НАУЧНО • ТЕХНИЧЕСКАЯ ИНФОРМАЦИЯ

Серия 2. ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ И СИСТЕМЫ
ЕЖЕМЕСЯЧНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ СБОРНИК

Издается с 1961 г.

№ 3

Москва 2014

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

УДК 004.89

М.И. Забежайло, Е.В. Синякова

К вопросу об интеллектуальности интеллектуального анализа данных

Выделены некоторые функциональные особенности программных инструментов компьютерного анализа данных, позволяющие отнести соответствующие программные средства такого типа к классу интеллектуальных систем (в традиционном для исследований в области искусственного интеллекта понимании этого термина). Обсуждаемые возможности систем интеллектуального анализа данных (ИАД) демонстрируются на примере формализации и решения средствами ДСМ-метода автоматического порождения гипотез задачи прогнозирования ресурсоемкости проектов конструирования и производства сложных программно-технических систем климатического оборудования для телекоммуникационной индустрии. Демонстрируются базовые дескриптивные и аргументационные возможности ДСМ-метода, позволяющие использовать его в качестве платформы ИАД.

Ключевые слова: интеллектуальный анализ данных, прогнозирование и принятие решений в условиях неполноты информации, ДСМ-метод автоматического порождения гипотез

Интеллектуальный анализ данных (ИАД) сегодня – это интенсивно развивающееся направление computer science & engineering, характеризующееся как разнообразными математическими моделями (в том числе – алгоритмикой), так и достаточно разнообразным арсеналом ориентированных на использова-

ние в промышленных масштабах прикладных программных систем.

Наиболее распространенным взглядом на обсуждаемую проблематику в настоящий момент является представление об ИАД как об области приложения мощных компьютерных программ – проблемно-

ориентированных «считалок» для поиска зависимостей, в неявном виде присутствующих в больших объемах эмпирических данных*. В большинстве случаев речь здесь идет о возможности анализа данных в ситуациях, где исследователя (в силу присущих ему естественных ограничений по скорости и объемам обрабатываемой информации) попросту заменяет компьютерная технология – программно-технический комплекс ИАД. (Прекрасный образец индустриальной инструментальной системы ИАД предлагает, например, компания SAS Institute – один из лидеров соответствующего сегмента рынка информационных технологий (ИТ) – см., например, [1]). Итак, достаточно часто инструментальные средства ИАД – это проблемно-ориентированная библиотека воплощенных в соответствующее программное обеспечение моделей и алгоритмов поиска зависимостей в накопленных данных, расширяющая (вычислительные!) возможности использующего ее исследователя в работе с большими объемами анализируемых данных.

Однако представляется уместным все же задаться вопросом: а где же здесь, собственно, «интеллект»? Речь ведь идет об интеллектуальном анализе данных!

В последние годы термин *интеллектуальный* становится все более часто встречающимся в связи с приложениями новых информационных технологий. Обилие в нашем сегодняшнем окружении именуемых† интеллектуальными устройств – интеллектуальных датчиков, интеллектуальных интерфейсов, интеллектуальных вычислителей, интеллектуальных сетевых middle-box'ов и т.п. – вынуждает задуматься о корректности использования термина *интеллектуальный*: со времен опубликования известного теста Тьюринга (см. например, [2]) в области computer science интеллектуальные решения (интеллектуальные системы) предполагают компьютерное моделирование тех или иных интеллектуальных функций специалиста (т.е. не просто расчеты, а в той или иной мере определенную «игру ума»).

В таком контексте, даже если не рассматривать текущую практику терминологических различий англоязычных и русскоязычных текстов по обсуждаемой проблематике‡, представляется полезным уточнить: в чем же, собственно, *интеллектуальность*, обсуждаемого (и активно используемого!) интеллектуального анализа данных? Подобное уточнение могло бы оказаться полезным для корректного отделения собственно интеллектуальных (в частности, в соответствии с тестом Тьюринга) компьютерных систем от, в том числе, полезных и важных (как для развития фундаментальных оснований, так и для использования при решении прикладных задач) систем и решений, применение наименования *интел-*

лектуальные к которым представляется, вообще говоря, некорректным (да, в общем-то, и ненужным: смена названия вряд ли сможет повлиять, например, на их функциональные возможности, удобство эксплуатации и т.п.).

Вслед за авторами [3, 4] мы считаем целесообразным развивать предложенный еще У.Р.Эшби [5] взгляд на интеллектуальные системы как на компьютерные инструменты (системы и решения), представляющие собой своего рода «усилители» интеллектуальной активности (познавательных возможностей) исследователя (специалиста).

Для собственно интеллектуального анализа данных обсуждаемый подход представляется интересным, в частности, в связи со следующими двумя группами функциональных возможностей.

1. Одна из важных особенностей реализуемого в рамках ИАД исследователем-экспертом процесса поиска зависимостей в данных – это возможность породить зависимости различных типов, сравнивая между собой соответствующие результаты экспертизы и, если можно, детально анализируя различия в них.

2. Одним из критически важных вариантов обсуждаемого анализа оказывается ситуация, когда одно из средств экспертизы дает результат одного типа (например, – позитивный), а другое – результат другого типа (например, негативный), однако, в текущем анализе в распоряжении исследователя может попросту не быть средств, чтобы предпочесть один из используемых инструментариев проведенной ИАД-экспертизы другому.

Именно по такой линии расширения «библиотеки» программных инструментов компьютерного анализа данных и автоматического порождения скрытых в них зависимостей развиваются сегодня продуктовые линейки лидеров рынка ИАД-решений (см. например, уже упоминавшийся выше инструментарий компании SAS Institute и др.). Подобный подход, разумеется, может рассматриваться как уже предложенный выше вариант расширения операционных возможностей ведущего ИАД-исследования специалиста на недоступные ему в ручном режиме объемы и скорости анализа данных (например, в области так называемых Big Data).

Тем не менее, несложно предложить и более «продвинутое» уточнение обсуждаемого термина *интеллектуальный* в рассматриваемом здесь контексте. Действительно, если продолжать использование возможных аналогий между «тактикой» выполнения ИАД-исследований специалистом и функциональными характеристиками компьютерной ИАД-системы, то можно было бы обратить внимание на характерные для эксперта возможности целенаправленного поиска подходящего решения в однотипных (по укрупненной схеме принятия решений), но, вообще говоря, различных (например, за счет выбора различных средств представления исходных обрабатываемых эмпирических данных§ и т.п.) формализованных моделей. В та-

* Один из наиболее популярных сегодня примеров подобных приложений – это область анализа так называемых Big Data.

† При ближайшем рассмотрении – всего лишь, фактически, за использование в их конструктивном решении микропроцессоров и/или некоторого набора программируемых функций.

‡ Например, для семейства, вообще говоря, существенно различных англоязычных терминов *business intelligence, intellectual, smart, intelligent* в русскоязычной версии мы зачастую вынуждены использовать лишь словосочетания с неизменным участием одного термина *интеллектуальный*.

§ Например, при работе с содержательными (неформализованными) исходными данными – эмпирической информацией – эксперт вынужден «погружать» эти данные в то или иное формализованное представление: в процессе формализации

кой ситуации эксперт обычно стремится использовать доступные ему возможности «настройки» задействованных им средств ИАД-экспертизы на содержательные особенности анализируемой предметной области и решаемой задачи, т.е. работает по классической схеме:

Задача в прикладной предметной области =>
=> формализованная модель =>
=> алгоритм =>
=> программная система (обеспечивающая решение выбранной прикладной задачи) =>
=> содержательная интерпретация полученных результатов.

В условиях неполноты информации об объекте исследования представляется желательным оперировать не единичными экземплярами, а классами соответствующих объектов – моделей, алгоритмов, программ. При этом в каждом таком классе желательно иметь соответствующие средства «навигации» - настройки инструментария ИАД на критически важные особенности решаемой прикладной задачи, т.е. фактически следует вести речь о модификации классической схемы к виду:

Задача в прикладной предметной области =>
=> класс взаимно согласованных формализованных моделей =>
=> семейство однотипных алгоритмов =>
=> комплекс программных систем (обеспечивающих решение выбранной прикладной задачи) =>
=> содержательная интерпретация полученных результатов.

Таким образом, при создании интеллектуальной системы анализа данных в каждом из трех названных классов взаимно-согласованных конструкций необходимо организовать управляемую (т.е. позволяющую оптимизировать качество получаемых содержательных результатов) «навигацию»: подстройку моделей, алгоритмов и программ с уже названной целью – улучшить конечные результаты проводимого ИАД.

При работе с такими классами объектов хорошо известны, по крайней мере, две «предельные» ситуации:

а) в соответствующем классе имеется всего один элемент заданного типа, реализуемый используемыми программными инструментами (ситуация с обычными ИАД-toolbox'ами);

б) в рассматриваемом классе все модели соответствующего типа обрабатываются одним способом (алгоритмом). Примеры: fuzzy sets (см. например, [6]) или же интервальные вычисления (см. например, [7]).

Предметом нашего интереса является «промежуточный вариант» обозначенной ситуации: возможность реализовать в соответствующем ИАД-

инструментарии направленный выбор нескольких моделей из класса, т.е. выбор, позволяющий «настроиться» на наиболее приемлемый результат (надежный, адекватно интерпретируемый, ...).

Дополнительно следует заметить, что в случае проблем с вычислительной сложностью** соответствующих алгоритмов ИАД для применения в реальных индустриальных приложениях, полезно иметь соответствующие возможности:

- быстро выделять эффективно разрешимые подклассы решаемых переборных задач восстановления зависимости,
- реализовать в том или иной виде формализованные приближенные вычисления.

В области информационных систем управления предприятием одной из критически важных сфер эффективного промышленного применения обсуждаемых нами инструментальных систем ИАД (т.е. систем – усилителей интеллектуальных возможностей специалиста) стали задачи поддержки принятия решений. В отличие от задач регистрационного типа – автоматизации бухгалтерских функций, бюджетирования и т.п. – эта область приложения информационных технологий имеет свои специфические особенности: здесь содержательные проблемы, требующие анализа и принятия разумных решений, как правило, плохо формализованы, отличаются необходимостью оперировать с данными сложной комплексной структуры, характеризуются неполнотой информации об объекте анализа и др.

В такой проблематике очевидным образом проявляются отличительные особенности (своего рода дифференциальные признаки, отличающие такие системы от «обычных» вычислительных инструментов компьютерного анализа данных) обсуждаемых нами систем – «усилителей» возможностей эксперта при решении им задач ИАД:

- выбор адекватных средств представления данных об объекте анализа (возможности варьирования типов данных в рамках одного и того же класса алгоритмов интеллектуального анализа данных), например: множеств признаков, кортежей, графов, числовых векторов, ...;
- настройка локальных средств анализа эмпирических данных в изучаемой предметной области на работу с соответствующими типами данных (например, выбор соответствующей алгебраической операции сходства при формализации сходства анализируемых объектов – множеств признаков, кортежей, графов, числовых векторов, ...);
- формулирование классов однотипных формализованных моделей анализа данных, позволяющих обеспечить варьирование инструментов принятия решений для их «настройки» на особенности данных в изучаемой предметной области (в том числе – варьирование как конкретных математических

постановки решаемой задачи анализируемые им объекты могут быть, в частности, представлены как множества признаков (множества), множества признаков с отношениями на них (кортежи, графы, последовательности, цепочки, ...), векторы числовых значений параметров (числовые векторы) и т.п.

** Как известно, ряд возникающих здесь комбинаторных задач являются доказуемо трудно-решаемыми (например, принадлежат к классам NPC, #PC и др.), что делает исчерпывающий поиск их решения в ряде приложений практически неосуществимым даже средствами современных суперкомпьютерных вычислительных комплексов.

средств поддержки принятия решений, так и «подстройки» соответствующей алгоритмики);

- построение содержательной интерпретации результатов, получаемых применением используемых формализмов (в том числе – содержательной интерпретации порождаемых зависимостей – каузальных, ассоциативных и т.п.), а также оценки достаточности оснований для принятия полученных содержательных результатов (их обоснованности, корректной интерпретируемости и т.п.) при неудовлетворительной ситуации в данном случае;

- возможности изучения причин фиксируемой неадекватности, чтобы вернуться к исходным шагам проводимого интеллектуального анализа (выбору формальных средств представления данных и т.д. – см. выше) и провести, опираясь на выполненную «работу над ошибками», новую – более адекватную – «настройку» используемых решающих средств ИАД на критически важные особенности изучаемой предметной области.

По-видимому, только если конкретный инструментарий (математические модели вместе с реализующими их алгоритмикой и программными системами) позволяет реализовать приведенные выше требования, его (этот инструментарий) *обоснованно* можно считать средством действительно *интеллектуального* анализа данных.

Теперь мы располагаем минимальной необходимой понятийной базой, чтобы продемонстрировать (на примере решения конкретной прикладной задачи анализа данных и принятия решений по управлению бизнесом) а затем – обсудить возможности ДСМ-метода автоматического порождения гипотез (см., например, [3, 4]) как платформы – *понятийной, методологической, дескриптивной и аргументационной, алгоритмической* – интеллектуального анализа данных. Названные возможности мы рассмотрим на примере применения ДСМ-метода для решения задачи прогнозирования ресурсоемкости проектов производства сложных высокотехнологичных изделий.

Рассматриваемая прикладная задача относится к специальному классу реальных прикладных задач – проблематике быстрого прогнозирования оценочных характеристик объектов в проектах производства сложных высокотехнологичных изделий на основе неполной информации о разрабатываемых изделиях, т. е. до выполнения (и завершения) их детального проектирования.

Обсуждаемая задача представляет собой один из востребованных типов задач поддержки принятия решения в области информационных систем управления предприятием. Актуальность рассматриваемой проблематики обусловлена тем, что ситуации, в которых возникает потребность в быстром оценивании соответствующих характеристик и всего разрабатываемого изделия (а именно – ситуации, когда срок принятия решения о запуске или же отказе от запуска соответствующего проекта оказывается заведомо меньше штатных сроков *выполнения полного проектирования* целевого технического объекта) достаточно распространены в производственной деятельности современных высокотехнологичных предприятий. Примерами таких ситуаций могут быть, в частности,

необходимость «быстрого» заключения договора с заказчиком, участие в различных конкурсных процедурах (разумеется, ограниченных в своей длительности по времени лишь требованиями заказчика) и т.д.

В рассматриваемой прикладной задаче техническим изделием, характеристики проектов производства которого необходимо прогнозировать, является специальный тип инженерного оборудования – климатические шкафы, используемые в телекоммуникационной индустрии. Характеристика, которую необходимо прогнозировать, – это класс длительности производства данного оборудования. Известной на момент принятия решения неполной информацией об объекте, на основе которой может быть осуществлён прогноз, является технический проект изделия (комплект проектной документации), отражающий внутреннюю структуру («архитектуру») изделия. Дополнительными исходными данными является накапливаемое множество фактов, представляющее собой данные о прецедентах производства подобных технических изделий за определённый срок.

Основные особенности решаемой прикладной задачи, характерные в общем случае для задач поддержки принятия решения в рассматриваемой области, это: необходимость оперировать техническими объектами сложной структуры, наличие неполной информации о структуре объекта на этапе принятия решения о запуске проекта, малая и, как правило, статистически незначимая выборка уже завершённых проектов, а также наличие скрытых (неявно выраженных в описании соответствующих прецедентов) причинно-следственных зависимостей в накапливаемом множестве фактов.

Представленные выше особенности рассматриваемой прикладной задачи позволяют рассчитывать, что наиболее перспективными для ее решения окажутся методы, применяющие для построения прогнозных моделей технику машинного обучения на прецедентах. Такими методами могут быть логические методы классификации (например, решающие деревья [8]), нейросетевые методы классификации (например, многослойные нейронные сети, сети Кохонена [9]), линейные методы классификации (например, метод опорных векторов [10, 11], логистическая регрессия [12]), метрические методы классификации (например, метод ближайших соседей и его обобщения [13]), а также ДСМ-метод автоматического порождения гипотез [3, 4]. Выбор наиболее подходящего метода для решения рассматриваемой прикладной задачи основан на анализе выполнимости следующих условий:

- 1) наличие корректной методологии решения поставленной задачи прогнозирования (в частности, надежных критериев достаточности оснований для принятия полученных результатов вычислений/прогнозирования);
- 2) возможность интерпретации результатов прогнозирования в содержательных терминах решаемой прикладной задачи;
- 3) возможность выявлять закономерности, в том числе и на небольших (статистически незначимых) обучающих выборках (в том числе – при обработке исходных данных со сложной структурой, например,

множества признаков с отношениями на них – графах, цепочках и т.п).

Анализ способов построения прогнозных моделей каждого из названных выше подходов показал, что все рассматриваемые методы, за исключением ДСМ-метода автоматического порождения гипотез, либо не обеспечивают выполнимость условия интерпретируемости результатов (см., например, нейросетевые методы классификации, линейные методы классификации, метрические методы классификации), либо сталкиваются с проблемой переобучения при уменьшении размера обучающей выборки (решающие деревья). По результатам проведенного сравнительного анализа оказалось, что из перечисленных выше полностью соответствует критериям отбора лишь ДСМ-метод автоматического порождения гипотез (АПГ). Основными достоинствами ДСМ-метода АПГ, повлиявшими на выбор данного инструментария в качестве средства решения рассматриваемой прикладной задачи, оказались возможности:

- учёта индивидуальных особенностей изучаемых объектов исследования;
- порождения гипотез на малых массивах данных, благодаря выявленному сходству объектов, характеризующих существенными параметрами;
- работы с открытыми (пополняемыми) массивами данных, распознавая необходимость управляемого расширения БД, если таковая возникает;
- порождения полученных правил классификации (причинных зависимостей) в явном виде, что обеспечивает прозрачную интерпретацию результатов обучения.

Для использования ДСМ-метода при решении обсуждаемой прикладной задачи прежде всего были выполнены настройки стандартных средств ДСМ-метода на особенности выбранной предметной области. Программная реализация разработанных алгоритмов была оформлена в виде законченного инструментального программного средства.

