способности снизу, бит/с

p_{base}	Число повторений *repeat*			
	1	2	3	4
0,1	0,0012	0,0080	0,0223	0,0436
0,2	0,0198	0,1050	0,2412	0,4013
0,3	0,1008	0,4340	0,8517	1,2567
0,4	0,3214	1,1379	1,9601	2,6434
0,5	0,7998	2,3570	3,6448	4,5868
0,6	1,7201	4,2686	6,0272	7,1795
0,7	3,4026	7,1727	9,3615	10,6631
0,8	6,5304	11,7282	14,2682	15,6577
0,9	13,2280	20,0382	22,8147	24,2406

ние оценки пропускной способности и представим их в табл. 3, 4 соответственно.

Замечание. Согласно табл. 4 верхние оценки пропускной способности совпали для всех значений числа повторений. Однако исследование теоретических объяснений данного экспериментального факта выходит за рамки настоящей работы.

Таблица 4

Оценка пропускной способности сверху, бит/с

p_{base}	Число повторений *repeat*			
	1	2	3	4
0,1	1,3057	1,3057	1,3057	1,3057
0,2	2,7653	2,7653	2,7653	2,7653
0,3	4,4202	4,4202	4,4202	4,4202
0,4	6,3305	6,3305	6,3305	6,3305
0,5	8,5899	8,5899	8,5899	8,5899
0,6	11,3552	11,3552	11,3552	11,3552
0,7	14,9204	14,9204	14,9204	14,9204
0,8	19,9452	19,9452	19,9452	19,9452
0,9	28,5352	28,5352	28,5352	28,5352

Заключение

В качестве заключения подведем итоги, перечислив поставленные в настоящей работе задачи, а также полученные результаты их решения. Прежде всего, представлено равномерное кодирование — конкретное описание протокола передачи информации через канал частичного стирания. Изучена математическая модель, получены результаты о среднем числе циклов передачи: точное значение, а также оценки сверху и снизу. Рассмотрен пример и получены конкретные оценки пропускной способности. Также приведено обсуждение методов борьбы с организацией скрытого канала, описанного в настоящей работе.

На рассматриваемом направлении предполагаются следующие исследования. Во-первых, заметим, что был построен всего лишь один довольно простой пример равномерного кода. Следовательно, существует задача перебора аналогичных кодов в целях максимизации пропускной способности.

Во-вторых, равномерное кодирование, описанное в настоящей статье, предполагает, что поведение Алисы не зависит от предыстории, т. е. от уже прочитанных с входной ленты символов алфавита S. Данное предположение можно отбросить, проведя, тем самым, обобщение понятия кода. Соответственно, возникает нетривиальная задача построения поведения Боба таким образом, чтобы на выходной ленте в конечном итоге отпечатывалось то же самое, что напечатано на входной, может быть, с заменой некоторых символов символом стирания.

Список литературы

1. **Казаков И. Б.** Передача информации в каналах, задаваемых структурами частичного стирания. Часть 1 // Программная инженерия. — 2020. — Т. 11, № 5. — С. 277—284.
2. **Казаков И. Б.** Разностный код и протокол циклической поблочной передачи в скрытом канале по памяти // Программная инженерия. — 2019. — Т. 10, № 5. — С. 204—218.

Transmission of Information in Channels Specified by Structures of Partial Erasure. Part 2

I. B. Kazakov, i_b_kazakov@mail.ru, Lomonosov Moscow State University, Moscow, 119234, Russian Federation

Corresponding author:

Kazakov Ilia B., Postgraduate Student, Lomonosov Moscow State University, Moscow, 119234, Russian Federation E-mail: i_b_kazakov@mail.ru

Received on May 06, 2020
Accepted on July 08, 2020

The paper is the second part of the research. The main notion introduced in the first part is a structure of partial erasure. The second part is devoted to construction of a protocol for transmitting information via a channel specified by such structure. We formulate the mathematical model of the protocol, give an explicit example of implementation and estimate throughput of this example.

Keywords: *covert channels, walk on a plane, structure of partial erasure, uniform coding, throughput*

For citation:

Kazakov I. B. Transmission of Information in Channels Specified by Structures of Partial Erasure. Part 2, *Programmnaya Ingeneria,* 2020, vol. 11, no. 6, pp. 322—329

DOI: 10.17587/prin.11.322-329

References

1. **Kazakov I. B.** Information transmitting in a Channel Definined by a Partial Erasure Structure. Part 1, *Programmnaya Ingeneria*, 2020, vol. 11, no. 5, pp. 277—284 (in Russian).

2. **Kazakov I. B.** Difference Code and a Protocol for Cyclic Blockwise Transmission in a Memory-Based Covert Channel, *Programmnaya Ingeneria*, 2019, vol. 10, no. 5, pp. 204—218 (in Russian).

ИНФОРМАЦИЯ

УДК 004.422 DOI: 10.17587/prin.11.330-334

Р. А. Карелова, канд. пед. наук, доц. кафедры, riya2003@mail.ru,
Е. Е. Игнатов, студент, levia4119@gmail.com, Нижнетагильский технологический институт (филиал) Уральского федерального университета имени первого Президента России Б. Н. Ельцина

Особенности реализации нейронной сети для автоматизации процессов распознавания дефектов стали

Представлен вариант реализации нейронной сети для распознавания дефектов на изображениях листов стали. В качестве средств разработки предложены язык Python, библиотека PyTorch, среда разработки Jupyter, архитектура сверточной нейронной сети Unet. Описаны особенности анализа входных изображений листов стали, особенности реализации самой нейронной сети. Точность полученной модели — 84 %. Предложенное решение может рассматриваться как часть программно-аппаратной системы автоматизации процессов распознавания дефектов на листах металла.

Ключевые слова: *цифровизация производства, нейронная сеть, дефекты стали, распознавание изображений, автоматизация производства, python, pytorch, архитектура unet*

Введение

Современные тенденции перехода экономики на новый технологический уклад обусловливают модернизацию отечественных производственных систем. На промышленных предприятиях активизируются процессы автоматизации и роботизации производства, растет уровень его цифровой интеллектуализации. В технологические процессы внедряют цифровые технологии, что позволяет сокращать временные и финансовые затраты на производство, уменьшать последствия влияния человеческого фактора [1, 2]. Не является исключением и металлургия, в частности, сталеплавильное производство. Сталь — это один из важнейших строительных материалов нашего времени. Множество предприятий по всему миру обрабатывают сталь. При этом произведенные листы стали могут содержать дефекты, отслеживание которых является важной составляющей всего производственного процесса.

Автоматизация выявления дефектов стали пока не является процессом унифицированным. В настоящее время, как правило, каждое предприятие, производящее листы стали, решает вопросы автоматизации выявления дефектов по-своему.

Одним из путей решения обозначенных вопросов является применение информационно-измерительной системы на основе рентгеновских лучей. Наличие дефектов на листе металла ослабляет лучи, и анализируя интенсивность, можно определять структурные неоднородности. Ультразвуковой метод использует упругие колебания ультразвукового диапазона. Нарушения однородности листа влияют на колебания и считываются датчиками.

Альтернативой указанным способам определения дефектов в листах стали, не требующей громоздкого оборудования, а также способной определять даже мелкие дефекты, является автоматизация с помощью внедрения искусственной нейронной сети [3]. Искусственная нейронная сеть — это математическая модель, функционирующая по принципу биологических нейронных сетей. В данном случае над стальной пластиной помещается камера, которая сканирует лист, а обученная нейросеть определяет, где и какой именно дефект присутствует на заготовке, после чего решает, что делать с ней дальше.

Постановка задачи

В настоящей работе рассматривается предлагаемый авторами подход к автоматизации процессов распознавания дефектов стали.

Объект исследования — подход к применению методов машинного обучения и нейронных сетей для автоматизации различных типов производств. Предмет исследования — методы использования нейронной сети для распознавания дефектов на листах стали. Целью работы, результаты выполнения которой представлены в статье, являлась разработка нейронной сети, способной распознавать дефекты на изображениях листов стали.

Чтобы максимально эффективно и быстро прийти к желаемому результату, необходимо было составить план. Поэтому работа была разбита на этапы, в течение которых решались следующие задачи:

- выбор среды разработки, языка программирования и необходимых для реализации библиотек;
- анализ данных, построение графиков, гистограмм, нахождение зависимостей;
- выбор подходящей по функциональным возможностям нейронной сети и ее архитектуры, выбор алгоритма оценивания качества работы и точности полученных результатов;

- программная реализация нейронной сети;
- обучение и проверка точности и качества результатов работы, проверка на оверфиттинг (переобучение);
- исправление недочетов, неточностей, а при необходимости, возвращение на предыдущие этапы.

Далее указанные этапы работы будут рассмотрены подробнее.

Выбор средств разработки

Для создания программного средства был выбран язык программирования Python, широко применяемый в настоящее время для машинного обучения ввиду своей высокой производительности при обработке больших данных. Этот язык имеет большое число библиотек машинного обучения, таких как Keras, TensorFlow, PyTorch, вычислительные библиотеки, средства построения графиков, например, numpy, matplotlib, удобную библиотеку обработки данных из программы Excel — Pandas [4]. Для реализации программного средства в рамках подлежащей решению задачи была выбрана библиотека PyTorch, которая позволяет строить вычислительный динамический граф.

Средой разработки программного обеспечения была выбрана платформа Jupyter. Она представляет собой веб-приложение с открытым исходным кодом, которое позволяет представлять проекты по машинному обучению в интерактивном виде, предоставляет возможность демонстрировать графики, математические уравнения.

Анализ данных

Следующим шагом стала оценка данных, которые подлежали обработке. Данные были представлены в виде двух таблиц и двух файлов с изображениями листов стали — тестовыми и тренировочными, соответственно. На рис. 1 приведен пример изображения листа стали.

На рис. 2 представлены первые пять строк фрагмента Excel-таблицы.

В представленной на рис. 2 таблице с данными три озаглавленных столбца: первый — идентификационный номер изображения листа стали (ImageId), второй — класс дефекта, принадлежащий диапазону от одного до четырех включительно (ClassId), третий — строка с закодированным местоположением дефекта (EncodedPixels). Данная строка зашифрована по такому алгоритму: первое число — это начальный пиксель, с которого начнется отсчет, следующее за ним — это длина пробега, т. е. сколько пикселей нужно отступить от начального. Так, если дано два числа 2 и 5, то имеется в виду, что начало идет со второго пикселя и так до пятого (2, 3, 4, 5). Пиксели нумеруются сверху вниз, затем слева направо.

Рис. 1. Фотография листа стали

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
1	0007a71bf.jpg	3	18661 28 18863 82 19091 110 19347 110 19603 11...
2	000a4bcdd.jpg	1	37607 3 37858 8 38108 14 38359 20 38610 25 388...
3	000f6bf48.jpg	4	131973 1 132228 4 132483 6 132738 8 132993 11 ...
4	0014fce06.jpg	3	229501 11 229741 33 229981 55 230221 77 230468...

Рис. 2. Фрагмент таблицы с данными о листах стали

Следует отметить, что на листах металлов могут присутствовать более 60 разных дефектов, появление которых обусловлено разными причинами. В настоящей работе рассматриваются четыре наиболее часто встречаемые в производстве дефекта: царапина, плена, расслоение и сквозной разрыв, которые были пронумерованы числами от одного до четырех.

Сначала данные были исследованы на число листов стали с дефектами и без дефектов. В результате проверки получилось, что 117 листов стали не имеют изъянов, а 6550 — имеют.

Составим гистограмму числа дефектов (листинг 1).

```
   fig, ax = plt.subplots() #размещение формы
гистограммы на экране
   sns.barplot(x=list(class _ dict.keys()),
y=list(class _ dict.values()), ax=ax)
   #заполнение гистограммы данными
   ax.set _ title("Количество изображений
с данным дефектом") #добавление заголовка графика
   ax.set _ xlabel("Класс дефекта") #добавление
названия параметра гистограммы
   class _ dict
```

Листинг 1. Гистограмма распределения числа дефектов по классам

Построенная с помощью библиотеки matplotlib гистограмма представлена на рис. 3.

Из гистограммы можно сделать вывод, что самым частым видом дефекта является класс 3 — около 70 % всех листов стали. По этой гистограмме можно отметить, что классы несбалансированы. Необходимо было установить, может ли на одном изображении быть несколько дефектов одновременно. Для этого был написан код, представленный на листинге 2,

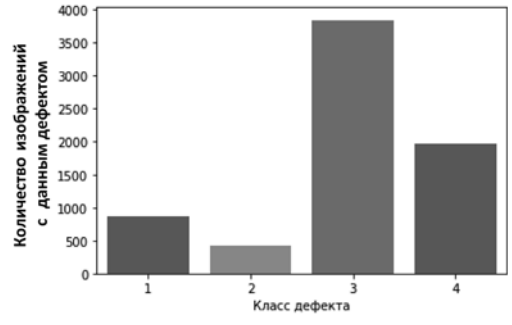

Рис. 3. Гистограмма распределения числа дефектов по классам

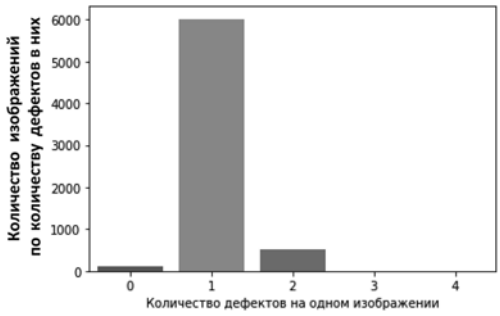

Рис. 4. Гистограмма числа дефектов к числу изображений

а полученный результат был визуализирован с помощью библиотеки matplotlib (рис. 4).

```
   fig, ax = plt.subplots() #размещение формы
гистограммы на экран
   sns.barplot(x=list(kind _ class _ dict.
keys()), y=list(kind _ class _ dict.values()),
ax=ax)#заполнение гистограммы данными
   ax.set _ title("Количество изображений
по количеству дефектов в них"); #добавление
заголовка графика
   ax.set _ xlabel("Количество дефектов на одном
изображений") #добавление названия параметра
гистограммы
   kind _ class _ dict
```

Листинг 2. Гистограмма числа дефектов к числу изображений

На оси абсцисс отложено число дефектов на изображении, а на оси ординат — число этих изображений. Из гистограммы можно сделать вывод, что несколько изображений имеют два дефекта.

Рассмотрим визуально, как выглядят дефекты по классам. Для этого была написана функция (листинг 3), которая отсортировала изображения по типам дефектов и обвела каждый из них определенным цветом. Под маской здесь подразумевается выделенная область изображения.

```
   def name _ and _ mask(col)
      labels = train _ df.iloc[col:col+4, 1]
#внесение значений тренировочных данных в
словарь определенного размера
      mask = np.zeros((256, 1600, 4), dtype=np.
uint8) #создание маски и заполнение ее нулями
   #проход по словарю тренировочных данных
   for idx, label in enumerate(labels.values):
      if label is not np.nan:
      mask _ label = np.zeros(1600*256, dtype =
      np.uint8) #заполнение
      label = label.split(" ")
      positions = map(int, label[0::2])
      length = map(int, label[1::2])
      #обвод дефектного участка
      for pos, le in zip(positions, length):
         mask _ label[pos-1:pos + le-1] = 1
      mask[:, :, idx] = mask _ label.reshape(256,
1600, order = 'F') #запись значений в массив
   return img _ names[0], mask
```

Листинг 3. Функция обведения дефектов на изображениях

Результат работы функции, описанной выше, представлен на рис. 5—8 (см. третью сторону обложки), где изображены листы стали с выделенными дефектами каждого класса (на осях дан размер в пикселях).

Далее было необходимо проанализировать данные и посмотреть у одного листа несколько дефектов сразу. Для этого функция была изменена, а цвета для классов остались прежними. На рис. 9 и 10 (см. четвертую сторону обложки) даны изображения сразу с двумя и тремя дефектами соответственно.

Для того чтобы нейронная сеть могла распознавать дефект и классифицировать его, необходимо было написать функцию препроцессинга изображений. В отмеченной ранее таблице данных (см. рис. 2) три строки: имя изображения, класс дефекта, а также пиксель дефекта, который представляет собой набор чисел, где на нечетном месте стоят координаты пикселя, а на четном — расстояние от этой координаты. Для того чтобы научить нейросеть распознавать дефекты, ей посылались изображения с выделенной маской брака. Для этого была написана функция, приведенная на листинге 4.

```
   def make _ mask(index, data):
      fname = data.iloc[index].name #запомина-
ние названия изображения
      labels = data.iloc[index][:4] #запоминание
данных изображения
      masks = np.zeros((256, 1600, 4), dtype =
np.float32) #создание массива, заполненного ну-
лями
      #проход по словарю данных
      for idx, label in enumerate(labels.values):
         if label is not np.nan:
            label = label.split(" ")
            positions = map(int, label[0::2])
            length = map(int, label[1::2])
            mask = np.zeros(256 * 1600, dtype=np.uint8)
            #обвод дефектной области
            for pos, le in zip(positions, length):
               mask[pos:(pos+le)] = 1
            masks[:, :, idx] = mask.reshape(256,
1600, order = 'F') #перезапись значений в массив
   return fname, masks
```

Листинг 4. Функция создания маски

На вход функции подается индекс строки, а также таблица с данными о листах. Функция запоминает имя изображения и строит маску. Для этого создается трехмерный массив, изначально заполненный нулям, размером $256 \times 1600 \times 4$. Первые два значения — это размер изображения, а последнее — измерение класса ошибки, т. е. число классов ошибок. Сделано это так как на одном изображении может быть больше одного дефекта. В итоге функция возвращает название изображения и маску с ее дефектом.

Реализация нейронной сети

Для решения задач распознавания и классификации изображений оптимальным вариантом является применение сверточной нейронной сети [5]. В нашем

случае число входов было равно $256 \times 1600 \times 1$, так как изображения были размером 256×1600 с одним дефектом. Создавалась нейронная сеть с помощью библиотеки машинного обучения PyTorch.

Архитектура сверточной нейронной сети была определена по аналогии с работой, посвященной распознаванию биомедицинских изображений [6]. Архитектура Unet состоит из сокращающегося пути для захвата контекста и симметричного расширяющегося пути, который обеспечивает точную локализацию. Написать подобную свертку можно и самостоятельно [7].

Далее была выбрана модель для сегментации. Необходима она для увеличения точности распознавания. Она имеет предварительно обученные веса, а также позволяет работать с Unet. Выбрана была модель resnet18 [8].

Обучение и проверка точности

В качестве критерия оценки точности работы нейросети была выбрана функция бинарной кросс-энтропии, так как она стремится приблизить распределение прогноза сети к целевому, штрафуя не только за ошибочные предсказания, но и за неуверенные [9]. Для дополнительной оценки также использовался метод DICE. Индекс известен под несколькими другими именами, а именно индекс Сёренсена—Дайса, индекс Соренсена и коэффициент Дайса [10]. Он показывает точность работы, основываясь на сравнении пикселей предсказанной модели и истинной модели.

Число эпох было задано равным 30. Каждая эпоха нейросети сохраняется и, если наблюдается оверфиттинг, нейросеть прекращает тренировки и возвращается к последнему сохраненному варианту нейросети.

В итоге нейросеть была натренирована до 84 %. Для проверки была написана функция (листинг 5), которая накладывает предсказанные пиксели на изображение.

```
def rle2mask(rle, imgshape):
   width = imgshape[0]
   height = imgshape[1]
   mask = np.zeros(width*height).astype(np.uint8)
#создание маски размером с входящее изображение

   array = np.asarray([int(x) for x in rle.
split()]) #перевод входной последовательности в
массив чисел
   starts = array[0::2]
   lengths = array[1::2]

   current_position = 0
   #выделение дефектной области
   for index, start in enumerate(starts):
      mask[int(start):int(start+lengths[index])] = 1
      current_position+ = lengths[index]

   return np.flipud(np.rot90(mask.
reshape(height,width), k = 1)) #поворот маски
на 90 градусов и отражение ее по вертикали

fig = plt.figure(figsize = (20,100))#создание области
размером 20×100
columns = 2
rows = 50
for i in range(1, 100+1):
   fig.add_subplot(rows, columns, i) #добавление
графики

   fn = train['ImageId_ClassId'].iloc[i].split('_')[0]
   img = cv2.imread('../second/data/test_
images/' + fn) #получение исходного изображения
   img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
#замена цветового пространства на RGB
   mask = rle2mask(train['EncodedPixels'].iloc[i],
img.shape) #наложение значения на изображения
   img[mask == 1,0]=255 #окрашивание предсказан-
ных значений красным

   plt.imshow(img) #отображение результата как
изображения
plt.show() #вывод на экран полученных результатов
```

Листинг 5. Функция отображения предсказанных дефектов

Представленная на листинге 5 функция преобразует полученные значения в массив, а затем накладывает его на изображение, окрашивая предсказанные значения красным цветом. Результат работы функции можно видеть на рис. 11 (см. четвертую сторону обложки).

Заключение

Как уже было отмечено ранее, точность модели достигла 84 %. В редких случаях программное средство не полностью выделяет маску дефектов, однако это в целом не оказывает негативного эффекта, так как факт наличия дефекта все равно устанавливается. Модель может быть улучшена путем исследования других архитектур сегментаций, описания нейросети с увеличенным или уменьшенным числом сверточных слоев.

В дальнейшем модель можно апробировать на производствах, где возникает потребность классифицировать и распознавать дефекты на стальных пластинах. Для этого необходимо собрать установку с камерой и организовать отправку изображений на компьютер, который будет осуществлять обработку полученных данных с помощью описанной нейронной сети. Полученные результаты могут быть экстраполированы и на производства любых других листов металла.

Список литературы

1. **Кузичкин А. А.** Нейронные сети в промышленности и информационных технологиях // Инновационный менеджмент. — 2016. — № 3. — С. 46—49.
2. **Bukht R., Heeks R.** Defining, Conceptualising and Measuring the Digital Economy // International Organisations Research Journal. — 2018. — No. 13. — P. 143—172.
3. **Трофимов В. Б., Кочев П. С.** О разработке интеллектуальной автоматизированной системы контроля качества листового проката на основе многоструктурного распознавания дефектов // Теплотехника и информатика в образовании, науке и производстве: сб. докладов IV Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых (ТИМ'2015) с международным участием, посвященной 95-летию основания кафедры и университета. — Екатеринбург: ООО "УЦАО", 2015. — С. 427—431.
4. **Созыкин А. В.** Обзор методов обучения глубоких нейронных сетей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. — 2017. — Т. 6, № 3. — С. 28—59. DOI: 10.14529/cmse170303.
5. **Сикорский О. С.** Обзор сверточных нейронных сетей для задачи классификации изображений // Новые информационные технологии в автоматизированных системах. — 2017. — № 20. — С. 37—42.

