УДК 004+005+519.7

О необходимости онтологии для классификации и навигации по универсуму компьютерных языков

Шилов Н.В. (Автономная некоммерческая организация высшего образования "Университет Иннополис")

В 2008-2013 гг. в Институте систем информатики им. А.П. Ершова развивался сначала проект по исследованию и классификации парадигм языков программирования, а затем — по разработке классификации компьютерных языков, виртуальный мир которых включает тысячи языков программирования, спецификаций, моделирования и множество других языков, которые можно отнести к (пользуясь терминологией, сложившейся в биологии) разным царствам, классам, семействам и так далее. В настоящей статье представлена сумма основных теоретических результатов исследований 2008-2013 гг. по классификации компьютерных языков (которые, по нашему мнению, остаются актуальными и на сегодняшний день) и обсуждаются новые подходы к задаче разработки портала знаний о компьютерных языках (вернее — его наполнении) с использованием научной периодики и современных лингвистических технологий (вместо размеченных источников в Интернете). Актуальность продолжения исследований по классификации компьютерных языков обусловлена познавательным значением такой классификации, а разработки портала знаний о классификации компьютерных языков — потребностями практики объективного выбора компьютерных языков по спецификации программных проектов.

Ключевые слова: языки программирования, языки разметки, языки запросов, компьютерные языки в целом, классификация, атрибуты, парадигма, портал знаний, навигация по классификации, логика Белнапа, дискипционная логика, темпоральная логика, логика со временем, верификация моделей

1. Введение

Классификация — необходимый шаг/результат при систематическом исследовании какой-либо области природы, техники, искусства, классификация состоит в группировке объектов исследования или наблюдения в соответствии с их общими признаками. При рассмотрении компьютерных языков можно выделить множество характеристик таких как парадигма (например, императивная, декларативная, объектно-ориентированная), дисциплина обработки (например, последовательная, недетерминированная, распределённая),

и виртуальная машина (например, машина Тьюринга, машин продукций, машина логического вывода) и так далее.

Поэтому в 2008-2010 и 2012-2013 гг. в Институте систем информатики им. А.П. Ершова СО РАН развивался сначала проект по исследованию и классификации парадигм языков программирования, а затем — по разработке концепции и автоматизации классификации компьютерных языков [1, 9, 15–17] как с целью изучения мира (универсума) компьютерных языков, так и с целью автоматизации классификации и повышения интеллектуального уровня выбора языков для научных и индустриальных проектов по описанию проекта на естественном языке посредством навигации по классификации при помощи формализованных запросов.

Исследования по парадиматизации (то есть по исследованию и классификации парадигм) языков программирования получили дальнейшее развитие в 2010-2018 гг. в работах [5, 6], причём, особое внимание уделено функциональной и параллельной парадигмам программирования, разработке "лексикона программирования" [8], основанного на выделении парадигм программирования.

По-другому сложилась ситуация с исследованиями задачи автоматизации классификации и навигации по классификации компьютерных языков. Для её решения в 2008-2013 гг. развивался подход, основанный на выделении "определяемых" (динамических) парадигм компьютерных языков по совокупности общих признаков (атрибутам), присущих языкам, (не ограничиваясь "парадигмой" в узком смысле историческом смысле), создании "нетаксономической" (не древовидной) онтологии компьютерных языков. Особое значение придавалось компьютерной поддержке разрабатываемой классификации в форме (прототипа) портала знаний по классификации компьютерных языков [15, 16]. Этот портал предполагалось наполнить сведениями о компьютерных языках из открытых размеченных и хорошо структурированных интернет-ресурсов (например, Wikipedia), организовав навигацию и доступа к собранной информации по логическим запросам, сформулированным на комбинированном языке дискрипционной логики [13, 18], темпоральной логики [10] со временем, построенном над четырехзначной логикой Бэлнапа [19], а в качестве алгоритма решения запросов предлагалось использовать метод верификации конечных моделей (model checking) [10]. Прототип такого портала был создан в 2012 г., но поддерживал только "статические" запросы (и не поддерживал работу со временем и темпоральными конструкциями), выраженными на языке дискрипционной логики над логикой Бэлнапа

[9, 15], но затем был оставлен и в настоящее время не существует.

Начало второго десятилетия XXI-го века совпало с началом “эпохи” целенаправленной массового создания проблемно-ориентированных языков программирования (Domain Specific Languages) [26, 33]. Появлением технологий быстрой разработки таких языков и сред программирования [7, 24, 32, 49], привело к бурному росту числа компьютерных языков, многие из которых не представлены в ни в Wikipedia, ни в других хорошо структурированных и размеченных интернет-ресурсах. (Заметим, что ранее существовала педагогическая методика “создай свой язык программирования”, используемая до сих пор в обучения программированию; активным пропагандистом такого подхода был математик, программист, психолог и педагог, один из основоположников теории искусственного интеллекта, создатель языка Logo — Сеймур Пейперт.) Необходимость осмыслить эту новую реальность в мире компьютерных языков привела участников исследований к решению отложить на время проект реализации портала классификации компьютерных языков.

Ситуация изменилась к настоящему времени — к концу второго десятилетия XXI-го века. Во-первых, появились работы по извлечению “утверждений” или “мнений” (хотя обычно ошибочно говорят об извлечении “знаний”) из плохо структурированных (и только частично размеченных средствами HTML) и — в перспективе — неструктурированных (и неразмеченных) источниках в Интернете [2, 14]. В частности, в цитируемых работах предлагается подход к автоматизации сбора информации о предметной области, использующие методы метапоиска и извлечения информации, базирующиеся на онтологиях и тезаурусах моделируемой области знаний. Во-вторых, получил развитие онтологический подход для формализации знаний в разных научных предметных областях, причём, подход онтологический оказался настолько перспективным, что появилась возможность реализации паттернов содержания при разработке онтологий разных научных областей [3, 50]. С помощью онтологии можно не только представить все необходимые понятия моделируемой области, но и обеспечить их единообразное и согласованное описание. “Заметим, что под научной предметной областью (НПО) здесь понимается предметная область, охватывающая некоторую научную дисциплину или область научных знаний во всех её аспектах, включая характерные для неё объекты и предметы исследования, применяемые в ней методы исследования, выполняемую в ней научную деятельность и полученные в рамках её исследований научные результаты.”[1]

[1]Цитируется по [3].