Одними из основных этапов решения задачи настройки стандартных средств ДСМ-метода на особенности выбранной предметной области являются формализация представления используемых эмпирических данных (структурных описаний соответствующих изделий – анализируемых объектов), а также формализация понятия сходства таких объектов в виде соответствующей алгебраической операции, где (следуя стандартной схеме применения ДСМ-метода):

I. Множеству объектов X поставлено в соответствие множество структурных схем (проектов) P изделий рассматриваемой прикладной задачи. Формализованное представление объектов (а именно структурных схем технических изделий) отражает множество элементов (подструктур стандартного вида) номенклатуры, входящих в состав рассматриваемого объекта, а также их количество. Так как возможно кратное вхождение элементов номенклатуры в состав изделия, формализованное описание объекта представляет собой нумерованное множество. Для представления объекта (нумерованного множества) стандартными теоретико-множественными средства-

ми было введено следующее формальное взаимно-однозначное соответствие:

O – множество всех объектов номенклатуры (множество образующих);

$$O^* = \{o_{i,n}\}, \quad (\text{где } i = 1, \dots, |O|, n \in N, N -$$

множество натуральных чисел) – множество, соответствующее расширению используемого множества образующих O для кодировки кратных вхождений элементов в нумерованное множество.

Таким образом, $P_i \subseteq O^*$.

II. Множеству признаков V -объекта поставлено в соответствие множество фрагментов структурных схем $F (F_i \subseteq O^*)$ изделий рассматриваемой прикладной задачи.

III. Множеству свойств W -объектов поставлено в соответствие множество классов длительности проектов $C : C = \{c_1, c_2, c_3, c_4, c_5\}$, где каждому свойству $c_i \in C$ соответствует упорядоченная пара $\{t_{\min}, t_{\max}\}$, соответствующая интервалу длительности $[t_{\min} \dots t_{\max}]$ производства изделий.

«Настройки» ДСМ-инструментария, подбираемые в ходе итераций (последовательно проводившихся при решении рассматриваемой прикладной задачи) ИАД-экспериментов, затронули в первую очередь используемое формальное определение сходства:

– в ситуации, когда объект рассматривается как множество типовых фрагментов его структуры (составных блоков), т.е. когда задействованный тип данных – множества, используемое сходство (рассматриваемых) объектов – теоретико-множественная операция пересечения $\cap$,

– в ситуации, когда объект рассматривается как множество типовых фрагментов его структуры (составных блоков) с дополнительным учетом кратности вхождения соответствующих составных блоков в структуру описываемого объекта (т.е. когда задействованный тип данных – нумерованные множества), уместно провести расширение исходного алфавита U по следующей схеме: каждой встречающейся кратности k вхождения составного блока S_i в описание текущего объекта A_j сопоставляется (в расширенном алфавите U^*) новая образующая S_{ik} . Легко видеть, что расширенный алфавит U^* для каждого конкретного множества примеров Ω восстанавливается достаточно легко. Далее: сходство объектов, представленных в расширенном алфавите $U^*(\Omega)$, вновь будет определяться как теоретико-множественная операция $\cap$ пересечения, что будет эквивалентно выделению для пары сходных объектов всех общих составных блоков S_{ik} совпадающей кратности.

Итак, в рассматриваемой ситуации отражение структурной «природы» обрабатываемых объектов в используемом языке представления данных об объектах (а именно – описанное представление их стандартными теоретико-множественными средствами) позволило определить сходство двух объектов как

сходство нумерованных множеств, формализуемое операцией теоретико-множественного пересечения^{††}.

Правила правдоподобного вывода (ППВ) ДСМ-метода при работе с объектами в обсуждаемой прикладной задаче выбирались сначала в стандартной конфигурации (см., например, [3, 4]), а в ситуациях, когда применение обычных ППВ не давало адекватно интерпретируемых результатов – расширялись дополнительными логическими условиями, отражающими некоторые специальные метрические соотношения (количества соответствующих случаев).

Следуя далее стандартной схеме применения ДСМ-метода, пришлось определиться с представлением в базе фактов позитивных примеров, негативных примеров, а также примеров неопределённости изучаемого свойства. В самом общем виде здесь множество позитивных примеров – множество объектов, принадлежащих определённому классу ((+)-примеры); множество негативных примеров – объекты, не принадлежащие определённому классу ((-)-примеры); множество недоопределённых примеров – объекты, класс длительности создания которых не определён.

Для рассматриваемой прикладной задачи выделение объектов, принадлежащих определённому классу, и объектов, класс которых не определён, не вызывает трудностей. Наличие пяти классов длительности ($C = \{c_1, c_2, c_3, c_4, c_5\}$), на которые подразделяются проекты выполненных изделий, накапливаемых во множестве фактов ДСМ-обучения, автоматически обеспечивает наличие позитивных примеров для каждого из классов длительности. Примерам неопределённости изучаемого явления соответствуют проекты ещё не выполненных изделий, для которых необходимо прогнозировать соответствующий класс длительности.

Однако выделение объектов, не принадлежащих определённому классу ((-)-примеров класса или контрпримеров), вызвало определенные затруднения. Оказалось, что можно предложить различные варианты определения контрпримеров, которые в общем случае могут иметь различное влияние на интерпретируемость получаемых результатов, и прежде всего – на содержательную интерпретируемость в терминах решаемой прикладной задачи выявленных гипотез (т.е. «реконструируемых» ДСМ-методом причин принадлежности/непринадлежности объектов к определённому классу длительности). Так, в частности, (-)-примерами определённого класса длительности можно считать проекты-представители классов длительности, отличных от рассматриваемого, или же проекты-представители классов длительности, отличных от рассматриваемого и относящихся к наименьшим (наибольшим) классам длительности по сравнению с рассматриваемым классом.

Для выбора наилучшего из возможных вариантов определения понятия *контрпример*, наиболее подхо-

дящего с точки зрения возможностей содержательной интерпретации полученных гипотез, был произведён ряд вычислительных экспериментов.

Так, при заданных пяти классах длительности анализируемых проектов $T_1, T_2, \dots, T_5$ и уточнении (для каждого класса T_i) множества примеров – как элементов этого класса T_i , а множества контрпримеров – как множества $\{T_1, T_2, \dots, T_5\} / T_i$, критически важным «дефектом» при порождении ДСМ-результатов оказались «межклассовые» сходства для классов слева и справа от T_i . (Разумной содержательной интерпретации таким сходствам предложить не удалось).

При отнесении к множеству примеров вместе с текущим T_i также всех классов меньшей длительности, а к множеству контрпримеров – оставшихся классов из множества $\{T_1, T_2, \dots, T_5\}$ объёмы не имеющих адекватной содержательной интерпретации ДСМ-гипотез хоть и сократились, однако, как и ранее, не удалось полностью избежать соответствующих «смысловых дефектов».

И только при явном запрете межклассовых сходств по схеме: внутри каждого класса сходство выражает операция пересечения $\cap$, а при применении операции сходства к представителям разных классов результат по определению полагается равным $\emptyset$, удалось (применением стандартных ДСМ-ППВ) сформировать полностью интерпретируемые результаты (обзор которых и представлен ниже).

В рамках проводимых экспериментов был реализован простейший вариант схемы ДСМ-ИАД (интеллектуального анализа данных средствами ДСМ-метода). Основными задачами, решёнными при реализации данной схемы, были:

- выбор варианта определения понятия *контрпример* определённого класса длительности. Так, в качестве первого варианта было принято решение рассматривать в качестве (-)-примеров фиксированного класса длительности все проекты-представители классов длительности, отличных от рассматриваемого;

- уточнение синтаксиса формулировки основных понятий ДСМ-метода, а именно синтаксиса формулировки решающих предикатов и правил правдоподобного вывода первого и второго рода (возможность использования стандартных формулировок, представленных при формализации метода сходства, лежащего в основе используемого для решения поставленной прикладной задачи ДСМ-метода АПГ без внесения модификаций, оказалась ограниченной узким кругом рассматриваемых здесь задач);

- выполнение модификации логических и алгоритмических средств ДСМ-метода в соответствии с особенностями прикладной задачи. Так была осуществлена настройка алгоритмики ДСМ-метода (а именно алгоритма порождения гипотез и алгоритма классификации), учитывающая содержательные особенности предметной области, отражённые в частности при уточнении синтаксиса формулировки основных понятий ДСМ-метода. Кроме того, была осуществлена разработка модифицированной версии быстрого алгоритма поиска глобальных сходств «Замыкай по одному [14]», нивелирующей недостатки

^{††} Более точно: нумерованные множества представлены обычными множествами (каждая возможная кратность вхождения конкретного элемента в нумерованное множество кодируется отдельным элементом обычного множества), сходство которых и задается теоретико-множественной операцией пересечения.

данного алгоритма и обеспечивающей возможности его применения в рамках ДСМ-метода;

- выполнение программной реализации разработанных алгоритмов в виде законченного программного средства.

Большинство гипотез, полученных с помощью разработанного начального варианта схемы ДСМ-ИАД в выбранной прикладной предметной области, было необоснованными и плохо интерпретируемыми: наблюдалось появление ошибочных (с точки зрения содержательной интерпретации в терминах исследуемой прикладной предметной области) (0)-гипотез (противоречивых гипотез, являющихся как причинами принадлежности, так и причинами непринадлежности объектов к определённому классу длительности) во всех классах длительности и ошибочных (–)-гипотез (гипотез, являющихся причинами непринадлежности объектов к определённому классу длительности) в старших классах длительности. Данные ошибочные (0)- и (–)-гипотезы должны были являться только (+)-гипотезами (гипотезами, отражающими причины принадлежности объектов к определённому классу длительности) для младших классов длительности. Выявленные трудности содержательной интерпретации полученных гипотез повлекли за собой также недостаточную обоснованность результатов классификации, полученных на основе применения данных гипотез («причинных» зависимостей).

Одной из особенностей порождения гипотез при поиске глобальных сходств объектов различных классов длительности оказалось выявление в старших классах сходств (т.е. подструктур изделий), характерных для младших классов. Этот факт, при использовании выбранного в данном варианте схемы ДСМ-ИАД способа определения понятий *пример* и *контрпример*, оказался причиной «определения» большей части (+)-гипотез младших классов (0)-гипотезами, а также появления в старших классах необоснованных (0)- и (–)-гипотез.

Для устранения влияния выявленных псевдопричин (в частности устранения возможности появления в старших классах необоснованных (0)- и (–)-гипотез) была предпринята попытка модификации используемых решающих средств, а именно модификации определения понятий «(+)-пример» и «(–)-пример» какого-либо класса длительности. При проведении второго эксперимента в качестве (+)-примеров фиксированного класса длительности были выбраны все проекты, длительность производства изделий по которым не превосходила длительность производства, характерную для рассматриваемого фиксированного класса. В качестве (–)-примеров фиксированного класса длительности были выбраны все проекты, длительность производства изделий по которым превосходила длительность производства изделий, характерную для рассматриваемого фиксированного класса. Введение данной модификации потребовало также внедрения некоторых изменений в правила правдоподобного вывода второго рода.

Анализ результатов применения модифицированной схемы ДСМ-ИАД в выбранной прикладной предметной области показал, что появилась возмож-

ность устранить один из недостатков, которые были характерны для предыдущего варианта: в старших классах стало возможным избежать появления необоснованных (0)- и (–)-гипотез, характерных для младших классов. Причиной для устранения этого недостатка стало изменение способа выбора (–)-примеров, а именно принадлежность (–)-примеров старшим относительно рассматриваемого классам. Однако для данного варианта реализации метода также оказался характерен ряд существенных недостатков. Например, сохранилось появление ошибочных (0)-гипотез в младших классах, наблюдавшееся в предыдущем варианте, принадлежность (+)-примеров к различным классам повлекла за собой появление в качестве необоснованных (+)-гипотез большого количества межклассовых сходств (т.е. сходств, в образовании которых участвуют по одному проекту-представителю как минимум двух разных классов длительности).

Тем не менее, недостатки приведенного варианта оказались достаточно легко устранимыми. Так, для предотвращения появления ошибочных (0)-гипотез в младших интервалах было решено определить эти гипотезы как (+)-гипотезы слабой силы. Для устранения появления межклассовых сходств в качестве необоснованных гипотез был введён запрет (как пустота по определению соответствующей операции сходства) на рассмотрение сходств, в образовании которых участвуют по одному проекту-представителю как минимум двух разных классов длительности. Кроме того, вариант модификации за счёт устранения понятия (0)-гипотезы и введения ранжирования по силе (+)-гипотез позволил также существенно упростить правила правдоподобного вывода 2-го рода (правила классификации).

Таким образом, в результате проведения ряда «настроечных» экспериментов удалось разработать проблемно-ориентированную ИС, результаты применения которой в выбранной прикладной предметной области (и прежде всего – полученные гипотезы и результаты классификации) являются обоснованными и ясно интерпретируемыми.

Достаточным основанием для принятия индуктивных выводов ДСМ-рассуждения является выполнение аксиомы каузальной полноты, которое требует корректной доопределимости всех объектов исходной базы фактов порожденными из нее (из этой базы фактов) ДСМ-гипотезами применением правил правдоподобного вывода второго рода. Если условие каузальной полноты не выполняется, то исходную базу фактов (примеров и контрпримеров для ДСМ-обучения) следует пополнять новыми примерами.

Результаты классификации, полученные с помощью применения разработанной схемы ИАД-ДСМ, показали, что из 69 проектов, составляющих обучающую выборку, 64 проекта были классифицированы верно, пять проектов классифицировано неверно, неклассифицированных проектов нет. Следовательно, несмотря на то, что была выявлена верно большая часть гипотез о принадлежности/непринадлежности проекта к определённому классу длительности, аксиома каузальной полноты, которая позволяет говорить о том, что индуктивный вывод был сделан на достаточном основании, осталась невыполненной.

(Комментарий: зато степень каузальной полноты 64/69 очень высокая – больше 0,9 с учетом того, что принято считать «хорошим» значение больше 0,8).

Однако анализ гипотез, которыми были неверно классифицированы объекты, позволяет направленным образом с помощью расширения базы фактов улучшать результаты прогноза. Так, для улучшения качества классификации следует формировать недостающие гипотезы с помощью направленного добавления новых проектов в те классы, где нет достаточного объема данных для корректного обучения ДСМ-системы, и понижать ранги некорректных гипотез в остальных классах также с помощью добавления новых фактов (также – вполне определенного вида). Предлагаемое в такой схеме направленное расширение обучающей выборки следует вести до ее «насыщения» – ситуации, когда все имеющиеся в ней (например, на текущий момент ДСМ-рассуждения) эмпирические зависимости о причинах изучаемых явлений полностью (пусть и в неявном виде – через описания соответствующих прецедентов) представлены имеющимися данными для ДСМ-обучения.

Таким образом, с помощью добавления новых фактов для формирования корректных гипотез и понижения ранга некорректных гипотез в нужных классах удалось обеспечить верную классификацию изначально неверно классифицируемых проектов обучающей выборки и, как следствие, добиться выполнимости аксиомы каузальной полноты.

Программная реализация разработанных алгоритмов в виде законченного программного средства обладает рядом интересных характеристик. Использование разработанного быстрого алгоритма поиска глобальных сходств позволяет обеспечить быстрый отклик ИС на запрос (о порождении каузальных зависимостей и прогнозировании свойств новых объектов), а также ее высокую производительность. Применение технологии Java повышает надёжность системы в целом за счёт встроенных в Java средств контроля типов, автоматического сборщика мусора и т.п. Организация независимости от используемой СУБД за счёт использования технологии JPA, наличие хорошо структурированной архитектуры ИС, а также использование объектно-ориентированного подхода предоставляют возможности для гибкого расширения или модификации функционала системы (например, при возникновении новых требований или же при появлении новых ДСМ-правил правдоподобного вывода). Также имеются возможности развития системы при необходимости до корпоративного уровня (т.е. до уровня информационной системы, используемой в качестве штатного инструментального средства при решении на предприятии производственных задач обсуждаемого типа), а также переноса части разработанного функционала системы на мобильные платформы.

* * *

ДСМ-метод автоматического порождения гипотез представляет собой удобную платформу для организации интеллектуального анализа данных. При ис-

пользовании ДСМ-метода в распоряжении исследователя оказываются возможности:

- адекватного представления неформальных моделей рассуждений в стиле индуктивных выводов Д.С.Милля, абдуктивных рассуждений в стиле Ч.С.Пирса, рассуждений по аналогии в стиле Д.Пойа;
- корректной математической формализации перечисленных содержательных типов рассуждений (неформальных семантик рассуждения) специальными логико-математическими средствами;
- варьирования выбираемых типов представления анализируемых данных;
- варьирования алгоритмики реализации ДСМ-правил правдоподобного вывода;
- организации приближенных вычислений при поиске ДСМ-гипотез (организации направленного перебора вариантов, в котором сначала может быть реализован поиск так называемых *полезных*^{**} гипотез, далее – если потребуется – дополненный построением всего множества ДСМ-гипотез, формируемых на заданном наборе примеров и контрпримеров);
- интеграции построенных компьютерных моделей рассуждений со средствами автоматизации достоверных выводов (системами автоматического доказательства теорем), а также вычислительными моделями и программными инструментами.

Важнейший прикладной результат, обеспечиваемый использованием инструментария ДСМ-метода в качестве платформы (т.е. комплекса, служащего основой для различных вычислительных систем) ИАД, – это возможности создавать многофункциональные прикладные программные системы-«усилители» познавательного потенциала специалиста, занятого интеллектуальным анализом данных.

СПИСОК ЛИТЕРАТУРЫ

1. Predictive Analytics and Data Mining. – URL: <https://www.sas.com/technologies/analytics/datamining/>
2. Turing A. Computing Machinery and Intelligence // *Mind*. –1950. October – Vol. LIX, № 236. – P. 433-460.
3. Финн В.К. Индуктивные методы Д.С. Милля в системах искусственного интеллекта // Искусственный интеллект и принятие решений. Часть I. – 2010. – №3. – С. 3-21; Часть II. – 2010. – № 4. – С.
4. Автоматическое порождение гипотез в интеллектуальных системах / под общ. ред. В.К. Финна. – М.: Книжный дом «Либроком», 2009.
5. Эшби У.Р. Введение в кибернетику. – М.: Иностранная литература, 1959. – URL: (<http://pcp.vub.ac.be/ASHBBOOK.html>)
6. Zadeh L. A. Fuzzy sets // *Information and Control*. – 1965. – Vol. 8 (3). – P.338–353.
7. Алефельд Г., Херцбергер Ю. Введение в интервальные вычисления. – М.: Мир, 1987. - 356 с.