6. **Ronneberger O., Fischer P., Brox T.** U-Net: Convolutional Networks for Biomedical Image Segmentation // Medical Image Computing and Computer-Assisted Intervention — MICCAI, Springer, LNCS. — 2015. — Vol. 9351. — P. 234—241. URL: https://doi.org/10.1007/978-3-319-24574-4_28.
7. **Intro** — chest xray, DICOM, viz, U-nets — full data. URL: https://www.kaggle.com/jesperdramsch/intro-chest-xray-dicom-viz-u-nets-full-data/notebook#Vanilla-Unet
8. **Segmentation** models pytorch + catalyst. URL: https://www.kaggle.com/interneuron/segmentation-models-pytorch-catalyst
9. **Godoy D.** Understanding binary cross-entropy / log loss: a visual explanation. URL: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a
10. **Tiu E.** Metrics to Evaluate your Semantic Segmentation Model. URL: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

Features of the Development of a Neural Network to Automate the Recognition of Steel Defects

R. A. Karelova, riya2003@mail.ru, **E. E. Ignatov,** levia4119@gmail.com, Nizhny Tagil Technological Institute (branch) of the Ural Federal University named after the first President of Russia B. N. Yeltsin, Nizhny Tagil, 622031, Russian Federation

Corresponding author:

Karelova Riya A., PhD, Associate Professor, Nizhny Tagil Technological Institute (branch) of the Ural Federal University named after the first President of Russia B. N. Yeltsin, Nizhny Tagil, 622031, Russian Federation
E-mail: riya2003@mail.ru

Received on July 06, 2020
Accepted on October 07, 2020

The article presents an embodiment of an artificial neural network for recognizing defects in images of steel sheets. Several stages of solving the problem are described: the choice of a development environment, a programming language, and libraries necessary for the implementation; features of data analysis, graphing, histograms, finding dependencies; the selection of a suitable neural network, the choice of neural network architecture, the selection of an algorithm for assessing quality and accuracy; neural network spelling; training and checking accuracy and quality, checking for overfitting (retraining).

As development tools, Python language, PyTorch library, Jupyter development environment, convolutional neural network architecture — Unet are proposed. Features of the analysis of input images of steel sheets, features of the implementation of the neural network itself are described. The function of binary cross entropy was chosen as a criterion for assessing accuracy, since it seeks to bring the distribution of the network forecast to the target, fine not only for erroneous predictions, but also for uncertain ones.

For additional evaluation, the DICE method was also used. The accuracy of the resulting model is 84 %. The proposed solution can become part of a hardware-software system for automating the recognition of defects on metal sheets.

Keywords: *production digitalization, neural network, steel defects, image recognition, production automation, python, pytorch, unet architecture*

For citation:

Karelova R. A., Ignatov E. E. Features of the Development of a Neural Network to Automate the Recognition of Steel Defects, *Programmnaya Ingeneria*, 2020, vol. 11, no. 6, pp. 330—334

DOI: 10.17587/prin.11.330-334

References

1. **Kuzichkin A. A.** Neural networks in industry and information technology, *Innovacionnyj menedzhment*, 2016, no. 3, pp. 46—49 (in Russian).
2. **Bukht R., Heeks R.** Defining, Conceptualising and Measuring the Digital Economy, *International Organisations Research Journal*, 2018, no. 13, pp. 143—172.
3. **Trofimov V. B., Kochev P. S.** On the development of an intelligent automated system for monitoring the quality of sheet metal based on multi-structure recognition of defects, *Teplotekhnika i informatika v obrazovanii, nauke i proizvodstve: sbornik dokladov IV Vserossijskoj nauchno-prakticheskoj konferencii studentov, aspirantov i molodyh uchyonyh (TIM'2015) s mezhdunarodnym uchastiem, posvyashchyonnoj 95-letiyu osnovaniya kafedry i universiteta*, Ekaterinburg, OOO "UCAO", 2015, pp. 427—431 (in Russian).
4. **Sozykin A. V.** Overview of deep neural network training methods, *Vestnik YUUrGU. Seriya: Vychislitel'naya matematika i informatika*, 2017, vol. 6, no. 3, pp. 28—59, DOI: 10.14529/cmse170303. (in Russian).
5. **Sikorskij O. S**. Overview of convolutional neural networks for the classification of images, *Novye informacionnye tekhnologii v avtomatizirovannyh sistemah*, 2017, no. 20, pp. 37—42 (in Russian).
6. **Ronneberger O., Fischer P., Brox T.** U-Net: Convolutional Networks for Biomedical Image Segmentation, *Medical Image Computing and Computer-Assisted Intervention — MICCAI*, Springer, LNCS, 2015, Vol. 9351, pp. 234—241, available at: https://doi.org/10.1007/978-3-319-24574-4_28
7. **Intro** — chest xray, DICOM, viz, U-nets — full data, available at: https://www.kaggle.com/jesperdramsch/intro-chest-xray-dicom-viz-u-nets-full-data/notebook#Vanilla-Unet.
8. **Segmentation** models pytorch + catalyst, available at: https://www.kaggle.com/interneuron/segmentation-models-pytorch-catalyst
9. **Godoy D.** Understanding binary cross-entropy / log loss: a visual explanation, available at: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.
10. **Tiu E.** Metrics to Evaluate your Semantic Segmentation Model, available at: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

УДК 004.912 DOI: 10.17587/prin.11.335-341

А. С. Козицын, канд. физ.-мат. наук, вед. науч. сотр., alexanderkz@mail.ru,
С. А. Афонин, канд. физ.-мат. наук, вед. науч. сотр., serg@msu.ru,
Д. А. Шачнев, программист, mitya57@gmail.com, Московский государственный университет им. М. В. Ломоносова

Метод оценки тематической близости научных журналов

Описан разработанный авторами алгоритм определения тематической близости журналов. Результаты работы его программной реализации могут использоваться для решения следующих актуальных задач: уточнение значений наукометрических показателей при получении рейтинговых показателей; оценка основных тенденций развития научных направлений; построение онтологий для определения правил безопасности в рамках модели ABAC; создание эффективных и удобных механизмов поиска научной информации. Особенностями представленного алгоритма являются использование графа соавторства для построения метрики тематической близости и возможность обработки журналов на разных языках, что сложно реализуемо для других алгоритмов тематического анализа, основанных на анализе полнотекстовой информации.

Программная реализация представленного алгоритма используется на практике в наукометрической системе "ИСТИНА".

Ключевые слова: *тематический анализ, наукометрия, информационные системы, библиография, автор, граф*

Введение

Для эффективного управления научными организациями и стимулирования их работников необходим перманентный анализ результатов их научной деятельности с применением методов наукометрии — дисциплины, изучающей науку через различные количественные показатели научно-технической и инновационно-технологической деятельности (далее для краткости будем называть научной деятельностью). Традиционно одним из основных количественных показателей оценки результатов такой деятельности является цитируемость публикаций. На измерении этого показателя основаны импакт-факторы журналов, индекс Хирша и g-индекс для авторов научных публикаций, i-индекс для организаций и другие важные наукометрические индикаторы [1, 2]. Технологическими платформами для расчета цитируемости публикаций являются системы научного цитирования, такие как Web of Science, Google Scolar, Scopus, РИНЦ и др. Разнообразие используемых в настоящее время систем цитирования неизбежно порождает неоднозначность оценки научных публикаций, журналов и авторов, поскольку каждая из систем цитирования использует свою область охвата для получения данных о цитируемости. Например, известный российский ученый, специалист в области лингвистики и древнерусского языка А. А. Зализняк согласно данным системы РИНЦ имеет 103 научных публикаций с общим числом цитирований 4291, его индекс Хирша равен 24 [3]. В то же время согласно данным Scopus он имеет только 10 публикаций с общим числом цитирований 11 и его индекс Хирша равен 2 [4].

Второй возможной причиной необъективности наукометрических показателей является разнообразие тематик научных исследований. При интерпретации полученных показателей цитирования необходимо анализировать тематическую область научных статей, поскольку показатели цитирования очень сильно зависят от тематики научных исследований [5]. Для получения более качественных результатов оценки научной деятельности требуется использовать наукометрические показатели, которые позволяют учитывать тематическую область анализируемых публикаций, например нормализованную среднюю цитируемость [6].

Таким образом, при создании автоматизированных систем расчета наукометрических показателей возникает задача выделения тематических классов научных журналов, которые можно учитывать при расчете и нормировке наукометрических показателей. Сложность задачи классификации обусловлена большим числом публикуемых в мире журналов на русском, английском и других языках. Например, в информационно-аналитической системе (ИАС) "ИСТИНА" [7, 8] зарегистрировано более 70 тыс. журналов и еще более 200 тыс. сборников научных публикаций.

Однако возможности использования тематического анализа результатов научной деятельности не ограничиваются только уточнением наукометрических показателей. Он позволяет решать и другие важные задачи. Применение методов тематического анализа помогает более детально оценить основные тенденции развития мировой науки на уровне государства или отдельной организации. В целях проведения такой оценки большинство разрабатываемых

наукометрических систем и систем цитирования имеют интерфейсы для визуализации тематического распределения данных. Например, ИАС "ИСТИНА" позволяет анализировать распределение показателей научных публикаций по тематическим рубрикаторам Scopus и ГРНТИ с фильтрацией по годам, по подразделениям и по другим параметрам [9].

Анализ основных тенденций развития различных тематических направлений науки и своевременное отслеживание их изменений необходимы в том числе для принятия правильных управленческих решений и выбора способов стимулирования научной деятельности [10, 11].

Выявление тематических зависимостей между журналами и другими объектами в информационных системах [12] позволяет решать такие важные технологические задачи, как автоматическое построение онтологий [13] для описания правил доступа к данным [14] при построении моделей логического разграничения доступа ABAC (*Attribute-Based Access Control*) [15].

Такие модели применительно к интеллектуальным системам больших данных приходят на смену традиционным моделям разграничения доступа (ролевая модель RBAC; мандатная модель MAC, дискреционная модель DAC [16—18]), ориентированным на применение в информационных системах с разделением обязанностей пользователей и фиксированными уровнями доступа. Основным преимуществом модели ABAC является возможность построения правил доступа на основе связей объектов и их атрибутов с возможностью автоматической проверки корректности и непротиворечивости этих правил [19].

Еще одной областью применения результатов автоматической тематической классификации научных публикаций и построения оценки тематической близости журналов является создание эффективных и удобных механизмов поиска научной информации. Подобный инструментарий позволит на основе аккумулированного опыта научного сообщества автоматически подбирать журналы, соответствующие заданным научным интересам пользователя, что поможет молодым сотрудникам увеличить свою публикационную активность и повысить показатели цитируемости.

Следует отметить, что существующие системы цитирования предоставляют интерфейс для проведения тематического анализа по заданным рубрикаторам (Scopus, ГРНТИ и др.). Однако уровень такой рубрикации слишком высок для проведения полноценного анализа. Например, РИНЦ предоставляет информацию по тематическим разделам "Автоматика. Вычислительная техника", "Информатика", "Математика", "Кибернетика", которые дают слишком общее описание области исследований индексируемой научной работы или журнала. Более детализированный рубрикатор ГРНТИ, также использующийся в РИНЦ, не позволяет оценить тематическую близость журналов. По рубрикатору ГРНТИ журнал "Программная инженерия" отнесен к рубрикам 20.23.17, 20.23.29, 28.23.29, 28.23.35, 28.23.39, 50.05.00, 50.05.09, 50.41.00, а журнал "Системное программирование" к рубрикам 50.05.17 и 50.41.17. Вместе с тем анализ публикационной активности авторов показывает, что журналы достаточно близки по тематике и имеют значительное число пересекающихся научных областей. Использование дополнительных источников информации позволяет в автоматическом режиме провести оценку тематической близости журналов и скорректировать тематические индексы, построенные на общих рубрикаторах.

Одним из наиболее распространенных подходов к решению задачи тематического анализа является использование текстовых элементов статей. Существует большое число различных алгоритмов тематического анализа документов и оценки их тематической близости с использованием как полных текстов, так и их отдельных частей — названий, аннотаций, ключевых слов [20]. Такие методы успешно применяют для анализа данных в социальных сетях, для кластеризации новостных потоков и в документообороте. Однако использование этого класса методов для анализа тематической близости журналов в наукометрических системах имеет ряд ограничений. Большая часть алгоритмов этого класса для проведения анализа требует использования полных текстов документов. При этом значительная часть журналов не размещает полные тексты статей в открытых источниках информации. Остальные журналы могут позволять размещать такие тексты в Интернете в разрозненных источниках без наличия каких-либо единых стандартов предоставления данных. В связи с этим сбор в единой системе текстов статей по значительной части мировых журналов представляет сложную техническую задачу.

Использование для тематического анализа только ключевых слов и названий не всегда позволяет получить достаточно точный результат, поскольку список ключевых слов содержит недостаточно информации для анализа. Для иллюстрации качества существующих методов анализа можно привести следующие данные из реально работающих систем.

Для оценки результатов тематического анализа в системе РИНЦ использовался поисковый запрос в области "статьи и доклады" по слову "библиография" в тематическом разделе "Информатика". В представленном системой списке среди первых 20 результатов оказалась только одна статья, косвенно связанная с информатикой — "Национальная библиография и интернет", в которой обсуждается возможность использования Интернета для размещения библиографических каталогов ("Авторы ищут ответы на вопросы: будет ли национальная библиография по-прежнему востребована или вольется в сетевое пространство?"). Остальные работы ("Национальная библиография и библиография местных документов", "Краеведческая библиография и библиография местной печати: структурно-семантический анализ понятий", "Библиография в эпохи великих перемен", "Избранная библиография по отечественной генеалогии", «Гимн библиографу, или размышления над книгой В. П. Леонова "Библиография как профессия"» и др.) к информатике не имеют никакого отношения.

Для оценки работы системы "Антиплагиат" в нее был загружен для анализа текст настоящей статьи. В результате тематического анализа системой "Антиплагиат" статья отнесена к следующим рубрикам. Рубрикатор ГРНТИ: "76 — Медицина и здравоохранение" 46,53 %; "50 — Автоматика. Вычислительная техника" 28,97 %; "06 — Экономика и экономические науки" 24,50 %. Рубрикатор УДК: "61 — Медицина. Охрана здоровья. Пожарное дело" 40,91 %; "00 — Наука в целом" 35,51 %; "33 —Экономика. Народное хозяйство. Экономические науки" 23,58 %.

Представленные выше примеры показывают, что используемые в промышленных системах методы полнотекстового тематического анализа имеют определенные недостатки в области точности распознавания тематики текстов. Дополнительные сложности возникают при сравнении тематики издающихся на разных языках журналов.

Несмотря на достаточно бурное развитие систем автоматического перевода, их результаты для терминов узкой тематики остаются недостаточно точными, особенно в области специальных научных терминов [21]. Для иллюстрации качества такого перевода можно привести следующие примеры. Название статьи "Rayleigh and Love surface waves in isotropic media with negative Poisson's ratio" система Prompt переводит как "Рэлей и Любовные волны поверхности в изотропических СМИ с отношением отрицательного Пуассона" ("Поверхностные волны Релея и Лява при отрицательном коэффициенте Пуассона изотропных сред").

Название "Self-Purification of Agrosoddy-Podzolic Sandy Loamy Soils Fertilized with Sewage Sludge" система Google переводит как "Самоочищение Агрозодди-Подзолик Сэнди глинистые почвы, оплодотворенные с отстоем сточных вод" ("Самоочищение агродерново-подзолистых супесчаных почв, удобренных осадком сточных вод").

Для устранения таких ошибок при поиске дубликатов и сравнении текстов на разных языках применяют различные методы коррекции переводов. Например, в работе [21] строятся все возможные переводы, и названия считаются близкими, если близкими оказываются слова хотя бы в одной построенной паре. В системе "Антиплагиат" [22] для сравнения текстов используются n-граммы классов слов.

Но эти методы неприменимы для тематической классификации, поскольку будут давать много ошибочных переводов терминов и, как следствие, большое число неправильно определенных тематических направлений.

В наукометрических системах для проведения тематического анализа можно использовать граф соавторства [23], который не требует наличия текстовой информации для статей и использует только их библиографические данные. Такой подход позволяет не только устранить проблему неточности систем перевода, но и проводить анализ всех библиографических записей без ограничений, накладываемых техническими возможностями получения полных текстов статей.

Отсутствие требования к наличию полных текстов статей значительно расширяет область поиска журналов. Для иллюстрации этого факта можно привести следующие цифры. В системе РИНЦ из 69 тыс. научных журналов только 7,5 тыс. имеют полные тексты (менее 11 %). В системе ИСТИНА 39 тыс. журналов включены в граф соавторства.

При проведении тематического анализа с использованием графа соавторства предполагается, что авторы публикуют свои материалы в нескольких тематически близких журналах. Как следствие, в близких по тематике журналах часто публикуются одинаковые авторы. Необходимые для анализа библиографические данные могут быть получены из наукометрических систем (например, ИАС "ИСТИНА") или систем цитирования (например, WoS). Следует отметить, что в подобных наукометрических системах и системах цитирования автору присваивается уникальный идентификатор, который определяется на этапе загрузки и разбора библиографических данных [24].

Использование уникальных идентификаторов при сравнении издающихся на разных языках журналов устраняет сложности, связанные с переводами фамилий авторов.

Алгоритм поиска похожих журналов по графу соавторства

Для проведения оценки тематической близости журналов необходимо создать отображение $F : J \times J \to \Re$, где J — множество журналов.

При этом значение для тематически близких пар журналов должно быть выше, чем для тематически несвязанных пар.

На первом этапе построения такого отображения создается множество $M(j_1, j_2)$ всех пар статей, которые были опубликованы в журналах j_1 и j_2 и имеют хотя бы одного совпадающего автора:

$$M(j_1, j_2) = \{d_1, d_2 \in D \mid d_1 \in D_{j_1}, \\ d_2 \in D_{j_2}, A_{d_1} \cap A_{d_2} \neq 0\},$$

где D — множество статей; D_j — множество статей в журнале j; A_d — множество авторов статьи d.

Для удаления лишнего шума журналы, для которых $|M(j_1, j_2)| < 2$, считаются несвязанными, и для таких пар $F(j_1, j_2) \equiv 0$. Остальные пары журналов, которым соответствует несколько пар статей, считаются связанными ребром с определенным весом. В настоящей работе рассматриваются несколько методов определения веса ребра.

Первый метод учитывает число уникальных соавторов для статей в каждой паре журналов:

$$F(j_1, j_2) \equiv \\ \equiv \left|\left\{a \in A \mid \exists d_1 \in D_{j_1} : a \in A_{d_1}, \exists d_2 \in D_{j_2} : a \in A_{d_2}\right\}\right|,$$

где a — авторы из всего множества авторов A.

Основным недостатком такого метода является отсутствие учета значимости авторов для каждой конкретной статьи. Этот недостаток характерен для всех аналогичных методов расчета соавторства,

например, с использованием меры исключительности, которая рассчитывается по следующей формуле: $\sum_{\{d \in D | a_1 \in A_d, a_2 \in A_d\}} \frac{1}{|A_d| - 1}$ [25].

Для значительной части статей вклад соавторов в написание текста статьи не одинаков. Один из соавторов выполняет большую часть работы, и его фамилия ставится на первом месте в библиографической ссылке. Научные результаты остальных соавторов используются в статье, но их основное направление научной деятельности может не совпадать с тематикой публикации. Оценка доли статей, в которых порядок авторов определяется степенью их научного вклада, проводилась по следующей методике.

В качестве корпуса данных для анализа использовались все опубликованные в 2014—2019 гг. статьи из наукометрической системы МГУ с числом авторов от 2 до 7.

Весь корпус статей был разбит на 6 блоков в соответствии с числом авторов $k \in \{2...7\}$. Для каждого корпуса были рассчитаны следующие значения:

r_k — число статей, в которых первый автор стоит в правильном лексикографическом порядке;

a_k — общее число статей.

Число статей, в которых первый автор не соответствует лексикографическому порядку, равно $a_k - r_k$.

Если бы первый автор определялся только долей участия в работе над статьей, и правильный лексикографический порядок был случайным событием, то выполнялось бы следующее равенство:

$$r_k = \frac{a_k - r_k}{k - 1}.$$

Тогда долю статей, для которых правильный набор авторов не случаен, можно оценить формулой

$$L = r_k - \frac{\frac{a_k - r_k}{k - 1}}{a_k}.$$

Результаты расчета для различного числа авторов представлены в табл. 1.

Приведенные в табл. 1 результаты расчетов показывают, что доля статей, в которых первый автор определяется лексикографическим порядком, значительно меньше доли статей, в которых порядок авторов определяется значимостью их вклада в публикацию. Это различие растет с увеличением общего числа соавторов в статье. Для учета этого факта необходимо использовать специальные методы нормализации весов авторов при аналитической обработке библиографических данных, в том числе при определении степени тематической близости журналов.

Таблица 1

Учет лексикографического порядка

Число соавторов	Значение L
2	0,24
3	0,15
4	0,08
5	0,06
6	0,05
7	0,03

Вес автора в каждой статье определяется как $w(a, d) = \frac{1}{2} + \frac{1}{2k}$ для первого автора и $w(a, d) = \frac{1}{2k}$ для остальных соавторов.

Тогда искомое отображение будет иметь вид

$$F(j_1, j_2) \equiv \sum_a W(a, j_1, j_2), \quad (1)$$

где

$$W(a, j_1, j_2) \equiv \min\left(\max_{d_1 \in D_{j_1}}\left(w(a, d_1)\right), \max_{d_2 \in D_{j_2}}\left(w(a, d_2)\right)\right).$$

Для проверки корректности представленного алгоритма его программная реализация была протестирована по следующей методике. На основе данных ИАС "ИСТИНА" был получен массив пар журналов, которые были определены как тематически близкие, и для случайных 200 пар из этого массива экспертами проставлено значение совпадения тематик по трехбалльной шкале (2 — точное; 1 — не совсем точное; 0 —ошибочное).

Общая сумма баллов делилась на удвоенное число анализируемых связей:

$$S = \frac{\sum_{i=1}^{N} c_i}{2N},$$

где $c_i \in \{0, 1, 2\}$ — оценка правильности связи i; N — число связей в тестовой выборке.

Оценка точности по этой методике составила 78 %.