И, наконец, задача создания и реализации онтологии компьютерных языков стала актуальной в рамках проекта РФФИ №17-07-01600 “Методы извлечения формальных спецификаций программных систем из текстов технических заданий и их верификация” [4, 27]. Этот “проект посвящён проблеме обеспечения качества программных систем с помощью формальных методов. В рамках проекта планируется разработать комплексный подход к извлечению формальных моделей и свойств распределенных программных систем из текстов технической документации с последующей их верификацией.”[2] Так как “правильный” выбор компьютерных языков для реализации программных проектов — одно из (возможно, производных) нефункциональных требований, предъявляемых техническим заданием (одной из разновидностей технической документации) к программному проекту, то автоматизированная система классификации компьютерных языков может стать инструментом верификации формализованных требований к компьютерным языкам, использованным при реализации проекта.

Настоящая статья имеет следующую структуру. В следующей части 2 мы напомним и обсудим основные идеи и понятия, выработанные ранее для создания классификации компьютерных языков. А в части 3 мы обсудим технические решения для реализации принятого подхода, и актуальные задачи, стоящие на пути компьютерной реализации разрабатываемой онтологии. В конце статьи в кратком заключении мы намечаем некоторые ближайшие задачи по развитию проекта.

Работа выполнена при поддержке гранта РФФИ №17-07-01600 “Методы извлечения формальных спецификаций программных систем из текстов технических заданий и их верификация”.

2. Концепция классификации компьютерных языков

Начнём с того, что несколько уточним основные определения, данное в работе [9]. Под компьютерным языком мы понимаем любой искусственный язык, разработанный для форматированного представления, автоматического преобразования, извлечения и управления данными, информацией и процессами. Под классификацией какого-либо универсума (компьютерных языков в частности) мы понимаем подход и инструментарий для выделения сущностей этого универсума (объектов, классов и ролей по отношению друг к другу), а также для навигации между этими сущностями по запросам. Среди всех компьютер-

[2] Цитируется по [4].

ных языков прежде всего выделяются языки программирования — компьютерные языки, предназначенные для организации преобразования данных.

Существует несколько известных (и не очень) де-факто попыток создания классификаций компьютерных языков. Это, прежде всего, популярный плакат [36], охватывающий историю возникновения и взаимного влияния 50 языков программирования, появившихся в период с 1954 г. по настоящее время. На этом плакате классификация проведена по времени возникновения, исторической преемственности между версиями (без определения понятия “версия”) и взаимному влиянию (без конкретизации в чём состояло “влияние”), а навигация сводится к следованию линиям развития языков во времени (вперед или назад) и стрелкам влияния.

На сайте [34] представлен глоссарий 19 основных категорий, относящихся к компьютерным языкам (например, “императивный язык”, “язык четвёртого поколения”), определены понятия “диалект”, “версия” и “реализация”, приведена хронология языков, оказавших наибольшее влияние на развитие других компьютерных языков, а также алфавитно-организованное описание 2500 языков (включая версии,диалекты и реализации) в терминах принятых категорий. Однако, [34] — это, скорее, интернет-справочник по истории и классификации компьютерных языков, а не классификация компьютерных языков и не портал знаний о классификации и компьютерных языках в силу следующих причин:

- единственное средство запросов и навигации между языками — это поиск языка по алфавиту, и переход по внутренним гиперссылкам на связанные языки;
- отсутствуют внешние гиперссылки на источники информации о категоризации языков и об языках (хотя есть ссылки на публикации по некоторым языкам).

В подтверждение сформулированного выше мнения, приведём следующую цитату с данного интернет-ресурса:

> It’s been suggested that the languages in this list should be arranged into categories, but to do so would be extremely difficult. For every classification scheme there will be a large proportion of languages that do not fit. The languages are therefore listed alphabetically, and in fact we think that this is the most useful organization. You’ll find that certain categories have been referred to in the list, but we must emphasize that most languages are not purely one or the other, and we are really categorizing language features.

Классификацию компьютерных языков можно пытаться строить в виде древовидной

таксономии аналогично естественными науками, например “по Линнею”: царство — тип (у растений — отдел) — класс — отряд (у растений порядок) — семейство — род — вид. Так, например, устроена энциклопедия языков программирования HOPL: an interactive Roster of Programming Languages [39], включающая сведения о более чем 8500 языках (однако, следует заметить, что эта таксономия выделяет очень странные классы компьютерных языков). Эта энциклопедия поддерживает навигацию между разнородной информацией, представленной в энциклопедии, по гиперссылкам с разными ролями (как-то, год создания, авторство, взаимное влияние и тому подобное) и поддерживает гиперссылки на внешние источники. Однако, этот уникальный ресурс не имеет языка запросов (кроме как поиск по ключевым словам как-то по названию, автору и так далее) и, кроме того, сама идея таксономии как основы классификации компьютерных языков представляется довольно-таки сомнительной (см. следующие два абзаца).

Во-первых, есть существенная разница между предметной областью в естественных науках и миром компьютерных языков: если в биологии, химии, физике “универсум” сравнительно статичен, то универсум компьютерных языков изменяется очень быстро. Так, например, только за последние два десятилетия мы наблюдали быстрый рост, как существовавших классов компьютерных языков, так и образование новых классов (например, языков представления знаний, языков кластерного/многоядерного программирования, проблемно-ориентированных языков программирования). Некоторые из этих новых компьютерных языков имеют свой специфический синтаксис (например графический), свою семантику (то есть модель обработки информации или виртуальную машину), свою прагматику (то есть сферу применения и распространения). Некоторые из давно существующих классов компьютерных языков сравнительно мало населены (например, языки проектирования СБИС), некоторые — густонаселенны (например, языки спецификаций), а некоторые классы пережили или переживают период роста. В тоже время разработка новых компьютерных языков (а следовательно и образование новых классов компьютерных языков) будет продолжаться по мере компьютеризации новых отраслей человеческой деятельности.