^{**} В том числе – полезных для доопределения свойств заданных новых объектов, для проверки выполнимости Условия Каузальной Полноты, содержащих образующие из заданного подмножества исходного алфавита или же «родителей» из заданного подмножества предложенных для ДСМ-обучения примеров и контрпримеров и др.

8. Баргесян А.А., Куприянов М.С., Холод И.И., Тесс М.Д., Елизаров С.И. Анализ данных и процессов: учеб. пособие. 3-е изд., перераб. и доп. – СПб.: БХВ – Петербург, 2009. – 512 с.
9. Головкин В. А. Нейронные сети: обучение, организация и применение. – М.: ИПР- ЖР, 2001.
10. Smola A., Schoelkopf B. A tutorial on support vector regression: Tech. Rep // NeuroCOLT2 NC2-TR-1998-030: Royal Holloway College. – London, 1998.
11. Cortes C., Vapnik V. Support-vector networks // Machine Learning. – 1995. – Vol. 20, № 3. – P. 273–297.
12. David W. Hosmer, Stanley Lemeshow. Applied Logistic Regression, 2nd ed. – New York, Chichester, Wiley. 2002. – 392 p.
13. Айвазян С.А., Бухштабер В.М., Енюков И.С., Мешалкин Л.Д. Прикладная статистика: классификация и снижение размерности. – М.: Финансы и статистика, 1989.
14. Кузнецов С.О. Быстрый алгоритм построения всех пересечений объектов из конечной полу-решётки // Научно-техническая информация. Сер.2. – 1993. – № 1. – С. 17-20.

Материал поступил в редакцию 06.09.13.

Сведения об авторах

СИНЯКОВА Елена Витальевна – магистр факультета вычислительной математики и кибернетики Московского государственного университета им. М.В.Ломоносова
e-mail: sinyakova-ev@yandex.ru

ЗАБЕЖАЙЛО Михаил Иванович – Кандидат физико-математических наук, старший научный сотрудник, Управляющий директор НП «Центр прикладных исследований компьютерных сетей» (Москва, Сколково).
e-mail: zmivan@gmail.com

Е.С. Панкратова, Д.А. Добрынин

Об одном способе выявления неинформативных признаков в интеллектуальных ДСМ-системах по медицине*

Описывается новый способ выявления неинформативных признаков в интеллектуальных ДСМ-системах по медицине. Приводятся результаты компьютерных исследований с исключением неинформативных признаков в различных медицинских системах и результаты сравниваются с предыдущими системами.

Ключевые слова: интеллектуальная система, ДСМ-метод, медицинская диагностика

За последнее десятилетие в секторе интеллектуальных систем ВИНТИ РАН создана серия из шести интеллектуальных систем типа ДСМ¹ (ИнтС ДСМ), применяемых для анализа медицинских данных.

ИнтС ДСМ являются инструментом поддержки медицинских исследований со сложноструктурированными данными и множеством фактов, необозримых без использования компьютерных технологий. Задача системы – помочь экспертам-врачам в принятии решения, когда нужно проанализировать большое количество результатов специальных обследований больного.

ИнтС ДСМ создавались специалистами ВИНТИ совместно с учеными медицинских учреждений для решения семи конкретных медицинских задач:

I. Диагностика двух заболеваний глаз: дегенеративного ретиношизиса и наследственных витреоретинальных дистрофий (Лаборатория клинической физиологии зрения Московского научно-исследовательского института (МНИИ) глазных болезней им. Гельмгольца) [1];

II. Диагностика системной красной волчанки (Отделение нефрологии Городской клинической больницы им. С.П. Боткина) [2];

III. Прогнозирование общей продолжительности жизни на основании анализа клинических и лабораторных данных больных меланомой (Российский онкологический научный центр (РОНЦ) им. Н.Н. Блохина РАМН) [3];

IV. Прогнозирования продолжительности жизни больных меланомой при вакцинации² (РОНЦ им. Н.Н. Блохина РАМН).

V. Прогнозирование времени до возможного рецидива рака молочной железы, а также прогнозирование послеоперационной продолжительности жизни (Лаборатория клинической онкогенетики РОНЦ им. Н.Н. Блохина РАМН);

VI. Прогнозирование осложнений после операций на брюшной полости (рак желудка и толстого кишечника) (РОНЦ им. Н.Н. Блохина РАМН);

VII. Прогнозирование конкретного осложнения SIRS после операции на брюшной полости (РОНЦ им. Н.Н. Блохина РАМН).

ИнтС ДСМ основаны на ДСМ-методе автоматического порождения гипотез [4], реализующем автоматизированные правдоподобные рассуждения (порождение гипотез о причинах свойств, вывод по аналогии, процедуры объяснения начального состояния базы данных).

Для осуществления возможности применения ДСМ-метода к решению перечисленных задач проводилась длительная совместная работа с экспертами-врачами. Врачи перечисляли все возможные признаки описания состояния больного или описания конкретного анализа больного, по которым надо прогнозировать: наличие или отсутствие конкретного заболевания; время до возможного рецидива; возможность осложнений. Задача врача – предоставить достаточное количество примеров положительного и отрицательного типа в терминологии ДСМ-метода, а также не пропустить значимых для диагноза признаков. Задача разработчиков системы – структурировать эти признаки таким образом, чтобы к ним было возможно применять операцию сходства и определять отношение вложения.

Первым этапом совместной работы было создание подсистемы представления знаний и базы фактов (БФ) – одной из составных частей ИнтС ДСМ. Атрибутами этой БФ являются сведения о пациентах, имеющиеся в медицинских картах и описанные языком представления данных. В соответствии с конкретными задачами эти атрибуты различны. Сущест-

* Работа выполнена при поддержке РФФИ (проект № 11-07-00618а) и программы фундаментальных исследований Президиума РАН (П15, проект № 209)..

¹ Название в честь английского мыслителя Джона Стюарта Милля

² Получено свидетельство о регистрации программного продукта.

вуют шесть различных наборов атрибутов – вариантов заполнения БФ – с присоединенным общим ДСМ-решателем. Для медицины создано шесть вариантов интеллектуальных ДСМ-систем.

Для возможности применения ИнтС ДСМ к конкретной задаче необходима настройка на предметную область [4], включающая:

- разработку языка представления данных;
- определение понятия «объект» и «свойство» в терминологии ДСМ-метода;
- задание операции сходства;
- задание отношения вложения;
- определение аксиом предметной области.

После выполнения всех требований настройки на предметную область врачи-эксперты заполняют БФ. Далее осуществляется настройка системы на экспериментальное исследование [4] по следующим параметрам:

1. Выбор стратегии: простой метод сходства, метод сходства с запретом на контрпримеры (невложение полученных методом сходства гипотез в исходные примеры противоположного знака) отдельно для (+)- и (-)-примеров.

2. Подбор нужного количества родителей (примеров, формирующих гипотезу).

3. Настройка фильтра конъюнктивных признаков.

4. Настройка фильтра дизъюнктивных признаков.

Возможный критерий оценки подбора параметров и стратегии эксперимента – применение процедуры «доопределение по одному»: последовательно каждому объекту представленного врачами массива данных присваивается значение «т» (недоопределенность), далее производится доопределение этого объекта средствами ДСМ-системы с выбранными параметрами и сравнивается доопределенное значение с существующим; подсчитывается общее количество правильных и неправильных доопределений; в соответствии с пп.1-4 выбираются параметры, при которых при применении процедуры «доопределение по одному» может быть получен наилучший результат, т.е. наибольшее количество правильных доопределений и наименьшее количество неправильных. Затем подсчитывается общее количество правильных и неправильных доопределений.

Эти параметры настройки логично использовать для доопределений вновь заносимых в базу фактов состоянии пациентов, которым необходимо поставить диагноз или сделать соответствующий прогноз.

В экспериментальных исследованиях имелись случаи неправильных и противоречивых доопределений.

Результаты работы ИнтС ДСМ для каждой из перечисленных задач приведем в табл. 1.

Перед нами встала задача: как увеличить число правильных доопределений и уменьшить число неправильных. Анализируя гипотезы, участвующие в правильных и неправильных доопределениях, мы предложили следующую модификацию исходных массивов данных.

Из всего множества порожденных гипотез G выделим подмножество гипотез T ($T \subset G$), которые участвуют в правильных доопределениях («правильные

гипотезы»), и подмножество гипотез F ($F \subset G$), которые участвуют в неправильных доопределениях («неправильные гипотезы»):

$T = \{hyp'_1, \dots, hyp'_i, \dots, hyp'_r\}$, $0 < i < R$, где R – количество гипотез, участвующих в правильных доопределениях.

$F = \{hyp''_1, \dots, hyp''_j, \dots, hyp''_s\}$, $0 < j < S$, где S – количество гипотез, участвующих в неправильных доопределениях или противоречивых доопределениях примеров противоположного знака.

Представим каждую гипотезу как множество входящих в нее атрибутов:

$hyp = \{at_k | 0 < k \leq 1, 1 \leq j \leq N, \}$, где $N = |hyp|$ – общее количество атрибутов в конкретной задаче.

$$X = \bigcup_{i=1}^r hyp'_i$$

$$Y = \bigcup_{j=1}^s hyp''_j$$

$Y \setminus X = Z$ множество признаков, которые участвуют в «неправильных» гипотезах, и не участвуют в «правильных». Наше предложение заключается в отнесении этих признаков к неинформативным и в дальнейших экспериментальных исследованиях мы исключаем их из объектов.

Поиск неинформативных признаков выполняется по следующему алгоритму.

(1) Проводится доопределение по одному, в ходе которого для каждого признака, входящего в левую часть гипотез, накапливается информация о том, в какие виды гипотез он попал. Возможные варианты:

- признак входит в гипотезы, которые определяют правильные доопределения (положительные «+» → «+» или отрицательные «-» → «-»);
- признак входит в гипотезы, которые определяют неправильные доопределения (положительные переходят в отрицательные «+» → «-», или отрицательные переходят в положительные «-» → «+»).

(2) Ищутся неинформативные признаки, которые обладают следующими свойствами: они входят в группу неправильных доопределений и не входят в группу правильных, т.е. признак может принадлежать только группам «+» → «-» или «-» → «+».

(3) Найденные неинформативные признаки исключаются из левой части гипотез и процесс повторяется с первого шага до тех пор, пока не будут исключены все неинформативные признаки.

Результаты работы ИнтС ДСМ без неинформативных признаков для каждой из перечисленных задач приведем в табл. 2.

Анализируя полученные результаты, сделаем вывод: что исключение из объектов неинформативных признаков в некоторых случаях приводит к улучшению результатов, т.е. повышается точность или полнота предсказаний.

Рассмотрим подробнее, как меняются доопределения при исключении неинформативных призна-

ков. Количество «правильных» доопределений увеличивается за счет противоречивых доопределений и «т»-доопределений. Количество «неправильных» доопределений уменьшается за счет превращения их в противоречивые и «т»-доопределения.

Результаты сравнения полноты и точности компьютерных исследований в обоих случаях приведены в табл. 3.

Таким образом, все предоставленные медиками признаки можно разделить на четыре группы:

1) необходимые признаки, без наличия которых гипотеза не имеет смысла. Например, нельзя прогнозировать результат лечения без наличия признака «терапия»;

2) существенные признаки, без наличия в гипотезе хотя бы одного из которых, гипотеза не имеет смысла;

3) сопутствующие признаки, не входящие в группы 1) и 2), например, пол, возраст;

4) неинформативные признаки, которые можно исключить из общего списка, предоставленного медиками.

Как видно из результатов компьютерных экспериментов, когда исключаются неинформативные признаки, имеет место повышение полноты и точности. В случае предоставления врачами новых массивов данных, будет проверена целесообразность методики исключения неинформативных признаков.

Таблица 1

№ задачи	Правильные доопределения («+»→«+»)	Неправильные доопределения «+»→«-»	Правильные доопределения («-»→«-»)	Неправильные доопределения «-»→«+»	Противоречивые доопределения
1	25	0	34	2	1
2	44	1	12	1	22
3	6	2	10	2	0
4	14	2	6	2	8
5	16	0	9	1	0
6	17	0	25	0	4
7	6	1	3	0	1

Таблица 2

№ задачи	Правильные доопределения («+»→«+»)	Неправильные доопределения «+»→«-»	Правильные доопределения («-»→«-»)	Неправильные доопределения «-»→«+»	Противоречивые доопределения
1	27	0	34	1	0
2	49	1	12	1	17
3	6	2	11	1	0
4	14	2	6	2	0
5	17	0	10	1	0
6	17	0	25	0	4
7	7	0	3	0	0

Таблица 3

№ задачи	Точность в 1 случае	Точность во 2 случае	Полнота в 1 случае	Полнота во 2 случае
1	96,3%	96,3%	78,6%	81,33%
2	97%	97%	63,6%	69,3%
3	80%	85%	42,1%	44,7%
4	83,3%	44,4%	83,3%	44,4%
5	96%	96,4%	64%	69%
6	100%	100%	85,7%	85,7%
7	70%	100%	32%	45,4%

СПИСОК ЛИТЕРАТУРЫ

1. Панкратова Е.С., Добрынин Д.А., Цапенко И.В., Зueva М.В. Интеллектуальная ДСМ-система для диагностики заболеваний органа зрения // Научно-техническая информация. Сер. 2. – 2007. – № 3. – С.14-18.
2. Добрынин Д.А., Захарова Е.В.; Панкратова Е.С. Общие принципы построения интеллектуальных ДСМ-систем для клинической и лабораторной диагностики на примере прогнозирования наилучшего способа лечения системной красной волчанки // Труды 9-ой нац. конф. по искусственному интеллекту с международным участием КИИ'2004; 28 сентября - 2 октября 2004 г. Тверь. – М.: Физматлит 2004, т.1, С. 132-139.
3. Михайлова И.Н., Панкратова Е.С., Добрынин Д.А., Самойленко И.В., Решетникова В.В., Шелепова В.М., Демидов Л.В., Барышников А.Ю., Финн В.К. О применении интеллектуальной компьютерной системы для анализа клинических данных больных меланомой // Российский биотерапевтический журнал. – 2010. – Т. 9, №2. – С.54.
4. Финн В.К. Эпистемологические основания ДСМ-метода автоматического порождения гипотез. Часть I // Научно-техническая информация. Сер.2. – 2013 – № 9. – С. 1-29.
5. Финн В.К. Эпистемологические основания ДСМ-метода автоматического порождения гипотез. Часть II // Научно-техническая информация. Сер.2. – 2013. – №12. – С. 1-26.
6. Панкратова Е.С. Интеллектуальная система типа ДСМ для анализа клинических данных // Научно-техническая информация. Сер.2. – 2011. – № 4. – С.8-16.
7. Панкратова Е.С., Виноградов Д.В. Формальное описание настройки интеллектуальных ДСМ-систем // Научно-техническая информация. Сер.2. – 2011. – № 9. – С.1-5.

Материал поступил в редакцию 29.11.13.

Сведения об авторах

ПАНКРАТОВА Елена Сергеевна – кандидат технических наук, старший научный сотрудник ВИНТИ РАН, Москва.
e-mail: pankr@viniti.ru

ДОБРЫНИН Дмитрий Анатольевич – кандидат технических наук, научный сотрудник ВИНТИ РАН, Москва.
e-mail: dobr@viniti.ru

Средства вывода фактов о типовых информационных объектах*

Рассматривается разработка средства логического вывода на онтологиях для платформы Microsoft .NET. Отличительной чертой рассматриваемого средства является возможность использования классов и объектов объектно-ориентированной среды программирования в роли, соответственно, концептов и индивидов. В результате достигается возможность построения рассуждения непосредственно над программными объектами, а также предоставляются типово-безопасные средства построения онтологических описаний информации. Рассмотрены ограничения, связанные с возможностями системы типов языков C# и F# для представления логических конструкций.

Ключевые слова: онтологии, объектно-ориентированное программирование, концептуальное моделирование, метаданные, системы абстрактных типов

ВВЕДЕНИЕ

В настоящей статье описана разработка аппликативной системы обработки онтологий дескрипционной логики, интегрированной с объектной средой программирования.

Дескрипционные логики (ДЛ) – семейство языков представления знаний, позволяющих описывать понятия предметной области в недвусмысленном, формализованном виде. Они сочетают в себе, с одной стороны, богатые выразительные возможности, а с другой – хорошие вычислительные свойства, такие как разрешимость и относительно невысокая вычислительная сложность основных логических проблем, что делает возможным их применение на практике. Таким образом, дескрипционные логики представляют собой компромисс между выразительностью и разрешимостью. Их можно рассматривать как разрешимые фрагменты логики предикатов, синтаксически же они близки к модальным логикам. Изначально они зародились как расширение фреймовых структур и семантических сетей механизмами формальной логики. В настоящее время дескрипционные логики составляют основу концепции семантического веба, являясь основой языка веб-онтологий OWL.

Иерархическая структура онтологий во многом похожа на иерархии классов в объектно-ориентированных (ОО) языках. Данное сходство породило многочисленные попытки автоматизации отображений онтологий на ОО-структуры и

наоборот. Однако вычислительные возможности, оснащающие дескрипционную логику, фундаментально отличаются от возможностей стандартных ОО-языков, таких как C# и Java, вследствие чего применение упомянутых средств влечет за собой потерю выводимости неявных фактов.

Данная работа посвящена построению средств представления ДЛ-онтологий и логического вывода на них, интегрированных с системой типов объектно-ориентированной среды Microsoft .NET. Отличительной особенностью данных средств являются возможности описания онтологических концептов с использованием классов, а также осуществления полноценного ДЛ-вывода над обычными объектами .NET. В статье исследованы дескрипционные логики разных уровней выразительности и возможности интеграции полученной системы с системами типов, а также рассмотрена возможность построения соответствия между дескрипционной логикой и системой типов языка программирования.

