УДК: 168.2 + 025.4 + 519.682 + 004.43 + 811.93

Название: Онтологический подход к проблеме классификации компьютерных языков: состояние и перспективы

Автор(ы):

Идрисов Р.И. (Институт систем информатики им. А.П. Ершова СО РАН),

Одинцов С.П. (Институт математики им. С.Л. Соболева СО РАН)

Шилов Н.В. (Институт систем информатики А.П. Ершова СО РАН, Новосибирский государственный университет, Новосибирский государственный технический университет)

Аннотация: За полувековую историю развития программирования и информационных технологий возникло несколько тысяч компьютерных языков. Виртуальный мир компьютерных языков включает языки программирования, спецификаций, моделирования и другие языки. В каждой из этих ветвей можно выделить разные подходы (например, императивный, декларативный, объектно-ориентированный), дисциплины обработки (например, последовательная, недетерминированная, распределённая) и формализованных моделей (от машин Тьюринга до машин логического вывода). Поэтому актуальной становится проблема классификации компьютерных языков. Для решения этой проблемы предлагается подход, основанный на парадигмах компьютерных языков и выделении общих атрибутов, присущих разным языкам. При этом особую роль призван сыграть разрабатываемый портал знаний о компьютерных языках. Этот портал предназначен для поиска сведений о компьютерных языках в открытых структурированных источниках в сети Интернет, организации хранения и доступа к собранной информации по логическим запросам с целью выявить законы мира компьютерных языков и помощи в выборе компьютерных языков для формирования новых программных проектов. Существующая в настоящее время версия портала далека от совершенства как с точки зрения представленной информации (количественно и качественно), так и с точки зрения выбора логического языка запросов. Поэтому в настоящей статье представлено общее описание проблемы классификации компьютерных языков, описание принципиального подхода к этой проблеме, текущее состояние портала, а также обсуждаются перспективы развития логического языка запросов.

Ключевые слова: компьютерные языки, компьютерные парадигмы, классификация, портал знаний, логика описаний понятий, паранепротиворечивость, устойчивость к противоречиям, верификация моделей.

1. Введение. Под *компьютерным языком* мы понимаем любой искусственный язык, разработанный для автоматической обработки информации, как то: представления,

преобразования и управления данными и процессами. Под *классификацией* какого-либо универсума (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.

Первый вопрос, на который надо дать ответ, почему мы считаем актуальной проблему классификации компьютерных языков. Простой ответ состоит в том, что необходимость классификации (и систематизации) следует уже из числа известных компьютерных языков, которых уже несколько тысяч. Так, например, на плакате *The History of Programming Languages* (http://oreilly.com/news/graphics/prog_lang_poster.pdf) издательства O'REILLY, специализирующегося на публикации ученой литературы по компьютерным языкам, представлены сведения о 2500 языках. Эти сведения включают названия, год рождения, авторство, влияние друг на друга и развитие версий компьютерных языков, появившихся в период с 1956 по 2006 гг.

Однако значение классификации компьютерных языков не исчерпывается желанием «навести порядок и систематику». Классификация уже существующих и новых компьютерных языков призвана помочь в выборе подходящих языков для разработки новых программных и информационных систем на основе требований к этим новым системам.

Классификацию компьютерных языков можно пытаться строить по аналогии с естественными науками, например «по Линнею»: царство – тип (у растений отдел) – класс – отряд (у растений порядок) – семейство – род – вид. Так, например, была устроена *Taxonomic system for computer languages* (которая была доступна с 2006 по 2011 г. на http://hopl.murdoch.edu.au/taxonomy.html). Эта таксономия включала множество сведений о 8512 языках, но выделяла очень странные классы компьютерных языков.

На наш взгляд, сама идея таксономии как основы классификации компьютерных языков представляется довольно-таки сомнительной хотя бы уже в силу большого отличия предметных областей в естественных науках и областью компьютерных языков: если предметные области биологии, химии, физики элементарных частиц сравнительно статичны, то область компьютерных языков высоко динамична.

Так, например, только за последнее десятилетие XX века мы наблюдали как быстрый рост 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 (т.е. модель обработки информации или виртуальную машину), свою прагматику (т.е. сферу применения и распространения).