Во-вторых, часто компьютерный язык трудно отнести к одному определённому классу. Например, функциональные языки программирования ML и Рефал появились ещё в 1960-ые годы как языки спецификаций вывода (логического и продукционного), но ещё в 1970-ые годы они стали языками программирования для задач искусственного интеллекта;

в то же время эти языки (в силу их декларативности) по-прежнему могут служить языками спецификаций для вычислительных программ (как, например, в проекте VeriFun [48] функциональный язык используется в качестве спецификации императивной программы, а система доказывает функциональную эквивалентность спецификации и программы). Кроме того, некоторые языки прямо по замыслу их разработчиков не могут быть отнесены к одному конкретному классу: например, активные пропагандисты языка Ruby позиционируют его как смесь (коктейль) из нескольких языков Perl, Smalltalk, Eiffel, Ada и Lisp с хорошим балансом между функциональными и императивными возможностями [44]. (Здесь, однако, нам кажется, происходить определённая путаница между классом языка программирования и стилем программирования [11]: программировать в функциональном стиле можно даже на Algol-60, а в императивном стиле — на языках Рефал и Lisp.)

Исходя из изложенной выше критики известных подходов, при разработке концепции и автоматизации классификации компьютерных языков в ИСИ СО РАН в 2012-2013 гг. [9, 15, 16] предлагалось основывать классификацию универсума компьютерных языков на гибком понимании *компьютерных парадигм*.

Вообще в современной методологии науки парадигмы — это разные подходы к постановке и решению задачи или проблемы. Современным пониманием этого термина мы обязаны широко известной диссертации Томаса Куна [35], впервые опубликованной в 1970 г. Роберт Флойд стал первым, кто употребил это понятие ввёл понятие в информатике и заговорил о *парадигмах программирования* в своей тьюринговской лекции *Paradigms of Programming* в 1978 г. [25], но, к сожалению, он не дал явного определения этого понятия. В 2009 г. Питер ванн Рой опубликовал в Интернете таксономию *The principal programming paradigms* [42], в которой выделил 27 различных парадигм, и предпринял попытку обосновать её в статье [43]. Само понятие парадигмы программирования определено в [43] следующим образом:

> A programming paradigm is an approach to programming a computer based on a mathematical theory or a coherent set of principles. Each paradigm supports a set of concepts that makes it the best for a certain kind of problem.

Это определение послужило основой для следующего определения (см. [9, 15, 16]):

- Компьютерные парадигмы — это альтернативные подходы (образцы) к формализации постановок, представления, инструментов и средств решения задач и проблем информатики.

- Компьютерные парадигмы обычно поддержаны строгими математическими теориями/моделями и аккумулированы в соответствующих компьютерных языках.
- Компьютерные парадигмы соответствуют классам компьютерных языков и наоборот:
  - каждый естественный класс компьютерных языков представляет компьютерную парадигму, которую языки этого класса реализуют;
  - каждая компьютерная парадигма естественно характеризуется классом компьютерных языков, реализующих эту парадигму.
- Онтологическое (для универсума компьютерных языков) значение компьютерной парадигмы характеризуется классом задач, для решения которой она подходит более всего или для решения которого она была разработана.
- Образовательное значение компьютерных парадигм состоит в том, что парадигмы учат мыслить по-разному, видеть под разным углом зрения, как конкретные задачи информатики, так и мировоззренческие проблемы информатики.

Синтаксиса, *семантики* и *прагматики* — это категории, которые используются для описания как естественных, так и искусственных языков (компьютерных языков в том числе):

- синтаксис — это "правописание" языка;
- семантика — это правила придания "смысла" правильно написанным фразам языка;
- прагматика — это практика использования корректных фраз языка.

Разумно предположить, что все эти три категории должны использоваться для выделения классов компьютерных языков.

Использование синтаксиса для классификации компьютерных языков должно отражать особенности формального синтаксиса и человеко-ориентированный аспекты: для разработки синтаксического анализатора важно знать, имеет ли язык контекстно-свободный синтаксис, является ли контекстно-свободным и тому подобное; следовательно, эти и другие формально-синтаксические свойства компьютерного языка должны быть атрибутами при классификации компьютерных языков. Однако неформальные (или прагматические) характеристики синтаксиса компьютерного языка также должны учитываться при классификации компьютерных языков: примерами могут служить неформализуемые атрибуты *гибкость* или *краткость*, а также формализуемые атрибут такие как понятие *стиль*, которое можно определить посредством библиотеки размеченных примеров "правильно-

го” и “неправильного”стиля (как, например, функциональные программы написанные в императивном стиле).

Несколько по-другому обстоит дело с использованием неформальной и формальной семантики компьютерных языков. Неформальная семантика мало пригодна для использования для классификации в силу отсутствия общеупотребимых и “понятных” характеристик семантики, а использование формальной семантики для классификации компьютерных языков наталкивается на следующие препятствия [9]:

- слабое знание формальной семантики среди пользователей компьютерных языков (разработчиков, руководителей и менеджеров проектов);
- распространённостью предубеждения, что формальная семантика малополезна и неэффективна на практике;
- избытком индивидуальных нотаций для формальной семантики с разным уровнем абстракции и “вхождения”.

Может быть, все эти препятствия можно преодолеть путём создания унифицированного формализма для описания семантик, развития компьютерного инстументария для поддержки этого формализма, продвижение его преподавания в университетах и популяризация среди разработчиков? Но по нашему мнению [9], этот путь требуют “всемирной” централизации принятия решений о стандартизации формальных семантик, больших капиталовложений и, кроме того, этот путь наверняка будет препятствовать появлению и внедрению новых вариантов формальных семантик.

Например, начиная с 1990-ых годов в Microsoft Research под руководством Юрия Гуревича активно разрабатывался такой унифицированный формализм для описания формальной семантики (прежде всего) языков программирования, сначала под названием эволюционирующие алгебры (evolving algebras), а позже — машины абсрактных состояний (abstract state machines) [23, 28, 29], в рамках которого была описана формальная семантика нескольких языков (например, языка программирования Java [45]). Однако, ни авторитет известного ученого, ни исследовательского подразделения компании Microsoft, ни примеры успешного использования не сделали данный популярный формализм доминирующим среди других формализмов описания семантики. Более того, за время, прошедшее после возникновения машин абстрактных состояний, в информатике появились новые идеи и подходы, которые могут быть использованы для описания формальной семантики, как, например, в работах [20, 21] развивается подход к описанию формальной

семантики, основанный на концептуальных системах переходов, предполагающий построение онтологии синтаксических и семантических понятий формализуемого компьютерного языка.