Расширение результатов поиска

Для расширения результатов поиска близких по тематике журналов можно использовать предположение, что если два журнала близки по тематике третьему, то они близки между собой. На основе формулы (1) для отображения $F(j_1, j_2)$ можно построить граф тематических связей журналов, заданного множеством вершин J и множеством ребер $\{(j_1, j_2) \in J \times J \mid F(j_1, j_2) > 0\}$.

Для улучшения качества анализа необходимо учитывать наличие в графе тематической близости междисциплинарных журналов. Такие журналы имеют много связей с журналами различных тематик, что значительно ухудшает результаты поиска. Для уменьшения влияния таких журналов на конечный результат работы алгоритма требуется перед проведением вычисления путей проводить предварительную нормировку функции веса ребер и построение новой матрицы близости $\hat{F}$. В рамках исследований, результаты которых представлены в работе, рассматривалось два способа нормировки — по сумме

$$\widehat{F}(j_i,\ j_k) \equiv \frac{F(j_i,\ j_k)}{\sum\limits_i F(j_i,\ j_k)}$$

и по количеству

$$\widehat{F}(j_i,\ j_k) \equiv \frac{F(j_i,\ j_k)}{\sum\limits_i \delta\left(F(j_i,\ j_k)\right)},$$

где $\delta(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0. \end{cases}$

Подсчет веса для путей длины n проводится по следующей рекуррентной формуле:

$$\widehat{F}^{(1)}(j_i,\ j_i) \equiv 1;$$

$$\widehat{F}^{(1)}(j_i,\ j_k) \equiv F(j_i,\ j_k),\ \text{при}\ \ i \neq k;$$

...

$$\widehat{F}^{(n)}(j_i,\ j_p) \equiv \sum_p \left(\widehat{F}^{(1)}(j_i,\ j_p)\,\widehat{F}^{(n-1)}(j_p,\ j_k)\right).$$

Тестирование программной реализации алгоритма проводилось на данных ИАС "ИСТИНА". После построения нового отображения для определения тематической близости были отобраны все ребра, для которых $F(j_i,\ j_k) = 0$, $\widehat{F}^{(n)}(j_i,\ j_k) > 0$, и из них выбраны 80 пар журналов, имеющих максимальное значение $\widehat{F}^{(n)}(j_i,\ j_k)$.

Были получены шесть наборов для $n \in \{2, 3, 4\}$ с критерием нормировки по сумме и по количеству. Сравнение качества полученных наборов проводили подсчетом суммарного коэффициента $s_i = k_i(80 - r_i)$,

Таблица 2

Сравнение качества результатов, s_i

Нормировка	Номер шага		
	2	3	4
По сумме	2528	2631	2589
По количеству	2458	2540	2226

где k_i — экспертная оценка степени правильности найденной тематической связи по шкале от 0 до 2, а r_i — позиция связи в отсортированном списке лучших 80 пар тематически связанных журналов. Результаты сравнения представлены в табл. 2.

Как видно из данных табл. 2, наилучший результат достигается при использовании нормировки по сумме на шаге 3 расширения. Функция $\widehat{F}^{(3)}$ с нормировкой по сумме выбрана для использования в программной реализации алгоритма для создания поискового интерфейса в ИАС "ИСТИНА".

Представление пользователю результатов тематического поиска журналов в ИАС "ИСТИНА" возможно как на основе отображения F, так и на основе отображения $\widehat{F}^{(3)}$. В специально разработанном интерфейсе пользователь может через форму поиска (https://istina.msu.ru/search/) найти интересующий его журнал и открыть его карточку. В карточке журнала отображается ссылка "Похожие по тематике журналы", нажав на которую можно вывести на экран список тематически похожих на него журналов (см. рисунок). Для удобства работы этот список содержит не только названия журналов и степень их тематической близости, но и дополнительные харак-

Show by 10 items Search:

N	Журнал	Вес	Статей за 5 лет	WS	SJR	RINC	Похожие журналы	Похожие конференции	Добавить в заметки
1	Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, № 2, ISSN 2075-9460)	91,07	176	-	-	.19 (2018)	журналы	конференции	+
2	Информационные технологии	69,77	20	-	-	.448 (2018)	журналы	конференции	+
3	Программирование	65,83	50	-	-	.685 (2018)	журналы	конференции	+
4	Programming and Computer Software	62,17	74	.637 (2019)	-	-	журналы	конференции	+
5	Проблемы информатики	38,4	1	-	-	.138 (2017)	журналы	конференции	+
6	Обозрение прикладной и промышленной математики	36,91	65	-	-	-	журналы	конференции	+

Список тематически похожих журналов для журнала "Программная инженерия"

теристики, такие как импакт-факторы WoS и РИНЦ, число публикаций и др.

В целях улучшения эргономики интерфейса пользователю предоставлена возможность добавлять журнал в свои заметки, которые можно просматривать и редактировать, а также использовать при последующем поиске. Для ускорения поиска и уменьшения времени отклика системы используется кеш результатов, который пересчитывается раз в сутки.

Заключение

Представленный алгоритм позволяет на основе только библиографических описаний статей определять тематически близкие пары журналов. Алгоритм не требует наличия полных текстов статей или ключевых слов, получить которые для большого объема журналов может быть сложной технической задачей. Представленный алгоритм также нечувствителен к языку и может успешно применяться для поиска тематически близких журналов на разных языках, что сложно реализуемо для алгоритмов, основанных на анализе полнотекстовой информации. Применение алгоритма возможно как автономно, так и в ансамбле с классическими алгоритмами полнотекстового тематического анализа.

Во втором случае представленный алгоритм будет дополнять алгоритмы полнотекстового тематического анализа, когда количество полнотекстовых данных оказывается недостаточным или тексты представлены на разных языках. Построение тематической близости возможно не только между журналами, но и между другими связанными с авторами объектами — конференциями, проектами, семинарами и др.

Полученные результаты тематического анализа могут использоваться для уточнения количественных значений наукометрических показателей при проведении анализа научной деятельности для принятия административно-управленческих решений, для построения правил определения политик разграничения доступа наряду с другими алгоритмами автоматического выделения связей между объектами [22], для создания удобных механизмов библиографического поиска и решения других аналогичных задач.

Работа выполнена при финансовой поддержке РФФИ (грант № 18-07-01055)

Список литературы

1. **Акоев М. А., Маркусова В. А., Москалева О. В., Писляков В. В.** Руководство по наукометрии: индикаторы развития науки и технологии. — Екатеринбург: Изд-во Урал. ун-та, 2014. — 248 с.

2. **Полянин А. Д.** Недостатки индексов цитируемости и Хирша. Индексы максимальной цитируемости. URL: http://eqworld.ipmnet.ru/ru/info/sci-edu/Polyanin_IndexH_2014.html (дата обращения 23.03.2020).

3. **Индекс** Хирша РИНЦ. URL: https://www.elibrary.ru/author_items.asp?authorid=104547 (дата обращения 01.10.2020).

4. **Индекс** Хирша Scopus. URL: https://www.scopus.com/authid/detail.uri?origin=resultslist&authorId=16461422800 (дата обращения 23.03.2020).

5. **Орлов А. И.** Наукометрия и управление научной деятельностью // Управление большими системами. Специальный выпуск 44: Наукометрия и экспертиза в управлении наукой. Институт проблем управления им. В. А. Трапезникова РАН. — 2013. — С. 538—568.

6. **Бричковский В. В.** Наукометрический анализ в информационном обеспечении инновационной деятельности // Наука и Инновации. — 2017. — № 8 (174). — С. 64—67.

7. **Интеллектуальная** система тематического исследования научно-технической информации (ИСТИНА) / под ред. В. А. Садовничего, В. А. Васенина и др. — М.: Изд-во Московского ун-та, 2014. — 262 с.

8. **Афонин С. А., Голомазов Д. Д., Козицын А. С.** Использование систем семантического анализа для организации поиска научно-технической информации // Программная инженерия. — 2012. — № 2. — С. 29—34.

9. **Анализ** статистических данных в системе ИСТИНА. URL:https://istina.msu.ru/statistics/organization/214524/dynamic/ (дата обращения 23.03.2020).

10. **Садовничий В. А., Васенин В. А., Афонин С. А., Козицын А. С., Голомазов Д. Д.** Информационная система "ИСТИНА" как big data — инструментарий в области управления на основе анализа наукометрических данных // Знания — Онтологии — Теории (ЗОНТ-2015). Материалы Всероссийской конференции с международным участием. Новосибирск, 2015. — С. 115—123.

11. **Васенин В. А., Зензинов А. А., Лунев К. В.** Использование наукометрических информационно-аналитических систем для автоматизации проведения конкурсных процедур на примере информационно-аналитической системы ИСТИНА // Программная инженерия. — 2016. — Т. 7, № 10. — С. 472—480.

12. **Афонин С. А., Козицын А. С., Шачнев Д. А.** Программные механизмы агрегации данных, основанные на онтологическом представлении структуры реляционной базы наукометрических данных // Программная инженерия. — 2016. — Т. 7, № 9. — С. 408—413.

13. **Платонов А. В., Полещук Е. А.** Методы автоматического построения онтологий // Программные продукты и системы. — 2016. — № 2. — С. 47—52.

14. **Afonin S.** Ontology models for access control systems// Proc. of the 3rd International Conference Russian-Pacific Conference on Computer Technology and Applications (RPC). 2018. — P. 1—6.

15. **Jin X., Krishnan R., Sandhu R.** A untied attribute-based access control model covering DAC, MAC and RBAC // Data and Applications Security and Privacy XXVI, Lecture Notes in Computer Science 7371. — 2012. — P. 41—55.

16. **Sandhu R. S., Samarati P.** Access control: principle and practice // IEEE Commun. Mag. — 1994. — Vol. 32, No. 9. — P. 40—48.

17. **Девянин П. Н.** Модели безопасности компьютерных систем: учеб. пособие для вузов. — М.: Академия, 2005. — 144 с.

18. **Гайдамакин Н. А.** Разграничение доступа к информации в компьютерных системах. — Екатеринбург: Изд-во Урал. ун-та, 2003. — 328 с.

19. **Afonin S., Bonushkina A.** Validation of safety-like properties for entity-based access control policies // Proc. of the international conference Advances in Soft and Hard Computing. Springer International Publishing, 2019. — P. 259—271.

20. **Witten I. H., Frank E., Hall M. A., Pal C. J.** Data Mining: Practical machine learning tools and techniques. — Morgan Kaufmann, 2016. — 558 p.

21. **Козицын А. С., Афонин С. А., Зензинов А. А.** Алгоритм определения переводов статей с использованием статистических данных // Электронные библиотеки. — 2018. — Т. 21, № 6. — С. 494—505.

22. **Плагиат** в научных статьях: трудности обнаружения перевода. URL: http://ai-news.ru/2018/01/plagiat_v_nauchnyh_statyah_trudnosti_obnaruzheniya_perevoda.html (дата обращения 19.03.2020).

23. **Краснов Ф. В.** Анализ методов построения графа соавторства: подход на основе двудольного графа// International Journal of Open Information Technologies. — 2018. — Vol. 6, No. 2. — P. 31—37.

24. **Козицын А. С., Афонин С. А.** Разрешение неоднозначностей при определении авторов публикации с использованием графов соавторства в больших коллекциях библиографических данных // Программная инженерия. — 2017. — Т. 8, № 12. — С. 556-562.

25. **Краснов Ф. В., Шварцман М. Е., Дементов А. В.** Сравнительный анализ коллекций научных журналов // Труды СПИИРАН. — 2019. — Т. 18, № 3. — С. 766—792.

Method for Assessing the Thematic Proximity of Scientific Magazines

A. S. Kozitsyn, alexanderkz@mail.ru, **S. A. Afonin,** serg@msu.ru, **D. A. Shachnev,** mitya57@gmail.com,
M. V. Lomonosov Moscow State University, Moscow, 119192, Russian Federation

Corresponding author:

Kozitsyn Alexander S., Leading Researcher, M. V. Lomonosov Moscow State University, Moscow, 119192, Russian Federation
E-mail: alexanderkz@mail.ru

Received on September 16, 2020
Accepted on October 01, 2020

The paper describes an algorithm for determining the thematic proximity of magazines. The results of its software implementation can be used to solve the following urgent tasks: clasification of the scientometric indicators values in the evaluation of scientific activity; assessment of the main trends in the development of the main scientific areas; determination of the relationships between data in information systems for constructing ontologies and defining safety rules within the framework of the ABAC model; creation of effective and convenient mechanisms for searching for scientific information.

A feature of the presented algorithm is the use of the co-authorship graph for constructing the thematic proximity metric and the ability to process magazines in different languages, which is difficult to implement for other thematic analysis algorithms based on the analysis of full-text information. The software implementation of the presented algorithm is used in practice in the scientometric system "ISTINA".

Keywords: *thematic analysis, scientometrics, information systems, bibliography, author, graph*

Acknowledgements. *This research was partly supported by RFBR (grant N18-07-01055).*

For citation:

Kozitsyn A. S., Afonin S. A., Shachnev D. A. Method for Assessing the Thematic Proximity of Scientific Magazines, *Programmnaya Ingeneria,* 2020, vol. 11, no. 6, pp. 335—341

DOI: 10.17587/prin.11.335-341

References

1. **Akoev M. A., Markousova V. A., Moskaleva O. V., Pisklyakov V. V.** *Guide to Scientometrics: Indicators of the Development of Science and Technology.* Ekaterinburg, Ural University Publishing House, 2014, 248 p. (in Russian).
2. **Polljanin A.** Disadvantages of citation and hirsch indices. Maximum Citation Indices, available at: http://eqworld.ipmnet.ru/ru/info/sci-edu/Polyanin_IndexH_2014.html (accessed 23.03.2020) (in Russian).
3. **Hirsch** Index RSCI, available at: https://www.elibrary.ru/author_items.asp?authorid=104547 (accessed 01.10.2020) (in Russian).
4. **Hirsch** Index Scopus, available at: https://www.scopus.com/authid/detail.uri?origin=resultslist\&authorId=16461422800 (accessed 23.03.2020) (in Russian).
5. **Orlov A. I.** Scientometrics and management of scientific activities, *Upravlenie bol'shimi sistemami. Special'nyj vypusk 44: Naukometriya i ekspertiza v upravlenii naukoj.* Institute of Management Problems V. A. Trapeznikova RAS, 2013, pp. 538—568 (in Russian).
6. **Brichkovsky V. V.** Scientometric analysis in the information support of innovative activity, *Nauka i Innovacii*, 2017, no. 8 (174), pp. 64—67 (in Russian).
7. ***Intelligent*** *system of case study of scientific and technical information (ISTINA)* / Eds V. A. Sadovnichy, V. A. Vasenin et al. Moscow, Moscow University Press, 2014, 262 p. (in Russian).
8. **Afonin S. A., Golomazov D. D., Kozitsyn A. S.** The use of semantic analysis systems to organize the search for scientific and technical information, *Programmnaya Ingeneria,* 2012, no. 2. pp. 29—34 (in Russian).
9. **Analysis** of statistical data in the system ISTINA, available at: http://istina.msu.ru/statistics/organization/214524/dynamic/ (accessed 23.03.2020) (in Russian).
10. **Sadovnichy V. A., Vasenin V. A., Afonin S. A., Kozitsyn A. S., Golomazov D. D.** Information system "ISTINA" as big data — a tool in the field of control based on the analysis of scientometric data, *Knowledge — Ontologies — Theories (ZONT-2015), Materials of the All-Russian Conference with international participation*. Novosibirsk, 2015, pp.115—123 (in Russian).
11. **Vasenin V. A., Zenzinov A. A., Lunev K. V.** Using scientometric information-analytical systems to automate competitive procedures using the example of the information-analytical system ISTINA, *Programmnaya Ingeneria*, 2016, vol. 7, no. 10, pp. 472—480 (in Russian).
12. **Afonin S. A., Kozitsyn A. S., Shachnev D. A.** Software data aggregation mechanisms based on the ontological representation of the structure of the relational database of scientometric data, *Programmnaya Ingeneria*, 2016, vol. 7, no. 9, pp. 408—413 (in Russian).
13. **Platonov A. V., Poleshchuk E. A.** Methods of automatic construction of ontologies, *Programmnye produkty i sistemy*, 2016, no. 2, pp. 47—52 (in Russian).
14. **Afonin S.** Ontology models for access control systems, *Proc. of the 3rd International Conference Russian-Pacific Conference on Computer Technology and Applications (RPC)*, 2018, pp. 1—6.
15. **Jin X., Krishnan R., Sandhu R.** A untied attribute-based access control model covering DAC, MAC and RBAC, *Data and Applications Security and Privacy XXVI, Lecture Notes in Computer Science, 7371,* 2012, pp. 41—55.
16. **Sandhu R. S., Samarati P.** Access control: principle and practice, *IEEE Commun. Mag.* 1994, vol. 32, no. 9, pp. 40—48.
17. **Devyanin P. N.** *Models of security of computer systems: textbook. manual for universities,* Moscow, Academy, 2005, 144 p. (in Russian).
18. **Gaydamakin N. A.** *Differentiation of access to information in computer systems*, Yekaterinburg, Publishing House Ural. Univ., 2003, 328 p. (in Russian).
19. **Afonin S., Bonushkina A.** Validation of safety-like properties for entity-based access control policies, *Proc. of the international conference Advances in Soft and Hard Computing.* Springer International Publishing, 2019, pp. 259—271.
20. **Witten I. H., Frank E., Hall M. A., Pal C. J.** *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2016, 558 p.
21. **Kozitsyn A. S., Afonin S. A., Zenzinov A. A.** Algorithm for Linking Translated Articles using Authorship Statistics, *Elektronnye biblioteki*, 2018, vol. 21, no. 6, pp. 494—505 (in Russian).
22. **Plagiat** v nauchnykh statiakh: trudnosti obnaruzheniia perevoda, available at: http://ai-news.ru/2018/01/plagiat_v_nauchnyh_statyah_trudnosti_obnaruzheniya_perevoda.html (accessed 19.03.2020) (in Russian).
23. **Krasnov F. V.** Analysis of co-authorship graph construction methods: bipartite graph approach, *International Journal of Open Information Technologies*, 2018, vol. 6, no. 2, pp. 31—37 (in Russian).
24. **Kozitsin A. S., Afonin S. A.** The Resolution of Ambiguities in the Identification of Authors of the Publication with the Use of Co-Authors' Graphs in Large Collections of Bibliographic Data, *Programmnaya Ingeneria,* 2017, vol. 8, no. 12, pp. 556—562 (in Russian).
25. **Krasnov F. V., Shvarcman M. E., Dimentov A. V.** Comparative analysis of scientific journals collections, *Trudy SPIIRAN*, 2019, vol. 18, no. 3, pp. 766—792 (in Russian).

УДК 004.825 DOI: 10.17587/prin.11.342-348

С. Д. Махортов, д-р физ.-мат. наук, зав. кафедрой, msd_exp@outlook.com,
Воронежский государственный университет

Методы решения продукционно-логических уравнений в нечеткой LP-структуре

Алгебраическая теория LP-структур развивает методологию управления знаниями в интеллектуальных системах продукционного типа, основанную на формально-логическом представлении. Ее важным достижением является метод релевантного обратного вывода, значительно снижающий число обращений к внешним источникам информации. В основе методологии лежит аппарат продукционно-логических уравнений. В настоящей статье введен и проанализирован расширенный класс таких уравнений для алгебраической модели, выразительные возможности которой охватывают интеллектуальные системы с нечеткой базой знаний. Предложены и обоснованы методы решения этих уравнений. Представленные теоремы создают теоретическую основу для дальнейших продвижений в области оптимизации нечеткого логического вывода.

Ключевые слова: *нечеткая продукционная система, релевантный обратный вывод, алгебраическая модель, нечеткая LP-структура, продукционно-логическое уравнение*

Введение

Алгебраические методы предоставляют эффективный формализм для построения и исследования моделей информационных систем широкого спектра, особенно — интеллектуальных [1, 2]. Это положение в полной мере относится к широко распространенным в информатике логическим системам продукционного типа [3—5].

В последнее десятилетие автором была создана алгебраическая теория LP-структур (*lattice production structures*) [6], позволяющая эффективно решать ряд важных задач, связанных с практическим применением продукционных систем. К таким задачам относятся эквивалентные преобразования, верификация, минимизация баз знаний, а также ускорение логического вывода. В частности, введен и исследован метод релевантного обратного вывода (LP-вывод) [7], существенно снижающий число обращений к внешним источникам информации. Впоследствии эта теория была расширена для моделирования распределенных продукционных систем [8, 9].

Важное свойство современных интеллектуальных систем — нечеткий характер знаний и рассуждений [10]. Поэтому возникает актуальная задача распространения преимуществ теории LP-структур на нечеткие продукционные системы. Начало этому направлению было положено в работах [11, 12]. Были введены понятия, описывающие нечеткость LP-структуры, и изучены отдельные полезные свойства нечеткого LP-вывода. В работах [13, 14] были представлены исследования, систематически обобщающее теорию LP-структур для управления нечеткими базами знаний. Введена базовая терминология FLP-структур с нечетким логическим отношением (Fuzzy LP-структуры), доказаны основные свойства — замкнутость, существование канонической формы и логической редукции.

Настоящая работа дополняет эту модель определением и исследованием аппарата продукционно-логических уравнений в FLP-структуре. В результате создается теоретическая основа для формального исследования и оптимизации обратного *нечеткого логического вывода*.

Статья состоит из следующих разделов. В разд. 1 введены необходимые базовые понятия и обозначения, сформулированы исходные математические результаты. В разд. 2 определен класс продукционно-логических уравнений в нечеткой LP-структуре, приведена теорема, открывающая возможности для их исследования. В разд. 3 установлены основные свойства уравнений и обоснованы методы их упрощения. В Заключении подведены итоги и указаны некоторые перспективы дальнейшего исследования.

1. Основы теории нечетких LP-структур

Исходные для настоящего раздела положения теории решеток, нечетких множеств и бинарных отношений изложены, например, в работах [15, 16]. Напомним некоторые из них, чтобы зафиксировать используемые здесь обозначения.

Нечеткое множество $A = (F, \mu_A)$ определяется функцией принадлежности $\mu_A : F \to [0, 1]$ на некотором обычном ("четком") множестве F. Величина $\mu_A(a)$ называется степенью принадлежности a к F. Носитель S_A нечеткого множества A ($S_A = \text{support } A$) — это четкое подмножество всех элементов $a \in F$, на которых $\mu_A(a) > 0$. Нечеткое множество называется конечным, если конечен его носитель.