Некоторые из давно существующих классов компьютерных языков сравнительно малочисленны (например, языки проектирования СБИС), некоторые – «густонаселенны» (например, языки спецификаций), а некоторые пережили или переживают сейчас период роста и миграций (например, языки разметки). В тоже время разработка новых компьютерных языков (а, следовательно, и образование новых классов компьютерных языков) будет продолжаться по мере компьютеризации новых отраслей человеческой деятельности.

Кроме того, часто специалисты по компьютерным языкам затрудняются отнести тот или иной язык 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 они (в силу их декларативности) по-прежнему могут служить языками спецификаций для вычислительных программ (см., например, проект VeriFun на http://www.verifun.org/, [21]). А некоторые языки изначально по замыслу их разработчиков не могут быть отнесены к одному конкретному классу. Вот, например, как позиционируют язык Ruby его активные пропагандисты [23]: *Its creator, Yukihiro "matz", blended parts of his favorite languages (Perl, Smalltalk, Eiffel, Ada, and Lisp) to form a new language that balanced functional programming with imperative programming*.

Мы полагаем, что современная классификация универсума компьютерных языков может быть основана на гибком понимании *компьютерных парадигм*. 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 [9], впервые опубликованной в 1970 г.

Роберт Флойд стал первым в информатике, кто ввёл понятие *парадигм программирования* в научный оборот в его тьюринговской лекции *Paradigms of Programming* в 1978 г. [Floyd, 1979]. К сожалению, он не дал явного определения этого понятия.

В 2009 г. Питер ванн Рой опубликовал в Интернете таксономию *The principal programming paradigms* (http://www.info.ucl.ac.be/~pvr/paradigms.html), в которой выделил 27 различных парадигм, и предпринял попытку обосновать её в статье [20]. Однако ни эта таксономия, ни соответствующая статья не дают развёрнутого определения понятия парадигм программирования. Можно процитировать только следующее краткое (и неполное, на наш взгляд) определение [20]: *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.

На основе краткого определения понятия парадигмы программирования, предложенного Питером ванн Роем [20], в 2011 г. нами было дано следующее более полное понятие компьютерных парадигм [3, 17]:

- 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 значение компьютерной парадигмы характеризуется классом задач, для решения которых 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.

2. Роль синтаксиса, семантики и прагматики. Категории *синтаксиса*, *семантики* и *прагматики* используются для описания как естественных, так и искусственных языков (компьютерных языков в том числе). Синтаксис – это правописание (орфография) языка. Смысл правильно написанных фраз (слов) языка определяет его семантика. Прагматика – это практика использования синтаксически корректных осмысленных фраз языка.

Разумно предположить, что все эти три категории (синтаксис, семантика и прагматика) должны использоваться для выделения классов компьютерных языков и, опосредовано, для выделения компьютерных парадигм.

Использование синтаксиса для классификации компьютерных языков должно отражать как особенности формального синтаксиса, так и человеко-ориентированный аспект восприятия синтаксиса языков. Для разработки синтаксического анализатора очень важно знать, является ли данный язык регулярным, имеет или нет контекстно-свободный

синтаксис, порождается ли он LR(k) грамматикой и тому подобное. Следовательно, эти и другие формально-синтаксические свойства компьютерного языка могут и должны быть атрибутами, учитывающимися при классификации компьютерных языков.

Однако неформальные (или прагматические) характеристики синтаксиса компьютерного языка также могут и должны учитываться при классификации компьютерных языков. Примерами могут служить неопределяемые атрибуты *гибкость*, *естественность*, *ясность*, или определяемые атрибуты такие как, например, *стиль*, определяемый через библиотеку (набор) примеров хорошего стиля. При этом эти неформализуемые характеристики синтаксиса компьютерных языков приобретают всё более и более важное значение, в то время как значение формальных характеристик синтаксиса (в силу развития методов и технологий синтаксического анализа) меняется сравнительно медленно.

Роль и значение семантики для компьютерных языков общепризнанны. Косвенным подтверждением признания этого является количество лауреатов премии имени А. Тьюринга (самой престижной международной премии в области информатики), заслуживших эту премию за вклад в развитие семантики компьютерных языков: Эдсгер Дейкстра (1972), Джон Бэкус (1977), Роберт Флойд (1978), Ч. Энтони Р. Хоар (1980), Никлаус Вирт (1984), Робин Милнер (1991) и Питер Наур (2005). Таким образом, за 46-летнюю историю присуждения этой ежегодной премии 7 раз она присуждалась именно за вклад в развитие языков программирования и их семантики.