Статья имеет следующую структуру: в первом разделе описываются основные понятия концептуального моделирования и их связь с онтологиями; во втором – рассматриваются способы встраивания онтологий в информационные системы на программном уровне; третий раздел посвящен разработке средства логического вывода – «решателя»; четвертый – связи систем типов с логическими системами и анализу возможностей системы типов языка C#. В «Заключении» описываются основные результаты представленной работы.

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 12-07-31090 мол_а.

КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ В ИНФОРМАЦИОННЫХ СИСТЕМАХ: ОНТОЛОГИИ И КЛАССЫ В ОБЪЕКТНО- ОРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ

Основными задачами, решаемыми программными системами, и, в частности, автоматизированными информационными системами (АИС), является сбор, запись, обработка и распределение данных, связанных с определенной предметной областью [1]. Данные представляются в виде информационных объектов – *индивидов*. Необходимость осуществления эффективной работы АИС с неоднородными объектами приводит к выявлению общих черт их коллективного поведения. Группировка объектов на основе схожих черт представляет собой их классификацию. Для проведения классификации информационных объектов необходимо наличие системы понятий – *концептов*, в соответствии с которой проводится разбиение на классы.

Концептами называются абстрактные сущности, обладающие интенционалом и экстенционалом. *Интенционалом (содержанием)* называется правило, в соответствии с которым вычисляется принадлежность индивидов к данному концепту. Индивиды, удовлетворяющие данному правилу, называются *экземплярами* концепта. *Экстенционалом (объемом)* называется множество индивидов – экземпляров концепта. На концептах задается отношение вложенности, соответствующее родовидовому отношению между понятиями и устанавливающее частичный порядок.

Концептуальное моделирование – это процесс организации знаний о предметной области в иерархическую структуру по степени абстракции, направленный на улучшение понимания рассматриваемых явлений [2]. *Концептуальная модель* обычно представляется в виде совокупности концептов и связей между ними – *концептуальных зависимостей* [1, 3].

Исторически использование концептуальных моделей при построении программных систем можно связать с введением типизации в языки программирования (ЯП), что дало возможность для появления декларативных методов представления знаний, где концепты играют роль типов [4].

С возникновением индустрии баз данных появилась необходимость наряду с объектами данных управлять и объектами метаданных. *Метаданные* – термин, который можно перевести как «данные о данных». Например, к метаданным документопубликации относятся сведения об авторе, дате создания, набор ключевых слов и т. д. Общим свойством метаданных является их относительный характер: информация, являющаяся метаданными в одной системе, в другой может рассматриваться как собственно данные.

Наличие объектов метаданных позволяет управлять интерпретацией объектов данных [5]. Таким образом, достигаются возможности работы с данными различной структуры, обеспечения взаимодействия разнородных объектов и т. п. Введение метаданных позволяет упростить работу с базами данных больших объемов и сложной структуры, делая процессы управления данными настраиваемыми.

Метаданные, получаемые при проведении классификации объектов данных, представляют собой *таксономию* – иерархическую структуру понятий. Таксономия, оснащенная системой декларативного представления знаний, представляет собой *онтологию*. В информатике и вычислительных науках онтологией предметной области называется явная спецификация ее концептуализации [6]. Знания, оснащающие таксономию, можно понимать как некоторый набор правил вывода, позволяющих выводить новые знания о концептах из уже имеющихся.

Типы являются выражением метаданных с точки зрения теории программирования. Более того, информация о типах дает возможность компилятору проводить статический анализ программ, направленный на выявление семантических ошибок. Соотношение, известное как изоморфизм Карри-Говарда, позволяет интерпретировать процесс типизации как доказательство теорем относительно программ, при этом различные системы типов соответствуют различным логическим системам. Логика предикатов первого порядка (ЛПП) соответствует довольно сложным системам типов, и сегодня лишь немногие языки программирования предоставляют данные возможности.

ДЕСКРИПЦИОННЫЕ ЛОГИКИ КАК СРЕДСТВО ИНФОРМАЦИОННОГО МОДЕЛИРОВАНИЯ

Дескрипционные логики – это языки представления знаний, сочетающие, с одной стороны, богатые выразительные возможности, а с другой – хорошие вычислительные свойства: разрешимость и относительно невысокую вычислительную сложность основных логических проблем. Указанные свойства делают возможным применение ДЛ на практике. Фактически ДЛ являются разрешимыми фрагментами логики предикатов, синтаксически же они близки к модальным логикам.

Дескрипционные логики оперируют понятиями «концепт» и «роль», соответствующими в других разделах математической логики понятиям «одноместный предикат» (или множество, класс) и «двухместный предикат» (или бинарное отношение). Интуитивно концепты используются для описания множеств некоторых объектов, например, «Люди», «Женщины», «Машины». Роли используются для описания двухместных отношений между объектами: например, на множестве людей имеется двухместное отношение « x есть родитель для y », а между людьми и машинами имеется двухместное отношение « x имеет в собственности y », где в качестве x и y можно подставлять произвольные предметы. С помощью языка ДЛ можно формулировать утверждения общего вида – о множествах в целом (всякая Женщина есть Человек, всякая Машина имеется в собственности не более чем у одного Человека) и частного вида – о конкретных объектах (Мария есть Женщина, Иван имеет в собственности Машину).

Приведем три основные идеи, лежащие в основе создания дескрипционной логики:

- Основными структурными блоками являются атомарные концепты (одноместные предикаты), роли

(двухместные предикаты) и индивиды (константы, конкретные объекты).

- Дополнительные концепты и роли строятся из атомарных при помощи конструкторов. Выразительная мощность языка ограничена так, что он использует довольно небольшой набор конструкторов для построения сложных понятий и ролей.

- Неявные знания могут быть получены автоматически, с помощью процедуры вывода. В частности, отношение вложенности между концептами или принадлежности между индивидом и концептом играют важную роль. Отношения вложенности и принадлежности выводятся из определения концепта и свойств индивида.

Базы знаний в ДЛ состоят из двух компонент: так называемых ТВох и АВох. ТВох представляет собой терминологию, т. е. словарь предметной области, в то время как АВох содержит утверждения о конкретных индивидах с точки зрения этого словаря. ТВох состоит из концептов и ролей. В дополнение к атомарным концептам и ролям дескрипционные логики позволяют пользователям создавать сложные описания концептов. ТВох может быть использован для присвоения имен сложным описаниям.

В наиболее общем случае аксиомы терминологии имеют вид:

$$C \sqsubseteq D \ (R \sqsubseteq S) \text{ или } C \equiv D \ (R \equiv S),$$

где C и D являются концептами, а R и S – ролями. Аксиомы первого вида называются вхождениями, второго – эквивалентностями.

Интерпретация $\mathcal{I}$ состоит из непустого множества $\Delta^{\mathcal{I}}$ (области интерпретации) и интерпретирующей функции $\cdot^{\mathcal{I}}$, ставящей в соответствие каждому концепту A множество $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, а каждой роли R – бинарное отношение $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Определим семантику аксиом ТВох. Интерпретация $\mathcal{I}$ удовлетворяет аксиоме вхождения $C \sqsubseteq D$, если $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, и она удовлетворяет эквивалентности $C \equiv D$, если $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$. Если $\mathcal{T}$ является набором аксиом, тогда $\mathcal{I}$ удовлетворяет $\mathcal{T}$, если $\mathcal{I}$ удовлетворяет каждому элементу T . Конечный набор аксиом $\mathcal{T}$ является терминологией или ТВох в том случае, если ни одно символьное имя не определено дважды.

Утверждения, или АВох, вводят константы (индивиды), задают им имена и определяют их свойства. Будем обозначать константы маленькими буквами $a, b, c, \dots$. Используя концепт C и роль R , мы можем составить следующие утверждения в АВох:

$$C(a), R(b, c).$$

Первое утверждение называется утверждением о концепте и означает, что a принадлежит C . Второе утверждение называется утверждением о роли и означает, что c является *наполнителем* роли R для b .

Введем семантику АВох, расширив область определения функции интерпретации именами индивидов. Таким образом, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ не только отображает атомарные концепты и роли во множества и отношения, но также каждому имени индивида a ставит в соответствие элемент $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Будем считать, что различные имена обозначают различные объекты, т. е., если a и b – несовпадающие имена, тогда $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. Интерпретация $\mathcal{I}$ удовлетворяет утверждению о понятии $C(a)$, если $a^{\mathcal{I}} \in C^{\mathcal{I}}$, и удовлетворяет утверждению о роли $R(a, b)$, если $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Интерпретация удовлетворяет АВох $\mathcal{A}$, если удовлетворяет каждому утверждению в $\mathcal{A}$. В этом случае говорят, что $\mathcal{I}$ – модель $\mathcal{A}$. Таким образом, $\mathcal{I}$ удовлетворяет утверждению α или АВох $\mathcal{A}$ с точки зрения ТВох $\mathcal{T}$, если она является как моделью α или $\mathcal{A}$, так и моделью $\mathcal{T}$.

Конструкторы концептов и ролей

Для включения описаний понятий (дескрипций) в область определения интерпретирующей функции в простейшем случае используются следующие индуктивные определения:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\ \perp^{\mathcal{I}} &= \emptyset \\ (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &= (C^{\mathcal{I}} \cap D^{\mathcal{I}}) \\ (\forall R C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b. (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\ (\exists R C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b. (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \end{aligned}$$

Два понятия C и D считаются эквивалентными, что записывается как $C \equiv D$, тогда и только тогда, когда $C^{\mathcal{I}} = D^{\mathcal{I}}$ для всех интерпретаций $\mathcal{I}$.

Описанный простейший язык, называемый $\mathcal{AL}$, может быть расширен добавлением новых конструкторов. Например, существуют следующие расширения:

- Конструктор объединения концептов записывается как $C \sqcup D$ и имеет следующую интерпретацию:

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

- Полное ограничение существования записывается как $\exists R.C$ и имеет следующую интерпретацию:

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b. (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$$

В данном случае накладывается ограничение не только на существование роли, но и на концепт, экземплярами которого она может заполняться.

• Ограничения численности записываются как $\geq nR$ и $\leq nR$, где n – неотрицательное целое число. Данные ограничения имеют следующую интерпретацию:

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \left| \{b \mid (a, b) \in R^{\mathcal{I}}\} \right| \geq n\}$$

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \left| \{b \mid (a, b) \in R^{\mathcal{I}}\} \right| \leq n\}$$

соответственно, где « $|\cdot|$ » обозначает мощность множества.

• Отрицание сложных понятий обозначается как $\neg C$ и интерпретируется следующим образом:

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

Расширение $\mathcal{AL}$ -языка указанными конструкторами дает язык, название которого имеет следующий вид: $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$, где $\mathcal{U}$ означает введение объединения понятий, $\mathcal{E}$ – полного ограничения существования, $\mathcal{N}$ – ограничений численности, а $\mathcal{C}$ – отрицания определенных понятий. Но, так как при наличии отрицания в логике автоматически становятся допустимы и объединения понятий, и полное ограничение существования, то обычно логику, расширенную всеми конструкторами, именуют $\mathcal{ALCN}$.

Роли, так же как и концепты, могут строиться из более простых при помощи конструкторов ролей. Пусть $R, S \in \mathcal{R}$ – имена ролей, тогда $R \sqsubseteq S$ – аксиома включения ролей. Иерархия ролей – это конечный набор аксиом включения ролей. Определим оператор $\sqsubseteq^*$ как рефлексивное транзитивное замыкание над некоторой иерархией ролей $\mathcal{R}$. Набор ролей $R^\downarrow = \{S \in \mathcal{R} \mid S \sqsubseteq^* R\}$ дает множество подролей роли R . Набор ролей $R^\uparrow = \{S \in \mathcal{R} \mid R \sqsubseteq^* S\}$ дает множество надролей роли R . Также определим набор $S = \{R \in \mathcal{P} \mid R^\downarrow \cap \mathcal{T} = \emptyset\}$ – множество простых ролей.

Логический вывод на онтологиях

Когда инженер по знаниям моделирует предметную область, он строит терминологию $\mathcal{T}$, определяя новые концепты, возможно, через уже существующие. Но где гарантия, что полученная система аксиом непротиворечива? С логической точки зрения, концепт является осмысленным для нас, если есть некоторая интерпретация этого концепта, удовлетворяющая аксиомам $\mathcal{T}$, и такая, что данному концепту в ней соответствует непустое множество. Концепт с этим свойством называется выполнимым (непротиворечивым) в соответствии с ТВох, концепт, не обладающий указанным свойством, называется невыполнимым (противоречивым). В непротиворечивой ТВох все концепты должны быть выполнимы. Невыполнимый концепт – концепт, в лю-

бой интерпретации соответствующий пустому множеству, – обозначается « $\perp$ ».

Запишем формальные определения основных свойств дескрипционных логик, важных при реализации вывода:

Выполнимость: Концепт C выполним в соответствии с $\mathcal{T}$, если существует модель $\mathcal{I}$ для $\mathcal{T}$, такая что $C^{\mathcal{I}} \neq \emptyset$. В этом случае говорят, что $\mathcal{I}$ – модель C .

Вложенность: Концепт C вложен в концепт D в соответствии с $\mathcal{T}$, если $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ для каждой модели $\mathcal{I}$ для $\mathcal{T}$. В этом случае записывают $C \sqsubseteq_{\mathcal{T}} D$ или $\mathcal{T} \models C \sqsubseteq D$

Эквивалентность: Два концепта C и D эквивалентны в соответствии с $\mathcal{T}$, если $C^{\mathcal{I}} = D^{\mathcal{I}}$ для каждой модели $\mathcal{I}$ для $\mathcal{T}$. В этом случае записывают $C \equiv_{\mathcal{T}} D$ или $\mathcal{T} \models C \equiv D$

Ортогональность: Два концепта C и D ортогональны в соответствии с $\mathcal{T}$, если $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ для каждой модели $\mathcal{I}$ для $\mathcal{T}$.

После того как инженер создал словарь проблемной области и проверил, что все понятия выполнимы, он приступает к наполнению АВох. Напомним, что в АВох существует два вида утверждений: $C(a)$ и $R(a, b)$. Конечно, все утверждения должны быть согласованными, иначе, с точки зрения логики, мы из них можем вывести абсолютно произвольное утверждение. Дадим сначала определение согласованности:

АВох $\mathcal{A}$ согласован (совместен) в соответствии с ТВох $\mathcal{T}$, если существует интерпретация, являющаяся как моделью $\mathcal{A}$, так и моделью $\mathcal{T}$. $\mathcal{A}$ согласован, если он совместен в соответствии с пустым $\mathcal{T}$.

Например, набор утверждений «Мама» («Маша») и «Папа» («Маша») согласован в соответствии с пустым ТВох, так как без ограничения на интерпретацию «Мама» и «Папа» два понятия могут быть интерпретированы так, что они имеют общий элемент. Однако утверждения несогласованны в соответствии с ТВох «Семья», т. к. в каждой его модели «Мама» и «Папа» ортогональны.

Существуют многочисленные расширения логики дополнительными конструкторами для построения концептов, ролей, а также дополнительными видами аксиом в ТВох. Имеется неформальное соглашение об именовании получающихся при этом логик – обычно путем добавления к имени логики букв, отвечающих добавленным в язык конструкторам.

Обзор алгоритмов логического вывода

Как уже было сказано выше, существует множество различных дескрипционных логик, которые базируются на трех основных принципах и различаются конструкторами, которые они используют для построения более сложных концептов. Естественно, что количество конструкторов влияет на выразитель-

ность логики, поэтому может появиться желание создать логику, включающую все мыслимые конструкторы. Но тогда возникает проблема вычислительной сложности реализуемого в ней логического вывода. Так, подобные логики могут либо быть неразрешимыми, либо принадлежать к таким классам сложности, как PSPACE, EXPTIME, NPSpace, P, NP и т. д.

При использовании выразительных дескрипционных логик наиболее подходящим решением будут табличные алгоритмы, которые вместо того чтобы исследовать вложенность концептов, сводят проблему вложенности к проблеме выполнимости и, с использованием специальных правил вывода, максимально «раскрывают» понятия в поисках противоречий.

Для лучшего понимания рассмотрим простой пример: пусть A и B – имена концептов, а R – имя роли. Предположим, что мы хотим узнать, является ли понятие $(\exists R.A) \cap (\exists R.B)$ вложенным в $(\exists R.(A \cap B))$. Эту задачу легко свести к выявлению непротиворечивости следующего концепта:

$$C = ((\exists R.A) \cap (\exists R.B)) \cap \neg(\exists R.(A \cap B))$$

Сначала приведем это понятие к отрицательной нормальной форме, т. е. преобразуем все отрицания с помощью правил де Моргана и обычных правил работы с кванторами так, чтобы они стояли как можно глубже в концепте. В результате мы получим:

$$C = ((\exists R.A) \cap (\exists R.B)) \cap (\forall R.(\neg A \sqcup \neg B))$$

Теперь попытаемся для этого понятия построить непустую интерпретацию. Это означает, что должна существовать по крайней мере одна константа в заданной интерпретации. Примем за такую константу некоторое b . Так как мы имеем пересечение трех понятий, то b должна удовлетворять каждому из них. Тогда, если b удовлетворяет первому понятию $(\exists R.A)$, значит существует некоторая константа c , такая что (b, c) принадлежит области интерпретации роли R , и c принадлежит области интерпретации концепта A . Аналогично для второго концепта $(\exists R.B)$ получаем, что должна существовать некоторая константа d , такая что (b, d) принадлежит области интерпретации роли R , и d принадлежит области интерпретации концепта B . Заметим, что c и d необязательно равны. Таким образом, у нас есть первое правило:

При столкновении с ограничением существования алгоритм вводит новую константу – наполнитель роли, и эта константа должна удовлетворять конструкциям, указанным в ограничении.