Нечеткое бинарное отношение R на множестве F — это нечеткое множество упорядоченных пар эле-

ментов из F с заданной функцией принадлежности $\mu_R : F \times F \to [0, 1]$. Отношение R на произвольном множестве F называется *рефлексивным*, если для любого $a \in F$ справедливо $\mu_R(a, a) = 1$.

Для моделирования нечеткого логического вывода можно использовать композицию нечетких отношений, в частности, в классической семантике, а именно — (max-min)-композицию. Композиция $R^2 = R \circ R$ определяется следующим образом:

$$\mu_{R^2}(a, c) = \max_b \left(\min\left(\mu_R(a, b), \mu_R(b, c)\right)\right),$$

где $a, b, c \in F$.

Нечеткое бинарное отношение R на множестве F *транзитивно*, если для любых $a, b, c \in F$ справедливо $\mu_R(a, c) \geqslant \min\left(\mu_R(a, b), \mu_R(b, c)\right)$. Таким образом, свойство транзитивности для отношения R эквивалентно вложению $R^2 \subseteq R$ в смысле нечетких множеств. Существует замыкание произвольного нечеткого отношения относительно свойств рефлексивности и транзитивности. Обзор алгоритмов его построения представлен в работе [17].

Известна также задача нахождения транзитивной редукции: для данного отношения R ищется минимальное нечеткое отношение R' такое, что его транзитивное замыкание совпадает с транзитивным замыканием R. Решению этой задачи посвящена, в частности, работа [18].

Пусть дана атомно-порожденная решетка $\mathbb{F}$, представляющая собой множество всех конечных подмножеств некоторого универсума F. Чтобы это подчеркнуть, вместо символов $\leqslant$, $\geqslant$, $\wedge$ и $\vee$, принятых в общей теории решеток [15], будем использовать знаки теоретико-множественных операций $\subseteq$, $\supseteq$ $\bigcap$ и $\bigcup$, а элементы решетки обозначать, как правило, большими буквами. Исключение составляют атомы, размечаемые маленькими буквами.

На $\mathbb{F}$ вводится нечеткое бинарное отношение R, содержащее $\supseteq$, а также обладающее транзитивностью и дистрибутивностью. Последнее свойство имеет следующую семантику [13].

Определение 1.1. Нечеткое бинарное отношение $R = (\mathbb{F}, \mu_R)$ называется дистрибутивным, если для любых $A, B_1, B_2 \in \mathbb{F}$ справедливо $\mu_R(A, B_1 \cup B_2) \geqslant \min\left(\mu_R(A, B_1), \mu_R(A, B_2)\right)$.

Отношение с указанными выше тремя свойствами будем называть продукционно-логическим, или просто логическим.

Определение 1.2. Под нечеткой LP-структурой (FLP-структурой) подразумевается алгебраическая система, представляющая собой решетку, на которой задано нечеткое продукционно-логическое отношение.

Как показано в работе [13], такая структура может служить алгебраической моделью интеллектуальной системы продукционного типа с нечеткими правилами. Рассматриваемое в ней бинарное отношение моделирует совокупность нечетких продукций. Его свойства естественным образом отражают возможности нечеткого логического вывода на решетке.

Для дальнейшего изложения потребуются некоторые связанные с FLP-структурами результаты, полученные в работах [13, 14].

Заданное на решетке исходное отношение R, как правило, не является логическим, однако может быть рассмотрено его логическое замыкание $\bar{R}$. Это замыкание по сути моделирует все потенциальные логические выводы в продукционной системе.

Логическим замыканием $\bar{R}$ нечеткого бинарного отношения R называется наименьшее логическое отношение, содержащее R. Заметим, что определяющее решетку отношение включения $\supseteq$ само является логическим отношением, наименьшим из всех возможных.

В работе [13] доказано существование логического замыкания и описана его структура. Это позволило ввести понятие эквивалентных отношений, т. е. в приложениях — формально эквивалентных нечетких баз знаний.

Два нечетких отношения R, P на общей решетке называются (*логически*) *эквивалентными* ($R \sim P$), если их логические замыкания совпадают. *Эквивалентным преобразованием* нечеткого отношения R называется такая модификация его функции принадлежности $(\mu_R \to \mu_P)$, что полученное в результате новое отношение P логически эквивалентно R.

Нечеткое отношение на атомно-порожденной решетке $\mathbb{F}$ называется *каноническим*, если его функция принадлежности положительна лишь на парах вида (A, a), где $A \in \mathbb{F}$, a — атом в $\mathbb{F}$. В работе [13] доказана теорема о существовании эквивалентного канонического отношения для произвольного R. В моделируемой продукционной системе каноническое отношение соответствует множеству правил так называемого *хорновского* типа.

В работе [14] исследовано содержание логических связей в нечеткой LP-структуре. Для нечеткого отношения R на решетке $\mathbb{F}$ вводится отношение $\tilde{R}$ (с функцией принадлежности $\mu_{\tilde{R}} = \tilde{\mu}_R$), полученное последовательным выполнением следующих действий (шагов).

1. Объединить R с отношением рефлексивности на решетке $\mathbb{F}$ и обозначить новое отношение R_1.

2. Расширить R_1 всевозможными парами (A, B), где $A = \bigcup A_t$, $B = \bigcup B_t$, $t \in T$ — объединения элементов $\mathbb{F}$, и обозначить новое отношение R_2. Точнее, для каждой такой пары доопределить функцию принадлежности μ_{R_1} отношения R_1 следующим образом:

$$\mu_{R_2}(X, Y) =$$

$$= \begin{cases} \max\left(\mu_{R_1}(A, B), \min\limits_t \left((\mu_{R_1}(A_t, B_t)\right)\right), \text{при } X = A, Y = B \\ \mu_{R_1}(X, Y), \text{иначе}. \end{cases}$$

3. Полученное отношение R_2 объединить с отношением $\supseteq$.

В работе [14] показано, что логическое замыкание нечеткого отношения R совпадает с транзитивным замыканием отношения $\tilde{R} \supseteq R$, построенного выше по R в виде "дистрибутивного многообразия". Сформулируем этот результат, так как он будет использован в настоящей работе.

Теорема 1.1. Логическое замыкание нечеткого отношения R совпадает с транзитивным замыканием $\tilde{R}^*$ соответствующего отношения $\tilde{R}$.

2. Продукционно-логические уравнения в FLP-структуре

В этом разделе введен связанный с нечеткими LP-структурами класс уравнений. Рассмотрен общий подход к вычислению решений.

Пусть дано нечеткое отношение R на атомно-порожденной решетке $\mathbb{F}$. Пусть также имеет место $\mu_R(A, B) > 0$ для некоторых элементов $A, B \in \mathbb{F}$. Тогда B может быть назван образом A, а A — прообразом B при отношении R. В FLP-структуре каждый элемент решетки может иметь много образов и прообразов, причем с различной степенью принадлежности (величиной $\mu_R(A, B)$).

Для данного $B \in \mathbb{F}$ *минимальным прообразом* при отношении R называется такой элемент $A \in \mathbb{F}$, что $\mu_R(A, B) > 0$ и A является минимальным в том смысле, что не содержит никакого другого $A_1 \in \mathbb{F}$, для которого $\mu_R(A_1, B) > 0$.

Определение 2.1. Атом $x \in \mathbb{F}$ называется начальным при нечетком отношении R, если нет ни одной пары $A, B \in \mathbb{F}$ такой, что $\mu_R(A, B) > 0$, причем x содержится в B и не содержится в A. Элемент X называется начальным, если все его атомы являются начальными. Подмножество $\mathbb{F}_0(R)$ (иногда будем обозначать $\mathbb{F}_0$), состоящее из всех начальных элементов $\mathbb{F}$, называется начальным множеством решетки $\mathbb{F}$ при отношении R.

Определенное выше начальное множество $\mathbb{F}_0$ образует подрешетку в $\mathbb{F}$.

Пусть $\bar{R}$ — логическое замыкание отношения R (см. разд. 1). Учитывая его структуру [13], нетрудно убедиться, что множества $\mathbb{F}_0(R)$ и $\mathbb{F}_0(\bar{R})$ совпадают.

Рассмотрим уравнение

$$\bar{R}(X) = B, \tag{1}$$

где $B \in \mathbb{F}$ — заданный элемент, $X \in \mathbb{F}$ — неизвестный.

Определение 2.2. *Приближенным решением* уравнения (1) называется любой прообраз элемента B в $\mathbb{F}_0$ (при отношении $\bar{R}$). *Решением (точным)* (1) называется любой минимальный прообраз элемента B в $\mathbb{F}_0$. *Общим решением* уравнения называется совокупность всех его решений $\{X_s\}$, $s \in S$.

Уравнения вида (1) будем называть *продукционно-логическими уравнениями* в нечеткой LP-структуре.

Замечание 2.1. По определению, точное решение уравнения (1) является и приближенным. Кроме того, приближенное решение всегда содержит хотя бы одно точное решение.

Основное обстоятельство, создающее трудности для процесса решения уравнения (1), состоит в том, что обычно задано лишь отношение R. Оно в моделируемой интеллектуальной системе соответствует известному множеству продукций — базе знаний. Решение же требуется найти как прообраз правой части (1) при отношении $\bar{R}$ — логическом замыкании R. При этом полное построение логического замыкания нецелесообразно, поскольку на практике потребует неприемлемый объем ресурсов, как вычислительных, так и в смысле занимаемой памяти.

Дополнительный фактор, усложняющий не только методы решения (1), но и саму постановку этой задачи — нечеткость отношения R. Кроме требуемой в определении 2.2 минимальности искомого прообраза X, необходимо учитывать и вторую его характеристику, а именно — значение функции принадлежности $\mu_{\bar{R}}(X, B)$.

Очевидно, что лучшими решениями будут те, которые дадут большее значение функции принадлежности, соответственно в продукционной системе — результаты обратного вывода с более высоким коэффициентом уверенности. Можно даже попробовать в некоторой мере пожертвовать точностью решения в пользу повышения коэффициента уверенности до практически приемлемых результатов. Подробному исследованию данного вопроса в будущем, возможно, помогут методы многокритериальной оптимизации.

В настоящей работе рассматривается простейшая постановка задачи для (1). Ищем решения, на которых функция принадлежности принимает положительные значения. Однако для каждого полученного решения X это значение величины $\mu_{\bar{R}}(X, B)$ должно быть вычислено.

Далее выясним вопрос о том, каким образом меняется общее решение уравнений вида (1) при объединении их правых частей. Точнее, можно ли для нечеткого отношения R вместо исходного уравнения решить несколько уравнений с более простыми правыми частями.

Лемма 2.1. Пусть X_1 — решение уравнения вида (1) с правой частью B_1, а Y_1 — решение уравнения того же вида с правой частью B_2. Тогда $X_1 \cup Y_1$ является приближенным решением уравнения

$$\bar{R}(X) = B_1 \cup B_2. \tag{2}$$

Доказательство. Поскольку $\bar{R}$ содержит отношение включения, то $\mu_{\bar{R}}(X_1 \cup Y_1, X_1) = 1$ и $\mu_{\bar{R}}(X_1 \cup Y_1, Y_1) = 1$. Отсюда, так как по условию леммы $\mu_{\bar{R}}(X_1, B_1) > 0$ и $\mu_{\bar{R}}(X_1, B_2) > 0$, в силу транзитивности $\bar{R}$ имеем $\mu_{\bar{R}}(X_1 \cup Y_1, B_1) > 0$ и $\mu_{\bar{R}}(X_1 \cup Y_1, B_2) > 0$. Из последних двух соотношений, пользуясь дистрибутивностью $\bar{R}$, получим $\mu_{\bar{R}}(X_1 \cup Y_1, B_1 \cup B_2) > 0$.

Теорема 2.1. Пусть $\{X_p\}$, $p \in P$ — общее решение уравнения вида (1) с правой частью B_1, а $\{Y_q\}$, $q \in Q$ — общее решение уравнения того же вида с правой частью B_2. Тогда общее решение уравнения (2) представляет собой множество всех элементов вида $X_p \cup Y_q$, из которого исключены элементы, содержащие другие элементы этого же множества.

Доказательство. По лемме 2.1 каждый элемент $X_p \cup Y_q$ является прообразом для $B_1 \cup B_2$, т. е. содержит хотя бы одно (точное) решение (2). Остается показать, что уравнение (2) не имеет решений, отличных от вида $X_p \cup Y_q$. Предположим противное — пусть некоторый $Z \in \mathbb{F}$, являясь решением (2), не совпадает ни с одним элементом вида $X_p \cup Y_q$. При этом Z не может и содержать ни одного другого элемента вида $X_p \cup Y_q$, иначе он не был бы решением (оно по определению минимально). Отсюда следует, что Z не содержит ни одного X_p (либо ни одного Y_q, что симметрично).

Вместе с тем поскольку $\mu_{\bar{R}}(Z, B_1 \cup B_2) > 0$, то в силу $\mu_{\bar{R}}(B_1 \cup B_2, B_1) = 1$ и $\mu_{\bar{R}}(B_1 \cup B_2, B_2) = 1$, а также транзитивности $\bar{R}$, получаем $\mu_{\bar{R}}(Z, B_1) > 0$ и

$\mu_{\bar{R}}(Z, B_2) > 0$. Последние неравенства означают, что Z содержит хотя бы по одному решению уравнений с правыми частями B_1 и B_2. Поскольку все эти решения присутствуют соответственно в множествах $\{X_p\}$ и $\{Y_q\}$, приходим к противоречию.

3. Методы упрощения процесса решения уравнений

Рассмотрим методы эквивалентного упрощения нечетких уравнений вида (1), дополняющие теорему 2.1. Будем предполагать, что R является конечным каноническим нечетким отношением на решетке $\mathbb{F}$, не содержащим пар отношения $\supseteq$, а правая часть B уравнения (1) представляет конечное объединение атомов.

Введем расслоение нечеткого отношения R на виртуальные слои $\{R^t \mid t \in T\}$. Его цель — облегчить исследование свойств логического замыкания $\bar{R}$. Кроме того, в отдельных слоях упрощается построение и исследование алгоритмов, связанных с процессом решения уравнения.

Построение слоев начнем в терминах обычных (четких) множеств. Рассмотрим носитель R — четкое отношение S_R. Напомним, что это множество всех упорядоченных пар элементов решетки, на которых функция принадлежности μ_R принимает положительные значения. Разобьем его на непересекающиеся подмножества, каждое из которых образовано всеми парами $(A, x_p) \in S_R$ с одним и тем же атомом x_p в качестве правой части. Такое разбиение возможно, поскольку исходное отношение R является каноническим. Обозначим эти подмножества $S_R(p)$ по их атому x_p, $p \in P$.

Согласно работе [6] слоем S_R^t в отношении S_R называется его подмножество, образованное упорядоченными парами, взятыми по одной из каждого непустого $S_R(p)$, $p \in P$. Для отношения S_R выполняется ряд полезных свойств, следующих из способа построения множества $\{S_R^t \mid t \in T\}$.

Замечание 3.1. Каждый слой S_R^t содержит максимально возможное подмножество пар в S_R с уникальными правыми частями. При добавлении к слою еще одной пары из S_R свойство уникальности нарушится.

Замечание 3.2. Любое подмножество пар в S_R с уникальными правыми частями содержится в некотором слое.

Замечание 3.3. В общем случае слои имеют непустые пересечения. Объединение всех слоев равно S_R.

Далее, возвращаясь к нечеткому отношению R, можно сформулировать следующее определение.

Определение 3.1. Слоем R^t в нечетком отношении R называется отношение, определяемое следующей функцией принадлежности:

$$\mu_{R^t}(A, b) = \begin{cases} \mu_R(A, b), \text{ если } (A,b) \in S_R^t \\ 0, \text{ иначе} \end{cases}.$$

Согласно сделанным построениям $R = \bigcup_t R^t$. Кроме того, каждое $S_R^t = S_{R_t}$ — носитель нечеткого отношения R^t.

Замечания 3.1—3.3 имеют очевидные аналоги для случая нечеткого отношения R и его расслоения $\{R^t \mid t \in T\}$. В частности, имеют место следующие свойства.

Замечание 3.1′. Каждый слой R^t представляет максимально возможное подмножество R, в котором функция принадлежности μ_R положительна только на парах элементов решетки с уникальными правыми частями. При расширении подмножества $R^t \subseteq R$ это свойство уникальности нарушится.

Замечание 3.2′. Любое подмножество в R, в котором функция принадлежности μ_R положительна только на парах элементов решетки с уникальными правыми частями, содержится в некотором слое.

Пусть $A \in \mathbb{F}$. Будем говорить, что элемент $B \in \mathbb{F}$ получен из A применением пары (Y, z), если $\mu_R(Y, z) > 0$, $Y \subseteq A$ и $B = A \cup z$. При этом очевидно, что $\mu_{\bar{R}}(A, B) \geqslant \mu_R(Y, z)$.

Рассмотрим некоторые свойства построенного выше расслоения $\{R^t \mid t \in T\}$ для отношения R. Здесь, в частности, окажется полезным следующее утверждение.

Лемма 3.1. Пусть $\mu_{\bar{R}}(A, B) > 0$. Тогда для вывода этого значения существует кортеж $(C_1, ..., C_m)$ элементов решетки $\mathbb{F}$ такой, что $A = C_0 \subset C_1 \subset ... \subset C_m \supseteq B$, причем каждый элемент $C_j, j > 0$ получен из C_{j-1} применением некоторой пары (X_j, z_j). При $\mu_R(A, B) > 0$ полагаем $m = 0$.

Доказательство. В условиях леммы по теореме 1.1 имеется упорядоченный конечный набор элементов $(B_1, ..., B_n)$ такой, что в последовательности (B_0, B_1), (B_1, B_2), ..., (B_n, B_{n+1}), где $B_0 = A$, $B_{n+1} = B$, для каждой пары (B_j, B_{j+1}) справедливо $\mu_{\tilde{R}}(B_j, B_{j+1}) > 0$ (см. разд. 1), причем $\mu_{\bar{R}}(A, B) \geqslant \min\limits_{0 \leqslant j \leqslant n} \mu_{\tilde{R}}(B_j, B_{j+1})$. При $\mu_{\tilde{R}}(A, B) > 0$ считаем, что $n = 0$.

Далее заметим, что по построению отношения $\tilde{R}$ имеет место $\mu_{\tilde{R}}(B_j, B_j \cup B_{j+1}) \geqslant \mu_{\tilde{R}}(B_j, B_{j+1}) > 0$ для всех $j = 0, ..., n$. Применяя такое свойство последовательно по $j = 0, ..., n$, получим новый набор $(\tilde{B}_1, ..., \tilde{B}_n)$, где $\tilde{B}_k = \bigcup\limits_{j=0,...,k} B_j$, $k = 0, ..., n$. Данный кортеж, наряду с исходным $(B_1, ..., B_n)$, также реализует транзитивную связь элементов решетки A и B. При этом он не уменьшает выводимое значение $\mu_{\bar{R}}(A, B)$ и обладает дополнительным свойством монотонности $A = \tilde{B}_0 \subset \tilde{B}_1 \subset ... \subset \tilde{B}_n \supseteq B_{n+1} = B$.

Рассмотрим далее структуру пар $(\tilde{B}_{j-1}, \tilde{B}_j)$, $j = 1, ..., n$. Учитывая содержание процесса построения отношения $\tilde{R}$ (см. разд. 1), нетрудно заметить следующий факт. Отношение R_1 (это R, дополненное отношением рефлексивности на решетке) образует базис при построении $\tilde{R}$ на основе операции объединения. Это означает, что каждое значение $\mu_{\tilde{R}}(\tilde{B}_{j-1}, \tilde{B}_j) > 0$ может быть выведено из некоторого (рефлексивно-дополненного) нечеткого подмножества R_j отношения R путем объединения левых (для $\tilde{B}_{j-1}$) и правых (для $\tilde{B}_j$) частей упорядоченных пар. Отсюда следует, что элемент $\tilde{B}_j$, включающий $\tilde{B}_{j-1}$, может быть по-

лучен из $\tilde{B}_{j-1}$ последовательным применением пар отношения $R_j \subseteq R$. Распространяя данное свойство на каждую пару $(\tilde{B}_{j-1}, \tilde{B}_j)$, $j = 1, ..., n$, при этом исключая избыточные (приносящие меньшие выводимые значения $\mu_{\bar{R}}$) применения (рефлексивно-дополненных) пар R, приходим к утверждению леммы при некотором $m \geqslant n$.

Замечание 3.4. Построенный в лемме кортеж $(C_0, ..., C_m)$ можно выбрать *точным* в следующем смысле. Правый атом z_j каждой примененной в кортеже для перехода от C_{j-1} к C_j пары (Y_j, z_j) должен содержаться в B либо в левой части Y_{j+k} некоторой применяемой позднее пары (Y_{j+k}, z_{j+k}), и не может содержаться в C_{j-1}. Переходы, не удовлетворяющие такому свойству, не являются необходимыми для получения B. Поэтому они могут быть исключены из кортежа вместе с C_j.

Далее докажем свойства, характеризующие расслоение $\{R^t\}$ отношения R с точки зрения решения логических уравнений вида (1).

Лемма 3.2. Если $\mu_{\bar{R}}(A, B) > 0$, то в R существует слой R^t такой, что $\mu_{\bar{R}^t}(A, B) = \mu_{\bar{R}}(A, B)$. Здесь $\bar{R}^t$ — логическое замыкание отношения R^t.

Доказательство. Пусть $\mu_{\bar{R}}(A, B) > 0$. Тогда по лемме 3.1 имеется упорядоченный "монотонно возрастающий" набор элементов $(C_1, ..., C_m)$ такой, что в последовательности (C_0, C_1), (C_1, C_2), ..., (C_{m-1}, C_m), где $C_0 = A$, $C_m \supseteq B$, каждый элемент C_j, $j > 0$ получается из C_{j-1} применением пары (Y_j, z_j). Кроме того, $\mu_{\bar{R}}(A, B) \geqslant \min\limits_{0 \leqslant j \leqslant m-1} \mu_{\bar{R}}(C_j, C_{j+1})$.

Из свойства монотонности $C_0 \subset C_1 \subset ... \subset C_m$ следует, что все переходы вида (C_{j-1}, C_j) $(j = 0, ..., m)$ могут быть получены применением пар элементов решетки с уникальными правыми частями. Действительно, любая пара (Y_j, z_j), впервые (слева направо) примененная к C_{j-1} для получения C_j, исключает необходимость еще одного применения любой пары с тем же атомом z_j в правой части, поскольку z_j уже содержится во всех элементах $(C_j, C_{j+1}, ..., C_m)$. Отсюда с учетом замечания 3.2′ нетрудно заметить, что существует слой R^t, содержащий все примененные пары в кортеже. Таким образом, получаем $\mu_{\bar{R}^t}(A, B) = \mu_{\bar{R}}(A, B)$.

