Компьютерные инструменты в образовании, 2021
№ 1: 68–84
УДК: 001.8:519.1:378
http://cte.eltech.ru
doi:10.32603/2071-2340-2021-1-69-85

ОНТОЛОГИЯ ДИСКРЕТНОЙ МАТЕМАТИКИ В ОБРАЗОВАНИИ

Молотков И. И.[1], студент, ✉ molotkov.ivan.igorevich@gmail.com
Новиков Ф. А.[2], доктор технических наук, профессор, ✉ fedornovikov51@gmail.com

[1]Санкт-Петербургский национальный исследовательский Академический университет им. Ж. И. Алфёрова Российской академии наук, ул. Хлопина, дом 8, корпус 3, литер А, 194021, Санкт-Петербург, Россия
[2]Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), ул. Профессора Попова, 5, корп. 3, 197376, Санкт-Петербург, Россия

Аннотация

В настоящее время онтологии широко используются в информатике для формализованного представления знаний о различных предметных областях. Разработаны и успешно применяются специальные формальные языки описания онтологий, которые позволяют описывать онтологии в форме, доступной для использования как человеком, так и компьютером. Среди разнообразных вариантов использования онтологий особое место занимает применение онтологий в образовании, поскольку систематизация и упорядочение знаний, будучи главным конкурентным преимуществом онтологического подхода, одновременно является одной из главных целей образовательного процесса. В статье предложены оригинальные приёмы построения онтологий для использования в образовательном процессе высшей школы. Центральной идеей является построение фасетных, иначе говоря, — многогранных онтологий, в которых различные аспекты одной и той же предметной области описываются концептуально схожими, но синтаксически различными средствами. Такой подход обеспечивает более точное и семантически адекватное описание при сохранении известной лаконичности и наглядности обозначений. В качестве языка описания онтологий предлагается использовать унифицированный язык моделирования UML 2, прекрасно зарекомендовавший себя при формализации во многих случаях. Изложение ведётся на примере построения онтологии дискретной математики, причём приводимые в статье диаграммы онтологий внедрены в учебные процессы Академического и Политехнического университетов Санкт-Петербурга.

Ключевые слова: *онтология, учебный процесс, формализация знаний, систематизация знаний, дискретная математика, обучение, UML, RDF, RDFS, OWL.*

Цитирование: Молотков И. И., Новиков Ф. А. Онтология дискретной математики в образовании // Компьютерные инструменты в образовании. 2021. № 1. С. 68–84. doi: 10.32603/2071-2340-2021-1-69-85

1. ВВЕДЕНИЕ. ЧТО ТАКОЕ ОНТОЛОГИЯ?

Термин «онтология», буквально «учение о сущем», возник в философии около 600 лет назад. Изначально это понятие было введено как эквивалент понятию «метафизика».

Теперь же термин «метафизика» используется для обозначения раздела философии, изучающего первоначальную природу реальности. Онтология — подраздел метафизики, занимающийся бытием, существованием [1].

Понятие «онтологии» в информатике в чем-то опирается на философию, но есть очень важные отличия. В нашем понимании, в информатике онтология — это способ целостной формализации знаний о какой-то предметной области, что вполне согласуется с представлениями Тома Грубера [2], родоначальника использования термина «онтология» в информатике: «онтология есть явная (имплицитная) спецификация концептуализации» [3]. Мы лишь позволяем себе рассматривать в онтологиях не только понятия (концепты) и отношения между ними, но и иные нужные объекты, например алгоритмы, а также употребляем термин формализация в качестве расширения идеи спецификации, поскольку формализация — это не только формальное описание, но и возможность что-то вычислять.

Онтологии прошли большой путь развития. На первых этапах в качестве онтологий использовались семантические сети и другие способы представления знаний. Особенно важно, что онтологии можно выразить в графической форме. На рис. 1 показан пример онтологии в графической форме из статьи [4].