На наш взгляд [9] эти препятствия могут быть преодолены на пути *многомерной стратификации* семантики наиболее типических компьютерных языков:

- в качестве измерений могут выступать, например, *образовательная* семантика, *реализационная*, *трансформационная* и так далее,
- а в каждом измерении можно выделить несколько *уровней* или *слоёв*, которые условно можно назвать *элементарным*, *основным* (или *базовым*) и *экспертным* (или *мастерским*).

Реализационная семантика компьютерного языка может делить язык на 2-3 или более слоя. Обычно можно условно выделить *ядро*, несколько *промежуточных* слоёв, и *полный* язык. Ядерный слой имеет эффективную реализационную семантику для класса *платформ* и достаточные средства для реализации *методом раскрутки*, *интерпретации* или *трансформаций* конструкций промежуточных слоёв. Полный язык может иметь только частичную трансформационную семантику в более низкие (промежуточные или ядерный) слои, а часть конструкций полного языка может иметь свою индивидуальную реализационную семантику для каждой конкретной платформы. В качестве примера слоёв языка можно сослаться на “уровни” языка C♯ [12].

Прагматика компьютерных языков (в отличие от синтаксиса и формальной семантики) — это плохо формализованные, плохо структурированные и плохо размеченные человеческие “знания” о компьютерных языках, извлекаемые из научных и учебных статей о конкретных языках, из форумов, каналов и порталов, посвященных практике использования языков в образовании, в разработке, в информационных технологиях и так далее. Здесь, однако, уместно напомнить, что мы употребляем слово “знания”, но на самом деле надо говорить только о представлениях или мнениях (beliefs), так как обычно в статьях, в постах и на страницах публикуют авторские мнения, а не знания в строгом научном смысле слова: согласно традиции, восходящей к Платону, знания — это представления о реальности доказательно соответствующие действительности (а, следовательно, непротиворечивые), а мнения людей, участвующих в жизненном цикле компьютерных языков, могут быть просто противоречивыми (даже ку одного человека), или могут не соответствовать действительности.

Такое понимание прагматики компьютерных языков естественно приводит [9, 15, 16] к идее использования *онтологии* для представления прагматики компьютерных языков [30]. Согласно [38]

> Ontology is the theory of objects and their ties. It provides criteria for distinguishing different types of objects (concrete and abstract, existent and nonexistent, real and ideal, independent and dependent) and their ties (relations, dependencies and predication).

Поэтому под онтологией предметной области (прагматики компьютерных языков в частности) мы будем понимать онтологию, реализованную в некотором формализме для представления мнения "экспертов" об объектах этой области, их классах и отношениях между объектами и объектами, объектами и классами, классами и классами, и так далее. Будем говорить, что онтология задана явно, если она явно представлена в виде графа, вершины которого — объекты онтологии, а отношения между объектами представлены дугами (или рёбрами); в противном случае будем говорить о неявной онтологии. Wikipedia (`www.wikipedia.org`) — это хорошо известный пример неявной онтологии. А популярный инструмент создания явных онтологий — это платформа Protégé [47].

Следуя [9], подчеркнём, что онтологическое представление прагматики компьютерных языков должна быть *публичной*, *открытой*, *эволюционирующей*, *темпоральной* и *со временем* онтологией. Под публичностью понимается доступ для получения данных и внесения изменений без ограничений и привилегий пользователей: любой участник жизненного цикла компьютерного языка может обратиться к онтологии (на правах анонимности или добровольной самоидентификации) с любым допустимым запросом для получения данных, удовлетворяющих запросу, и может добавить любые данные, удовлетворяющие поддерживаемым форматам. Открытость означает, что в онтологии принята модель открытого мира, означающая, что не все существующие объекты и отношения уже представлены в онтологии. Под эволюционностью понимается не только развитие онтологии во времени, но и поддержку историй правок, хронику её развития так, что в любой момент можно получить справку о её состоянии в любой момент в прошлом. И, наконец, темпоральность и использование времени означает наличие "штампов времени" у правок, возможность формулировки запросов с привязкой ко времени (неформальный пример: "что думали в 1970 году лауреаты премии Тьюринга о перспективах развития языков программирования, созданных в 1960-ые годы, к 1980-ому году?") и использованием темпоральных конструкций

(например: *до тех пор пока* и так далее). Wikipedia может служить хорошим примером публичной открытой эволюционирующей онтологии, однако, язык запросов к этой неявной онтологии — это язык позитивных булевских запросов без темпоральных конструкций.

3. На пути к реализации онтологической классификации

В работах [9, 15, 16] идея использовать аппарат формальных онтологий для представления "знаний" о прагматике компьютерных языков была распространена на онтологию "знаний" о компьютерных языках вцелом. Объектами этой онтологии могут быть все компьютерные языки, причём, слои и уровни следует считать отдельными компьютерными языками, "диалектами" друг друга. Отношения между языками в этой онтологии — это отношения, типичные для области компьютерных языков, например: *быть диалектом*, *быть версией*, *наследует синтаксис* и другие "связи" между языками (исторические, реализационные, гносеологические, и так далее). Атрибутами объектов и отношений могут выступать (формальные и неформальные) характеристики синтаксиса и семантики, (неформальные) характеристики прагматики компьютерных языков, временны́е теги (например, год первой публикации), ссылки на проеты, выполненные с использованием этих языков, и так далее.

Заметим, что постер Computer Languages History [36] и справочник The Language List [34] уже реализуют онтологический подход к классификации (хотя явно это нигде не сказано). Ещё в большей степени понятию онтологии соответствует HOPL: an interactive Roster of Programming Languages [39]: в этой онтологии представлено более 8500 объектов (языков программирования в данном случае), почти 1800 библиографических ссылок (в качестве атрибутов языков) и почти 5500 отношений (точнее — связей) между языками (влияние, диалекты, версии и реализации, связи по авторам, по году и месту рождения и так далее), поддержана (уже упоминавшейся нами довольно-таки спорной) таксономия классов языков программирования. Средства навигации по этой онтологии представлены связями языков и таксономией классов. Недостатком этой онтологии является её непубличность (закрытость для редактирования) и фиксированное состояние (на 2006 г.).