Следствие 3.1. Логическое замыкание канонического нечеткого отношения R равно объединению логических замыканий его слоев, т. е. $\bar{R} = \bigcup\limits_{t \in T} \bar{R}^t$.

Доказательство. Поскольку $R^t \subseteq R$, то согласно работе [13] (замечание 3.1) имеем $\bar{R}^t \subseteq \bar{R}$, т. е. $\mu_{\bar{R}^t}(A, B) \leqslant \mu_{\bar{R}}(A, B) (\forall t \in T)$. Обратно, если $\mu_{\bar{R}}(A, B) > 0$, то по лемме 3.2 справедливо $\mu_{\bar{R}}(A, B) = \mu_{\bar{R}^t}(A, B)$ при некотором $t \in T$, что означает $\mu_{\bar{R}}(A, B) \leqslant \max \mu_{\bar{R}^t}(A, B)$.

Будем говорить, что решение X уравнения (1) (точное или приближенное) порождается в R некоторым слоем R^t, если $\mu_{\bar{R}}(X, B) = \mu_{\bar{R}^t}(X, B)$.

Замечание 3.5. Согласно лемме 3.2 любое решение уравнения (1) порождается в R некоторым слоем R^t.

Замечание 3.6. Очевидно, что для нахождения решения уравнения (1) в слое R^t достаточно вместо (1) решить аналогичное уравнение с отношением $\bar{R}^t$.

Последние замечания не гарантируют, что два различных слоя не могут порождать одного и того же решения. Кроме того, могут существовать слои, дающие точное решение в R^t, но приближенное в R. Некоторые слои могут вообще не давать решений. Однако справедливо утверждение о том, что один слой не может порождать более одного точного решения.

Лемма 3.3. Ни один слой R^t отношения R не может порождать двух различных точных решений уравнения (1).

Доказательство. Предположим противное, что некоторый слой R^t порождает два точных решения $X_1 \neq X_2$. Пусть для определенности существует начальный атом $z_0 \in X_1$ и $z_0 \notin X_2$. Согласно леммам 3.1 и 3.2 для вывода величины $\mu_{\bar{R}^t}(X, B) > 0$ существует "монотонно возрастающий" кортеж $(X_1 = C_0, C_1, ..., C_m)$, каждый переход j в котором осуществляется соответствующей парой элементов (Y_j, z_j), для которой $\mu_{\bar{R}^t}(Y_j, z_j) > 0$. Выберем кортеж точным в соответствии с замечанием 3.4.

Далее будем просматривать $(C_0, ..., C_m)$ слева направо, и при этом строить некоторую специальную последовательность пар элементов решетки. Поскольку рассматривается точное решение X_1, оно не содержит "лишних" атомов. Его атом z_0, как и любой атом точного решения: *а*) содержится в левой части хотя бы одной пары (Y_1, z_1), осуществляющей переход в кортеже $(C_0, ..., C_m)$; либо *б*) z_0 содержится в B — правой части уравнения (1). Если верно *а*, добавим соответствующую пару (Y_1, z_1) к накапливаемой последовательности пар, в противном случае завершим просмотр. На следующем шаге (переместившись в кортеже вправо до применения пары (Y_1, z_1)) ищем аналогичную пару (Y_2, z_2) для атома z_1. Для него имеем ту же альтернативу: *а* либо *б*, поскольку просматриваемый кортеж является точным (см. замечание 3.4). Рано или поздно, закончив просмотр кортежа, получим последовательность пар-переходов $\{(Y_i, z_i) \mid i = 1, ..., n\}$, причем $z_i \in Y_{i+1}$, $(0 \leqslant i \leqslant n-1)$, $z_n \in B$. Если эта последовательность окажется пустой $(n = 0)$, то имеем $z_0 \in B$.

Рассмотрим теперь *точный* кортеж вывода $(\tilde{C}_0, ..., \tilde{C}_k)$, соответствующий другому решению — X_2. Его будем просматривать справа налево, пытаясь в переходах в обратном порядке выделить ту же самую последовательность $\{(Y_i, z_i)\}$. Поскольку $z_n \in B$, то по построению кортежа имеем и $z_n \in \tilde{C}_k$. Далее циклически выполняем однотипные шаги. Для атома z_n возможны два варианта: *а*) существует пара с правой частью z_n, применение которой вызвало некоторый переход в кортеже $(\tilde{C}_0, ..., \tilde{C}_k)$; либо *б*) z_n содержится в $\tilde{C}_0 = X_2$. В случае *а* упомянутой парой может быть лишь (Y_n, z_n), так как слой R^t не может содержать двух пар с одинаковой правой частью z_n, дающих положительное значение функции принадлежности (замечание 3.1′). В этом случае переходим к предыдущей паре в выделяемой последовательности $\{(Y_i, z_i)\}$, исходя из отправной точки $z_{n-1} \in Y_n$ — правой части

очередной пары. В случае b получим противоречие: если $n > 0$, то атом z_n не является начальным и, соответственно, не может содержаться в решении X_2; вариант $n = 0$ противоречит сделанному в начале доказательства предположению $z_0 \notin X_2$. Продолжая данный процесс, в силу конечности последовательности пар $\{(Y_i, z_i)\}$ рано или поздно придем к ситуации *б*, противоречащей сделанным предположениям, что и доказывает лемму.

Объединяя установленные выше результаты, можно сформулировать следующее утверждение.

Теорема 3.1. Для нахождения общего решения уравнения (1) достаточно найти (единственное) решение X_t в каждом слое R^t, в котором оно существует. Далее из полученного множества решений необходимо исключить элементы, содержащие другие элементы этого же множества.

Утверждение теоремы следует из замечания 3.5 и леммы 3.3.

Теоремы 2.1, 3.1 и замечание 3.6 позволяют свести решение исходного уравнения (1) к нахождению решений уравнений вида

$$\bar{R}^t(X) = b, \qquad (3)$$

где b — не начальный атом решетки $\mathbb{F}$, R^t — слой в R.

Вопросам существования и конечному способу решения уравнения (3) будет посвящена отдельная статья.

Заключение

В настоящей работе определен и исследован класс продукционно-логических уравнений в нечеткой LP-структуре, расширяющей область применения этой алгебраической теории до моделирования нечетких интеллектуальных систем продукционного типа.

Предложены и обоснованы методы решения этих уравнений. Нахождение решения продукционно-логического уравнения соответствует обратному нечеткому логическому выводу. Представленные теоремы создают теоретическую основу для дальнейших продвижений в области оптимизации нечеткого логического вывода.

В качестве продолжения работы планируются к рассмотрению вопросы о непосредственной разрешимости упрощенных уравнений и числе их решений.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00037.

Список литературы

1. **Oles F. J.** An Application of Lattice Theory to Knowledge Representation// Theor. Comput. Sci. Oct. — 2000. — Vol. 249, No. 1. — P. 163—196.

2. **Бениаминов Е. М.** Алгебраические методы в теории баз данных и представлении знаний. — М.: Научный мир, 2003. — 184 с.

3. **Жожикашвили А. В., Стефанюк В. Л.** Алгебраическая теория продукционных систем // VIII национальная конференция по искусственному интеллекту с международным участием КИИ-2002. 7—12 октября 2002, г. Коломна: Тр. конференции. Т. 1. — М.: Физматлит, 2002. — С. 428—436.

4. **Maciol A.** An application of rule-based tool in attributive logic for business rules modeling Expert Systems with Applications. — 2008. — Vol. 34, No. 3. — P. 1825—1836.

5. **Дородных Н. О., Юрин А. Ю.** Использование диаграмм классов UML для формирования продукционных баз знаний // Программная инженерия. — 2015. — № 4. — С. 3—9.

6. **Махортов С. Д.** Математические основы искусственного интеллекта: теория LP-структур для построения и исследования моделей знаний продукционного типа / Под ред. В. А. Васенина. — М.: Изд-во МЦНМО, 2009. — 304 с.

7. **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.

8. **Махортов С. Д.** Алгебраическая модель распределенной логической системы продукционного типа // Программная инженерия. — 2015. — № 12. — С. 32—38.

9. **Махортов С. Д.** Продукционно-логические уравнения в распределенной LP-структуре // Программная инженерия. — 2016. — Т. 7, № 7. — С. 324—329.

10. **Батыршин И. З., Недосекин А. О., Стецко А. А.** и др. Нечеткие гибридные системы: Теория и практика / Под ред. Н. Г. Ярушкиной. — М.: Физматлит, 2007. — 208 с.

11. **Махортов С. Д., Шмарин А. Н.** Нечеткий LP-вывод и его программная реализация // Программная инженерия. — 2013. — № 12. — С. 34—38.

12. **Махортов С. Д., Шмарин А. Н.** Оптимизация метода LP-вывода // Нейрокомпьютеры. Разработка, применение. — 2013. — № 9. — С. 59—63.

13. **Махортов С. Д.** Алгебраическая модель интеллектуальной системы с нечеткими правилами // Программная инженерия. — 2019. — Т. 10, № 11—12. — С. 457—463.

14. **Махортов С. Д., Клейменов И. В.** О логической редукции алгебраической модели интеллектуальной системы с нечеткими правилами // Вестник Воронежского государственного университета. Серия: Физика. Математика. — 2019. — № 3. — С. 67—78.

15. **Биркгоф Г.** Теория решеток: пер. с англ.— М.: Наука, 1984. — 568 с.

16. **Рыжов А. П.** Элементы теории нечетких множеств и ее приложений. М.: Диалог-МГУ, 2003. — 81 с.

17. **Garmendia L., Del Campo R. G., López V., Recasens J.** An Algorithm to Compute the Transitive Closure, a Transitive Approximation and a Transitive Opening of a Fuzzy Proximity // Mathware & Soft Computing. — 2009. — Vol. 16. — P. 175—191.

18. **Hashimoto H.** Reduction of a Nilpotent Fuzzy Matrix // Information Sciences. — 1982. — Vol. 27. — P. 233—243.

Methods for Solving Production-Logical Equations in a Fuzzy LP-Structure

S. D. Makhortov, msd_exp@outlook.com, Voronezh State University, Voronezh, 394018, Russian Federation

Corresponding author:

Makhortov Sergey D., Head of Department, Voronezh State University, Voronezh, 394018, Russian Federation
E-mail: msd_exp@outlook.com

Received on August 12, 2020
Accepted on September 14, 2020

Algebraic methods provide an effective formalism for constructing and researching models of a wide range of information systems, especially intelligent ones. This provision fully applies to the production-type logical systems widespread in computer science.

In the last decade, the author has created an algebraic theory of LP-structures (lattice production structures), which makes it possible to effectively solve a number of important problems related to production systems. Such tasks include equivalent transformations, verification, minimization of knowledge bases, and acceleration of logical inference. In particular, the method of relevant backward inference (LP-inference) was introduced and investigated, which significantly reduces the number of calls to external sources of information. Subsequently, this theory was expanded to model of distributed production systems.

An important property of modern intelligent systems is the fuzzy nature of knowledge and reasoning. Therefore, an urgent problem arises of extending the advantages of the theory of LP-structures to fuzzy production systems. The beginning of this direction was laid in the previous articles of the author. Concepts describing the fuzzy LP-structure were introduced, and some useful properties of fuzzy LP-inference were studied. In recent works, studies are presented that systematically generalize the theory of LP-structures for managing fuzzy knowledge bases. The basic terminology of FLP-structures with a fuzzy logical relation (Fuzzy LP-structures) is introduced, the basic properties are proved — closedness, the existence of a canonical form and logical reduction.

This work complements this model by defining and investigating the apparatus of production-logical equations in the FLP-structure. Methods for solving these equations are proposed and substantiated. Finding a solution to the production-logical equation corresponds to the backward fuzzy inference. The presented theorems provide a theoretical basis for further advances in the field of optimization of fuzzy inference.

As a continuation of the work, it is planned to consider questions about the direct solvability of simplified equations and the number of their solutions.

Keywords: *fuzzy production system, relevant backward inference, algebraic model, fuzzy LP-structure, production-logical equations*

For citation:

Makhortov S. D. Methods for Solving Production-Logical Equations in a Fuzzy LP-Structure, *Programmnaya Ingeneria,* 2020, vol. 11, no. 6, pp. 342—348

DOI: 10.17587/prin.11.342-348

References

1. **Oles F. J.** An Application of Lattice Theory to Knowledge Representation, *Theor. Comput. Sci*., Oct. 2000, vol. 249, no. 1, pp. 163—196.

2. **Beniaminov E. M.** *Algebraic methods in the theory of databases and knowledge representation*, Moscow, Nauchnyj mir, 2003, 184 p. (in Russian).

3. **Zhozhikashvili A. V., Stefanjuk V. L.** Algebraic theory of production systems, *VIII nacional'naja konferencija po iskusstvennomu intellektu s mezhdunarodnym uchastiem KII-2002*, Kolomna, October 7—12, 2002, Trudy konferencii. T. 1. Moscow, Fizmatlit, 2002, pp. 428—436 (in Russian).

4. **Maciol A.** An application of rule-based tool in attributive logic for business rules modeling, *Expert Systems with Applications,* 2008, vol. 34, no. 3, pp. 1825—1836.

5. **Dorodnykh N. O., Yurin A. Yu.** The Use of Diagrams of UML Classes for Production Knowledge Bases Formation, *Programmnaya Ingeneria*, 2015, no. 4, pp. 3—9 (in Russian).

6. **Makhortov S. D.** *Mathematical Foundations of Artificial Intelligence: The LP structures theory for the knowledge models of production type construction and research* / Eds. by V. A. Vasenin. Moscow, MCCME, 2009, 304 p. (in Russian).

7. **Bolotova S. Yu., Makhortov S. D.** Algorithms of the relevant backward inference that is based on production-logic equations solving, *Iskusstvennyiy intellekt i prinyatie resheniy*, 2011, no. 2, pp. 40—50 (in Russian).

8. **Makhortov S. D.** The algebraic model of the distributed logical system of the production type, *Programmnaya Ingeneria*, 2015, no. 12, pp. 32—38 (in Russian).

9. **Makhortov S. D.** Production-logical equations in the distributed LP-structure, *Programmnaya Ingeneria*, 2016, vol. 7, no. 7, pp. 324—329 (in Russian).

10. **Batyrshin I. Z., Nedosekin A. O., Stecko A. A.** et al. *The Fuzzy Gibrid Systems: Theory and Practice,* Moscow, Fizmatlit, 2007, 208 p. (in Russian).

11. **Makhortov S. D., Shmarin A. N.** Fuzzy LP-inference and its software implementation, *Programmnaya Ingeneria,* 2013, no. 12, pp. 34—38 (in Russian).

12. **Makhortov S. D., Shmarin A. N.** Optimizing LP-inference method, *Nejrokomp'jutery. Razrabotka, primenenie*, 2013, no. 9, pp. 59—63 (in Russian).

13. **Makhortov S. D.** An Algebraic Model of the Intelligent System with Fuzzy Rules, *Programmnaya Ingeneria*, 2019, vol. 10, no. 11—12, pp. 457—463 (in Russian).

14. **Makhortov S. D., Kleymenov I. V.** On Logical Reduction of an Algebraic Model of the Intellectual System with Fuzzy Rules, *Vestnik VGU. Serija Fizika. Matematika*, Voronezh, 2019, no. 3, pp. 67—78 (in Russian).

15. **Birkhoff G.** *Lattice Theory,* Rhode Island, 1995, 420 p.

16. **Ryzhov A. P.** *Elements of the Theory of Fuzzy Sets and of its Applications,* Moscow, Dialog-MGU, 2003, 81 p. (in Russian).

17. **Garmendia L., Del Campo R. G., López V., Recasens J.** An Algorithm to Compute the Transitive Closure, a Transitive Approximation and a Transitive Opening of a Fuzzy Proximity, *Mathware & Soft Computing*, 2009, no. 16, pp. 175—191.

18. **Hashimoto H.** Reduction of a Nilpotent Fuzzy Matrix, *Information Sciences*, 1982, vol. 27, pp. 233—243.

УДК 004.82 DOI: 10.17587/prin.11.349-360

К. И. Костенко, канд. физ.-мат. наук, доц. кафедры, kostenko@kubsu.ru,
Кубанский государственный университет, Краснодар

Моделирование замыканий онтологий в формализмах семантических иерархий

Представлены результаты изучения схем моделирования структур сложных знаний для разных этапов жизненных циклов и потоков знаний в интеллектуальных системах. Структуры знаний конструируются с использованием универсального и унифицированного формата представления. Основой такого формата являются алгебраические структуры знаний в формализмах представления знаний. Для синтеза знаний применяются операции специальных классов. Классы исходных данных (базы) процессов синтеза составляют простые знания. В них реализуются основы представления содержания (онтологии) моделируемых областей. Многообразия знаний, конструируемых из элементов таких множеств классов (замыкания онтологий), определяют выразительные возможности онтологий. Для моделирования основных конструктов и операций над онтологиями, принятых в дескрипционных логиках, использованы полные структурные представления знаний в формализме семантических иерархий. В операциях отражаются функциональные аспекты разных фундаментальных математических систем, адаптированные к атрибутам формализмов знаний. Построенные примеры баз операций (морфизмов) над знаниями составляют часть системы унифицированных типов многообразий алгебраических структур знаний. Показана возможность переноса схем представления и обработки знаний в формализме семантических иерархий на случай произвольных формализмов знаний. Это позволяет считать формализмы семантических иерархий универсальной платформой для моделирования потоков знаний и процессов их обработки в интеллектуальных системах. Рассмотрены схемы формирования замыканий онтологий применениями последовательностей операций над знаниями в формате семантических иерархий.

Ключевые слова: *замыкание онтологии, алгебраическая структура знания, вложение знаний, формализм семантических иерархий, когнитивная операция, гомоморфизм, гомоморфное расширение*

Введение

Теоретический фундамент интеллектуальных систем (ИС) составляют форматы представления и методы обработки абстрактных знаний. Они расширяются в прикладные модели содержания областей знаний и управления знаниями [1—3]. Индуктивная природа прикладных ИС проявляется в использовании эмпирических, слабоструктурированных семейств знаний. Они составляют содержание соответствующих областей. Системы таких знаний вкладываются в многообразия структурных представлений знаний из абстрактных математических моделей. Обработка получаемых семейств формализованных знаний связана с реализацией когнитивных целей. Цели определяют общее содержание процессов решения профессиональных задач. Используемые для этого базовые знания интегрируются в семантические структуры, согласованные с эмпирическими представлениями о схемах решений задач. Конструирование структур сложных знаний связано с моделированием слабо формализованных представлений о процессах мышления и структурной организации памяти.

Системы общих знаний об указанных понятиях формируются в разных областях. В первую очередь они развиваются в лингвистике, когнитивной психологии, системной инженерии. Применение моделей приведенных областей обеспечивает содержательную полноту формально-математического описания интеллектуальных процессов и систем с использованием сущностей фундаментальных математических моделей.

Решение профессиональных задач в ИС заключается в построении структур сложных знаний, основанном на простых знаниях. Последние составляют онтологии предметных областей. В них реализуется полная декомпозиция содержания таких областей. Например, в виде троек *RDF*. Всякая тройка представляет отдельное простое знание. Универсальным форматом таких знаний являются иерархии глубины один. Сложные знания соответствуют фрагментам содержания предметных областей. Они представляются иерархиями, которые составляются из простых знаний с помощью специальных операций. Многообразия таких сложных знаний составляют замыкания онтологий.

Для моделирования онтологий используют специальные языки. Они основаны на дескрипционных логиках и применяют ограниченные семейства конструкций построения фрагментов содержания предметных областей. Языки работы с онтологиями используют теоретико-множественные операции для извлечения фрагментов онтологий как основы процессов решения профессиональных задач. Применяемых операций недостаточно для полноты моделирования процессов мышления и структур памяти. Для этого востребованы дополнительные операции. Они связаны с инвариантами теории систем, лингвистики, когнитивной психологии, а также других разделов математики. Унификация функциональных аспектов разных разделов математики с сущностями приведенных областей составляет основу абстрактного моделирования концепции ИС. Порождающими элементами моделей ИС являются формализмы представления знаний, компоненты пространственной архитектуры и агенты реализации потоков и процессов знаний.

Дедуктивно определяемое понятие формализма представления знаний реализует обобщающее уточнение разных представлений о моделях знаний. Оно опирается на систему инвариантов и порождающих принципов, позволяющих рассматривать известные модели знаний как примеры таких формализмов. Функциональные элементы формализмов знаний позволяют конструировать потоки и процессы обработки знаний, основываясь на алгебраических, логических, топологических и кибернетических аналогах для элементов мышления и структур памяти. Моделирование онтологий в инвариантах формализмов знаний позволяет использовать указанные аналоги. Это расширяет выразительные и аналитические возможности применения онтологий, включая последние в орбиты разных разделов абстрактной математики. Онтологии составляют базисные фрагменты формализмов знаний. При моделировании процессов решения задач в соответствующих областях фрагменты расширятся в многообразия знаний формализмов, синтезируемых из элементов онтологий.

Для моделирования конструктов дескрипционных логик далее будут использованы представления знаний в специальном формализме представления знаний семантическими иерархиями, близкими к конструкциям *RDF*. Это не приводит к потере общности, поскольку указанные иерархии сохраняют свойства алгебраической и семантических структур фрагментов знаний в произвольных формализмах. Таким образом, формализмы семантических иерархий составят основу математического моделирования представления содержания областей знаний и процессов его обработки, определяемого целями соответствующих ИС.

1. Формализмы представления знаний

В понятии формализма представления знаний (формализма) обобщается многообразие применяемых моделей представления знаний. Указанное многообразие развивается преимущественно индуктивно и становится все более разнородным. Формализмы знаний представляют такие модели с помощью систем абстрактных знаний. Они основаны на фундаментальных структурных и семантических инвариантах знаний.

Определение. Формализм знаний — это четверка $\Im = (M, D, \circ, \prec)$, где M — перечислимое множество абстрактных знаний, содержащее пустое знание Λ; D — множество фрагментов знаний, в котором M является разрешимым подмножеством; $\circ : D \times D \to D$ — вычислимая операция композиции; $\prec \subseteq D \times D$ — разрешимое отношение вложения фрагментов знаний, для которого $\forall z \in D(\Lambda \prec z)$ [2].