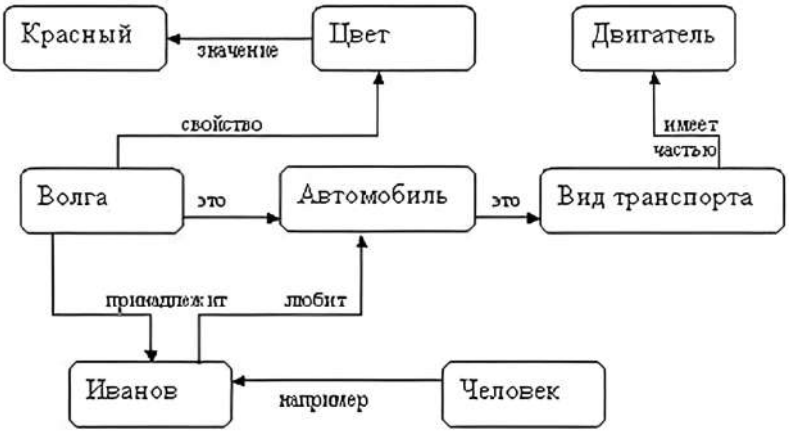

Рис. 1. Пример семантической сети

На приведённом рисунке есть понятия: «Человек», «Волга», «Автомобиль», «Красный», есть отношения между понятиями: «Иванов — любит — автомобиль». Так мы формализовали знание о том, что у Иванова есть красная Волга, которую он любит.

Уже этот наивный пример показывает, что существуют проблемы, которые необходимо решить для успешного применения онтологий. Среди этих проблем:

— неоднозначность понятий (например, «Волга» — это название реки, или «Волга» — это марка автомобиля?);
— неясность обобщений (нет различий между классами и экземплярами классов; неясно, например, в чем отличие отношения «Человек — например — Иванов» от отношения «Волга — это — автомобиль»);
— произвольность обозначений (например, нет указаний на то, какими могут быть объекты, отношения, их обозначения).

Существуют и с успехом используются различные способы решения как перечисленных, так и оставленных за кадром проблем, связанных с прагматикой онтологий. Далее

мы кратко перечисляем известные подходы и показываем предлагаемый нами способ решения этих проблем с помощью унифицированного языка моделирования UML.

2. СПОСОБЫ ПРЕДСТАВЛЕНИЯ ОНТОЛОГИЙ

Наш подход к построению онтологий не единственный и не первый — человечество изобрело и с успехом применяет различные способы представления областей знаний.

Таксономия — учение о принципах и практике классификации и систематизации сложноорганизованных иерархически соотносящихся сущностей.

Таксономии используют понятия и отношения наследования между ними. Можно воспринимать это так — у какого-то класса есть атрибуты, все его дочерние классы наследуют эти атрибуты, кроме этого, могут иметь еще какие-то свои. На языке дискретной математики можно сказать, что математической моделью таксономии является лес ориентированных корневых деревьев.

Таксономии используются в биологии [5], медицине, информационном поиске. Таксономии также могут использоваться в объектно-ориентированном программировании, так как эта парадигма оперирует классами, которые могут наследовать друг другу — то есть, существует иерархия.

Однако таксономии имеют свои существенные ограничения. Рассмотрим концептуальную модель торговли. В ней есть понятия «Магазин», «Товар», «Поставщик». Эти три понятия не являются обобщениями друг друга, но при этом между ними есть зависимость. Эту зависимость не может показать таксономия — нужно что-то более общее.

Реляционная модель данных — хорошо известный способ решить возникшую проблему. В реляционной модели существуют классы, атрибуты у классов, а также любые отношения между классами (кроме этого, для отношений можно указывать, сколько экземпляров каждого класса участвует в отношении).