Ещё один существующий проект классификации — это двуязычная (имеет русскую и английскую версии) открытая эволюционирующая wiki-подобная неявная онтология языков программирования Progopedia [41]. Она отслеживает три вида связей между языками (диалект, версия, реализация), предлагает две простых таксономии языков (из 31 парадиг-

мы программирования и 13 вариантов типизации в языках программирования), но менее населена объектами чем The Language List [34] и тем более чем HOPL [39]: в Progopedia представлены сведения о 171 языках программирования, 83 их диалектах, 349 реализациях и 734 версиях (то есть, в нашей терминологии, о 1337-ми объектах). Средства навигации между объектами — алфавитный порядок, диалекты, версии и реализации, таксономии по парадигмам и вариантам типизации.

Однако общим недостатком классификаций [34, 36, 39, 41] компьютерных языков (языков программирования в частности) мы считаем отсутствие явного понимания, что они имеют дело с онтологией соотвествующего универсума и, следовательно, неиспользование современных технологий Semantic Web для онтологий предметных областей.

Поэтому при разработке публичной открытой эволюционирующей темпоральной онтологии для классификации компютерных языков в 2012-2013 гг. в работах [9, 15, 16] в качестве основы была выбрана логика описаний понятий (известная так же как дискрипционная логика — Description Logic, DL) [13, 18, 30] с использованием технологий языка OWL (Web Ontology Language) [30, 31]. Поэтому ниже мы в описании подхода [9, 15, 16] будет использована терминология DL и OWL (что будет обозначаться через слеш — термин DL / термин OWL).

Объектами разрабатываемой онтологии являются компьютерные языки, включая слои и уровни как отдельные объекты. Например, Pascal, LISP, PROLOG, SDL, а также OWL-Lite, OWL-DL и OWL-full — все эти языки могут быть объектами. Связи между объектами задаются ролями/свойствами, которые могут быть специфицированы ролевыми термами DL, построенных из явно заданных элементарных ролей/свойств. Например, *наследовать синтаксис* или *быть диалектом* могут быть элементарными ролями/свойствами, которые задаются явным указанием пар объектов (компьютерных языков), связанных соответствующим отношением. Понятиями/классами являются множества объектов, которые могут быть специфицированы (выделены) посредством понятийных термов DL, построенных из явно заданных элементарных понятий. Примерами элементарных понятий могут быть *является функциональным языком* или *является языком спецификаций*, их содержание должно быть задано явным указанием объектов (компьютерных языков), входящих в соответствующий класс.

В работах [13, 18, 30] компьютерные парадигмы выделяются посредством понятийных термов DL: каждый понятийный терм определяет компьютерную парадигму посредством

выделения понятия/класса компьютерных языков. (В такой трактовке парадигмы компьютерных языков перестают быть древовидной таксономией, а становятся подрешёткой решётки множеств компьютерных языков.) Объекты (компьютерные языки) могут быть аннотированы разнообразными формальными атрибутами (например, характеристиками формального синтаксиса) и неформальными атрибутами (например, библиотеками примеров хорошего стиля для языка); список возможных атрибутов не фиксирован, но некоторые атрибуты обязательны (то есть автоматически связываться с любым объектом при его инициализации), например:

- дата рождения языка,
- авторство языка,
- рекомендуемая библиография,
- ссылка на доступную пробную версию.

(Заметим, что значения этих атрибутов могут быть неопределенны.)

Так же как и некоторые атрибуты, некоторые элементарные понятия/классы должны автоматически присутствовать в нашей онтологии (поскольку являются классикой предметной области), например: функциональные языки, языки спецификаций, исполняемые языки, языки со статическими типами и так далее (см., например, списки парадигм на [34, 39, 41, 42]). Кроме того, по-видимому обязательным является элементарный класс *типических* или *парадигмальных* языков, который должен содержать хотя бы по одному языку (но немного) из каждого другого элементарного понятия/класса (который автоматически добавляется при создании нового элементарного понятия/класса).

Наполнение элементарных понятий/классов должно происходить на основе явно указанных атрибутов объектов (компьютерных языков), например, язык попадает в языки спецификаций, если у этого языка значение соответствующего атрибута установлено явно. Таким образом, семантика элементарных понятий/классов имеет вполне определённый экстенсиональный характер: с одной стороны, содержание каждого элементарного понятия/класса в каждый конкретный момент времени задаётся путём явного перечисления, но, стороны, у объекта могут быть как недоопределённные так и переопределённые атрибуты. Например, у языка C♯ может быть не указано мнение, что язык является функциональным, а может быть указано два противоречивых мнения, что этот язык является функциональным, и что не является. А вот неэлементарные понятия/классы определяются чисто экстенсионально через элементарные понятия/классы посредством понятий-

ных термов DL, их содержание может быть не полностью определено (даже в каждый конкретный момент времени). Например, *исполнимые языки спецификации* — это пересечение понятий/классов *языков спецификаций* и *исполнимых языков*, недоопределенность содержания этого понятия/класса может произойти только из-за недоопределенности содержания каждого из онятий/классов *языков спецификаций* и *исполнимых языков*; а вот *неисполнимые языки спецификации* — это пересечение понятий/классов *языков спецификаций* и дополнения класса *исполнимых языков*, недоопределенность содержания этого понятия/класса может произойти не только из-за недоопределенности содержания каждого из онятий/классов *языков спецификаций* и *исполнимых языков*, но из-за неопределённости операции дополнения в модели открытого мира (см. следующий абзац).

Обсудим подробнее вариант определения отрицания в модели открытого мира (то есть всегда неполной информации) для универсума компьютерных языков. В [9] предлагался подход, в котором одновременно с введением какого-либо нового *позитивного атрибута* будет автоматически порождаться его *негативный* напарник. (Например, при создании позитивного атрибута *является исполняемым языком* будет автоматически порождён и его негативный атрибут-напарник *не является исполняемым языком*.) Введение негативных атрибутов (которые также могут иметь неопределённые значения) позволяет частично решить проблему дополнения для понятий/классов: в дополнение до какого-либо элементарного класса, определённого позитивным атрибутом, попадают только те языки, у которых явно проставлено значение соответствующего негативного атрибута. Например, язык программирования C не является языком спецификаций в онтологии, если только явно проставлено значение его соответствующего негативного атрибута (то есть такое мнение представлено в онтологии).