Знание называется элементарным, если оно не может быть представлено в виде композиции других знаний. Композиция элементарных знаний, задающая заданный фрагмент знания, называется алгебраической структурой (АС) этого фрагмента [4]. В общем случае формализмов знаний операция композиции может оказаться не инъективной. Поэтому возможны формализмы, в которых отдельные фрагменты знаний имеют несколько разных АС. Дублированием элементов множества D формализма $\Im = (M, D, \circ, \prec)$ можно переопределить $\Im$ в новый формализм, в котором операция композиции является инъективной.

Специальный класс формализмов знаний (обозначается как *SH*) составляют формализмы семантических иерархий. Эти формализмы представляют специальные модели представления знаний, называемые пространствами конфигураций. Отдельные знания в таких пространствах называются конфигурациями. Они представляются нагруженными бинарными деревьями. Вершинами дерева являются двоичные наборы. Они определяют пути, ведущие из корня в вершины. Множество всех возможных вершин (наборов) обозначается как I. Пустой набор λ соответствует корню дерева. Висячие вершины дерева размечаются элементарными знаниями (конфигурациями), а внутренние — разрешимыми отношениями. Отношение, приписанное произвольной внутренней вершине дерева, выполняется между конфигурациями, представленными левым и правым поддеревьями этой вершины. Множество отношений R является перечислимым. Для R разрешимо вложение отношений. Иерархические структуры произвольных конфигураций определяют два вычислимых отображения: $\varepsilon : M \to M \times M$ и $\psi : M \to R$. Они называются разложением и связыванием конфигураций. Если $z \in M$ и $\varepsilon(z) = (z_1, z_2)$, то пара конфигураций (z_1, z_2) образует разложение конфигурации z. Если $z \in M = (\Lambda, \Lambda)$, то z называется элементарной конфигурацией. При этом $z = \Lambda$ — также элементарная конфигурация. Множество элементарных конфигураций обозначается как M_0. Если $z \in M \setminus M_0$, то $(z_1, z_2) \in \psi(z)$. Неэлементарная конфигурация $z \in M$ называется простой конфигурацией, если элементы пары $\varepsilon(z)$ принадлежат M_0. Множество простых конфигураций обозначается как M_1. Отображения ε и ψ определяют дерево полного структурного представления (ПСП) всякой конфигурации. Листья этого дерева размечены элементами M_0, а внутренние

вершины — элементами R. Множество всех вершин (висячих вершин) ПСП $z \in M$ обозначается как $D(z)$ ($O(z)$). Если $z \in M$, то обозначения $[z]_\alpha$ и $(z)_\alpha$ применяются для разметки вершины $\alpha \in D(z)$ и конфигурации, представимой поддеревом дерева ПСП z с корнем α. Дополнительное требование к отображениям ε и ψ состоит в условиях конечности глубины ПСП всякой конфигурации.

Возможны разные схемы представления пространств конфигураций формализмами семантических иерархий. Они основаны на уточнениях операции композиции конфигураций, позволяющей формировать ПСП всякой конфигурации с помощью АС этой конфигурации [4]. Если $z \in M \setminus M_0$ и $\varepsilon(z) = (z_1, z_2)$, то будем рассматривать случай, когда алгебраическая структура z $\left(\sum(z)\right)$ получается из алгебраических структур z_1 и z_2 $\left(\sum(z_1)$ и $\sum(z_2)\right)$ как композиция $\left(\sum(z_1) \circ \sum(z_2)\right) \circ \psi(z)$. Знания (фрагменты знаний) в получаемых формализмах называют конфигурациями (фрагментами) конфигураций.

Отношения из R в пространствах конфигураций являются специальными элементарными знаниями формализмов семантических иерархий. Композиции $z_1 \circ z_2$, где $z_1, z_2 \in M$ определяют разные фрагменты конфигураций, а $(z_1 \circ z_2) \circ r$ соответствует конфигурации z, если $\varepsilon(z) = (z_1, z_2)$ и $\psi(z) = r$. Если же $(z_1, z_2) \notin r$, то композиция $(z_1 \circ z_2) \circ r$ соответствует пустой конфигурации. При этом операция композиции определяется так, что всякая конфигурация, отличная от Λ, представляется единственной алгебраической структурой [2].

Понятие вложения фрагментов знаний в произвольном формализме $\Im \in SH$ моделируется понятием трассирования конфигураций [2]. Трассирование $z_1 \in M$ в $z_2 \in M$ имеет место (обозначается как $z_1 \leqslant_t z_2$), если существует изотонное (монотонное в отношении вложения двоичных последовательностей) отображение $\xi: I \to I$, для которого:

1) $\forall \alpha \in D(z_1)\left(\alpha \in O(z_1) \Leftrightarrow \xi(\alpha)) \in O(z_2)\right)$;

2) $\forall \alpha \in D(z \forall \sigma \in \{0,1\}(\xi(\alpha) \subset \xi(\alpha\sigma) \to \xi(\alpha\sigma) = \xi(\alpha)\beta\sigma\gamma)$;

3) $\forall \alpha \in D(z_1) \setminus O(z_1)\left([z_1]_\alpha \rho_1 [z_2]_{\xi(\alpha)}\right)$;

4) $\forall \alpha \in O(z_1)\left([z_1]_\alpha \rho_0 [z_2]_{\xi(\alpha)}\right)$.

Здесь ρ_1 — это отношение вложения отношений из R, а ρ_0 — разрешимое отношение порядка на множестве M_0, моделирующее сравнение содержания элементарных знаний [4]. Важными случаями (видами) трассирования являются сжатия ($\beta = \gamma = \lambda$, где λ — пустая последовательность из I), растяжения (ξ — инъективное и $\beta = \lambda$), а также o-трассирования, если $\beta = \lambda$.

Трассирования конфигураций удобны для моделирования сравнений содержания знаний, представленных конфигурациями. Они применяются для конструирования точных определений структурных и функциональных свойств знаний в формате ПСП конфигураций.

Перенос понятий и результатов исследования пространств конфигураций на общий случай формализмов знаний предполагает уточнение понятия трассирования в элементах алгебраической структуры знаний произвольных формализмов.

2. Трассирование алгебраических структур знаний

Формализмы семантических иерархий соответствуют пространствам конфигураций. Они составляют класс формализмов, допускающих возможность глубокого математического исследования. Результаты такого исследования могут быть перенесены на произвольные формализмы знаний. Последнее возможно вследствие близости форматов бинарных деревьев для алгебраических структур знаний в произвольных формализмах и ПСП конфигураций. Висячим вершинам АС всякой конфигураций и формализма $\Im \in SH$ приписываются элементарные конфигурации или отношения, а внутренние вершины размечаются операцией композиции. При этом различия в форматах ПСП и АС конфигурации допускают возможность трансформации структур таких представлений, сохраняющих их свойства. Алгебраическая структура всякой конфигурации в произвольном формализме естественно интерпретируется как ПСП некоторой конфигурации, внутренние вершины которого размечены отношением, соответствующим операции композиции и обозначаемым как $\circ$.

Возможность использования АС знаний произвольных формализмов из SH вместо ПСП конфигураций соответствующих пространств основывается на трансформации понятия трассирования ПСП конфигураций в его аналог для АС конфигураций. Для этого применяется специальное преобразование ПСП конфигураций в АС конфигураций. Общая схема такого преобразования приведена на рис. 1. Она составлена двумя правилами.

Верхнее правило относится к элементарным конфигурациям. Если $z \in M_0$, то $h(z)$ представляется алгебраической структурой, совпадающей с ПСП z. Нижнее правило применяется к неэлементарным конфигурациям. Здесь $z \in M \setminus M_0$ — такая конфигурация, для которой $\varepsilon(z) = (z_1, z_2)$ и $\psi(z) = r$. Выражение

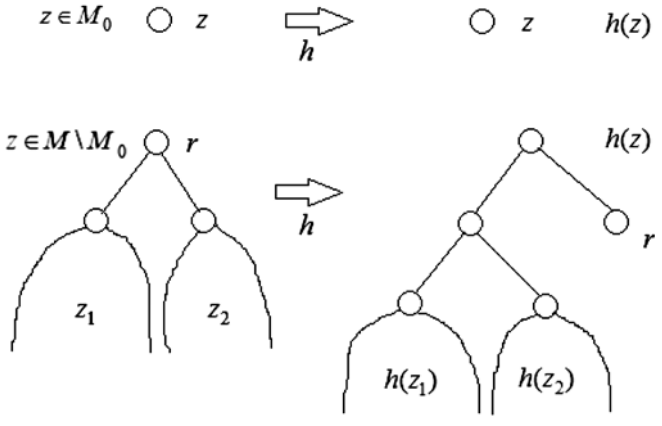

Рис. 1. Трансформация ПСП в АС конфигурации

$h(z)$ далее обозначает алгебраическую структуру z с внутренними вершинами, размеченными операцией композиции $\circ$.

Для трансформации понятий трассирования в формате ПСП конфигураций пространства конфигураций в формат АС конфигураций приведенного формализма далее будут применены элементы обозначений трансформации h, приведенные на рис. 1. Множество внутренних вершин $h(z)$, которые являются корнями фрагментов, представляющих отдельные вершины ПСП z, обозначим как $D_0(h(z))$. То есть $D_0(h(z)) = (D(h(z)) \cap \{00, 01\}^*) \setminus O(h(z))$. Здесь $D(h(z))$ $(O(h(z)))$ — множество вершин (висячих вершин) АС $h(z)$, а $\{00, 01\}^*$ — множество слов в алфавите $\{00, 01\}$, включающее пустое слово λ. Переопределим понятие трассирования произвольных конфигураций в алгебраических структурах конфигураций.

Определение. Пусть $z_1, z_2 \in M \setminus M_0$. Тогда $h(z_1)$ t-трассируется в $h(z_1)$ (обозначается как $h(z_1) \leqslant_t h(z_2)$), если существует такое изотонное отображение $\xi : I \to I$, для которого выполняются следующие соотношения:

1) $\forall \alpha \in I(\alpha \in D_0(h(z_1)) \to \xi(\alpha) \in D_0(h(z_2)))$;
2) $\forall \alpha \in D_0(h(z_1))(\xi(\alpha 0) = \xi(\alpha)0 \;\&\; \xi(\alpha 1) = \xi(\alpha)1)$;
3) $\forall \alpha \in D(h(z_1))(\alpha \in O(h(z_1)) \leftrightarrow \xi(\alpha) \in O(h(z_2)))$;
4) $\forall \alpha \in D_0(h(z_1))\forall \sigma \in \{0,1\}(\xi(\alpha 0) \subset \xi(\alpha 0\sigma) \to \exists \beta,$ $\gamma \in I(\xi(\alpha 0\sigma) = \xi(\alpha)0\beta\sigma\gamma))$;
5) $\forall \alpha \in D_0\left([z_1]_{\alpha 1} \subseteq [z_2]_{\xi(\alpha)1}\right)$;
6) $\forall \alpha \in D_0(h(z_1))\forall \sigma \in \{0,1\}\times$ $\times\left(\xi(\alpha 0\sigma) \in O(h(z_2)) \to [h(z_1)]_{\alpha 0\sigma}\rho_0[h(z_2)]_{\xi(\alpha 0\sigma)}\right)$;

Здесь ρ_0 — сравнение вложения на множестве элементарных конфигураций, а $\subseteq$ — обозначение вложения отношений на множестве M.

Соотношения 1—6 последнего определения представляют соотношения из определения трассирования конфигураций с точностью до представления отдельных вершин $\alpha \in D(z) \setminus O(z)$ тройками вершин $h(\alpha)$, $h(\alpha)0$, $h(\alpha)1$, где $h(\alpha)$ — вершина, получаемая заменой символов 0 и 1 в α на пары 00 и 01. Поэтому свойства, формулируемые для конфигураций в форматах их ПСП или АС в формализме $\Im \in SH$, переносимы с учетом последнего соответствия.

Теорема. Для любых конфигураций $z_1, z_2 \in M$ соотношения $z_1 \leqslant_t z_2$ и $h(z_1) \leqslant_t h(z_2)$ равносильны.

Доказательство. Пусть M — множество конфигураций в формализме $\Im \in SH$ и для $z_1, z_2 \in M$ выполняется соотношение $z_1 \leqslant_t z_2$. Возьмем такое изотонное отображение ξ, для которого выполняется условие трассирования z_1 в z_2. Определим t-трассирование ξ', для которого $h(z_1) \leqslant_t h(z_2)$. Для этого применим вспомогательное отображение $g : I \to I$, которое ставит в соответствие вершинам ПСП конфигураций вершины АС этих конфигураций. Оно определяется следующими соотношениями:

$$g(\lambda) = \lambda \;\text{ и }\; \forall \alpha \in I,\ \sigma \in \{0,1\}\left(g(\alpha\sigma) = g(\alpha)0\sigma\right).$$

Тогда условия выполнимости отношения $h(z_1) \leqslant_t h(z_2)$ удовлетворяются для t-трассирования ξ':

$$\xi'(\alpha) = \begin{cases} g(\xi(\beta)),\ g(\beta) = \alpha \;\&\; \beta \in D(z_1); \\ g(\xi(\beta))1,\ \alpha = \beta 1 \;\&\; \beta \in D_0(h(z_1)); \\ g(\xi(\beta)),\ \alpha = \beta 0 \;\&\; \beta \in D_0(h(z_1)); \\ \xi'(\beta 0),\ \alpha = \beta 0\sigma \;\&\; \beta \in D_0(h(z_1)) \;\&\; \beta = \\ = g(\gamma) \;\&\; \sigma \in \{0,1\} \;\&\; \xi(\gamma) = \xi(\gamma\sigma); \\ g(\xi(\gamma\sigma)),\ \alpha = \beta 0\sigma \;\&\; \beta \in D_0(h(z_1)) \;\&\; \beta = \\ = g(\gamma) \;\&\; \sigma \in \{0,1\} \;\&\; \xi(\gamma) \subset \xi(\gamma\sigma). \end{cases}$$

Устанавливаемое последней схемой отображение вершин $h(z_1)$ в $h(z_2)$ является изотонным. Первая строка схемы отображения определяет ξ' для g-образов элементов $D(z_1)$. Две следующие строки схемы определяют ξ' для левого и правого потомков g-образов таких элементов. Последние две строки уточняют ξ' для случаев $\xi(\gamma) = \xi(\gamma\sigma)$ и $\xi(\gamma) \subset \xi(\gamma\sigma)$ в определении ξ при переходе от γ к $\gamma\sigma$. При этом отображение ξ' определяется по ξ так, что для него выполнены условия определения отношения $h(z_1) \leqslant_t h(z_2)$.

Проверим справедливость обратного утверждения. Пусть $h(z_1) \leqslant_t h(z_2)$ и ξ' — подходящее t-трассирование, для которого выполняется последнее отношение. Тогда $z_1 \leqslant_t z_2$ для t-трассирования ξ, определяемого соотношениями:

$$\xi(\alpha) = \begin{cases} \omega,\ g(\alpha) = \beta \;\&\; \xi'(\beta) = \gamma \;\&\; g(\omega) = \gamma; \\ \alpha,\ \text{иначе.} \end{cases}$$

Отображение является t-трассированием z_1 в z_2. *Теорема доказана.*

Отмеченная связь между t-трассированиями ПСП конфигураций пространств конфигураций и t-трассированиями АС конфигураций в соответствующем формализме $\Im \in SH$ означает, что подходящим образом определенные сравнения семантического вложения алгебраических структур произвольных конфигураций допускают корректную трансформацию в сравнения конфигураций, основанные на понятии трассирования их ПСП. Поэтому утверждения, связанные с трассированиями и вложениями конфигураций, допускают преобразование в утверждения, основанные на АС конфигураций. Это позволяет использовать ПСП конфигурации в качестве унифицированного формата моделирования сравнения алгебраических структур знаний в произвольных формализмах знаний. Данный формат удобен для представления иерархических семантических структур. В нем для представления семантических отношений между фрагментами знаний не требуются дополнительные вершины, применяемые в алгебраических структурах конфигураций. Формализм семантических иерархий занимает особое положение в многообразии формализмов знаний, поскольку в его рамках оказывается возможным применение конструкции ПСП конфигураций.

3. Морфизмы и процессы синтеза знаний в семантических иерархиях

Основу абстрактного моделирования операций и процессов в интеллектуальных системах образует система классов морфизмов над множествами абстрактных знаний, представляемых их алгебраическими структурами в подходящих формализмах. Указанные классы интегрируют функциональные сущности фундаментальных математических моделей из разных областей математики, применяемые для множеств ПСП конфигураций [4]. Морфизмы соответствуют этапам процессов синтеза знаний, составляющих реализации когнитивных целей (задач) для предметных областей, моделируемых интеллектуальными системами. Комбинации морфизмов рассматриваемых классов составляют диаграммы процессов в ИС. Последние осуществляют обработку знаний, реализующую отдельные когнитивные цели. Морфизмы могут соответствовать операциям над знаниями, исследуемым в лингвистике, когнитивной психологии, системной инженерии. Семантика таких морфизмов позволяет использовать понятия и конструкции из указанных областей. Это расширяет возможности моделирования интеллектуальных объектов и процессов [5].

Всякая диаграмма процесса может быть определена как двудольный ориентированный граф. Вершины графа соответствуют морфизмам рассматриваемых классов и областям определения и значения (базам) морфизмов. Ребра диаграммы соединяют вершины областей определения морфизмов с вершинами соответствующих классов морфизмов (классов морфизмов и областей их значения). Пути диаграммы представляют композиции морфизмов, реализующих процессы обработки знаний. Ребра диаграммы размечаются дополнительными условиями, определяющими способы управления конкретными реализациями таких композиций [6]. Вариантами управления комбинациями морфизмов являются: композиция, расщепление и интеграция.

Для концептуального моделирования и управления схемами процессов в интеллектуальных системах важны классификации морфизмов и взаимосвязи морфизмов разных типов. Примерами общих классов морфизмов являются гомоморфизмы и гомоморфные расширения. Морфизмами первого типа моделируются преобразования конфигураций, сохраняющие их структурные и семантические свойства без расширения дедуктивного содержания знаний. Гомоморфными расширениями абстрактных знаний моделируются преобразования, расширяющие содержание знаний включением в них дополнительных элементов, которые могут быть удалены из них с помощью подходящих гомоморфизмов.

Гомоморфизмы и гомоморфные расширения полезны для моделирования процессов переноса фрагментов знаний между компонентами многомерной архитектуры интеллектуальных систем [7, 8]. Семейства знаний внутри отдельных компонентов обладают согласованными свойствами, что позволяет обрабатывать такие знания совместно. Гомоморфизм (гомоморфное расширение) реализует трансформацию знаний, синтезированных в одном компоненте, в знания, обрабатываемые в другом компоненте. Для переносимого знания изменяются значения аспектов переносимых знаний (квантов). При этом формируется новое знание, сохраняющее свойства исходных знаний и адаптированное к значениям аспектов знаний компонента результата операции. Например, гомоморфизмом является преобразование, трансформирующее иерархическое представление математической формулы в ее символьную запись. Приводимые далее определения согласованы с инвариантами формализмов знаний.

Определение Вычислимое отображение $h : D_{M_1} \to D_{M_2}$ называется гомоморфизмом формализма $\Im = (M_\Im, D_\Im, \circ, \prec)$ в формализм $\Re = (M_\Re, D_\Re, \circ, \prec)$, если:

1) $\forall z_1, z_2 \in D_\Im (z_1 \subseteq z_2 \to h(z_1) \subseteq h(z_2))$;

2) $\forall z_1, z_2 \in D_\Im (h(z_1 \circ z_2) = h(z_1) \circ h(z_2))$.

Определение. Вычислимое отображение $h^- : D_\Im \to D_\Re$ называется гомоморфным расширением, если существует такой гомоморфизм $h^+ : D_\Re \to D_\Im$, что $\forall z \in D_\Im (h^+ h^-(z) = z)$.

Соответствия гомоморфизмов и связанных с ними гомоморфных расширений являются многозначными. Всякому гомоморфизму, применяемому при моделировании операций обработки знаний, соответствует семейство возможных гомоморфных расширений. Гомоморфные расширения применяются для пополнения содержания отдельных знаний дополнительным содержанием, которое либо детализирует имеющееся общее знание, либо добавляет в него новые элементы, расширяющие возможности работы со знаниями.

4. Базы морфизмов формализма семантических иерархий

Базы морфизмов составляют множества конфигураций, составляющих области определения и значений морфизмов. Многообразие баз определяется системами морфизмов, применяемых в ИС. Они адаптированы к структурам организации памяти при обработке знаний, изучаемым в лингвистике, когнитивной психологии, системной инженерии.

Примерами общих абстрактных баз для пространств конфигураций являются множества неструктурированных конфигураций M, ПСП конфигураций $\Sigma(M)$, АС конфигураций $\Upsilon(M)$, а также конфигураций, ПСП которых имеют глубину $k = 0, 1, ..., (M_k)$. Множество простых M_1 конфигураций соответствует базе полной декомпозиции содержания области знаний. Ее составляет многообразие знаний в этой области, представимыми тройками *RDF*. Перечислимые фрагменты этого многообразия составляют онтологии моделируемой области знаний. Специальные базы, моделирующие области определения и значения многоместных операций из фундаментальных математических моделей, составляют классы параллельных и последовательных серий $S(\Sigma)$

и $\bar{S}(\Sigma)$ [4]. Сериями представляются упорядоченные и неупорядоченные множества конфигураций, принадлежащие специальным классам конфигураций. Областями определения морфизмов конструирования серий из элементарных и простых знаний являются фрагменты M_1 (онтологии). Общезначимые базы морфизмов конфигураций составляют окрестности знаний определенных радиусов. Всякая конфигурация рассматриваемой базы синтезируется из элементов онтологий областей знаний с использованием морфизмов селекции [3]. Окрестность радиуса один заданного $z \in M_0$ в отношении $\rho \in R$ (обозначается как $O_\rho^P(z)$) — это параллельная серия, составленная для множества $\{z' | z \rho z' \,\&\, z' \in M_0 \,\&\, P(z, z', \rho)\}$. Она составлена элементарными знаниями, связанными с z отношением ρ. Здесь $P(z, z', \rho)$ – предикат условия выбора элементов онтологии, который должен выполняться для всех $z' \in M_0$, составляющих границу окрестности. Общая схема ПСП конфигурации $O_\rho^P(z)$ приведена в левой части рис. 2.

На рис. 2 символом ⋮ обозначено вспомогательное отношение, применяемое для составления параллельных серий [5].

Границу рассматриваемой окрестности составляют вершины, соответствующие наборам вида 10*1. В рассмотренном примере такие вершины размечены элементарными конфигурациями, с которыми z связано отношением ρ.