У такого подхода есть явные плюсы — можно выразить информацию о любых отношениях между понятиями. Но при этом есть проблема — нет почти никаких ограничений. Возможно, для задач обучения нам не нужны абсолютно любые отношения в онтологиях, а достаточно будет всего нескольких. Из-за отсутствия ограничений на отношения представление может быть менее наглядным. Кроме этого, не существует проверки на корректность (если нет ограничений, ничего нельзя нарушить).

Поэтому для конкретных задач зачастую требуется найти компромисс между выразительностью и ограничениями — нужно решить, какие отношения оставить, а какие убрать, какие свойства должны быть у отношений, какими могут быть объекты и т. д. Для целей описания онтологий разработана целая серия специальных языков.

2.1. Язык RDF (Resource Description Framework)

Самый простой их всех языков, изобретен в 1990-х годах [6]. Вся информация представляется в нем в виде набора триплетов «субъект — предикат — объект».

Субъекты и объекты — это сущности, которые можно визуализировать вершинами некоторого графа, а предикат — это бинарное отношение между сущностями, которое можно показать дугой. Таким образом, всю онтологию, то есть множество триплетов RDF, можно показать в виде диаграммы ориентированного графа.

RDF — слишком примитивный язык, в котором нет типизации отношений, все дуги одинаковые — это не очень хорошо с точки зрения выразительности при построении онтологии.

2.2. Язык RDFS (RDF Schema)

Расширение языка RDF [7]. В языке RDF все возможности отображать отношения между понятиями исчерпываются именованными предикатами, при этом про сами предикаты ничего конкретного сказать нельзя. В языке RDFS появились новые возможности. Можно указать, что один класс является подклассом другого, одно свойство (предикат) — частным случаем другого свойства. Таким образом, RDFS имеет возможность специфицировать обобщения как для сущностей, так и для отношений. Также для свойства можно указывать область определения и значения.

2.3. Язык OWL (Web Ontology Language)

Это язык [8] включает в себя все функции RDFS. Кроме этого, можно накладывать еще очень много ограничений и на свойства (отношения), и на классы (сущности). Можно, например, сказать, что отношение должно быть транзитивным, рефлексивным, симметричным и т. д.

В языке OWL также присутствуют конструкторы классов — можно создавать новые классы объединением, пересечением, дополнением старых. Можно задавать классы через ограничения свойств. Видимо, OWL — наиболее богатый язык для описания онтологий из специализированных языков на сегодняшний день.

2.4. Унифицированный язык моделирования UML (Unified Modeling Language)

Для описания онтологий подходит унифицированный язык моделирования UML [9]. Диаграмма классов UML включает в себя все возможности RDF, RDFS и OWL, хотя некоторые конструкции в UML выглядят более запутанными, а потому реже используются.

В диаграммах классов в UML есть классы с атрибутами, есть разные отношения между классами: ассоциация, обобщение, реализация, зависимость, агрегация, композиция. Обобщение — то же, что и подклассы из RDFS и OWL, реализация, как и ассоциация, — тоже есть в RDFS и OWL. Агрегации/композиции (отношение целое — часть) в OWL и RDFS нет, хотя это достаточно полезное отношение. Также наличие такого встроенного типа отношений как зависимости в UML нам представляется очень полезным. Можно показать, что какой-то класс выражается через некоторые другие, но, в отличие от конструктора классов OWL, это выражение не обязательно приводить явно. И это намного полезнее — строгий вывод зависимого класса из некоторых независимых зачастую может быть очень сложен, поэтому нам более важна информация, что он в принципе возможен, чем то, как конкретно этот вывод делается. Поэтому мы полагаем, что наличие отношения «зависимость» в UML для онтологий намного полезнее конструктора классов в OWL. Таким образом, агрегация, композиция и зависимость — отношения, которые часто применимы при описании онтологий, при этом присущи они только UML.

3. ОНТОЛОГИИ ДЛЯ ОБУЧЕНИЯ

В настоящее время общепризнано, что онтологии в информатике должны быть машиночитаемыми и должны быть направлены на достижение определенной цели. Нашей целью является использование онтологий для обучения. Онтологии могут быть полезны в образовательном процессе. Мы предлагаем использовать их в качестве визуального материала наряду с презентациями для лекций.