Однако, неформальная семантика мало пригодна для использования при классификации в силу нечёткости терминов и понятий, а использование формальной семантики для классификации компьютерных языков до сих пор наталкивалось на следующие препятствия:

- слабое знание формальной семантики среди пользователей компьютерных языков (разработчиков, руководителей и менеджеров проектов);
- распространённое предубеждение, что формальная семантика малополезна и неэффективна на практике;
- широкий спектр индивидуальных нотаций для формальной семантики с разным уровнем абстракции.

Может показаться, что все эти препятствия можно преодолеть путём унификации разных формальных семантик, развития алгоритмических инструментов для унифицированной семантики, преподавания унифицированной семантики в университетах и популяризации унифицированной семантики среди разработчиков.

Этот путь, по нашему мнению, требует чрезвычайно высокой централизации принятия решений о стандартизации формальных семантик, больших капиталовложений и, кроме того,

этот путь наверняка будет препятствовать появлению и внедрению новых вариантов формальных семантик.

Мы считаем, что затруднения, подобные перечисленным выше, могут быть преодолены путём *многомерной стратификации* семантики наиболее типических компьютерных языков. Подчеркнём, что мы ведём речь именно о многомерной стратификации семантики, когда в качестве одного из измерений может выступать, например, *образовательная* семантика, в качестве другого – *реализационная* и так далее.

Так, образовательная семантика компьютерного языка может делить язык на 2-3 образовательных *уровня*. Эти уровни можно условно назвать *элементарным*, *основным* (или базовым) и *экспертным* (или мастерским), причём элементарный и основной уровни или основной и экспертный уровни могут иногда совпадать. Элементарный образовательный уровень компьютерного языка – это учебный диалект для изучения основ языка человеком (возможно, даже без использования компьютера), с очень простой и прозрачной формальной семантикой в виде *виртуальной машины* языка. Основной образовательный уровень компьютерного языка – это диалект языка для регулярного использования, предполагающий понимание со стороны пользователя, как основные конструкции языка могут быть определены (в принципе) в терминах виртуальной машины языка. Экспертный образовательный уровень компьютерного языка – это язык в полном объеме с пониманием того, какие конструкции языка соответствуют формальной семантике, а какие нет, но имеют семантику обусловленную особенностями и эффективностью реализации (т.е. практикой использования). Обращаем внимание, что здесь речь идёт не о машинных языках, автокодах (или ассемблерах) и языках высокого уровня, а об образовательном аспекте языка.

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, 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# [1].

Прагматика компьютерных языков (в отличие от синтаксиса и формальной семантики) – это неформализованные представления людей, вовлечённых в жизненный цикл языка (авторов и реализаторов, «экспертов» и простых пользователей) о происхождении,

назначении и использовании этого языка. Другими словами, прагматика компьютерных языков – это человеческие «знания» о компьютерных языках.

Здесь, однако, уместно напомнить, что хотя мы и ведём речь о «знаниях», но на самом деле надо говорить только о представлениях или мнениях (beliefs), т.к. (согласно традиции, восходящей к Платону) знания – это представления о реальности, соответствующие действительности (т.е. истинные представления); «знания» разных людей, вовлечённых в жизненный цикл компьютерного языка, могут быть просто противоречивыми (и, следовательно, не могут соответствовать действительности одновременно).

Тем не менее, описанная интерпретация прагматики компьютерных языков естественно приводит к идее *явного* использования *формальных онтологий* для представления прагматики компьютерных языков.

Формальная «*онтология – это теория объектов и их связей. Онтология предусматривает критерии для выделения различных типов объектов (конкретных и абстрактных, существующих и несуществующих, реальных и идеальных, зависимых и независимых) и связей между ними (отношений, зависимостей и предшествования)*»[1] [22]. Поэтому под онтологией предметной области (прагматики компьютерных языков в том числе) мы будем понимать формальную онтологию, представляющую мнения «экспертов» об объектах этой области, их классах и отношениях между ними. Будем говорить, что онтология задана явно, если она явно представлена в виде графа, вершины которого – объекты онтологии, а дуги – отношения между объектами; в противном случае будем говорить о неявной онтологии. Наиболее известная неявная онтология – это Wikipedia (www.wikipedia.org). Наиболее популярный инструмент создания явных онтологий – это платформа Protégé (http://protege.stanford.edu/) [19].