Окрестность элементарной конфигурации радиуса два в отношениях ρ_1 и ρ_2 для предикатов P_1 и P_2 получается из окрестности $O_{\rho_1}^{P_1}(z)$ заменой всякой вершины ее границы на окрестность радиуса один элементарной конфигурации этой вершины в отношении ρ_2 для предиката P_2. Данная окрестность обозначается как $O_{\rho_2}^{P_2}\left(O_{\rho_1}^{P_1}(z)\right)$ или $O_{\rho_2,\rho_1}^{P_2 P_1}(z)$. Пример такой окрестности приведен в правой части рис. 2. Аналогично определяются окрестности произвольных радиусов. Для обозначения таких окрестностей применяют выражения вида $O_{\rho_k \ldots \rho_1}^{P_k \ldots P_1}(z)$.

Комбинации баз морфизмов составляются с помощью операций суммы и произведения баз. Суммой баз B_1 и B_2 в $\Im \in SH$ называется множество $B_1 \oplus B_2 = \{z_1 \oplus z_2 | z_1 \in B_1 \,\&\, z_2 \in B_2\}$. Здесь $z_1 \oplus z_2$ — прямая сумма конфигураций [3]. Суммы позволяют конструировать базы, элементы которых применяются для моделирования областей определения многоместных отображений. Этим обеспечивается достаточность применения одной двуместной операции (морфизма прямой суммы баз) для моделирования операций обработки знаний. Все остальные морфизмы можно считать унарными. Области определения таких морфизмов моделируются прямыми суммами подходящих баз морфизмов. Морфизмы декомпозиции баз моделируют операции проекции конфигураций, принадлежащих суммам нескольких баз на фрагменты таких сумм. Они позволяют извлекать фрагменты значений морфизмов как исходные данные для последующих морфизмов в схемах последовательностей выполнения морфизмов.

Общий случай морфизмов извлечения фрагментов конфигураций реализуется комбинациями морфизмов проекции. Такие морфизмы связаны с выбором вершин ПСП исходных конфигураций. Они являются корнями поддеревьев, из которых составляется ПСП результирующей конфигурации.

5. Моделирование формализмов семантических иерархий в дескрипционных логиках

Среди моделей представления знаний важную роль играют дескрипционные логики и связанные с ними онтологии предметных областей.

Такие логики являются инструментом моделирования содержания предметных областей множествами знаний, представляемых тройками *RDF* и называемых далее онтологиями этих областей. Возможности дескрипционных логик определяются наборами операций конструирования (синтеза) фрагментов знаний

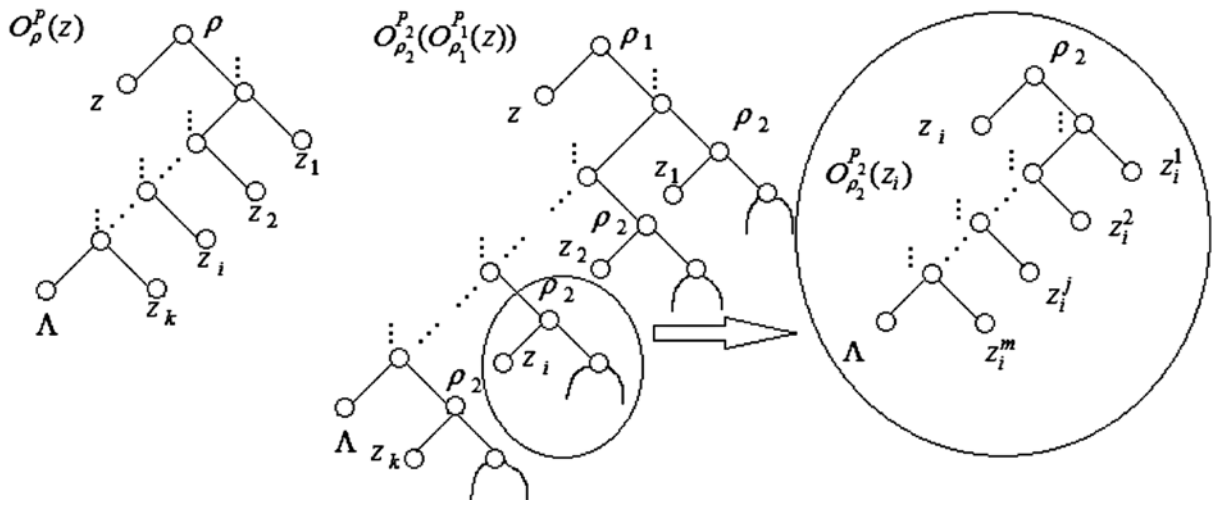

Рис. 2. Окрестности конфигураций заданной глубины

из элементов онтологий. Детальность и полнота онтологии определяется многообразием задач предметной области, решаемых на основе онтологии.

Построение онтологий реализуется как полная декомпозиция слабоструктурированного содержания предметных областей, извлекаемого из первичных ресурсов. Первоисточники представляют содержание областей знаний в форматах, принятых среди специалистов. Операции декомпозиции первичных ресурсов реализуют последовательное разложение на части вплоть до элементарных фрагментов (обычно имен понятий или математических выражений). Фрагменты связываются с помощью подходящих семантических отношений. Они соответствуют простым конфигурациям, включаемым в онтологии [3]. Декомпозиции содержания, реализуемые разными способами, позволяют конструировать разные онтологии одной области. Полнота онтологий связана с возможностью решения соответствующих классов задач, реализуемых операциями и процессами синтеза знаний сложной структуры, моделирующих схемы мышления специалистов.

Универсальность онтологий означает возможность представления фрагментов содержания абстрактных или прикладных областей с помощью семейств знаний унифицированного формата как результата анализа и декомпозиции этого содержания.

В частности, рассмотрим представление моделей формализмов семантических иерархий в формате онтологии. Пусть $\Im = (M, D_M, \circ, \prec)$ — такой формализм. Он соответствует пространству конфигураций, представляя применяемое в этом пространстве семейство отношений R и отображения разложения и связывания конфигураций $\varepsilon : M \to M \times M$ и $\psi : M \to R$. Определим абстрактную онтологию, соответствующую $\Im$. Пусть $D_M = \{a_1, ..., a_i, ...\}$ и $R = \{r_1, ..., r_i, ...\}$ — перечислимые множества имен фрагментов конфигураций и отношений между конфигурациями. Поставим им в соответствие перечислимые множества $T = \{k_1, ..., k_i, ...\}$ и $P = \{\gamma_1, ..., \gamma_i, ...\}$. Эти множества определяют семейства элементарных знаний формализма, соответствующих фрагментам конфигураций и отношениям. Пусть $h : D_M \to T$ и $g : R \to P$ — вычислимые биекции, для которых $\forall i \left(h(a_i) = k_i \ \& \ g(r_i) = \gamma_i\right)$.

Для моделирования отображений ε и ψ применим три дополнительных отношения φ_2, φ_3 и φ_4. Эти отношения составляются по следующему правилу.

Если $a_i \in M$, $\varepsilon(a_i) = (a_j, a_l)$ и $\psi(a_i) = r_m$, то в онтологию включаются конструкции: $k_i\varphi_3 k_j$, $k_i\varphi_4 k_l$, и $k_i\varphi_2\gamma_m$.

Приведенные тройки реализуют инъективное представление отображений ε и ψ простыми знаниями, составляющими онтологию.

Дополнительные предположения о множествах D_M и R в формализмах семантических иерархий моделируются специальными элементами онтологии. Отношение ρ_0 (вложения элементарных знаний из $\Im$) моделируется отношением φ_0, а отношение ρ_1 на R — отношением φ_1. Перечисленные конструкции представления абстрактных знаний составляют область аксиом (*A-box*) онтологии формализма $\Im$ [3]. Содержания этой области достаточно для восстановления ПСП произвольных конфигураций. Поэтому эта область полностью определяет $\Im$. Это позволяет проводить абстрактное исследование свойств данного формализма, используя множество ПСП конфигураций, конструируемых из элементов онтологии.

Практическое применение онтологий формализмов семантических иерархий связано с моделированием ограниченных множеств элементарных и простых знаний, представляемых иерархиями глубины 0 и 1. Конфигурации, ПСП которых имеют глубину больше, чем 1, допускают неявное представление в онтологии. Они конструируются из элементов онтологии с помощью индуктивных правил синтеза [5]. Правила моделируют процессы синтеза знаний в формате конфигураций, составляющих решения отдельных профессиональных задач. В случае моделирования процессов мышления правилами представляются конкретные схемы когнитивного синтеза, состоящие в связывании фрагментов онтологий в фрагменты ПСП конфигураций, а также схемы последующих преобразований таких фрагментов.

Преобразование конфигураций в их полные структурные представления реализует абстрактную схему полной декомпозиции. Другие схемы декомпозиции неструктурированных конфигураций моделируют возможные способы частичного извлечения из них элементарных и простых конфигураций. Практическая значимость многообразия схем декомпозиции слабо структурированных первоисточников объясняется существованием разных качественных аспектов отражения содержания областей знаний формализованными знаниями. Примерами аспектов являются: абстрактность, атомарность, уровень и время.

Моделирование реализаций целей на основе обработки знаний из онтологий реализуется процессами формирования (эволюции) ПСП конфигураций, синтезируемых из элементов онтологий. Начало синтеза реализуется операциями селекции элементов онтологии, интегрированных в структуры серий и окрестностей и связанных с начальными данными целей. Последующая обработка ПСП конфигураций выполняется как композиция морфизмов трансформации получаемых структур. Начальные данные процессов также извлекаются из онтологий. Например, это может быть специальная серия простых конфигураций, представляющих значения условий решаемой задачи. Они содержатся в онтологии первичных данных (или *E*-онтологии) ИС [7].

Для постановок задач применяют унифицированные форматы. Например, это могут быть шаблоны структур синтезируемых конфигураций, а также предикаты, истинные для конфигураций решений [4].

Когнитивные операции над знаниями в формате конфигураций моделируются классами абстрактных морфизмов. Такое моделирование является высокоуровневым в случае, если морфизмы соответствуют операциям общих моделей мышления [8]. Низкоуровневое моделирование реализации когнитивных целей основывается на использовании операций, соответствующих конструктам дескрипционных логик.

6. Моделирование конструктов дескрипционных логик в формализмах семантических иерархий

Полнота класса формализмов знаний означает возможность моделирования онтологий и семантических структур, синтезируемых из элементов онтологий с использованием инвариантов формализмов. Представления онтологий классов и отношений между классами далее реализуются как АС сложных знаний, основанные на конструкции неупорядоченных серий простых знаний. Моделирование операций в дескрипционных логиках осуществляется специальными морфизмами. Ими реализуются процессы когнитивного синтеза, связанного с семантикой решаемых задач. В качестве баз морфизмов используются подходящие классы АС знаний.

Приведем пример моделирования конструктов языка *ALC* в формате ПСП конфигураций формализмов из *SH*. Это позволяет использовать такие формализмы в качестве теоретического фундамента изучения процессов когнитивного синтеза, основанного на онтологиях. При этом моделирование конструкции и операции онтологий приводит к расширению представлений о специальных классах конфигураций формализмов из *SH*. Прикладное значение моделирования высокоуровневых процессов когнитивного синтеза связано с возможностью расширения семейства конструктов для онтологии области знаний, включая морфизмы трансформации сложных знаний.

Рассмотрим возможности представления элементов логики *ALC* и их комбинаций с помощью ПСП конфигураций. К ним относятся классы и отношения, а также формулы, определяющие соотношения между классами и отношениями. Формулы составляют раздел утверждений онтологии (*T-box*). Отдельные классы представляются окрестностями радиуса один для элементарного знания, соответствующего имени класса и отношения $\in$. Пример такой конфигурации приведен на рис. 3, *а*.

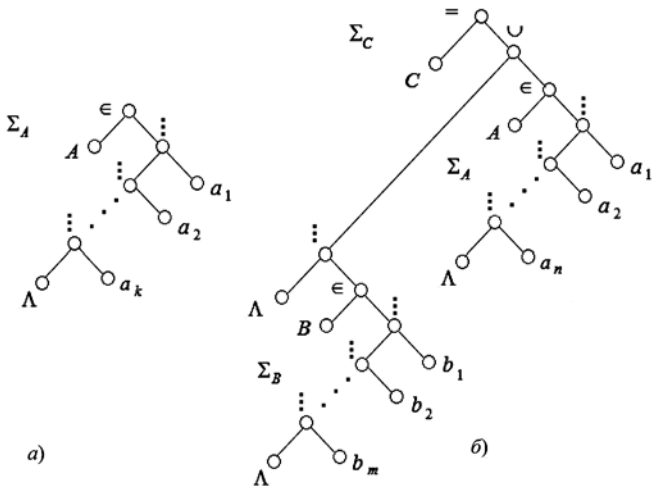

Рис. 3. Представление классов и формул конфигурациями

Выражение Σ_A на этом рисунке обозначает ПСП конфигурации, представляющий класс *A*. Элементарные конфигурации a_1, ..., a_k — это все разные первые элементы троек вида $z \in A$, содержащиеся в онтологии.

Представление формул, составленных с помощью операций объединения, пересечения и дополнения классов онтологий, возможно с помощью конфигураций специальной структуры. Пример соответствующей конфигурации приведен на рис. 3, *б*. Обозначаемая как Σ_C структура конфигурации соответствует конструкции дескрипционных логик, задаваемой выражением $C = A \cup B$, где *A*, *B* и *C* — классы онтологии. Символ $\cup$ обозначает отношение. Оно связывает ПСП конфигураций, представляющих объединяемые классы. Объединение нескольких классов формируется как неупорядоченная (параллельная) серия конфигураций, представляющих классы. Серия, представленная на рис. 3, *б*, соответствует двум классам. Приведенная конфигурация конструируется из представлений классов *A* и *B* как прямая сумма ПСП конфигураций $C \oplus_{=} ((\Lambda \oplus_{\in} \Sigma_B) \oplus_{\cup} \Sigma_A)$. Здесь $\oplus_r$ — операция прямой суммы пар ПСП конфигураций, связываемых отношением *r*. Полные структурные представления конфигураций для формул пересечения классов онтологии $(C = A \cap B)$ и дополнения класса $(C = Thing \backslash A)$ определяются аналогично рассмотренному случаю объединения классов. Здесь *Thing* — класс всех элементарных знаний онтологии.

Преобразование конфигураций для формул, составленных из представлений классов в формулах, в ПСП конфигураций, представляющих классы результатов выполнения теоретико-множественных операций, можно реализовать с помощью эндоморфизмов трассирования конфигураций [4]. Морфизм конструирования класса объединения двух заданных классов определяется как композиция морфизмов, последовательно применяемых к прямой сумме классов начальных данных.

Результат выполнения первого морфизма (вставки ПСП конфигурации, представляющей класс *B* в неупорядоченную серию, представляющую класс *A*), изображен на рис. 4, *а*. Построение данной ПСП возможно несколькими способами. Например, если использовать морфизм правой вставки (*Insert*) ПСП Σ_B в вершину α ПСП Σ_A, определяемую из условия $\alpha = 10 * \& \alpha \in O(z)$. Затем применяется морфизм прямой суммы элементарной конфигурации *C* и построенной ПСП, связываемых отношением $\in$.

Второй морфизм композиции реализует преобразование полученной ПСП в значение *p*-эндоморфизма, определяемого специальным отображением *p*-трассирования растяжения ξ полученной конфигурации $\Sigma(C)$ [2]. Приведенная схема отображения ξ является *p*-трассированием ПСП конфигурации (рис. 4, *а*) в ПСП конфигурации *z* (рис. 4, *б*). Выражение $|A|$ в записи схемы обозначает мощность класса *A*.

$$\xi(\alpha)=\begin{cases}\alpha,\alpha\in\{\lambda,0\}; & (1)\\ 1\alpha,\ \alpha=10^{k}\ \&\ \alpha1\in D(\Sigma_A)\ \&\ k\in\{0,\ 1,\ 2\ ...\}; & (2)\\ 1\alpha,\alpha=10^{k}1\ \&\ \alpha\in D(\Sigma_A)\ \&\ k\in\{0,1,2...\}; & (3)\\ 1\alpha110^{t},\alpha=10^{k}\ \&\ [z]_{1\alpha110^{t}1}\neq\Lambda\ \&\ t=\mu d([z]_{\xi(10^{k-1})0^{d}1}\notin A)\&\mid A\mid=k; & (4)\\ 1\alpha110^{t}1,\alpha=10^{k}1\ \&\ [z]_{1\alpha110^{t}1}\neq\Lambda\ \&\ t=\mu d([z]_{\xi(10^{k-1})0^{d}1}\notin A)\&\mid A\mid=k; & (5)\\ \xi(10^{k-1})0^{t},\alpha=10^{k+s}\ \&\ k=\mid A\mid\ \&\xi(10^{k+s-1})=10^{k}110^{p}\ \&\ t=\mu d(d>p\ \&\ [z]_{\xi(10^{k-1})0^{d}1}\notin A); & (6)\\ \xi(10^{k-1})0^{t}1,\alpha=10^{k+s}1\ \&\ k=\mid A\mid\ \&\ \xi(10^{k+s-1})=10^{k}110^{p}\ \&\ t=\mu d(d>p\ \&\ [z]_{\xi(10^{k-1})0^{d}1}\notin A); & (7)\\ \xi(\beta)\sigma,\text{ в остальных случаях, где }\alpha=\beta\sigma\ \&\ \sigma\in\{0,1\}. & (8)\end{cases}$$

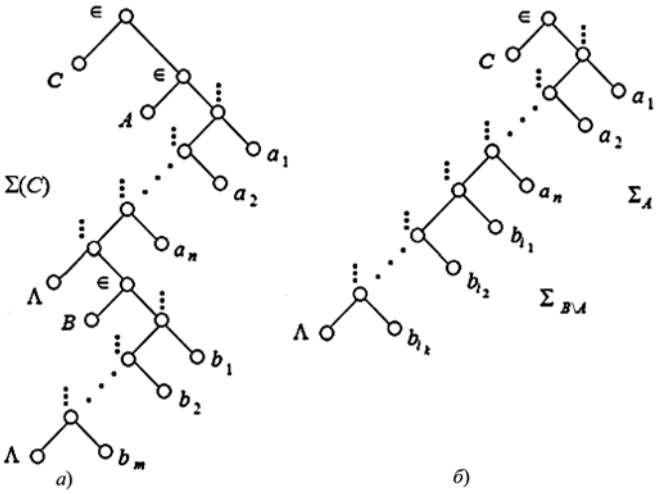

Рис. 4. Моделирование морфизма объединения классов онтологии

Значения отображения ξ реализуют трассирование в ПСП z вершин ПСП конфигурации значения морфизма. Она представляет класс, в который включаются все элементы серии, представляющей A (соотношения (2) и (3)). Соотношения (3) и (4) соответствуют переходу в $\Sigma(C)$ к части, составленной элементами $B\backslash A$. Построение остальной части класса C продолжается далее добавлением в C элементов B, которых нет в A (соотношения (6) и (7)).

Схемы моделирования конструктов пересечения и дополнения классов дескриптивной логики ALC в формализмах из SH определяются аналогично. Рассмотрим пример схемы для моделирования конструкта пересечения классов. Она начинается с интеграции исходных классов в конфигурацию, изображенную на рис. 4, *а*. Приводимая далее схема отображения p-трассирования определяет p-эндоморфизм конфигураций. На его основе реализуется преобразование последней конфигурации в ПСП конфигурации, представляющей пересечение классов A и B.

Значения отображения ξ определяют такой фрагмент ПСП (рис. 3, *а)*, который составляет часть серии, представляющей класс A, составленную только теми ее элементами, которые содержатся в серии, представляющей класс B.

$$\xi(\alpha)=\begin{cases}\alpha,\ \alpha\in\{\lambda,0\};\\ 110^{k},\ \alpha=1\ \&\ 110^{k}\in O(z)\ \&\ A\cap B=\varnothing;\\ 110^{k},\ \alpha=1\ \&\ k=\mu t([z]_{110^{k}1}\in B);\\ 110^{k}1,\ \alpha=11\ \&\ k=\mu t([z]_{110^{k}1}\in B);\\ 110^{k},\ \alpha=10^{d}\ \&\ d>0\ \&\ \xi(10^{d-1})=\\ =110^{q}\ \&\ k=\mu t(t>q\ \&\ [z]_{110^{t}1}\in B);\\ 110^{k}1,\ \alpha=10^{d}1\ \&\ d>0\ \&\ \xi(10^{d-1})=\\ =110^{q}\ \&\ k=\mu t(t>q\ \&\ [z]_{110^{t}1}\in B);\\ \xi(\beta)\sigma,\text{ в остальных случаях,}\\ \text{где }\alpha=\beta\sigma\ \&\ \sigma\in\{0,1\}.\end{cases}$$

Для моделирования конструкта дополнения заданного класса A до семейства всех элементарных знаний онтологии достаточно сформировать конфигурацию, интегрирующую класс всех элементарных знаний (соответствует классу всех элементарных знаний *Things*) и класс A. Структура такой конфигурации получается из структуры, приведенной на рис. 3, заменой имен множеств, приписанных вершинам, A на *Things* и B на A. Конфигурация, представляющая дополнение A, может быть извлечена из построенной конфигурации с помощью p-трассирования ξ, получаемого из схемы последнего трассирования, заменой условий $\in B$ (принадлежности элементов пересечения множеству B) на условие $\notin A$.

Для построения конфигураций, представляющих заданное отношение R, можно использовать конфигурации, структура которых аналогична окрестности радиуса два для имени отношения. Такая окрестность конструируется в два этапа. На первом этапе составляется окрестность отношения радиуса один, границу которой составляют элементарные знания из множества первых элементов пар, составляющих R. Затем для каждого элемента границы построенной окрестности конструируется окрестность радиуса один, составленная всеми вторыми элементами пар с заданным значением первого элемента. Результат второго шага составляет окрестность R радиуса два. Пример ПСП такой конфигурации приведен на рис. 5.