Онтологии позволяют:

1) вынести все изученные понятия в одну схему — так учащимся проще понимать, что вообще было пройдено;
2) показать иерархию понятий — какие у рассматриваемых сущностей есть специализации и обобщения;
3) отразить, как одни математические объекты могут преобразовываться в другие, то есть, какие существуют алгоритмы.

Наш опыт использования онтологий в образовании распространяется на дискретную математику [10] и смежные дисциплины. Приводимые примеры взяты из поддерживаемого нами ресурса [11]. Рассмотрим диаграмму, которой завершается лекция «Функции» (рис. 2).

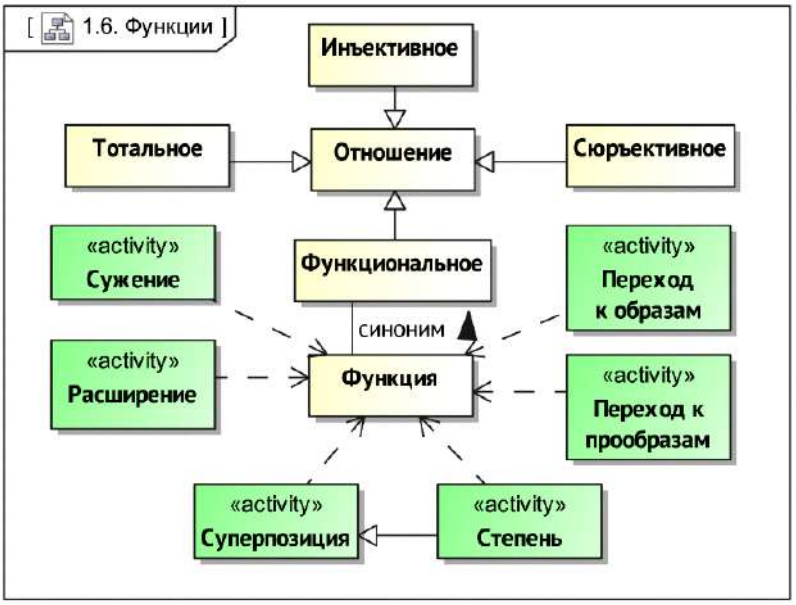

Рис. 2. Пример диаграммы, показывающей фрагмент онтологии дискретной математики

Сразу видно, о чем была лекция — это виды отношений, операции с функциями. Кроме этого, наглядно видна иерархия понятий: тотальные, сюръективные, инъективные и функциональные отношения суть подклассы класса отношений.

Таким образом, студентам онтологии могут упростить процесс изучения нового материала, структурировать знания по предмету, помочь подготовиться к экзамену. Для преподавателя онтологии полезны, так как они могут быть использованы как план лекций, могут быть выданы студентам в качестве универсального конспекта и т. д.

Важным обстоятельством является то, что онтологии на языке UML могут и должны быть подготовлены с помощью соответствующего программного инструмента. Это означает, что онтологии на UML являются не просто картинками, а являются машиночитаемыми артефактами, моделями, которые допускают компьютерную обработку.

Вообще говоря, обработка моделей в UML, в том числе онтологий, ничем не ограничена. Наличие в UML встроенного логического языка первого порядка OCL и наличие метамодели UML, то есть исчерпывающей спецификации UML на UML, позволяет строить программные инструменты для автоматической генерации исполняемого кода, авто-

матического доказательства свойств моделей и многое другое [12]. Эти возможности реализованы и используются во многих инструментах в специфических проявлениях.

В применении онтологий в обучении мы активно используем возможность управлять презентацией онтологии за счет фильтрации моделей (такая возможность есть во всех инструментах, поддерживающих UML). Имея единую модель онтологии, составленную с максимальной степенью подробности, преподаватель в конкретной презентации может показывать или скрывать на диаграммах те или иные подробности и детали в зависимости от уровня подготовки аудитории и задач лекции.