Теперь перейдем к третьему концепту $(\forall R.(\neg A \sqcup \neg B))$: все наполнители роли R должны удовлетворять концепту $\neg A \sqcup \neg B$. В том числе и введенные ранее c и d . Таким образом, получаем второе правило:

При столкновении с ограничением значения вместе с уже определенными наполнителями ролей, алгоритм предлагает новые конструкции на этих ролях.

Таким образом, c принадлежит $\neg A \sqcup \neg B$, т. е. c принадлежит либо области интерпретации $\neg A$, либо $\neg B$, но c принадлежит области интерпретации A , следовательно c может принадлежать только области интерпретации $\neg B$. Аналогично для d . То есть получили третье правило:

При столкновении с дизъюнкцией пытаемся проверить обе возможности на успех. Если мы получили противоречие, необходимо вернуться и пойти вторым путем.

Таким образом, мы получили непустую интерпретацию с областью определения $\{b, c, d\}$, областью определения роли $R = \{(b, c), (b, d)\}$, областью определения концепта $A = \{c\}$, областью определения концепта $B = \{d\}$. То есть первый концепт не был вложен во второй.

Но если мы добавим к первому концепту числовое ограничение $\leq 1R$, то в изложенном выше рассуждении можно будет сделать вывод, что $c = d$. В этом случае мы получим противоречие, из чего будет сделан вывод о том, что первый концепт вложен во второй. Теперь мы имеем четвертое правило:

При столкновении с ограничением «не более» алгоритм отождествляет некоторые наполнители ролей.

Таким образом, мы объяснили основной принцип работы табло-алгоритма «на пальцах». Конечно, для логик, поддерживающих достаточно большое число конструкторов, алгоритм становится сложнее, и к описанным правилам необходимо добавлять новые, а также вводить определенный порядок действий, делить константы на «новые» и «старые», ограничивать правила применения ролей и т. д. Но об этом речь пойдет ниже.

ПРИМЕНЕНИЕ ОНТОЛОГИЙ В ОБЪЕКТНО-ОРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ

Разработка информационных систем на основе концептуальных моделей подразумевает наличие некоторого метода, позволяющего связывать программный код с данной моделью. Существует, по крайней мере, два способа решения данной задачи:

- использование программной библиотеки, предоставляющей интерфейс для доступа к информации, хранимой в концептуальной модели;
- использование концептов модели на уровне конструкций языка программирования.

Первый способ является более простым для реализации и предоставляет программисту большую свободу действий. Однако среда программирования не может использовать концептуальную модель в

процессе типизации, т. е. многие ошибки не могут быть выявлены статически на этапе компиляции.

Второй способ, совмещая возможности языка программирования и концептуальной модели, позволяет использовать концепты в качестве типов, что дает возможность применения проверки типов для выявления многих ошибок.

К средствам первого вида относится библиотека Jena [7]. Jena предоставляет доступ к семантической информации посредством классов Java, таких как `OntClass` для концептов онтологии, `OntProperty` для ролей, `Individual` для индивидов и т. п. (см. рис. 1). Эти классы аналогичны классам рефлексии (reflection) [8] в Java. Так как в качестве идентификаторов концептов, ролей и экземпляров используются строки, их корректность не проверяется во время компиляции, что ведет к риску возникновения ошибок во время выполнения программы.

Средства второго типа чаще всего используют сходство понятий концепта в концептуальном моделировании и класса в ООП. Связывание концептов с классами позволяет использовать их при перегрузке функций и диспетчеризации.

Вариантом реализации подхода второго типа можно считать метод, описанный в работе [9], по-

добный подход используется в Интегрированной системе информационных ресурсов (ИСИР) РАН [10]. Он подразумевает генерацию на основе веб-онтологии библиотеки классов Java. Все классы и интерфейсы, содержащиеся в генерируемой библиотеке, а также их свойства, создаются на основе онтологии. Классы и интерфейсы соответствуют концептам, их свойства – свойствам этих концептов.

При присоединении сгенерированной из концептуальной модели программной библиотеки к основной сборке программист получает доступ к концептам через сгенерированные классы. То есть он может расширять сгенерированные классы собственными методами, привязывая таким образом функционал приложения к концептуальной модели. Осуществляя диспетчеризацию вызовов подобных методов, среда Java будет использовать иерархию классов, которая повторяет иерархию концептов. То есть делается попытка совмещения возможностей языка Java с веб-онтологиями. По сравнению с использованием таких библиотек, как Jena, рассматриваемый подход дает более компактный код (см. рис. 2), который, кроме того, проверяется компилятором на соответствие типов: программист не может использовать свойство или класс, не заданные в онтологии, компилятор сообщит об ошибке.

```
OntModel model = ... // загрузка онтологии CATALOG
// и данных на основе этой онтологии
OntClass personClass = model.getOntClass(catalogBaseURI+"#Person");
DatatypeProperty fnProp =
    model.getDatatypeProperty(catalogBaseURI+"#FullName");
DatatypeProperty emailProp =
    model.getDatatypeProperty(catalogBaseURI+"#EMAIL");
Iterator i = personClass.listIndividuals();
while (i.hasNext()) {
    Individual vcard = i.next();
    Iterator i2 = vcard.listPropertyValues(emailProp,true);
    while (i2.hasNext()) {
        Literal email = i2.next().as(typeOf(Literal));
        if (email.getString().equals("amanda_cartwright@example.org") ) {
            Literal fullname = vcard.getPropertyValue(fnProp). as(typeOf(Literal));
            if ( fullname != null )
                System.out.println("Name: " + fullname.getString());
        }
    }
}
```

Рис. 1. Код для доступа к онтологии с помощью системы Jena

```
Iterator i = Person.listIndividuals();
while (i.hasNext()) {
    VCard vcard = i.next();
    Iterator i2 = vcard.getEmails();
    while (i2.hasNext()) {
        Literal email = i2.next();
        if (i2.next() == "amanda_cartwright@example.org") ) {
            if ( vcard.getFullName() != null )
                System.out.println("Name: " + vcard.getFullName());
        }
    }
}
```

Рис. 2. Код, использующий онтологию через сгенерированные классы

Однако после генерации классов из иерархии концептов теряется возможность логического вывода, а также возможность описывать анонимные дескрипции концептов. Указанные потери связаны с фундаментальными различиями онтологий и системы классов Java:

- роли в онтологии определяются отдельно от концептов, дескрипции концептов формируются с использованием заранее определенных ролей; в Java свойства классов (их поля и методы) определяются внутри классов, одно свойство не может быть атрибутом различных классов;

- концепты представляют собой предикаты, один и тот же индивид при изменении его свойств может становиться или прекращать быть экземпляром каких-либо концептов; объекты Java с момента создания связаны с классом, экземпляром которого являются. Исключением являются динамические языки, такие как Ruby, на который ориентирована библиотека ActiveRDF. Однако в них теряются преимущества статической типизации: ошибки типов не выявляются на этапе компиляции.

Метод перегрузки и диспетчеризации функций на основе концептов предложен в работе [11]. В данной работе использован язык Lisp, концепты представляются в виде объектов среды CLOS (Common Lisp Object System, Объектная система языка Common Lisp). Возможность логического вывода не теряется за счет интеграции с системой Classic. Однако в работе не уделено должного внимания однозначности диспетчеризации. Кроме того, в качестве индивидов могут быть использованы только объекты системы Classic, а не любые объекты языка Lisp.

Целью нашей работы является разработка средств построения логического вывода в дескрипционных логиках, позволяющих использовать классы и объекты среды .NET в качестве, соответственно, концептов и индивидов. Наличие такого средства позволит использовать в качестве концептов статически проверяемые конструкции среды программирования и в то же время не терять возможностей логического вывода.

РЕАЛИЗАЦИЯ РАССУЖДАТЕЛЯ И ИНТЕГРАЦИИ С СИСТЕМОЙ ТИПОВ

Интеграция с системой типов

Для того чтобы использовать ДЛ-решатель для вывода на типах, необходимо провести сопоставление объектов системы типов и объектов дескрипционной логики. Также для того чтобы вывод был корректен, необходимо извлечь всю необходимую информацию о типах и их взаимосвязи. Для начала произведем сопоставление:

Типы	↔	Концепты
Объекты	↔	Утверждения о концептах
Методы без параметров	↔	Роли

В связи с тем, что в существующих языках объектно-ориентированного программирования (ООП) у каждого класса существует множество методов и далеко не все из них необходимы нам в качестве ролей,

вводится список методов, которые могут быть использованы в качестве ролей.

Также из-за того, что у конкретного объекта .NET может быть только один тип или несколько вложенных друг в друга типов, то мир объектов .NET является замкнутым, а значит необходимо вручную производить замыкание мира типов, при этом оставляя мир концептов в их первоначальном понимании открытым. Для этого перед использованием таблоалгоритма составляем отображение типа на неравные (невложенные в него) типы, которые присутствуют в рассматриваемой модели. Также добавляем для каждого атомарного концепта-типа отрицание концептов-типов из этого отображения. В процессе вывода, при добавлении нового атомарного концепта, повторяем это действие, тем самым закрывая мир концептов-типов, оставляя мир ДЛ-концептов открытым.

Вдобавок ко всему, оперируя с концептами, мы предполагаем, что все определения, а также все вложенности концептов передаются нам в явном виде, но для объектов аппликативной среды эта информация не присутствует в явном виде в объекте, и нам необходимо заранее извлечь все отношения вложенности и эквивалентности концептов-типов, до осуществления алгоритма.

Кроме того, чисто теоретически в дескрипционной логике могут быть «подвешенные» константы – индивиды, не принадлежащие конкретному концепту. Для систем типизации такое недопустимо: каждый объект имеет тип, поэтому для всех объектов, переданных в решатель, мы извлекаем информацию об их типе и добавляем этот тип в список оперируемых концептов.

В результате описанных выше преобразований и преобразования неявной информации в явную мы сможем свести набор выражений о типах к пригодному для вывода в ДЛ списку утверждений, который может быть исследован на совместность и непротиворечивость.

Алгоритм работы

Алгоритм предобработки входных значений

В данном разделе будут конкретизированы основные тезисы, описанные выше, а также составлен алгоритм предобработки типов для их использования в текущем выводе:

1. Получаем список всех концептов-типов.
2. Ищем методы без параметров и выбираем из них те, что могут быть использованы в качестве ролей.
3. Формируем список новых утверждений о ролях на основе сгенерированного списка методов-ролей и добавляем его к уже существующему набору утверждений.
4. Получаем список всех индивидов-объектов.
5. Добавляем новые утверждения об индивидах-термах за счет вычисления всех концептов-типов, к которым они принадлежат.
6. Формируем список аксиом включения из неявных знаний о вложенности типов, а также из предположения, что для всех индивидов-объектов должны выполняться методы концепта-типа, к которому он

принадлежит, т. е. для объекта типа C , у которого есть метод R с возвращаемым значением B , должно выполняться

$$C \sqsupseteq \forall R.B.$$

7. Далее строим модель, в которой каждому типу-концепту и объекту-индивиду ставятся в соответствие концепты и индивиды TBox и ABox.

8. Запускаем алгоритм вывода на полученной модели.

Алгоритм вывода

Список правил

Упорядочим новые индивиды отношением $\prec$. То есть, если $b \in \mathcal{O} _ \mathcal{N}$ вводится в $\mathcal{A}$, тогда отношение $a \prec b$ верно для всех новых индивидов a , уже существующих в $\mathcal{A}$.

9. Правило пересечения:

Условие: A содержит $(C_1 \sqcap C_2)(a)$, но не содержит либо $C_1(a)$, либо $C_2(a)$.

$$\text{Действие: } A = A \cup \{C_1(a), C_2(a)\}.$$

10. Правило объединения:

Условие: A содержит $(C_1 \sqcup C_2)(a)$, но не содержит ни $C_1(a)$, ни $C_2(a)$.

Действие:

$$A = A \cup \{C_1(a)\}, A = A \cup \{C_2(a)\}.$$

11. Правило ограничения существования:

Условие: A содержит $(\exists R.C)(a)$, a не заблокирована, и A не содержит константу $b \in \mathcal{O}$, такую что $C(b)$ и $S(a, b)$ принадлежат A , где $S \in R^\downarrow$.

Действие: $A = A \cup \{C(b), R(a, b)\}$, где b имя константы, не входящей в A .

12. Правило ограничения значения:

Условие: A содержит $(\forall R.C)(a)$, и существует такая $b \in \mathcal{O}$, что $S(a, b)$, где $S \in R^\downarrow$, но A не содержит $C(b)$.

$$\text{Действие: } A = A \cup \{C(b)\}$$

13. Правило ограничения значения транзитивных ролей:

Условие: A содержит $(\forall R.C)(a)$, и существует такая $b \in \mathcal{O}$, что $S(a, b)$, где $S \in T^\downarrow, T \in \mathcal{T}, T \in R^\downarrow$, но A не содержит $(\forall T.C)(b)$.

$$\text{Действие: } A = A \cup \{\forall T.C(b)\}$$

14. Правило аксиом включения:

Условие: A содержит $(\forall x.C)(x)$, и существует такая $a \in \mathcal{O}$, упомянутая уже в A , и A не содержит $C(a)$.

$$\text{Действие: } A = A \cup \{C(a)\}$$

15. Правило числового ограничения «не менее»:

Условие: A содержит $(\geq nR)(a)$, a не заблокирована, и A не содержит констант $b_1, \dots, b_n \in \mathcal{O}$ таких, что правило $S_i(a, b_i) (1 \leq i \leq n)$ и $b_i \neq b_j (1 \leq j \leq n)$, где $S_1, \dots, S_n \in R^\downarrow$.

Действие:

$A = A \cup \{R(a, b_i) | 1 \leq i \leq n\} \cup \{b_i \neq b_j | 1 \leq i < j \leq n\}$, где $b_1 \dots b_n \in \mathcal{O} _ \mathcal{N}$ – различные имена констант, не содержащихся в A .

16. Правило числового ограничения «не более»:

Условие: A содержит различные константы $b_1, \dots, b_{(n+1)}$, такие что $(\leq nR)(a)$ и $S_1(a, b_1), \dots, S_{(n+1)}(a, b_{(n+1)})$ содержатся в A , и $b_i \neq b_j$ не содержится в A для некоторых $i \neq j$, где $S_1, \dots, S_{(n+1)} \in R^\downarrow$.

Действие: Для каждой пары b_i, b_j , таких что $i > j$ и $b_i \neq b_j$ не содержится в A , получаем ABox $A_{ij} = [b_i / b_j]A$, полученный заменой b_i на b_j в A .

17. Правило ограничения существования предиката:

Условие: A содержит $(\exists f_1, \dots, f_n.P)(a)$, и не существует таких $(x_1, \dots, x_n \in \mathcal{O} _ \mathcal{C})$, что $f_1(a, x_1), \dots, f_n(a, x_n), P(x_1, \dots, x_n) \sqsupseteq A$.

Действие:

$$A = A \cup \{f_1(a, x_1), \dots, f_n(a, x_n), P(x_1, \dots, x_n)\},$$

где $(x_1, \dots, x_n \in \mathcal{O} _ \mathcal{C})$ не входят в A , также после выполнения этого преобразования необходимо исключить все вилки в полученном A , т. е. для выражений вида $f_i(a, x), f_i(a, x_i) \sqsubseteq A$ произвести замену x_i на x , если $x \prec x_i, i \in 1 \dots n$.

Правила «Правило ограничения существования предиката», «Правило числового ограничения "не более"», «Правило ограничения существования» называются *генерирующими*, так как при их выполнении образуются новые индивиды или конкретные объекты.

Отметим основные ситуации, приводящие к противоречию:

1. $\perp(a)$ содержится в A для некоторой переменной a .
2. $A(a)$ и $\neg A(a)$ содержатся в A для некоторой переменной a .
3. $\{(\leq nR)(a)\} \sqcup \{S_i(a, b_i) \mid 1 \leq i \leq n+1\} \cup \cup \{b_i \neq b_j \mid 1 \leq i < j \leq n+1\} \subseteq A$

для некоторых констант $a, b_1, \dots, b_n$ с неотрицательным индексом n и ролями $S_1, \dots, S_n \in R^\downarrow$.

4. $f(a, x)$ и $\forall f. \perp(a)$ содержится в A для некоторой переменной a .
5. $P_1(x_1^{(1)}, \dots, x_n^{(1)}) \in A$ и $P_k(x_1^{(k)}, \dots, x_n^{(k)}) \in A$,

но пересечение этих предикатов невыполнимо.

Таким образом, алгоритм заключается в трансформации начального состояния в некоторый набор конечных состояний. При этом, если хотя бы одно из состояний (АВох-ов) непротиворечиво (совместно), значит непротиворечив и первоначальный концепт. Если же все состояния несовместны, то и первоначальный концепт противоречив. Заметим, что процесс применения правил конечен. Если к АВох нельзя применить больше ни одно правило, то он называется *завершенным* [12]. Но при использовании лишь указанных выше правил могут возникнуть ситуации, в которых наш алгоритм будет нетерминальным, поэтому, во избежание таких проблем, вводят строгую последовательность действий:

18. Применяем все негенерирующие правила столь долго, насколько это возможно.

19. Применяем одно генерирующее правило и начинаем снова с шага 1.

При этом:

- Применяем правила для индивида из множества $\mathcal{O}_N$, только если больше не применимы правила для индивида из множества $\mathcal{O}_O$.
- Применяем правила для индивида из множества $b \in \mathcal{O}_O$, только если больше не применимы правила для индивида $a \in \mathcal{O}_O$, такого что $a < b$.

Отметим также, что приведенный выше алгоритм универсален как для исследования выполнимости концепта, так и для совместности АВох. Несмотря на то, что получившаяся логика является EXPTIME-полной, алгоритм для нее NEXPTIME-труден.

Логический вывод фактически позволяет проводить структурный анализ объекта ОО-среды и его типа: проверять наличие тех или иных методов, проверять значения, получаемые в результате их выполнения и т. д.