Элементарные роли/свойства — это естественные отношения между компьютерными языками: *быть диалектом*, *наследовать синтаксис* и так далее. Например: “C-light является промежуточным слоем C”, “OWL наследует синтаксис XML” и так далее. Для построения сложных ролей/свойств из элементарных можно использовать стандартный набор монотонных операций алгебры бинарных отношений: объединение, пересечение, инверсию (взятие обратного отношения), транзитивное замыкание. Например, роль/свойство *использует синтаксис диалекта* — это просто композиция элементарных ролей/свойств *использует синтаксис* и *является диалектом*. (Однако с операцией дополнения ролей/свойств возникают те же трудности, что и с операцией дополнения понятий/классов, но для ча-

стичного преодоления этих затруднений использовать подход, основанный на создании негативного напарника для каждой позитивной элементарной роли/свойства.)

В мире компьютерных языков есть три роли/свойства, специфические для этой предметной области: *быть диалектом*, *быть версией*, *быть реализацией*. (Эти роли/свойства используются в [36, 41].) Разумеется, эти роли/свойства должны быть элементарными в этой онтологии и задаваться явным перечислением пар языков. Но для пользователя онтологией надо предусмотреть правило, которыми следует руководствоваться при причислении пар языков к одному из этих отношений. К сожалению, нет консенсуса по определению этих отношений между языками. Так, например, Progopedia [41] считает, что реализация имеет несколько версий, а The Language List [36] считает наоборот, что версия может иметь несколько реализаций. Поэтому в [9] было предложено закрепить следующие определения:

- Диалекты — это языки у которых общий элементарный уровень. Например, Common LISP и Home LISP — это диалекты друг друга.
- Версии (или варианты) — это языки с общим основным уровнем. Например, Visual C и Borland C — это, наверное, версии друг друга.
- Реализация — это платформа-специфическая версия языка. Например, Visual C — это версия языка C Карнигана и Риччи для платформы Windows.

В 2012 г. альфа-версия онтологии компьютерных языков (ограниченной функциональности) была реализована А.А. Акининым в виде портала знаний по классификации компьютерных языкаов и доступна в течении нескольких месяцев для ознакомления в сети Интернет (по адресу `http://complang.somee.com/Default.aspx`). Этот портал был реализован как веб-приложение, интерфейс позволял пользователю просматривать и редактировать основные элементы онтологии — это компьютерные языки (объекты), элементарные свойства/классы объектов, элементарные роли/отношения между объектами, атрибуты (аннотации объектов) и базу знаний (совокупность предложений DL, которые претендуют на то, что бы быть законами универсума компьютерных языков). Для внутреннего представления данных использовался RDF-репозиторий. Были реализованы два основных сервиса: это решатель запросов как основное средство выделения классов, навигации по онтологии и проверки совместности, и визуализация онтологии в виде графа. Решатель — это верификатор моделей с явным представлением состояний (explicit state model checker), но для варианта DL, построенного не над булевской двузначной логикой,

а над четырехзначной логикой Бэлнапа [19], обогащённого алгебраическими конструкциями из анализа формальных понятий (Formal Concept Analysis — FCA) [13, 30]. Первоначальное наполнение портала было произведено путём импорта данных о компьютерных языках из открытых хорошо структурированных и размеченных источниках в Интернете — Progopedia [41] и DBpedia (проект, направленный на извлечение структурированной информации из данных, созданных в рамках проекта Википедия); фиксированная структура и разметка данных в Progopedia и DBpedia позволяли осуществить поиск данных простым парсингом страниц Progopedia и DBpedia.

Однако, (как уже было сказано в части 1) к 2013 г. проект создания классификации компьютерных языков и соответсвующего компьютерного инструментария столкнулся с рядом вызовов, на которые в 2013 г. участники не смогли в то время ответить. Это, во-первых, наступление эры проблемно-ориентированных компьютерных языков, о большинстве которых никогда не появится информация ни в Progopedia, ни в Wikipedia, ни в DBpedia и даже в научной периодике по компьютерным языкам, но только в литературе по конкретной предметной области, для которой создавался тот или иной компьютерный язык.

Во-вторых, для компьютерных языков имеет значение аспект *популярности* языка, которую можно определить по-разному. Например, популярность можно оценить при помощи так называемого индекса TIOBE (TIOBE programming community index) [46]. Этот индекс оценивает популярность языков программирования на основе подсчёта результатов поисковых запросов, содержащих название языка, на порталах Google, Blogger, Wikipedia, YouTube, Baidu, Yahoo!, Bing, Amazon. (Можно предположить, что существует связь между количеством найденных страниц и количеством программных проектов, разработчиков и вакансий, связанных с тем или иным языком.) Однако, такое определение популярности не учитывает развитость социальной сетей (сообществ) профессионалов, студентов и любителей, вовлечённых в жизненный цикл конретных языков, сервиса (поддержки) ответов на вопросы со стороны сообществ и компаний.

В результате проект созданий классификации компьютерных языков и соотвествующего компьютерного инструментария был на время отложен (причём, сожалению, когда ещё не было опубликовано ни формальное описание использованной логики, ни алгоритма решателя логических запросов.)

Однако с 2013 г. прошло достаточно времени, и сейчас в 2018 г. мы надеемся, что сумеем

дать ответ на вызовы, ранее остановившие проект классификации компьютерных языков и создания портала знаний о классификации компьютерных языков. Во-первых, для автоматического поиска и извлечения информации по компьютерным языкам можно попытаться применять методы извлечения информации из текстов [22] (используя для первоначального наполнение портала импорт данных из Progopedia и DBpedia). И, во-вторых, для определения популярности компьютерных языков — методы анализа социальных сетей [37] (на ряду с индексом TIOBE).

4. Заключение

И так, подведём итог данной статьи. Во-первых, настоящая статья относится к жанру position paper: "Position papers in academia enable discussion on emerging topics without the experimentation and original research normally present in an academic paper. Commonly, such a document will substantiate the opinions or positions put forward with evidence from an extensive objective discussion of the topic."[3] Однако, особенность настоящей статьи является то, что, с одной стороны, она написана по материалам предыдущих исследований, выполненных в период 2008-2013 гг., а, с другой стороны, — после значительного перерыва в пять лет перед возобновлением приостановленных исследований, но уже с использованием новых подходов.