На рис. 5 изображена окрестность радиуса два для элементарного знания, представляющего отношение R. Окрестность организована как неупорядоченная

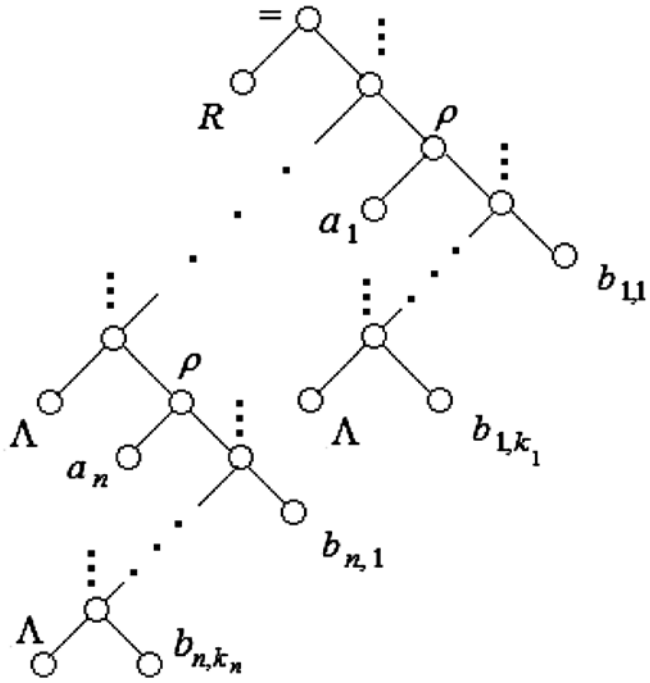

Рис. 5. Конфигурация представления отношения в онтологии

серия неупорядоченных серий, связанных с именем отношения отношением равенства =. Последние формируются для первых элементов пар из рассматриваемого отношения (может быть представлено выражением $\exists R.Things$). Для каждого такого элемента составляется окрестность радиуса один во вспомогательном отношении ρ, которая включается в серию конфигураций, связанных с R отношением =.

Построение ПСП последней конфигурации можно выполнить с использованием проекций отношений. В *ALC* проекции моделируются конструкциями кванторов существования и всеобщности. Указанные конструкты применяют для построения классов, составляемых по отношению и классу онтологии. Символьные представления указанных конструктов имеют вид $\exists R.C$ и $\forall R.C$. Здесь R и C — имена отношения и класса онтологии. При этом выражение $\exists R.C$ определяет класс элементарных знаний $\{x \mid \exists y \in C((x, y) \in R)\}$. Выражению $\forall R.C$ соответствует класс $\{x \mid \forall y \in M((x, y) \in R \to y \in C\}$.

Построение конфигураций, представляющих классы, задаваемых приведенными выражениями, реализуется специальными морфизмами. Они моделируются процессами, основанными на сценариях, использующих операции селекции, вставки и трассирования [4].

Рассмотрим пример синтеза класса, представляемого выражением $\exists R.C$. Для этого из конфигурации, представляющей онтологию, с помощью подходящего отношения трассирования извлекается неупорядоченная серия, составленная первыми элементами простых знаний, связанных отношением R с элементами класса C. Эта серия составляет правое поддерево корня ПСП синтезируемой конфигурации. Левое поддерево корня этой конфигурации представляет алгебраическую структуру формулы $\exists R.C$, задаваемую композицией $(\exists \circ R) \circ C$. Конфигурация, представляющая множество $\forall R.C$, конструируется аналогично.

7. Замыкания онтологий

Онтология, представляемая конечным множеством простых знаний $\delta \subseteq M_1$, является основой индуктивных процессов построения сложных конфигураций, синтезируемых из элементов δ. Для моделирования процессов синтеза применяются предопределенные семейства классов морфизмов. Замыкание онтологии δ образуют множества знаний сложной структуры − результаты применения последовательностей морфизмов.

Обозначим как $F = \{\Phi_1, ..., \Phi_k\}$ ($B = \{B_1, ..., B_r\}$) семейства классов морфизмов (баз морфизмов), структурированных отношением вложения классов (баз). Здесь B_1 — это класс онтологий, задаваемых как параллельные серии простых конфигураций. Пусть $L \subseteq \Sigma$ — это перечислимое множество.

Определение. Перечислимая последовательность ПСП конфигураций $z_1, ..., z_k, ...$ образует вывод в формализме семантических иерархий, основанный на L, если:

$$\forall i(z_i \in L \vee \exists j_1, ..., j_m < i \; \exists f \in \in F\left(z_i = f_{j+1}\left(\bigoplus_{t=1,...,m} z_{j_t}\right)\right).$$

Конфигурация $z \in \Sigma$ называется синтезируемой из онтологии δ, если существует вывод из конфигурации, представляющей δ, содержащий z. Замыканием онтологии δ в заданных классах морфизмов называется множество всех таких конфигураций, которые выводятся из δ с помощью морфизмов классов указанных классов. Для моделирования выводов удобно применять диаграммы последовательностей морфизмов [8]. Вершинам таких диаграмм соответствуют ПСП конфигураций, а ребрам — морфизмы, преобразующие конфигурацию вершины начала ребра в конфигурацию вершины, являющейся концом этого ребра. Особый случай составляют фрагменты диаграммы, моделирующие прямые суммы конфигураций, составляющие одну конфигурацию из нескольких с помощью прямой суммы конфигураций из нескольких баз морфизмов $\left(z = \bigoplus_{t=1,...,m} z_{j_t}\right)$.

Фрагмент диаграммы приведен на рис. 6.

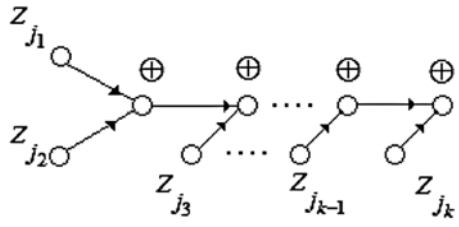

Рис. 6. Диаграммы прямой суммы элементов баз морфизмов

Заключение

Унификация моделей интеллектуальных систем, основанная на абстрактных математических инвариантах, является актуальной, но сложной для исполнения задачей. Формализмами знаний реализуются категории содержания и формы знаний, представляемые семантической и алгебраической структурами абстрактных знаний. Непосредственное применение инвариантов формализмов знаний для представления и управления знаниями в ИС связано с ограничениями, определяемыми природой математических знаний [9]. Адаптация указанных инвариантов к конкретным предметным областям предполагает необходимость использования специальных атрибутов, трансформирующих как сами абстрактные модели, так и представления об их свойствах. Универсальное решение проблемы адаптации состоит в конструировании специальной формальной системы. В ней аккумулируются теоретические представления об общих атрибутах и свойствах знаний и процессов их обработки. Такие представления исследуются и формируются в разных областях знаний и могут быть слабо формализованными.

Указанная формальная система может основываться на формализмах семантических иерархий, дополненной моделью многомерной структуры компонентов ИС [7]. Эта структура связана с классами знаний, имеющих близкие свойства. Такие классы реализуются подходящими формализмами знаний. Знания в этих классах обрабатываются совместно при реализации этапов потоков знаний в ИС [1].

Формализмы семантических иерархий позволяют моделировать конструкции знаний и процессы в онтологиях, реализуемые с использованием разных дескрипционных логик. Такое моделирование определяет выразительные возможности онтологий произвольных онтологий, реализуемые как результаты процессов синтеза конфигураций применяемого формализма.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-01-00289.

Список литературы

1. **Burgin M.** Theory of Knowledge: Structures and Processes. World Scientific, 2017. — 948 p.

2. **Костенко К. И.** Формализмы представления знаний и модели интеллектуальных систем. — Краснодар: Кубанский гос. ун-т, 2015. — 300 с.

3. **Baader F., Calvanese D., McGuinness D. L., Nardi D., Patelchneider P. F.** The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003. — 505 c.

4. **Костенко К. И.** Операции когнитивного синтеза формализованных знаний // Программная инженерия. — 2018. — Том 9, № 4. — С. 174—184.

5. **Костенко К. И.** Моделирование оператора вывода для иерархических формализмов знаний // Программная инженерия. — 2016. — Том 7, № 9. — С. 424—431.

6. **Ковалев С. П.** Теоретико-категорный подход к проектированию программных систем // Фундаментальная и прикладная математика. — 2014. — Том 19, № 3. — С. 111—170.

7. **Kostenko K., Lebedeva A., Levitskii B.** The intelligent office engineering by rational and reactive mind invariants // Proceedings of the 6th International Conference Actual Problems of System and Software Engineering, Moscow, Russia, 12—14 November, 2019. — P. 106—116.

8. **Larue O., Poirier P., Nkambou R.** A tree-level cognitive architecture for the simulating of human behavior // Canadian AI, LNAI 7310. — 2012. — P. 337—342.

9. **Янов Ю. И.** Математика и метаматематика // Математические вопросы кибернетики. Вып. 16: Сб. статей / Под ред. Н. А. Карповой. — М.: Физматлит, 2007. — С. 129—154.

Ontologies' Closures Modeling by Formalisms of Semantic Hierarchies

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

Corresponding author:

Kostenko Konstantin I., Associate Professor, Kuban State University, Krasnodar, 350040, Russian Federation
E-mail: kostenko@kubsu.ru

Received on June 26, 2020
Accepted on September 22, 2020

Schemes are studied for modeling the complex knowledge structures as synthesized from knowledge areas ontologies elements. These structures' applications relate to knowledge life cycles and knowledge flows stages within intelligent systems. The format of semantic hierarchies is proposed as unified and universal for the synthesis of these structures. This allows replacing the general case of knowledge algebraic structures by special case of semantic hierarchies. Constructing the synthesized knowledge structures is performed by special operations knowledge algebraic structures in any knowledge representation formalisms. These operations simulate fundamental mathematical systems' functional aspects. They are adapted to the knowledge formalisms' attributes. Attributes are explored within different knowledge areas. They associated generally with thinking operations and mind memory structure. Datasets (operations' bases) of synthesis processes operations are constructed as special classes of knowledge with a uniform structure. Ontologies for considered knowledge areas are used as source of such bases constructing. Ontologies closures are defined as sets of knowledge that may be constructed off ontologies elements by synthesis operations.

They determine the ontologies' expressive capabilities. The constructs and operations over ontologies, that proposed at descriptive logics, can be modelled by knowledge' complete structural representations, adopted for semantic hierarchies formalisms. Such formalisms are convenient for simulating knowledge presentation and processing. They form foundation for constructing intelligent systems' abstract and applied models.. The possibility is proved for transferring the knowledge properties and knowledge processing schemes at semantic hierarchies to the general case of knowledge algebraic structures. The schemes for modeling the ontological constructions as semantic hierarchies are given. This proves possibility of applying such formalisms as the basis for modeling synthesis processes in ontologies. Such schemes allow constructing the ontologies' closures as generated by knowledge-processing operations' sequences. Last fact means possibility for formalisms of semantic hierarchies to be uniform foundation of modelling the knowledge flows and knowledge transforming processes by intelligent systems.

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

Acknowledgements:

The reported study was funded by RFBR. Grant project number № 20-01-00289

For citation:

Kostenko K. I. Ontologies' Closures Modeling by Formalisms of Semantic Hierarchies, *Programmnaya Ingeneria,* 2020, vol. 11, no. 6, pp. 349—360

DOI: 10.17587/prin.11.349-360

References

1. **Burgin M.** *Theory of Knowledge: Structures and Processes.* World Scientific, 2017, 948 p.

2. **Kostenko K. I.** *Knowledge representation formalisms and intelligent systems models,* Krasnodar, Kuban state university, 2015, 300 p. (in Russian).

3. **Baader F., Calvanese D., McGuinness D. L., Nardi D., Patelchneider P. F.** *The Description Logic Handbook: Theory, Implementation, and Applications,* Cambridge University Press, 2003, 505 p.

4. **Kostenko K. I.** Operations of Formalized Knowledge Cognitive Synthesis*, Programmnaya Ingeneria,* 2018, vol. 9, no. 4, pp. 174—184 (in Russian).

5. **Kostenko K. I.** Simulation of Inference Operator for Hierarchical Knowledge Representation Formalisms, *Programmnaya Ingeneria,* 2016, vol. 7, no. 9, pp. 424—431 (in Russian).

6. **Kovalev S. P.** Category-theoretic approach to software systems design, *Fundamentalnaya i prikladnaya matematika*, 2014, vol. 19, no. 3, pp. 111—170 (in Russian).

7. **Kostenko K., Lebedeva A., Levitskii B.** The intelligent office engineering by rational and reactive mind invariants, *Proceedings of the 6th International Conference Actual Problems of System and Software Engineering,* Moscow, Russia, 12—14 November, 2019, pp. 106—116.

8. **Larue O., Poirier P., Nkambou R.** A tree-level cognitive architecture for the simulating of human behavior*, Canadian AI, LNAI 7310,* 2012, pp. 337—342.

9. **Janov J. I.** Matematika i Metamatematika, *Matematicheskie voprocy kibernetiki*. Vol. 16. / Eds. N. A. Karpova, Moscow, Fizmatlit, 2007, pp. 129—154 (in Russian).

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

Сдано в набор 08.10.2020 г. Подписано в печать 25.11.2020 г. Формат 60×88 1/8. Заказ PI620
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**

Рисунки к статье **Р. А. Кареловой, Е. Е. Игнатова**

«ОСОБЕННОСТИ РЕАЛИЗАЦИИ НЕЙРОННОЙ СЕТИ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ РАСПОЗНАВАНИЯ ДЕФЕКТОВ СТАЛИ»

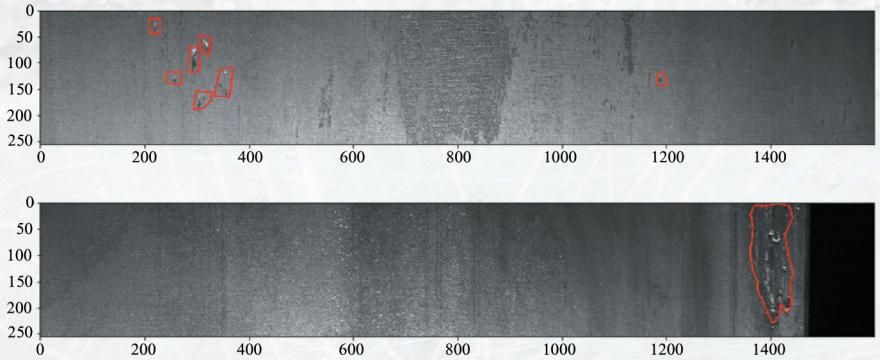

Рис. 5. Листы стали с дефектом класса 1

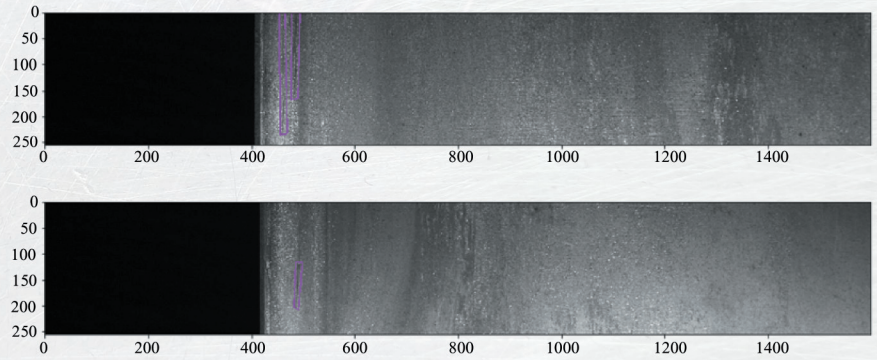

Рис. 6. Листы стали с дефектом класса 2

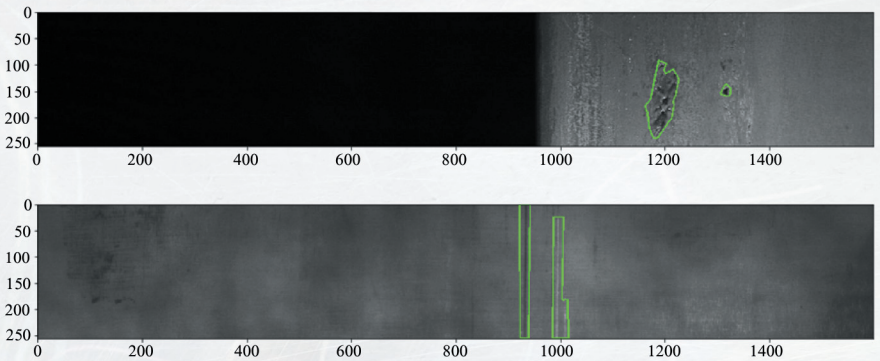

Рис. 7. Листы стали с дефектом класса 3

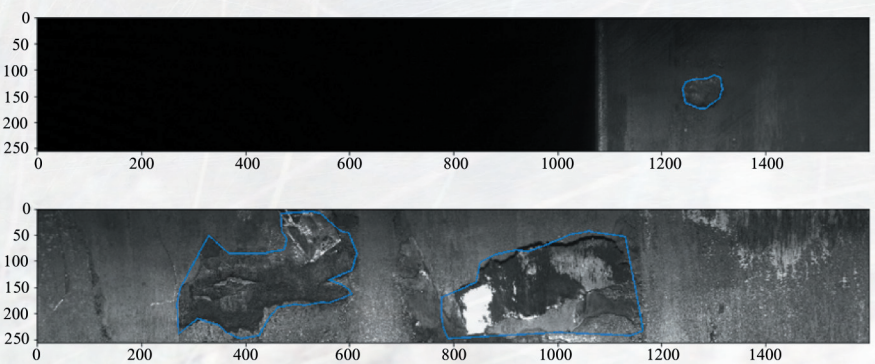

Рис. 8. Листы стали с дефектом класса 4

Рисунки к статье **Р. А. Кареловой, Е. Е. Игнатова**
«ОСОБЕННОСТИ РЕАЛИЗАЦИИ НЕЙРОННОЙ СЕТИ
ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ РАСПОЗНАВАНИЯ ДЕФЕКТОВ СТАЛИ»

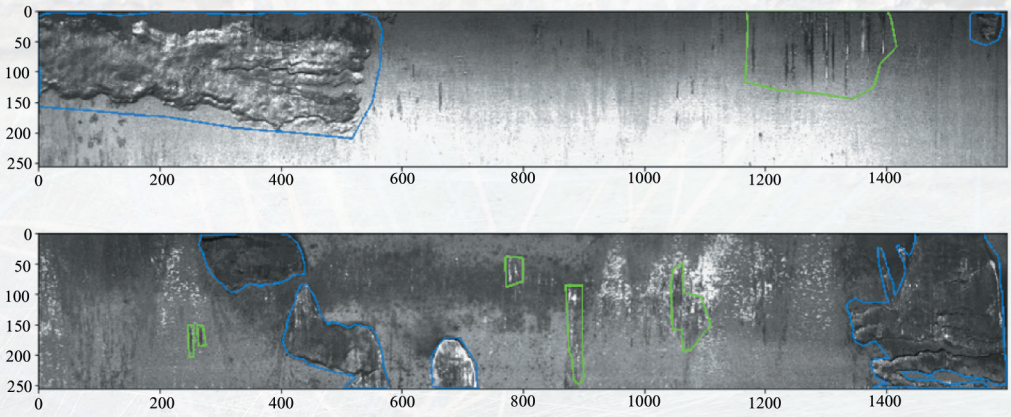

Рис. 9. Листы стали с двумя дефектами одновременно

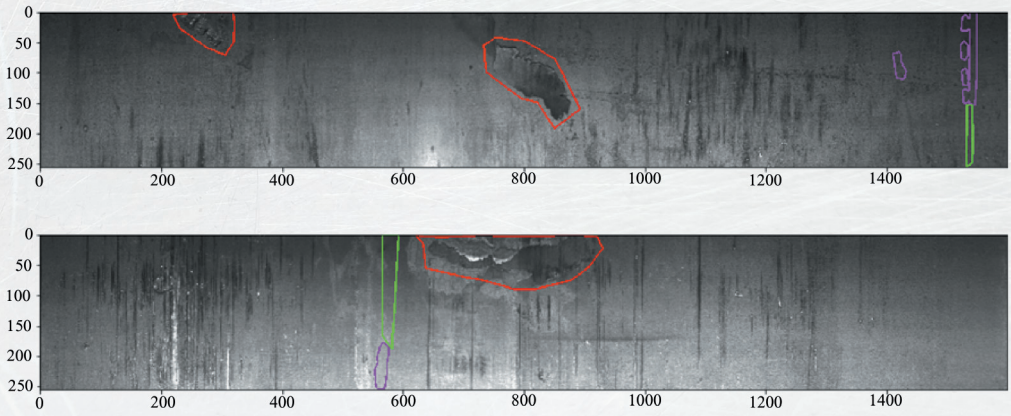

Рис. 10. Листы стали с тремя дефектами одновременно

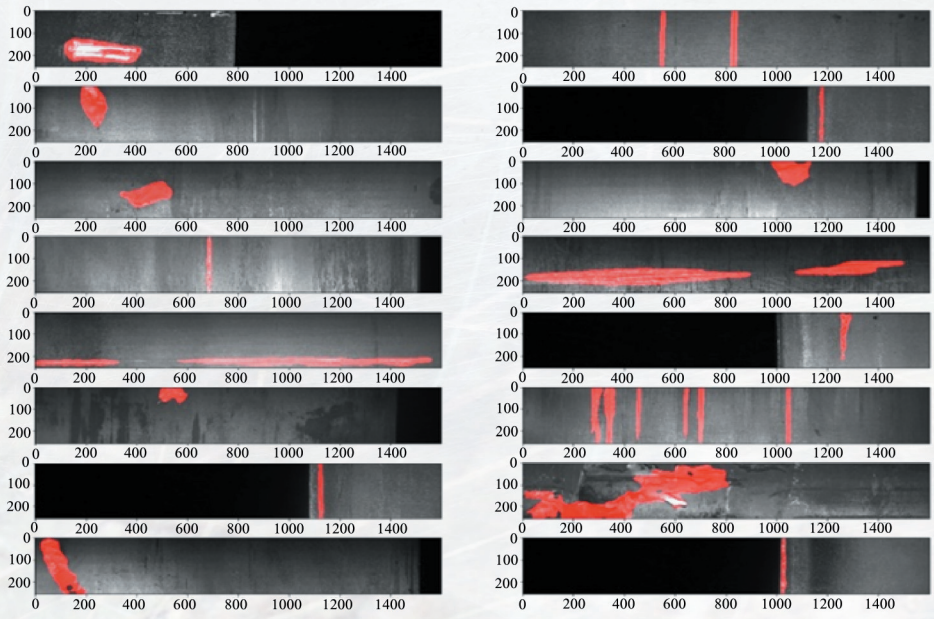

Рис. 11. Полученные маски дефектов

ISSN 2220-3397. Программная инженерия. 2020. Том 11. № 6. 309 – 360. Подписной индекс по Объединенному каталогу "Пресса России" – 22765