ОГРАНИЧЕНИЯ СИСТЕМЫ ТИПОВ ЯЗЫКОВ C# И F#

Теория типов является средством выражения семантики, позволяющим наиболее полно использовать возможности среды программирования. Сегодня на-

блюдается постоянный рост выразительности систем типов, используемых в современных языках программирования. Система типов является формализацией процесса проверки компилятором семантической корректности программы. Другими словами, в процессе типизации проводится доказательство некоторых свойств программы. Подобный взгляд ведет к более глубокой интерпретации системы типов – как некоторой логической системы. В вычислительных науках данное соответствие носит название «изоморфизма Карри-Говарда» [13], утверждающего, что типы соответствуют теоремам, в то время как программы (вычислительные объекты), имеющие (населяющие) некоторый тип, – доказательствам соответствующей теоремы. Тип, для которого невозможно написать программу, соответствует теореме, у которой не существует доказательств, – противоречию. Тип, соответствующий любой программе, соответствует тавтологии. В ООП данный тип обычно называется *Object* (в языке Scala – *Any*).

Изоморфизм Карри-Говарда дает дополнительную мотивацию для погружения дескрипционных логик в систему типов, так как утверждает, что данная задача разрешима: обе системы являются изоморфными друг другу. Этот же результат позволяет предсказывать применимость тех или иных языков программирования для погружения в них ДЛ: системы типов различных языков отличаются по выразительности и соответствуют различным логическим системам. Основная классификация систем типов известна как « λ -куб Барендрегта» [14].

Одной из основных конструкций в логике является импликация $A \rightarrow B$, которой в программировании соответствует функциональный тип $A \rightarrow B$ – тип функций из A в B . Таким образом, для соответствия более или менее выразительной логической системе язык программирования должен являться *функциональным*: он должен поддерживать функциональные типы как типы простых значений – типы параметров и результатов функций.

Далее, уровень выразительности логики, соответствующей системе типов, во многом определяется разрешенными зависимостями между типами и терминами (значениями, программами, объектами). Простейшие логические высказывания связывают между собой другие высказывания. Такие выражения соответствуют типам, зависящим от типов – «джереникам» (generics). Например, функциональный тип $A \rightarrow B$ определяется в C# через дженерик-тип $\text{Func}\langle A, B \rangle$.

Предикаты представляют собой логические высказывания, зависящие от значений предметных переменных. Таким образом, моделирование ЛПП требует наличия «зависимых типов» – типов, зависящих от термов. Данная особенность значительно повышает сложность системы типов и поэтому сегодня присутствует лишь в немногих языках. Примером зависимого типа является тип векторов длины n : подстановка вместо n различных чисел дает различные типы векторов. Система типов языка C# не поддерживает данную возможность, что говорит о невозможности полного погружения в нее ДЛ.

В дальнейшем планируется перенос описанных разработок на язык Scala. Данный язык основан на платформе Java, которая имеет еще большее распространение, чем .NET. В отличие от C# в Scala присутствует поддержка некоторой формы зависимых типов. Кроме того, в системе типов Scala присутствует тип `Nothing`, который присваивается ошибочным программам. С точки зрения изоморфизма Карри-Говарда этот тип соответствует противоречию, т. е. тождественен ложному высказыванию. Таким образом, вместе с функциональным типом, моделирующим импликацию, данный тип образует имплективный базис $\langle \rightarrow, 0 \rangle$. Являясь базисом, данная система высказывательных функций позволяет выразить любую другую функцию, что потенциально позволяет использовать систему типов Scala как логически полную систему.

Недостаток Scala, из-за которого изначально была выбрана платформа .NET, заключается в более сложных методах обработки дженериков. В то время как в байт-коде среды CLR типовые параметры дженерик-типов (например, `A` и `B` в `Func<A, B>`) сохраняются и после компиляции программы, в виртуальной машине Java используется так называемое *стирание типов*: типовые параметры существуют только при компиляции программы и в байт-коде не присутствуют. Восстановление значения типовых параметров для их использования при выполнении программы требует особых приемов, усложняющих программный код.

ЗАКЛЮЧЕНИЕ

Задачей данной работы являлась разработка системы обработки онтологий дескрипционной логики, интегрированной с объектной средой программирования. Была построена система, осуществляющая вывод неявных зависимостей между явно представленными знаниями, т. е. отвечающая на вопрос: существует ли интерпретация данного понятия, или интерпретация, являющаяся моделью для системы утверждений. В качестве концептов могут использоваться типы/классы объектной среды C#, а в качестве индивидов – объекты. Таким образом, одним из результатов выполненной работы стала система построения логического вывода над объектами среды .NET. Разработанная система логического вывода в дескрипционных логиках является уникальной для платформы .NET и превосходит подобные средства для других платформ по степени интеграции с системой типов и компактности получаемого программного кода. В результате исследования были выявлены ограничения системы типов C#, что делает предпочтительным выполнение дальнейших исследований и разработок на других языках программирования.

СПИСОК ЛИТЕРАТУРЫ

1. Вольфенгаген В.Э., Яцук В.Я. Аппликативные вычислительные системы и концептуальный метод проектирования систем знаний. – М.: МО СССР, 1987.

2. Taivalsaari A. On the notion of inheritance // ACM Computing Surveys (CSUR). – 1996. – Vol. 28, № 3. – P. 438–479.
3. Шенк Р. Обработка концептуальной информации. – М.: Энергия, 1980.
4. Hindley J.R. Basic simple type theory. – Cambridge Univ. Pr., 1997.
5. Исмаилова Л., Косиков С. Соотнесение семантик против семантики соотнесений // Технологии информационного общества, интернет и современное общество // Тр. VI всерос. объедин. конф. – СПб.: Изд-во филологического факультета СПбГУ, 2003. – С. 17–22.
6. Gruber T.-R. et al. A translation approach to portable ontology specifications // Knowledge acquisition. – 1993. – Vol. 5. – P. 199–199.
7. McBride B. Jena: Implementing the RDF Model and Syntax Specification // Proc. of the 2001 Semantic Web Workshop. – 2001.
8. Smith B.C. Procedural reflection in programming languages: PhD thesis. – Massachusetts Inst. of Technology, 1982.
9. Goldman N.M. Ontology-oriented programming: static typing for the inconsistent programmer // Lecture Notes on Computer Science, The Semantic Web, ISWC. – 2003. – P. 850–865.
10. Бездушный А.А. и др. RDFS как основа среды разработки цифровых библиотек и web-порталов // Электронные библиотеки. – 2003. – Т. 6, № 3.
11. Moeller R. A functional layer for description logics: knowledge representation meets object-oriented programming // Proceedings of the 11th ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications. – New York, USA: ACM, 1996. – P. 198–213.
12. Baader F. et al. The description logic handbook: theory, implementation, and applications. – Cambridge University Press, 2007.
13. Sørensen M.H., Urzyczyn P. Lectures on the Curry-Howard isomorphism. – Elsevier, 2006. – Vol. 149.
14. Barendregt H. Introduction to generalized type systems // Journal of functional programming. – 1991. – Vol. 1, № 2. – P. 125–154.

Материал поступил в редакцию 25.12.13.

Сведения об авторах

ШАПКИН Павел Александрович – кандидат технических наук, старший преподаватель кафедры «Кибернетика» НИЯУ МИФИ, зам. зав. направлением «Компьютерные науки и информационные технологии» СКИБ 22
e-mail: pashapkin@mephi.ru

ДЕМЧЕНКО Анна Павловна – студент Национального исследовательского ядерного университета МИФИ (НИЯУ МИФИ)
e-mail: Ann1137@yandex.ru

Может ли отрицание отрицать презумпцию?*

Рассматривается конструкция, в составе которой отрицание, на первый взгляд, отрицает презумпцию. Показано, что презумпция и в этой конструкции сохраняет свое главное свойство, состоящее в том, что она не подвергается отрицанию при отрицании предложения.

Ключевые слова: презумпция, ассерция, отрицание стандартное, радикальное, глобальное

1. ОБЩИЕ СВОЙСТВА ПРЕЗУМПЦИЙ

Понятие **презумпция** (говорят также **пресуппозиция**; ср. англ. термин *presupposition*; нем. термин *Voraussetzung*) пришло в лингвистику из логики и стало незаменимым инструментом семантического и синтаксического анализа. Оно широко используется как в лексической семантике, так и в семантике синтаксиса. В частности, понятие презумпции существенно продвинуло решение проблемы отрицания и отрицательного предложения: было обнаружено свойство отрицания в естественном языке, состоящее в том, что оно воздействует на **ассерцию** высказывания и не распространяется на его презумпции.

1.1. Презумпция и ассерция

Истинность презумпций предложения является необходимым условием его осмысленности. И наоборот, ложность презумпций делает предложение семантически аномальным. В логике, где главным свойством предложения считается его истинность или ложность, а главным отношением между предложениями – отношение логического следования, возникло следующее определение презумпции:

Определение 1. Предложение Р является **презумпцией** предложения S, если Р следует как из S, так и из отрицания S.

Это определение позволяет диагностировать наличие презумпции в предложении. Так, предложение (1) имеет презумпцию ‘Фишер отказался от игры’, что видно из его отрицания (2):

(1) Отказ от игры уронил Фишера в глазах шахматистов ⊃ ‘Фишер отказался от игры’;

(2) Отказ от игры *не* уронил Фишера в глазах шахматистов ⊃ ‘Фишер отказался от игры’.

Тест на отрицание позволил выявить презумпции как значимые семантические компоненты многих

слов и синтаксических конструкций разных языков, например:

• **фактивных** глаголов и, шире, фактивных предикатов:

Иван *жалеет /не жалеет*, что уехал ⊃ ‘Иван уехал’;

• **фазовых** глаголов, типа *перестать*, *продолжать*:

Иван *перестал /не перестал* бить свою жену ⊃ ‘бил’;

• **конативных** глаголов (например, глагола *удаться*):

Ивану *удалось /не удалось* попасть на концерт Макаревича ⊃ ‘пытался’;

• **адвербиалов-модификаторов** – со значением образа действия (типа *быстро*, *громко*, *с трудом*), цели и проч.:

Они *разговаривали громко /не громко* ⊃ ‘они разговаривали’;

Трамвай *остановился, чтобы высадить пассажиров* ⊃ ‘трамвай остановился’.

Слова, категории и конструкции, которые порождают презумпции, называют **триггерами презумпций** (*presupposition triggers*). Например, триггерами презумпций являются фактивные и фазовые глаголы.

Важная роль понятия презумпции в лингвистике определяется, в частности, тем, что оно вызвало к жизни понятие **ассерции**. Если презумпция – это то, что «само собой разумеется», а потому избегает действия отрицания, то ассерция (утверждаемое) – это, наоборот, то, что ему подвергается. Иначе говоря, ассерция составляет **сферу действия** отрицания – равно как и других семантических операторов (см. о сфере действия отрицания и кванторных слов в [1, с. 85, 144] и о сфере действия в более широком смысле – в [2]). Так, в примере (3) ассерция – ‘Иван жалеет’, поскольку именно эта пропозиция подвергается отрицанию при отрицании предложения в целом:

(3) а. Иван *жалеет*, что поехал на конференцию в Марокко;

б. Иван *не жалеет*, что поехал на конференцию в Марокко.

* Данная работа была выполнена при финансовой поддержке РФНФ, грант № 14-04-00604а.

Наряду с фактивными глаголами, которые гарантируют истинность подчиненной пропозиции, существуют **имплицативные**. В русском языке имплицативными являются глаголы *суметь, успеть, добиться, удалиться, осмелиться* и др.

Определение 2. Семантический компонент Р является в предложении S **имплицативом**, если из S следует Р, а из отрицания S следует не-Р.

Так, из того, что Иван успел купить билет, следует, что он купил билет, а из того, что не успел, следует, что не купил. Компонент 'Иван купил билет' является в структуре предложения *Иван успел купить билет* имплицативом.

1.2. Презумпция и так называемое «внешнее отрицание»

Хотя презумпция – это, почти по определению, то, что не поддается отрицанию, в литературе исследовалась возможность **подавления презумпций в контексте отрицания**. В этой связи было введено в рассмотрение понятие «внешнего отрицания» (**external negation, presupposition-denying negation**, в противоположность обычному **presupposition-preserving negation**, см. [3, с. 137]). Якобы в языке есть отрицание, которое не различает в структуре предложения презумпций и ассерций.

Пример из [4]. Если понимать *не* в (4) как «внешнее» отрицание (т.е., по Расселу, как конъюнктивно присоединенную ассерцию), то предложение (4) будет иметь смысл (5), а (6) не будет аномальным:

(4) Джон *не огорчен* тем, что провалился на экзамене;

(5) 'Джон либо провалился и не огорчен, либо не провалился';

(6) Джон *не огорчен* тем, что провалился на экзамене, – он его сдал.

Иначе говоря, допускается, что фраза (6) может пониматься как истинная в контексте, когда Джон НЕ провалился на экзамене, – хотя при этой интерпретации фраза должна отрицать свою презумпцию.

Ясно, что такое употребление отрицания не соответствует реальному узусу и допустимо разве что в режиме языковой игры.

1.3. Нейтрализация презумпций в контексте снятой утвердительности

Если триггер презумпции попадает в контекст **снятой утвердительности**, презумпция исчезает. Вернемся к предложению (1). В нем пропозиция 'Фишер отказался от игры' – презумпция; между тем предложение (1'), где глагол *уронить* в сослагательном наклонении, не имеет этого следствия, т.е. не имеет этой презумпции:

(1) Отказ от игры уронил Фишера в глазах шахматистов;

(1') Отказ от игры уронил *бы* Фишера в глазах шахматистов.

О снятой утвердительности (*suspended assertion*) см. [5], а также [6, 7]; ср. термин *non-veridicality* в [8]. Утвердительность финитного глагола может сниматься косвенной модальностью, будущим временем,

союзами типа *чтобы*, пропозициональными установками и проч.

В контексте снятой утвердительности триггера пропозиция Р приобретает нейтральную модальность (можно сказать, **нейтрализуется**), но она **не отрицается**. Нейтрализация презумпций в контексте снятой утвердительности триггера не имеет ничего общего с «подавлением» презумпции под действием внешнего отрицания, которого не бывает. «Presuppositional inferences are typically subject to cancellation by direct denial only when the presupposition trigger is embedded under some other operator» [9].

2. РАДИКАЛЬНОЕ ОТРИЦАНИЕ

Теперь, на фоне всего того, что было сказано о неподавляемости презумпций, обратимся к конструкции, в составе которой отрицание *на первый взгляд* отрицает презумпцию. Эта конструкция была обнаружена Г.И. Кустовой, см. [10], а в последнее время внимание было вновь привлечено к этой проблеме благодаря работе В.Ю. Апресян [11].

2.1. Событийная и расщепленная каузативная конструкция

В [10] (см. также [12, с. 214–231]) были рассмотрены две конструкции с каузативным глаголом. Ср. предложения (1а) и (1б):

- (1) а. Его женитьба на служанке *опозорила* семью;
б. Он *опозорил* семью женитьбой на служанке.

В примере (1а) обычная каузативная конструкция с событийной каузацией: каузативный глагол в норме связывает две ситуации; в (1а) это женитьба, **каузатор**, и позор, **эффект**. А каузативная конструкция в (1б) была названа в [12] **личной**. Можно думать, особенность каузативной конструкции в (1б) состоит в том, что она является результатом своего рода **расщепления** событийной каузативной конструкции в (1а), см. [13, с. 283]. При расщеплении может, действительно, возникать подлежащее-лицо и дополнительный агентивный смысл, ср. различие между (2а) и (2б):

- (2) а. Ваня приход *обрадовал* нашу маму;
б. Ваня *обрадовал* нашу маму своим приходом.

Но это не обязательно. Предложение (3) (пример из [13, с. 283]) может иметь две интерпретации. Интерпретация (i) агентивная; семантическая роль подлежащего – Агенса, а именная группа в твор. падеже обозначает поступок. А интерпретация (ii) неагентивная; семантическая роль подлежащего – Мишень:

(3) Родственники *раздражали* его бесконечными вопросами.

(i) Родственники задавали много вопросов, имея целью его раздражить;

(ii) Родственники задавали много вопросов, и это его раздражало.

А в примерах (4) и (5) подлежащее в расщепленной конструкции однозначно неагентивно:

- (4) а. Мой проигрыш *огорчил* отца;
б. Я *огорчил* отца своим проигрышем.

Это тем более ясно для примера (5), где дополнение в твор. падеже выражает свойство, а подлежащее – носитель свойства и вовсе не лицо:

(5) Письма Сельвинского молодым поэтам удивляют своей щедростью (← Щедрость писем Сельвинского молодым поэтам удивляет).

Граница между контролируемой и неконтролируемой каузацией может быть размытой. Так, в контексте примера (6) каузация, скорее всего, неконтролируемая: люди, которые вели разговоры об искусстве, не имели цели расстроить Маяковского (при том даже, что они объявляются ответственными за произведенный эффект).

(6) Его <Маяковского> легко было огорчить. Виноваты литературные бывшие люди. Они *расстраивали* его разговорами о настоящем искусстве. [Л. Я. Гинзбург. Записные книжки. Воспоминания. Эссе (1920–1943)].

В событийной каузативной конструкции различаются следующие компоненты. Подлежащее Р, пропозициональный актанта, – это **каузатор**; с этой именной группой связана ФАКТИВНАЯ ПРЕЗУМПЦИЯ, что естественно для каузативного глагола. Сказуемое Q (каузативный глагол) выражает эффект (радость, раздражение, позор и проч.), касающийся компонента Y, прямого дополнения.

В расщепленной каузативной конструкции появляется новый компонент – X: в агентивном случае это Агенса, но в принципе это может быть любой участник, обозначенный подлежащим. Например, в (5) это носитель свойства, выражаемого компонентом Р, т.е. даже не лицо. Компонент Р, маркированный в расщепленной конструкции твор. падежом, обозначает поступок Агенса, как в (26), или свойство участника-подлежащего, как в (5). Группа сказуемого Q по-прежнему выражает эффект.

Хотя расщепленная конструкция изучалась в основном на глаголах эмоции, она свойственна и другим каузативным глаголам – таким как *защитить, задержать, затруднить, избавить, испортить, омрачить, помешать, спасти, утолить, унижить, уронить* (ср. *Отказ уронил его* и *Он уронил себя отказом*) и др., см. более подробный перечень в [10].