Здесь уместно сформулировать ещё несколько требований к онтологическому представлению прагматики компьютерных языков: это должна быть *открытая эволюционирующая темпоральная* формальная онтология. Под открытостью мы понимаем 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, соответствующие поддерживаемым форматам. Под эволюционностью мы

[1] “ontology is the theory of objects and their ties. Ontology provides criteria for distinguishing various types of objects (concrete and abstract, existent and non-existent, real and ideal, independent and dependent) and their ties (relations, dependencies and predication)” [22]

понимаем 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 означает не только наличие штампов времени у правок, но и возможность формулировки запросов с привязкой ко времени и темпоральных конструкций (например «*до тех пор пока*», «*в следующей правке*» и так далее). Wikipedia может служить хорошим примером открытой эволюционирующей онтологии, язык запросов к этой неявной онтологии не поддерживает темпоральных конструкций.

3. На пути к онтологии компьютерных языков. Идея использовать аппарат формальных онтологий для представления «знаний» о прагматике компьютерных языков естественно обобщается на онтологию «знаний» о компьютерных языках. Объектами этой онтологии могут быть все компьютерные языки, причём слои и уровни каждого языка следует считать отдельными объектами (т.е. компьютерными языками), диалектами друг друга. Отношения между языками в этой онтологии – это отношения, типичные для данной предметной области, например, «*быть диалектом*», «*быть версией*», «*наследует синтаксис*». Атрибутами объектов и отношений могут выступать (формальные и неформальные) характеристики синтаксиса, семантики и (неформальные) характеристики прагматики компьютерных языков и связей между ними. Подробнее наш подход к разработке этой онтологии и её использованию для классификации компьютерных языков мы опишем ниже, а пока остановимся на существующих онтологиях компьютерных языков.

3.1 Существующие онтологии компьютерных языков. В первую очередь следует вспомнить уже упоминавшийся плакат *The History of Programming Languages* (http://oreilly.com/news/graphics/prog_lang_poster.pdf) издательства O'REILLY. Это явная онтология языков программирования не является ни открытой, ни эволюционирующей. Объектами этой онтологии являются языки программирования, но, к сожалению, в этой онтологии не выделены никакие классы языков программирования. Отношения между языками – это «*повлиял на*» (серые стрелки на плакате) и «*стал развитием*» (стрелка своего цвета для каждой линии развития). Языки аннотированы атрибутами «*год рождения*» и «*авторство*». Средства навигации по этой онтологии – по линии времени, по линиям развития или линиям влияния языков.

History of Programming Languages (HOPL, http://www.scribd.com/doc/6915615/An-interactive-Roster-of-Programming-Languages). Значительно более проработанная интерактивная явная онтология языков программирования. Она насчитывает 8512 объектов (т.е. языков программирования), 17837 библиографических ссылки (в качестве атрибутов для

языков), 5445 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) таксономией классов языков программирования. Средства навигации по этой онтологии представлены связями языков и таксономией классов. Недостатком этой онтологии является её закрытость для редактирования и фиксированное состояние (на 2006 г.).

Progopedia (http://progopedia.ru/) – это двуязычная (поддерживает русский и английский языки) открытая эволюционирующая wiki-подобная неявная онтология языков программирования. Она отслеживает три вида связей между языками (диалект, версия, реализация), предлагает две простых таксономии языков (из 31 парадигмы программирования и 13 вариантов типизации в языках программирования), но значительно «беднее» по числу объектов, чем постер от O'REILLY или HOPL. В Progopedia представлены сведения о 160 языках программирования, 76 их диалектов, 332 реализациях и 700 версиях (то есть 1258 объектах). Средства навигации между объектами – алфавитный порядок, диалекты, версии и реализации, таксономии по парадигмам и вариантам типизации.

3.2 Основы развиваемого подхода. Мы разрабатываем открытую эволюционирующую темпоральную онтологию на основе логики описаний понятий (Description Logic – DL) [2, 4, 18] с использованием технологий языка OWL (Web Ontology Language) [18, 8]. Поэтому мы в описании нашего подхода будем использовать терминологию DL и OWL через слеш DL/OWL.