Сумма основных результатов исследований в период 2008-2013 гг.:

- Была выработана концепция онтологического подхода к классификации компьютерных языков, основанного на гибком понимании парадигм языков, которые определяются всей совокупностью характеристик языка и его связей с другими языкам.
- Была осознано значение компьютерной поддержки онтологии компьютерных языков и эффективного языка запросов для навигации по этой онтологии, причём, язык запросов должен поддерживать дискрипционные темпоральные временны́е запросы в модели открытого мира (всегда неполной информации).
- Был реализован прототип портала реализующий статический фрагмент онтологии компьютерных языков (содержавший сведения по более чем 1200 языкам программирования, извлечёнными из структурированных и размеченных DBpedia и Progopedia) и языка запросов на основе дискрипционной логики над четырёхзначной логикой Белнапа.

[3] Цитируется по [40].

Задачи, которые надо решить в ближайшее время для продолжение работ по проекту классификации и навигации по миру компьютерных языков:

- Разработать инструменты (основанные на методах анализа текстов на естественном языке и машинного обучения) для извлечения сведения о компьютерных языках из текстовых интернет-ресурсов и социальных сетей (используя в качестве начального "знания" (обучающей выборки) "знания" из структурированных и размеченных DBpedia и Progopedia).
- Разработать схему публичной, открытой, эволюционирующей, темпоральной и со временем онтологии и архитектуру портала знаний о классификации компьютерных языков, а затем — приступить к прототипированию этого портала (используя в качестве начального наполнения сведения о языках программирования из DBpedia и Progopedia) и интеграции его с инструментами извлечения сведений о компьютерных языках из интернет-ресурсов.
- Формально описать дискрипционный темпоральный временно́й логический язык запросов над четырёхзначной логикой Белнапа, исследовать алгоритмы верификации запросов на этом языке в конечных открытых моделях, прототипировать решатель таких запросов и интегрировать его с порталом знаний о классификации компьютерных языков.

Список литературы

1. Андреева Т. А., Ануреев И. С., Бодин Е. В., Городняя Л. В., Марчук А. Г., Мурзин Ф. А., Шилов Н. В. Образовательное значение классификации компьютерных языков // Прикладная информатика. 2009. № 6(24). С.18-28.
2. Ахмадеева И.Р., Боровикова О.И., Загорулько Ю.А., Сидорова Е.А. Сбор онтологической информации для интеллектуальных научных Интернет-ресурсов // Системная информатика. 2014. № 3. С. 13–23.
3. Боровикова О.И., Загорулько Ю.А. Подход к реализации паттернов содержания при разработке онтологий научных предметных областей // Системная информатика. 2018. № 12. С. 27–40.
4. Гаранина Н.О., Зюбин В.Е., Лях Т.В. Онтологический подход к организации шаблонов требований в рамках системы поддержки формальной верификации распределенных программных систем // Системная информатика. 2017. № 9. С. 111–132.
5. Городняя Л.В. Парадигмы программирования: анализ и сравнение. Новосибирск: Издательство СО РАН, 2017. 232 с.
6. Городняя Л.В. Парадигмальный подход к факторизации определений языков и систем про-

граммирования // Системная информатика. 2018. №12. С. 1-26.

7. Дмитриев С. Языково-ориентированное программирование // RSDN Magazine. 2005, №5. [Электронный ресурс]. Систем. требования: любой интернет-браузер. http://rsdn.org/article/philosophy/LOP.xml (дата обращения: 15.12.2018).
8. Ершов А.П. Предварительные соображения о лексиконе программирования // Избранные труды. Новосибирск. 1994. С. 395–406.
9. Идрисов Р.И., Одинцов С.П., Шилов Н.В. Онтологический подход к проблеме классификации компьютерных языков: состояние и перспективы // Системная информатика. 2013. № 1. С. 63-78.
10. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. СПб.: БХВ-Петербург, 2010. 552 с.
11. Непейвода Н.Н. Стили и методы программирования. Курс лекций. Учебное пособие. М.: Бином. Лаборатория знаний / Интернет-Университет Информационных Технологий (ИНТУИТ), 2005. 320 с.
12. Непомнящий В.А., Ануреев И.С., Дубрановский И.В., Промский А.В. На пути к верификации C♯-программ: трехуровневый подход // Программирование. 2006. № 4. С. 4–20.
13. Шилов Н.В. Формализмы и средства создания и поддержания онтологий. В "Модели и методы построения информационных систем, основанных на формальных, логических и лингвистических подход" коллективная монография под ред. Марчука А.Г.. Новосибирск: из-во СО РАН. 2009. С.10-48.
14. Akhmadeeva I.R., Zagorulko Y.A., Mouromtsev D.I. Ontology-based information extraction for populating the intelligent scientific internet resources // Communications in Computer and Information Science. 2016. Vol. 649. P. 119–128
15. Shilov N.V., Akinin A.A., Zubkov A.V., and Idrisov R.I. Development of the Computer Language Classification Portal // Springer Lecture Notes in Computer Science. 2012. Vol. 7162. P. 340-348.
16. Akinin A.A., Zubkov A.V., Shilov N.V. New Developments of the Computer Language Classification Knowledge Portal // Proceedings of the 6th Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2012), May 30-31, Perm, Russia. Moscow: Institute of System Programming. 2012. P. 54-58.
17. Anureev I., Bodin E., Gorodnyaya L., Marchuk A., Murzin F., Shilov N. On the problem of computer language classification // Computer Science. 2008. №28 P. 31–42.
18. Baader F., Calvanese D., McGuinness D.L., Nardi D., Patel-Schneider P.F. The Description Logic Handbook: Theory, Implementation, Applications. Cambridge: Cambridge University Press, 2003. 624 p.
19. Belnap N.D. How a computer should think // Contemporary Aspects of Philosophy: Proceedings of the Oxford International Symposium. Oriel Press. 1977. P.30–56. (Есть русский перевод: Бэлнап Н. Как нужно рассуждать компьютеру // Белнап Н., Стил Т. Логика вопросов и ответов. М.: Прогресс. 1981. [Электронный ресурс]. Систем. требования: любой интернет-браузер. URL: http://scicenter.online/logika-scicenter/kak-nujno-rassujdat-62531.html (дата обращения:

15.12.2018).)