В событийной каузативной конструкции статус пропозиционального актанта Р всегда **презумпция** – Р следует и из данного предложения, и из его отрицания:

(7) а. Ваня приход *обрадовал* нашу маму ⊃ ‘Ваня пришел’;

б. Ваня приход *не обрадовал* нашу маму ⊃ ‘Ваня пришел’.

Что же касается расщепленной конструкции, то тут ситуация иная.

2.2. Как возникает радикальное отрицание

Как было обнаружено в [10], в расщепленной каузативной конструкции пропозиция, соответствующая актанта-каузатору, может подвергаться отрицанию. В самом деле, отрицание при глаголе в составе расщепленной конструкции может иметь две интерпретации:

(8) Иван *не порадовал* нас своим возвращением =
(i) ‘То, что Иван вернулся, нас не порадовало’;
(ii) ‘Иван не вернулся, тем самым нас не порадовал’.

При интерпретации (i), со **стандартным** отрицанием, отрицается компонент эффект, ассерция утвердительного предложения, а пропозиция Р, каузатор, ведет себя как презумпция, т.е. является истинной и в исходном предложении, и в его отрицании:

(9) а. Иван *порадовал* нас своим возвращением ⊃ ‘Иван вернулся’;

б. Иван *не порадовал* нас своим возвращением ⊃ ‘Иван вернулся’.

А отрицание с интерпретацией (ii) можно назвать **радикальным** (в [10] оно названо **имплицативным**), поскольку оно отрицает не только пропозицию Q, эффект, которая была в (9а) ассерцией, но и пропозицию ‘Иван вернулся’, каузатор, которая составляла в предложении (9а) презумпцию. Путь к радикальной интерпретации отрицания в расщепленной каузативной конструкции, можно представить так.

Прежде всего следует обратить внимание на то, что интерпретация (8ii), в отличие от (8i), предполагает фразовый акцент на пропозициональном актанта:

(8i) Иван *не порадовал* нас своим возвращением;

(8ii) Иван *не порадовал* нас своим возвращением .

Следовательно, это отрицание относится к классу **СМЕЩЕННЫХ** (см. о смещенном отрицании [1, с.149ff; 2, с. 40–53]): отрицание стоит при главном предикате, но отрицается та пропозиция, которая соответствует пропозициональному актанта, поскольку на нем фразовый акцент.

Однако обычное смещенное отрицание дало бы отрицание акцентно выделенного актанта Р, каузатор, а остаток, т.е. компонент Q, эффект, отодвинулся бы в презумпцию, т.е. не подвергся отрицанию. Так, предложение *Я не начал книгу с предисловия*, со смещенным отрицанием, означает: ‘я начал не с предисловия’ [ассерция], ‘я начал с чего-то другого’ [презумпция]. Для (8ii) смещенное понимание отрицания дало бы смысл: ‘Иван не вернулся’ [ассерция, которая приняла на себя отрицание], ‘Иван нас порадовал чем-то другим’ [презумпция, которая не отрицается]. Между тем радикальное отрицание отрицает не только пропозициональный актанта Р, каузатор, но и сам каузативный глагол, т.е. компонент Q, эффект.

Можно думать, второе отрицание, в составе компонента Q, эффект, возникает за счет контекстной **идентифицирующей имплицатуры**: для (8ii) это ‘вернуться – значит порадовать’. Именно в силу этой имплицатуры мы получаем второе отрицание и интерпретацию (8ii) ‘не вернулся, ТЕМ САМЫМ нас не порадовал’. Т.е. в предложении, которое является исходным утвердительным при радикальном отрицании, не только пропозициональный актанта Р меняет статус презумпции на статус ассерции, но и компонент Q меняет статус презумпции, который

был бы положен ему при чисто коммуникативной перестройке, на статус имплицатива:

(10) Иван *не порадовал* нас своим возвращением $\surd$ = НЕВЕРНО, ЧТО (Иван вернулся, тем самым нас порадовал) =

Иван не вернулся, тем самым нас не порадовал.

В отличие от стандартного отрицания при каузативном глаголе, которое отрицает пропозицию глагола потому, что она ассерция, радикальное отрицание отрицает эту пропозицию только в силу идентифицирующей имплицатуры.

Показателен пример (11):

(11) Мы *не оскорбим* нашего учителя покупным подарком.

Радикальная интерпретация отрицания в (11) – ‘не подарим покупного подарка, **ТЕМ САМЫМ** не оскорбим’. Она основана на имплицатуре ‘подарить человеку покупной подарок – значит оскорбить его’, так что не подарить покупного подарка – это единственная возможность не оскорбить¹. Вне этой имплицатуры предложение понимается со стандартным отрицанием.

Итак, в отличие от «внешнего отрицания», которое, якобы, подавляет пресуппозицию утвердительного предложения, сводя ее ко второй ассерции, радикальное отрицание в расщепленной каузативной конструкции **не отрицает пресуппозиции**. Радикальное отрицание – это отрицание предложения, в котором пропозициональный актант каузативного глагола в результате коммуникативной перестройки получил ассертивный статус. А отрицание пропозиции самого каузативного глагола возникает как следствие идентифицирующей имплицатуры, которая порождается в контексте глаголов с определенной семантикой.

В составе расщепленной конструкции каузативный глагол утрачивает фактивность как неотъемлемую характеристику: пропозициональный актант, выраженный твор. падежом, не обязательно презумпция. Именно этим объясняется неоднозначная интерпретация приглагольного отрицания в расщепленной конструкции.

Подчеркнем отличие радикального отрицания от внешнего. Внешнее отрицание – это отрицание конъюнкции, т.е. дизъюнкция отрицаний, см. толкование (5), которое дается в разделе 1 предложению (4). А радикальное отрицание – это конъюнкция отрицаний.

2.3. Радикальное отрицание в расщепленной конструкции

В [10] в качестве примера имплицативной (иначе – радикальной) интерпретации отрицания приводится предложение (12а); для сравнения – в (12б) стандартное отрицание, сохраняющее презумпцию:

(12) а. Он хоть и проиграл в карты целое состояние, но по крайней мере *не опозорил* семью женитьбой на служанке $\surd$;

б. Он *не опозорил* $\surd$ семью женитьбой на служанке.

В (12а) для однозначно радикальной интерпретации отрицания требуется контекст, который сместит акцент на компонент Р. Возможны разные способы смещения акцента. См. однозначно радикальную интерпретацию отрицания в примерах (13)–(16), на которых видно, что отрицание, хотя оно стоит при каузативном глаголе, отрицает прежде всего пропозицию Р, выраженную подчиненным существительным в твор. падеже, которая ассертивна; отрицание предикации самого каузативного глагола порождается идентифицирующей имплицатурой. Так, в (13) отрицается шарканье, в (14) – крики и мольбы, в (15) – заем денег; в (16) отрицается наличие пуповины; в примерах ниже пропозиция Р (или ее вершина) подчеркнута (примеры из Русского национального корпуса, <http://www.ruscorgpora.ru>):

(13) Редкие невыписанные родильницы *не тревожат шарканьем* $\surd$ пустынные коридоры... [Т. Соломатина. Акушер-ХА! Байки (2009)];

(14) Брови его не сдвинуты, и тучи рассеялись, лицо его просветлело и сияет покоем, взгляд его не летит вдаль, и шум городов *не беспокоит* его криками и мольбами $\surd$. [Улья Нова. Инка (2004)];

(15) Занимать денег у Фили Герой не стал — и тем самым остался на равных со старым приятелем, *не испортил дружбу деньгами* $\surd$. [М. Чулаки. Примус // «Звезда», 2002];

(16) Еще один ребенок удивил Бритоголового тем, что вышел самым естественным образом, через родовые пути, но и он *не порадовал* его наличием пуповины $\surd$. [Л. Улицкая. Казус Кукоцкого (2000)].

Для сравнения приведем примеры стандартного отрицания в расщепленной конструкции. Пропозициональный актант Р может быть презумпцией, и тогда он, естественно, не отрицается (следует обратить внимание на то, что в (17)–(19) фразовый акцент – не в составе пропозиционального актанта, а на глаголе):

(17) Воланд рассмеялся громовым образом, но никого не испугал и смехом этим никого *не удивил* $\surd$. [М. А. Булгаков. Мастер и Маргарита. Ч. 2. (1929–1940)];

(18) По крайней мере сын-изверг *не спас* $\surd$ себя отцеубийством: он был казнен вместе с другими. [Н. М. Карамзин. История государства Российского. Т. 9. (1816–1820)];

(19) Кого я уже *не обижу* $\surd$ дотошностью своего недоброго глаза? [Нина Воронель. Без прикрас. Воспоминания (1975–2003)].

Радикальная интерпретация глагольного отрицания возникает чаще всего в контексте снятой утвердительности, см. раздел 1.3. В примерах, приведенных ниже, из контекста ясно, что отрицание каузального эффекта, выраженного в пропозиции главного предиката, есть следствие того, что не будет, не может, не должна и т.д. иметь места ситуация, заданная пропозициональным актантом Р, который является ассерцией, т.е. составляет сферу действия всех модальностей. Так, в (20) верно, что ‘он не испугает хозяйку перепадами настроения, потому что перепадов настроения у него не будет’:

¹ За этот пример я благодарна А.А. Гиппиусу, который обратил внимание на его нетривиальную имплицатуру.

(20) Он *не испугает* хозяйку перепадами настроения, оно всегда приподнято. [«100% здоровья», 2002.12.11];

(21) Нет уж, что бы там ни было <...>, она *не обрадует* узурпатора расспросами и виду не подаст, насколько ей сейчас тревожно. [Т. Соломатина. Десять месяцев, или «Комедия женских положений» (2010)];

(22) Квартирохозяин мой, почтальон, учится играть на скрипке, потому что любит свою мамашу и не хочет *огорчить* ее женитьбой. “Жена все-таки чужой человек”, – говорит он. [Максим Горький. Жизнь Клима Самгина. Ч. 1. (1925)];

(23) Кстати, чтобы *не обижать* миллионеров предложениями принять причитающиеся им за интервью деньги, исследователи предлагали им возможность внести эти средства в благотворительные фонды. [«Бизнес-журнал», 2004.01.22];

(24) Потом твой друг возвращает ее, и вы договариваетесь *не тревожить* пространство информированием третьих лиц. [Т. Соломатина. Отойти в сторону и посмотреть (2011)].

Известный призыв не унижать человека жалостью – это призыв не жалеть:

(25) Надо уважать человека! Не жалеть... *не унижать* его жалостью... уважать надо! Выпьем за человека, Барон! [Максим Горький. На дне (1902)].

Нетривиальная идентифицирующая импликатура – ‘жалеть – значит унижать’.

Противопоставление стандартного и радикального отрицания в контексте снятой утвердительности демонстрирует пример (26). В (26а), с событийной каузативной конструкцией и стандартным отрицанием, пропозиция Р, которая была презумпцией в утвердительном контексте, оказывается условием, т.е. не имеет истинностного значения. А в примере (26б), с расщепленной конструкцией, пропозиция Р отрицается, т.е. ведет себя как ассерция; отрицание радикальное:

(26) а. Отказ от игры не уронит Фишера в глазах шахматистов =

‘если Фишер откажется от игры, он не упадет в глазах шахматистов’;

б. Фишер не уронит себя отказом от игры =

‘Фишер не откажется от игры и потому не упадет в глазах шахматистов <а если откажется, то упадет>’.

Хороший тест на радикальное отрицание – невозможность замены расщепленной отрицательной конструкции на событийную. Расщепленная неотрицательная конструкция может быть заменена на событийную, см. пример (2). И расщепленная отрицательная со стандартным отрицанием тоже может, см. примеры (17)–(19). А расщепленная конструкция с радикальным отрицанием не может – поскольку в событийной конструкции, отрицание может быть только стандартное, см. пример (7). Так, она *не обрадует* узурпатора расспросами в контексте примера (21) нельзя заменить без изменения смысла на *ее расспросы не обрадуют узурпатора*.

2.4. Радикальное отрицание в контексте агентивных каузативов

Как видно из примеров в разделе 2.3, радикальная интерпретация свойственна, прежде всего, неагентивным каузативным глаголам. Отсюда внимание к глаголам эмоции: особенностью глаголов эмоции является их преобладающая неагентивность (см. [14, с. 156; 13, с. 276], там же ссылки на литературу). Основные каузативные глаголы эмоции, такие как *огорчить*, *обрадовать*, *удивить*, в норме не употребляются как глаголы действия: их парный несом. вид обозначает не деятельность (activity), а состояние. Однако есть глаголы эмоции, которые допускают употребление в агентивном значении, т.е. в значении контролируемого действия: в несом. виде они обозначают деятельность. Например, *оскорблять*, *обижать*, *развлекать*; сюда же относятся *злить*, *пугать*, *веселить*, о которых идет речь в [11]. Эти глаголы не фактивные: именной группе Р в твор. падеже в их актантажной структуре соответствует пропозиция в ассертивном статусе. Эта именная группа обозначает не автономный каузатор эмоции, а своего рода ресурс в распоряжении каузатора-Агенса: это действие, которое Агенса совершает с целью достичь определенного эффекта. Предложение (27а) имеет смысл (27б) (это **глобальное отрицание**, см. о нем в разделе 3):

(27) а. Он больше не развлекает меня *анекдотами* =

б. ‘НЕ (он по-прежнему рассказывает мне анекдоты, чтобы меня развлечь)’.

Радикальное понимание отрицания – ‘не рассказывает, чтобы не развлекать’ – в контексте *развлекать* и других агентивных глаголов положительной эмоции – по прагматическим причинам маловероятно, см. об этом различии [12, с. 224] Но при глаголах отрицательного воздействия радикальное отрицание при агентивном каузативе вполне возможно:

(28) Граф же Строганов <...> внимательно слушал каждую [лекцию] с начала до конца, никогда *не оскорбляя профессора преждевременным выходом* из аудитории. [Ф.И. Буслаев. Мои воспоминания (1897)] (= ‘не выходил преждевременно, чтобы не оскорблять’).

В неагентивном случае перефразировка для предложения с радикальным отрицанием – ‘не произошло Р, тем самым не произошло Q’, а в агентивном – ‘Х не сделал Р, чтобы не произошло Q’.

В предложении (27’) именная группа инструментального дополнения может быть понята как определенная, и тогда ее пропозиция получает статус презумпции – рассказывание анекдотов имеет место. Презумпция, естественно, не входит в сферу действия стандартного отрицания: анекдоты рассказываются, но меня не развлекают. Но в этом случае *развлекать* уже не обозначает контролируемого действия. Каузативная конструкция здесь может трактоваться как расщепленная событийная; интерпретация отрицания стандартная:

(27’) Он не развлекает меня *своими анекдотами* ≈ ‘Его анекдоты меня не развлекают’.

Итак, пример (28) показывает, что радикальное отрицание возможно также в контексте конструкции с каузативным глаголом в агентивном употреблении, которая не является результатом расщепления событийной.

2.5. Что способствует радикальной интерпретации отрицания?

Как сказано в [12, с. 214], каузативные глаголы, допускающие радикальное отрицание, «не образуют однородного класса». Но глаголы, которые допускают радикальную интерпретацию отрицания, обычно допускают и стандартную.

Интерпретация отрицания может зависеть от ДЕНОТАТИВНОГО СТАТУСА пропозиционального актанта Р (см. об этом в [11]). Так, в примере (29) с глаголом *удивить* пропозициональный актант Р определенный и ведет себя под отрицанием как презумпция, так что интерпретация отрицания стандартная, а в (30), где пропозициональный актант неопределенный, – как ассерция, и интерпретация отрицания радикальная:

(29) Новый проект *удивил* / *не удивил* нас своей беспомощностью ⊃ ‘беспомощность имеет место’ (стандартное отрицание);

(30) а. Новый проект *удивил* нас кое-какими интересными деталями ⊃ ‘были детали’;

б. Новый проект *не удивил* нас какими-либо интересными деталями ⊃ ‘не было деталей’ (радикальное отрицание).

Примеры (31), (32) тоже демонстрируют возможность двойкой интерпретации отрицания при одном и том же глаголе.

(31) а. Он *не помог* нам своим советом ⊃ ‘дал совет’ (стандартное отрицание);

б. Он *не помог* нам советом ⊃ ‘не дал совета’ (радикальное отрицание).

(32) а. На этот раз он *не раздражал* меня своими глупыми шутками ⊃ ‘глупые шутки были’ (стандартное отрицание);

б. Оставь его дома, чтобы он *не раздражал* меня глупыми шутками ⊃ ‘чтобы не было глупых шуток’ (радикальное отрицание).

Глагол *спасти* не допускает радикальной интерпретации отрицания в утвердительном контексте, см. (33а). Но в контексте снятой утвердительности отрицание все-таки может распространяться на обе пропозиции:

(33) а. Он *не спас* семью отказом от наследства ⊃ ‘отказался, но не спас’;

б. Он *не спасет* утопающего прыжком в воду ⊃ ‘он не прыгнет в воду, чтобы спасти утопающего’.

Радикальной интерпретации способствует порядок слов – поскольку он влияет на место фразового ударения, ср.:

(34) а. Отказом от игры он *не уронит* себя (скорее стандартное отрицание: не уронит, *если* откажется);

б. Он *не уронит* себя отказом от игры (скорее радикальное отрицание: не откажется, тем самым, не уронит).

Заметим, что в событийной каузативной конструкции порядок слов не влияет на интерпретацию; пропозициональный актант остается семантической презумпцией в любой позиции – и в тематической, как в (35а), и в речатической, как в (35б):

(35) а. *Ванина двойка* не огорчила маму ⊃ ‘Ваня получил двойку’;

б. Маму не огорчила *Ванина двойка* ⊃ ‘Ваня получил двойку’.

Влияние на интерпретацию отрицания оказывает также глагольный вид (роль вида рассматривается, в несколько ином плане, в [11]):

(36) а. Я *не обременю* его своей просьбой;

б. Я *не буду обременять* его своей просьбой.