Объектами разрабатываемой онтологии являются компьютерные языки, включая слои и уровни языков как отдельные объекты. Например, Pascal, LISP, PROLOG, SDL, LOTOS, UMLT, а также C, C-light and C-kernel, OWL-Lite, OWL-DL и OWL-full – эти все языки могут быть объектами. Связи между объектами задаются ролями/свойствами, которые могут быть специфицированы ролевыми термами DL, построенными из явно заданных элементарных ролей/свойств. Например, «*наследовать синтаксис*» или «*быть диалектом*» могут быть элементарными ролями/свойствами, которые должны быть заданы явным перечислением 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. Например, элементарными понятиями могут быть «*является функциональным языком*» или «*является языком спецификаций*»; их содержание должно быть задано явным перечислением объектов (компьютерных языков), входящих в соответствующий класс.

Т.к. мы считаем, что компьютерные парадигмы соответствуют классам компьютерных языков и наоборот, то сказанное выше о выделении классов компьютерных языков посредством понятийных термов DL означает, что каждый понятийный терм определяет компьютерную парадигму посредством выделения понятия/класса компьютерных языков. В такой трактовке парадигмы компьютерных языков перестают быть древовидной таксономией и становятся подрешёткой решётки множеств компьютерных языков.

Объекты (компьютерные языки) могут быть аннотированы разнообразными формальными атрибутами (например, характеристиками формального синтаксиса языка) и неформальными атрибутами (например, библиотеками примеров хорошего стиля для языка). Список возможных атрибутов не фиксирован, но мы полагаем, что некоторые атрибуты должны присутствовать обязательно (т.е. автоматически связываться с любым объектом при его инициализации), например:

- дата рождения языка (с разной степенью гранулярности – от точной даты до интервала в несколько лет);
- авторство языка;
- рекомендуемые библиографические ссылки (заданные URI или другим образом);
- ссылка на доступную пробную версию (простую и компактную в установке или сетевую реализацию языка, распространяемую свободно или условно-свободно).

(При этом мы не запрещаем, что значения этих атрибутов могут быть неопределенны, т.е. соответствующие поля незаполнены.)

Мы также полагаем, что некоторые элементарные понятия/классы должны автоматически присутствовать в нашей онтологии, например: языки с контекстно-свободным синтаксисом, функциональные языки, языки спецификаций, исполняемые языки, языки со статическими типами, языки с динамическим связыванием и другие. Кроме того, по-видимому имеет смысл предусмотреть элементарный класс *парадигмальных* языков, который должен содержать хотя бы по одному языку (но немного) из каждого другого элементарного понятия/класса. Мы также предполагаем заимствовать другие идеи для элементарных понятий/классов, представленные, например, в The Language List (http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm).

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.

Как уже было сказано, неэлементарные понятия/классы определяются через элементарные посредством понятийных термов DL. Например, *исполнимые языки спецификации* – это пересечение понятий/классов *языков спецификаций* и *исполнимых языков*.

Т.к. наша онтология строится в модели *открытого мира* (т.е. неполной информации), то, естественно, возникают трудности в определении операции дополнения для понятий/классов: например, если для какого-либо языка не сказано, что он имеет контекстно-свободный синтаксис, это еще не означает, что этот язык не имеет контекстно-свободного синтаксиса, но может означать, что значение соответствующего атрибута пока не проставлено. Поэтому мы предполагаем, что в нашей онтологии одновременно с введением какого-либо нового *позитивного* атрибута будет автоматически порождаться его *негативный* напарник. Например, при создании позитивного атрибута «*имеет контекстно-свободный синтаксис*» будет автоматически порождён негативный атрибут «*не имеет контекстно-свободный синтаксис*». Введение негативных атрибутов (которые также могут иметь неопределённые значения) позволяет 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 нашей онтологии, если только явно не проставлено значение его соответствующего негативного атрибута (иначе мы ничего не можем сказать на этот счёт, исходя из представленных в онтологии «знаний»).

Элементарные роли/свойства – это естественные отношения между компьютерными языками: «*быть диалектом*», «*наследовать синтаксис*» и т.д. Например: «C-light *является промежуточным слоем* C», «OWL *наследует синтаксис* XML» и так далее. Для построения сложных ролей/свойств из элементарных мы разрешаем 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, которые обусловлены неполнотой информации в модели открытого мира. Для преодоления этого мы намерены использовать тот же подход, основанный на создании негативного напарника для каждой позитивной элементарной роли/свойства.