20. Anureev I.S., Promsky A.V. Conceptual transition systems and their application to development of conceptual models of programming languages // System Informatics. 2017. N 9. P. 133–154.
21. Anureev I.S. Operational conceptual transition systems and their application to development of conceptual operational semantics of programming languages // System Informatics. 2017. N 9. P. 155–200.
22. Baeza-Yates R., Ribeiro-Neto B. Modern Information Retrieval: The Concepts and Technology behind Search (second edition). Addison-Wesley, 2011. 944 p.
23. Börger E., Stärk R. Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag, 2003. 438 p.
24. Efftinge S., Köhnlein J., Friese P. Build your own textual DSL with Tools from the Eclipse Modeling Project. 2008. [Электронный ресурс]. Систем. требования: любой интернет-браузер. http://www.eclipse.org/articles/article.php?file=Article-BuildYourOwnDSL/index.html (дата обращения: 15.12.2018).
25. Floyd R.W. The paradigms of Programming // Communications of ACM. 1979. Vol. 22. P. 455–460. (Есть русский перевод: Флойд Р. О парадигмах программирования // Лекции лауреатов премии Тьюринга за первые двадцать лет. 1966–1985. М.: Мир, 1993. 560 с.)
26. Fowler M. Domain-Specific Languages. Addison-Wesley, 2011. 640 p. (Есть русский перевод: Фаулер М. Предметно-ориентированные языки программирования. Вильямс, 2017. 576 с.)
27. Garanina N.O., Zubin V., Lyakh T., Gorlatch S. An Ontology of Specification Patterns for Verification of Concurrent Systems // In: New Trends in Intelligent Software Methodologies, Tools and Techniques. Proceedings of the 17th International Conference SoMeT-18, Granada, Spain, 26-28 September 2018. H. Fujita and E. HerreraViedma (Eds.). Frontiers in Artificial Intelligence and Applications. Vol. 303. Amsterdam: IOS Press. 2018. P. 515–528.
28. Gurevich Yu. Evolving Algebras: An Attempt to Discover Semantics // Current Trends in Theoretical Computer Science / Eds. G. Rozenberg and A. Salomaa. World Scientific, 1993. P. 266–292.
29. Gurevich Yu. Sequential Abstract State Machines capture Sequential Algorithms // ACM Transactions on Computational Logic. 2000. Vol. 1. №1. P. 77–111. (Есть русский перевод: Гуревич Ю. Последовательные машины абстрактных состояний охватывают последовательные алгоритмы // Системная информатика. Вып. 9 / Пер. П.Ф. Емельянова. Новосибирск: Изд-во СО РАН, 2004. С. 7–50.)
30. Handbook on Ontologies. International Handbooks on Information Systems / Editors Staab S., Studer R. Springer. 2009. 2nd edition. 660 p.
31. Hitzler P., Krötzsch M., Rudolph S. Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009. 456 p.
32. JetBrains MPS. 2006-2017. [Электронный ресурс]. Систем. требования: любой интернет-браузер. http://www.jetbrains.com/mps/ (дата обращения: 15.12.2018).
33. Internationa Conference on Software Languages Engineering. 2018. [Электронный ресурс]. Си-

стем. требования: любой интернет-браузер. URL: http://www.sleconf.org (дата обращения: 15.12.2018).

34. Kinnersley B. The Language List. [Электронный ресурс]. Систем. требования: любой интернет-браузер. https://web.archive.org/web/20160506170543/http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm (дата обращения: 15.12.2018).
35. Kuhn T.S. The structure of Scientific Revolutions. Univ. of Chicago Press, 3rd Ed., 1996. (Есть русский перевод: Кун Т. Структура научных революций. М.: Издательство АСТ, 2003. 605 с.)
36. Levenez E. Computer Languages History. [Электронный ресурс]. Систем. требования: любой интернет-браузер. https://www.levenez.com/lang/ (дата обращения: 15.12.2018).
37. McCulloh I., Armstrong H., Johnson A. Social Network Analysis with Applications. Wiley, 2013. 320 p.
38. Ontology: Theory and History. [Электронный ресурс]. Систем. требования: любой интернет-браузер. https://www.ontology.co. (дата обращения: 15.12.2018).
39. Pigott D. HOPL: an interactive Roster of Programming Languages. 1995-2006. [Электронный ресурс]. Систем. требования: любой интернет-браузер. https://web.archive.org/web/20080511193953/http://hopl.murdoch.edu.au/ (дата обращения: 15.12.2018).
40. Position paper. From Wikipedia, the free encyclopedia. [Электронный ресурс]. Систем. требования: любой интернет-браузер. https://en.wikipedia.org/wiki/Position_paper (дата обращения: 15.12.2018).
41. Progopedia. [Электронный ресурс]. Систем. требования: любой интернет-браузер. http://progopedia.ru/ (дата обращения: 15.12.2018).
42. van Roy P. Classification of the principal programming paradigms. 2009. [Электронный ресурс]. Систем. требования: любой интернет-браузер. http://www.info.ucl.ac.be/∼pvr/paradigms.html (дата обращения: 15.12.2018).
43. van Roy P. Programming Paradigms for Dummies: What Every Programmer Should Know // New Computational Paradigms for Computer Music. IRCAM/Delatour, France. 2009. P. 9–38.
44. Ruby. A Programmer's best friend. [Электронный ресурс]. Систем. требования: любой интернет-браузер. URL:http://www.ruby-lang.org/en/about/ (дата обращения: 15.12.2018).
45. Stärk R., Schmid J., Börger E. Java and the Java Virtual Machine. Springer, 2001. 381 p.
46. TIOBE Index | TIOBE — The Software Quality Company [Электронный ресурс]. Систем. требования: любой интернет-браузер. https://www.tiobe.com/tiobe-index/ (дата обращения: 15.12.2018).
47. Tudorache T., Nyulas C. I., Musen M. A., Noy N. F. WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web // Semantic Web Journal. 2011. Vol. 11. № 165. P. 1–11.
48. Walther Ch., Schweitzer St. About VeriFun // Springer Lecture Notes in Computer Science. 2003. Vol. 2741. P. 322–327 .
49. Xtext 2006-2017. [Электронный ресурс]. Систем. требования: любой интернет-браузер.

http://www.eclipse.org/Xtext/ (дата обращения: 15.12.2018).

50. Zagorulko Y., Borovikova O., Zagorulko G. Pattern-Based Methodology for Building the Ontologies of Scientific Subject Domains. In: New Trends in Intelligent Software Methodologies, Tools and Techniques. Proceedings of the 17th International Conference SoMeT-18, Granada, Spain, 26-28 September 2018. H. Fujita and E. HerreraViedma (Eds.). Frontiers in Artificial Intelligence and Applications. Vol. 303. Amsterdam: IOS Press. 2018. P. 529–542.