Предложение (36а), с глаголом в сов. виде (будущего времени), допускает и стандартную, и радикальную интерпретацию отрицания, а в (36б), с глаголом в несов. виде, интерпретация отрицания однозначно радикальная. При несов. виде речь идет о действии (просьбе), каузирующий эффект которого (обременение) Агенсу заранее известен, так что предупредить эффект можно только несовершением действия: (36б) значит ‘не буду просить, чтобы не обременять’. А при сов. виде будущее время может пониматься и в значении намерения, и в значении предсказания: ‘моя просьба, если она будет иметь место, не обременит’. Так что сов. вид совместим и с радикальной, и со стандартной интерпретацией отрицания.

Похожее противопоставление возможно и в прош. времени: опять-таки дело в том, что предупредить эффект можно только несовершением действия. Так, (37б), с глаголом в несов. виде в узуальном значении, означает ‘не задавал вопросов, чтобы не расстраивать’; так что вопросов не было, отрицание радикальное. А в (37а) отрицание стандартное:

(37) а. Я *не расстроил* его своими вопросами;

б. Волька прикидывался, будто ничего не замечает, и *не расстраивал* старика бестактными вопросами. [Л. Лагин. Старик Хоттабыч (1955)].

2.6. От каузации к интерпретации

Группа глаголов, в которой Г.И. Кустовой было обнаружено нестандартное отрицание, охарактеризована ею как глаголы **интерпретации**; в частности, это глаголы, которые дают **косвенную номинацию** событию (см. о косвенной номинации [15]). К глаголам косвенной номинации отнесены, среди прочих, *наказать*, *отблагодарить*, *отомстить*, *поощрить*, которые часто употребляются без указания конкретного действия, являющегося средством достижения цели². Указание действия не исключено (ср. пример из НКРЯ: *поклонился женищинам, которые по шведскому обыкновению отблагодарили меня книксеном*), но в отличие от других глаголов интерпретации, например, *испортить*, см. пример (15), для глаголов косвенной номинации радикальная интерпретация отрицания является единственно возможной.

² В [16] способность самостоятельно употребляться в качестве косвенной номинации самого действия отмечается для глагола *порадовать*.

Можно думать, переход от каузальной связи между ситуациями к интерпретативной и лежит в основе перехода от стандартного понимания отрицания к радикальному. Каузальная связь предполагает существование причины, несет презумпцию отдельного существования причины. А интерпретация как бы отождествляет причину и следствие, превращает разные события в разные номинации одного и того же. Это и есть та идентифицирующая импликатура, которая дает возможность отрицать две пропозиции с помощью одной отрицательной частицы.

Можно сказать иначе: идентифицирующая импликатура сводится к тому, что событие Р из достаточного условия ситуации Q становится также необходимым. Этот переход достаточно широко распространен в разных языках, см. [17]. Русский союз *если* обычно понимается в значении ‘если и только если’; например, фраза *Если попросишь, приду* подразумевает ‘а если нет, то нет’, см. [6, с. 44]: «Понимание условного предложения как выражающего не только достаточное, но и необходимое условие возникает как следствие постулата информативности» (имеются в виду коммуникативные постулаты, *conversational implicatures*, по П. Грайсу, [18]).

В агентивном случае то, что было названо идентифицирующей импликатурой каузативной ситуации, сводится к убеждению говорящего (или Агенса) в том, что Р неминуемо вызовет Q, так что избежать Q можно только не совершив Р. Это объясняет радикальную интерпретацию отрицания в контексте примера (36б), где несом. вид глагола в Q выражает тот факт, что Р – контролируемое действие с предсказуемым эффектом.

О каузативной ситуации с предсказуемым эффектом упоминает Г.И. Кустова – на примере глаголов «косвенного результата», типа *опозорить, разорить*, которые связывают две ситуации: контролируемое Р и нежелательное Q. В контексте отрицательного императива несом. вида таких глаголов «речь идет об определенном и конкретном Р, и избежать Q – целиком во власти Х-а (поскольку известно, что Р вызывает Q, достаточно просто не делать Р)» ([12, с. 226]). Так, императив от глагола в составе Q – *Не позорь отца / Не разоряй семью* – на самом деле означает ‘не делай Р’ / ‘прекрати делать Р’ (например: *Не женись на служанке, Перестань играть в карты*), поскольку Р приведет к Q.

Отметим, однако, что в предложении с радикальным отрицанием каузатор Р может не быть действием, как в примерах (13) или (16); не быть действием с предсказуемым для Агенса эффектом, как в (8); не быть действием, отказ от которого имеет целью произвести эффект. Так, в строчке Б. Окуджавы (пример из [13, с. 299]) имеется в виду ‘не буду лить слез, и тем самым не оскорблю’, а не ‘не буду лить слез, чтобы не оскорбить’:

(38) *Не оскорблю* своей судьбы слезой поспешной и напрасной.

2.7. Радикальное отрицание за пределами каузативной конструкции

В [19] описано нетривиальное поведение под отрицанием глагола *повезти*, в безличном употреблении. Этот глагол употребляется в двух разных конструкциях (имеется третья, которую мы здесь опускаем). В (39), где пропозициональный акт глагола вводится союзом *что*, обычное отрицание, сохраняющее презумпцию.

(39) а. Ему *повезло* \ , что он встретил эту женщину = ‘от события Y, которое произошло с X-ом, X-у <было> хорошо’;

б. Ему *не повезло* \ , что он встретил эту женщину = ‘от события Y, которое произошло с X-ом, X-у не <было> хорошо’.

А отрицание в (40) можно трактовать как радикальное:

(40) а. Ему *повезло* \ родиться богатым \ = ‘с X-ом произошло хорошее событие Y, тем самым, ему <было> хорошо’;

б. Ему *не повезло* \ родиться богатым = ‘с X-ом не произошло хорошее событие Y, тем самым, ему не <было> хорошо’.

Радикальная интерпретация отрицания возникает за счет того, что каузальная связь между двумя разными событиями, которая ощущается в (39), в предложении (40) вырождается в их идентификацию: каузация одного события другим переходит в разные наименования одного и того же события.

3. ГЛОБАЛЬНОЕ ОТРИЦАНИЕ

Итак, радикальное отрицание отрицает пропозицию, которая при других употреблениях того же глагола является презумпцией. Похожими свойствами обладает и **отрицание с широкой сферой действия** [2]. (В [7] используется в качестве синонима термин **глобальное отрицание**.)

З а м е ч а н и е. Термин **широкая сфера действия** не вызывает возражений, поскольку *широкая* может пониматься как расширенная. Но термин **узкая сфера действия** не представляется уместным в применении к стандартному отрицанию, которому естественно не воздействовать на презумпцию.

Отрицание с широкой сферой действия хорошо демонстрируется на предложениях с обстоятельством цели, пример из [2]:

(1) Трамвай остановился, чтобы посадить пассажиров.

У предложения (1) есть два разных отрицания – в силу неоднозначности его коммуникативной структуры.

При одной структуре в предложении различаются две пропозиции, связанные аргументно-предикатным отношением, причем пропозиция адвербиала является ассерцией, а пропозиция финитного глагола – пропозициональным актантом адвербиала и пресуппозицией: адвербиал является модификатором ситуации, описываемой финитным глаголом. При этой коммуникативной структуре предложению соответствует стандартное отрицание, сохраняющее пресуппозицию. См. (2а) и его отрицание (2б):

(2) а. Трамвай остановился[↗] чтобы высадить пассажиров[↘] =

[‘трамвай остановился’ – презумпция, ‘цель – высадка пассажиров’ – ассерция]

б. Трамвай остановился *не* для того, чтобы высадить пассажиров.

При другой коммуникативной структуре две пропозиции объединяются в одну: модификатор пропозиции финитного глагола превращается в актант глагола, а сама пропозиция теряет свою отдельность и презумптивный статус. Эта единая пропозиция и отрицается. Это отрицание с широкой сферой действия, иначе – глобальное:

(3) Трамвай *не* остановился, чтобы высадить пассажиров[↘]

= НЕ (Трамвай остановился, чтобы высадить пассажиров).

Предложение (27) из раздела 2 демонстрировало именно глобальное отрицание.

Радикальное и глобальное отрицание, будучи разными конструкциями, обладают следующим сходством. В обоих случаях отрицание расширяет свою сферу действия, потому что две пропозиции объединяются в одну. В случае глобального отрицания объединение происходит на синтаксическом уровне: модификатор глагольной пропозиции превращается в актант глагола. А в случае радикального отрицания – на семантическом, за счет идентификации события-каузатора и эффекта каузации.

ЗАКЛЮЧЕНИЕ

Итак, имеются следующие варианты интерпретации отрицания в контексте каузативного глагола.

В событийной каузативной конструкции пропозициональному актанту каузативного глагола всегда соответствует презумпция, и в этом контексте единственно возможным является стандартное отрицание, которое сохраняет пропозицию каузатор, презумпцию, и отрицает пропозицию эффект, ассерцию (*Его женитьба на служанке не опозорила семью*).

В расщепленной каузативной конструкции возможно, помимо стандартного отрицания, сохраняющего презумпцию, радикальное отрицание, отрицающее и пропозицию каузатор, и пропозицию эффект (*Иван не порадовал нас возвращением*).

Радикальное отрицание в контексте агентивного каузатива отличается тем, что каузатор – это не просто событие, а действие, сознательный отказ от которого имеет целью предупредить эффект (*Я не буду обременять его своей просьбой*).

Один из выводов работы состоит в том, что фактивность глагола, т.е. способность порождать однозначно презумптивный статус пропозиционального актанта, зависит от конструкции: в контексте расщепленной каузативной конструкции глагол утрачивает свою фактивность, так что будет ли пропозициональный актант презумпцией, зависит от денотативного статуса этого актанта.

В [10] рассматривается еще один класс каузативных глаголов, поведение которых под отрицанием в существенной степени определяется тем, что один из компонентов в их семантической структуре имеет

статус презумпции. Это глаголы типа *решишь*, *уговорить*, *поймать*; про них в [20, с. 64] говорится, что у них деятельность, описываемая глаголом несов. вида, составляет пресуппозицию в семантике глагола сов. вида, у которого ассерцией является достижение результата. Так, *не решил* означает ‘решил, но не получил результата’, так что *не решил* значит отчетливо не то же самое, что *не решил*. В [21, с. 89] приводится представительный список глаголов с этой особенностью.

В [22, с. 112] в применении к глаголам, описанным Ю.Д. Апресяном и М.Я. Гловинской, используется термин **конатив**, от лат. *conatus* ‘попытка’. (Примерно те же глаголы называются конативами в [23, с. 79–88; с. 110–132ff])

Конативы противопоставлены глаголам с **накоплением результата** [21, с. 83], таким как *забить* <гвоздь> или *открыть* <окно>. У них компонент «деятельность» является ассертивным – как и компонент достижения результата. Можно думать, в семантике не-конативных глаголов компонент «результат», который связан каузальной связью с обращенной на него деятельностью, является имплицативом, так что отрицание в контексте не-конативного глагола будет радикальное. Правда, компонент «недостигнутый результат» тут практически избыточен: *не открыл окно* = ‘не действовал с целью <предполагаемой этим действием>, тем самым, не достиг ее’. Отсюда квазисинонимия видов: *не открыл окно* ≈ *не открывал окна*, см. [25].

В [9] утверждается, что презумптивный статус компонента «деятельность» в семантике конативов обеспечивается модификатором «большое усилие». Однако в таком случае конативы не являются классом глаголов, противопоставленным действиям с накоплением результата. Так, в семантику глагола *приволачить* входит компонент «большое усилие»; но он не конатив – *не приволок* не предполагает ‘приволакивал’. Глагол сов. вида *дотащить* имеет в контексте отрицания презумпцию деятельности, тоже требующей специальных усилий – из *не дотащил* следует ‘тащил’; но несов. вид от *дотащить* – *дотаскивать*, а не *тащить*, так что *тащить* и *дотащить* не видовая пара.

В принципе, почти любое действие, например, открывание окна, может окончиться неудачей, и тогда предполагаемая его семантикой деятельность окажется (неудавшейся) попыткой. Так что правомерно также использование термина конатив для характеристики одного из частных видовых значений (как в [24, с. 27]), т.е. для употребления глагола, а не для класса глаголов.

Так или иначе, отрицание в контексте неконативного каузативного глагола действия (физического, ментального и проч.) – это вариант радикального отрицания, хотя и вырожденный.

Радикальное отрицание в расщепленной каузативной конструкции – это нестандартная интерпретация смещенного предикатного отрицания, которая закономерно возникает в контексте идентифицирующей имплицатуры, когда эффект отождествляется с событием – является косвенной номинацией события. Отрицается прежде всего ассерция. Так что ответ на во-

прос, поставленный в заглавии, отрицательный: презумпции не подвергаются отрицанию в составе общепризнательного предложения.*

СПИСОК ЛИТЕРАТУРЫ

1. Падучева Е. В. О семантике синтаксиса. Материалы к трансформационной грамматике русского языка. – 2-е изд. – М.: Наука, 1974; М.: УРСС, 2007.
2. Богуславский И. М. Исследования по синтаксической семантике. – М.: Наука, 1985.
3. Horn L. R. A natural history of negation. – Chicago: University of Chicago press, 1989.
4. Kempson R. M. Presupposition and the delimitation of semantics. – Cambridge: Univ. Press, 1975.
5. Weinreich U. On the Semantic Structure of Language // Universals of Language / ed. J. Greenberg. – Cambridge, MA: MIT Press, 1963. – P. 114–171; переизд.: Weinreich U. On Semantics. – Philadelphia: Univ. of Pennsylvania Press, 1980. – P. 37–96; рус. пер.: Вейнрейх У. О семантической структуре языка // Новое в зарубежной лингвистике. Вып. V. Языковые универсалии / ред. В. А. Звегинцев. – М.: Прогресс, 1970. – С. 163–249.
6. Падучева Е. В. Высказывание и его соотнесенность с действительностью. – М.: Наука, 1985; 6-е изд. – М.: ЛКИ, 2010. – URL: <http://lexicograph.ruslang.ru/TextPdf1/paducheva1985.pdf>.
7. Падучева Е. В. Эффекты снятой утвердительности: глобальное отрицание // Русский язык в научном освещении. – 2005. – № 2(10). – М.: Языки славянской культуры, 2005. – С. 17–42. – URL: <http://lexicograph.ruslang.ru/TextPdf2/rjns2005.pdf>.
8. Giannakidou A. Licensing and Sensitivity in Polarity Items: From Downward Entailment to (Non)Veridicality // Chicago Linguistic Society. – 2001. – Vol. 39.
9. Beaver D., Geurts B. Presupposition // The Stanford Encyclopedia of Philosophy / ed. E. Zalta. – Stanford, CA, 2011. – URL: <https://webpace.utexas.edu/dib97/sep-v3.2.pdf>.
10. Кустова Г. И. О коммуникативной структуре предложений с событийным каузатором // Московский лингвистический журнал. Т. 2. – М.: РГГУ, 1996. – С. 240–261.
11. Апресян В. Ю. Семантика эмоциональных каузативов: статус каузативного компонента // Компьютерная лингвистика и интеллектуальные технологии: По материалам Международной конференции «Диалог 2013». Вып. 12(19). – М.: РГГУ, 2013. – С. 43–57.
12. Кустова Г. И. Типы производных значений и механизмы языкового расширения. – М.: Языки русской культуры, 2004.
13. Падучева Е. В. Динамические модели в семантике лексики. – М.: Языки славянской культуры, 2004. – URL: <http://lexicograph.ruslang.ru/TextPdf1/PaduDinamMod2004.pdf>.
14. Арутюнова Н. Д. Предложение и его смысл. – М.: Наука, 1976.
15. Зализняк Анна А. *Считать и думать*: два вида мнения // Логический анализ языка. Культурные концепты. – М.: Наука, 1991. – С. 187–194.
16. Зализняк Анна А., Левонтина И. Б., Шмелев А. Д. Лексика радости // Константы и переменные русской языковой картины мира. – М.: Языки славянских культур, 2012. – С. 462–470.
17. Geis Michael L., Zwicky Arnold M. On invited inferences // Linguistic Inquiry. – 1971. – Vol. 2(4). – P. 561–566. – URL: <http://www.jstor.org/stable/4177664>.
18. Grice H. P. Logic and conversation // Syntax and Semantics. Vol. 3. Speech Acts. – N. Y.: Academic Press, 1975; рус. пер.: Грайс Г. П. Логика и речевое общение // Новое в зарубежной лингвистике. Вып. 16. Лингвистическая прагматика. – М.: Прогресс, 1985. – С. 217–237.
19. Апресян Ю. Д. Семантическая структура слова и его взаимодействие с отрицанием // Международная конференция «Диалог», 26–30 мая 2010. – М.: РГГУ, 2010.
20. Апресян Ю. Д. Типы информации для поверхностно-семантического компонента модели «Смысл ↔ Текст» // Wiener Slawistischer Almanach, SBd 1. – Wien, 1980.
21. Гловинская М. Я. Семантические типы видовых противопоставлений русского глагола. – М.: Наука, 1982.
22. Падучева Е. В. Семантические исследования: Семантика времени и вида в русском языке. Семантика нарратива. – М.: Языки русской культуры, 1996. – URL: <http://lexicograph.ruslang.ru/TextPdf1/PaduSemantIssl1996.pdf>.
23. Томмола Х. Аспектуальность в финском и русском языках // Neuvostoliitto instituutin vuosikirja. № 28. – Helsinki, 1986.
24. Бондарко А. В. Вид и время русского глагола. – М.: Просвещение, 1971.
25. Имперфектив отрицания в русском языке // Вопросы языкознания. – 2008. – № 3. – С. 3–21. – URL: http://lexicograph.ruslang.ru/TextPdf1/imperf_negation_VJa1.pdf.

Материал поступил в редакцию 13.11.13

Сведения об авторе

ПАДУЧЕВА Елена Викторовна – доктор филологических наук, профессор, главный научный сотрудник ВИНТИ РАН, Москва
e-mail: elena.paducheva@yandex.ru

* Выражаю огромную благодарность Барбаре Парти за конструктивную критику этой работы на разных ее этапах.