Здесь стоит заметить, что в мире компьютерных языков есть три роли/свойства, специфические для этой предметной области: «*быть диалектом*», «*быть версией*», «*быть реализацией*». (Об этих ролях/свойствах мы упоминали при обсуждении онтологий HOPL и Progopedia.) Разумеется, эти роли/свойства являются элементарными в этой онтологии и должны задаваться явным перечислением пар языков. Но для пользователя онтологией надо предусмотреть правила, 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 считает, что реализация имеет несколько версий, а The Language List, наоборот, считает, что версия может иметь несколько реализаций. Поэтому для определённости мы закрепляем для нашей онтологии компьютерных языков следующие определения, которыми следует руководствоваться:

- 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 версии друг друга;
- реализация – это платформа-специфическая версия языка. Например, Visual C – это версия языка C Карнигана и Риччи для платформы Windows.

Универсальные и экзистенциальные (кванторные) ограничения также могут использоваться для конструирования новых понятий/классов. Например, если пользоваться нотацией DL, то понятийный терм

$$(\textit{markup language}) \sqcap \exists(\textit{uses syntax of}): (\neg\{\text{XML}\})$$

охватывает класс языков разметки (*markup language*), которые используют синтаксис не XML. Т.к. дополнение понятий в нашей онтологии реализуется при помощи негативных атрибутов, то, если предположить, что у языка разметки LATEX этот негативный атрибут был кем-то явно указан, то LATEX будет примером языка из этого понятия. Примером использования универсального ограничения может служить следующее предложение (из T-бокса): $\{\text{XML}\} \sqsubseteq \forall(\textit{is dialect of}): (\neg\{\text{ML}\})$; оно выражает тот факт, что XML является диалектом какого угодно языка, но не ML.

3.3 Программная реализация. Мы приступили к реализации портала знаний о компьютерных языках (который со временем должен развиться в явную открытую эволюционирующую темпоральную онтологию) около двух лет назад [17]. В 2011–12 гг. он был доступен для публичного тестирования в сети Интернет по адресу http://complang.somee.com/Default.aspx. Он был реализован как веб-приложение, поэтому

познакомиться с ним можно было с использованием браузера. Интерфейс позволял пользователю просматривать и редактировать любые данные портала. Однако этот прототип не поддерживал эволюционность и темпоральность. Для внутреннего представления данных использовался RDF-репозиторий.

Прототип портала предоставлял два основных сервиса: это решатель DL-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, но для паранепротиворечивого варианта DL [10, 11, 12], обогащённого алгебраическими конструкциями из анализа формальных понятий (Formal Concept Analysis – FCA) [7, 18. 2]. В основе этого варианта DL лежит четырёхзначная пропозициональная логика Белнапа [5, 13, 14, 15]. Конструкции. Заимствованные из FCA – это верхние и нижние производные. Такой выбор варианта DL обусловлен открытостью онтологии и неполнотой информации в модели открытого мира. Использование алгоритма верификации модели в качестве основы решателя запросов (вместо логического вывода) обусловлено тем, что мы намерены создать эволюционирующую онтологию для очень динамичного универсума компьютерных языков, а не инструмент для вывода следствий из заранее сформулированных законов, сложенных в базе знаний портала.

Благодарности. Работа подготовлена в рамках междисциплинарного проекта СО РАН №3 «Принципы построения онтологии на основе концептуализаций средствами логических дескриптивных языков» на 2012–2014 гг.

Список литературы

1. 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.
2. Шилов Н.В. Формализмы и средства создания и поддержания онтологий. // Модели и методы построения информационных систем, основанных на формальных, логических и лингвистических подхода: монография / под ред. Марчука А.Г.. Новосибирск: из-во СО РАН, 2009. С.10-48.
3. 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, 2012, Perm, Russia. P. 54-58.

4. Baader F., Calvanese D., Nardi D., McGuinness D., and Patel-Schneider P. (editors) The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.
5. Belnap N.D. How a computer should think. Contemporary Aspects of Philosophy: Proceedings of the Oxford International Symposium, 1977. P. 30-56.
6. Floyd R.W. The paradigms of Programming // Communications of ACM. 1979. V. 22. P. 455-460. (Русский перевод: Флойд Р. О парадигмах программирования. // В кн.: Лекции лауреатов премии Тьюринга. М: Мир, 1993.)
7. Ganter B. and Wille R. Formal Concept Analysis. Mathematical Foundations. Springer Verlag, 1996.
8. Hitzler P., Krötzsch M., Rudolph S. Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009.
9. Kuhn T.S. The structure of Scientific Revolutions. Univ. of Chicago Press, 3rd Ed., 1996. (Русский перевод: Кун Т. Структура научных революций. Издательство АСТ, 2003.)
10. Ma Y., Hitzler P., and Lin Z. Algorithms for paraconsistent reasoning with owl. Proc. of European Semantic Web Conference 2007, Springer Lecture Notes in Computer Science, V. 4519. 2007. P. 399-413.
11. Ma Y., Hitzler P., and Lin Z. Paraconsistent reasoning for expressive and tractable description logics. Proc. of the 21st International Workshop on Description Logic, CEUR Electronic Workshop Proceedings. V. 353, 2008.
12. Ma Y. and Hitzler P. Paraconsistent Reasoning for OWL 2. Proc. of Web Reasoning and Rule Systems, Springer Lecture Notes in Computer Science. V. 5837. 2009. P. 197-211.
13. Odintsov S.P., Wansing H. Inconsistency-Tolerant Description Logic. Motivation and Basic Systems, in: V.F. Hendricks, J. Malinowski (eds.) “Trends in Logic: 50 Years of Studia Logica”, Kluwer Academic Publishers, Dordrecht, 2003. P. 287-321.
14. Odintsov S.P, Wansing H. Inconsistency-Tolerant Description Logic. Part II: A Tableau Algorithm for $CALC^{C}$, Journal of Applied Logic. 2008. V. 6. P. 343-360.
15. S.P. Odintsov, H. Wansing. Modal logics with Belnapian truth values. Journal of Applied Non-Classical Logics. 2010. V. 20. P. 279-301.
16. Ritchie D.M. The Development of the C Language. The second ACM SIGPLAN History of Programming Languages Conference (HOPL-II), ACM, 1993. P. 201-208.
17. Shilov N.V., Akinin A.A., Zubkov A.V., and Idrisov R.I., Development of the Computer Language Classification Portal // Lecture Notes in Computer Science. 2012. Vol.7162. P.340-348.

18. Staab S. and Studer R. (editors) Handbook on Ontologies. International Handbooks on Information Systems. Springer, 2nd edition, 2009.
19. Tudorache T., Nyulas C. I., Musen M. A., and Noy N. F. WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web // Semantic Web Journal. 2011. V. 11, n.16. P. 1-11.
20. van Roy P. Programming Paradigms for Dummies: What Every Programmer Should Know // In: New Computational Paradigms for Computer Music. IRCAM/Delatour, France. 2009. P. 9-38.
21. Walther Ch., Schweitzer St. About VeriFun // Lect. Not. in Comp. Sci. 2003. V. 2741. P. 322-327.
22. Ontology. A Resource Guide for Philosophers // [Электронный ресурс]. URL: http://www.formalontology.it/ (дата обращения: 19.11.2012).
23. Ruby. A Programmer's best friend // [Электронный ресурс]. URL: http://www.ruby-lang.org/en/about/ (дата обращения: 19.11.2012).

UDK: 168.2 + 025.4 + 519.682 + 004.43 + 811.93

Title: Ontology approach to classification of Computer Languages: current state and challenges

Author(s):

Renat I. Idrisov (A.P. Ershov Institute of Informatics Systems),

Sergey P. Odintsov (S.L. Sobolev Institute of Mathematics),

Nikolay V. Shilov (A.P. Ershov Institute of Informatics Systems, Novosibirsk State University, Novosibirsk State Technical University)

Abstract: During the semicentennial history of Computer Science and Information Technologies, several thousands of computer languages have been created. The computer language universe includes languages for different purposes (programming, specification, modeling, etc.). In each of these branches of computer languages it is possible to track several approaches (imperative, declarative, object-oriented, etc.), disciplines of processing (sequential, non-deterministic, distributed, etc.), and formalized models, such as Turing machines or logic inference machines. The listed arguments justify the importance of of an adequate classification for computer languages. Computer language paradigms are the basis for the classification of the computer languages. They are based on joint attributes which allow us to differentiate branches in the computer language universe. We present our computer-aided approach to the problem of computer language classification and paradigm identification. The basic idea consists in the development of a specialized knowledge portal for automatic search and updating, providing free access to

information about computer languages. The primary aims of our project are the research of the ontology of computer languages and assistance in the search for appropriate languages for computer system designers and developers. The paper presents our vision of the classification problem, basic ideas of our approach to the problem, current state and challenges of the project, and design of query language.

Keywords: computer languages, computer paradigm, classification, knowledge portal, description logic, paraconsistency, inconsistency-tolerance, model checking