4. ТРЕБОВАНИЯ К ОНТОЛОГИИ ДИСКРЕТНОЙ МАТЕМАТИКИ

Мы используем онтологии для конкретной области знаний — дискретной математики. При этом онтологии создаются преподавателями, используются студентами технических вузов в качестве конспектов. Исходя из этого, нужно понять, что мы хотим от нашей модели.

Так как онтологии используются в учебном процессе, отдельные диаграммы не должны быть слишком объёмными (практическое правило — 7 ± 3 сущностей на одной диаграмме). При этом их будут смотреть студенты, которые уже должны быть знакомы с материалом. Диаграммы нужны студентам не для первоначального знакомства, а для повторения, структурирования изученного. Соответственно, онтологии должны быть достаточно общими, абсолютно точными, но не очень подробными.

Теперь нужно определиться с тем, какие элементы онтологий наиболее полезны для формализации дискретной математики.

- **Именованные сущности.** Дискретная математика, как и большинство других областей знания, оперирует понятиями, которые адресуют множества однотипных объектов. Например, «множество», «граф», «функция».
- **Типизированные атрибуты.** У математических объектов могут быть различные атрибуты. Например, «мощность множества», «область определения функции».
- **Часть — целое.** Некоторые объекты могут быть частью других. Например, «вершина графа», «элемент множества».
- **Общее — частное.** В математических понятиях много иерархий. Например, «монотонная функция» — подкласс класса «функция», а «строго возрастающая функция» — подкласс класса «монотонная функция».
- **Зависимости определений.** У понятий есть определения. Эти определения индуцируют зависимости между понятиями, причём эти зависимости не должны образовывать циклов. Например, определение «простой цикл в графе — это такое множество вершин графа, в котором любая вершина является смежной по отношению в точности к двум другим вершинам из этого множества», апеллирует к понятиям «множество», «граф», «вершина» и «смежность», которые должны быть определены ранее. При этом порядок определений, что определять раньше, а что позже, во многом является субъективным выбором автора учебного курса, а потому явная демонстрация этого порядка важна для студента.
- **Алгоритмические аспекты.** Наряду со статической системой понятий, в дискретной математике существуют различные алгоритмы. Например, есть много алгоритмов обхода и анализа графов, алгоритмов проверки выполнимости булевых формул и т. д. Студенту необходимо уверенно идентифицировать алгоритмы, а также по-

нимать, какие объекты подаются на вход алгоритмов и какие объекты получаются в результате выполнения алгоритмов.

- **Специфические ассоциации.** Встречаются отношения между понятиями, которые выносить в отдельный класс нежелательно (так как такой класс будет очень немногочисленным), но отразить их в онтологии желательно. Для этого следует иметь некоторое «обобщенное» отношение ассоциации, которое можно уточнять, чем оно является в каждом конкретном случае. Например, заводить специальное обозначение для отношения «инцидентно» — плохо, потому что встречается это только в некоторых темах, но показать такую информацию иногда хочется, например, «вершина инцидентна ребру».
- **Комментарии.** Полезно иметь возможность комментировать, оставлять заметки. Это позволяет в любом случае включить в онтологию какую-то важную информацию, даже если специальных формальных средств для представления этой информации не предусмотрено.

5. ИСПОЛЬЗОВАНИЕ UML ДЛЯ ПОСТРОЕНИЯ ОНТОЛОГИЙ

Покажем, как можно использовать UML в нашей задаче. Для этого рассмотрим, какие сущности есть в диаграммах классов UML, и соотнесем их с упомянутыми выше требованиями к онтологиям.

1. **Классы и атрибуты.** Классами являются математические объекты — понятия вроде «граф», «бинарное отношение» и т.д. Как и говорилось в требованиях, атрибуты могут быть полезны для указания тех свойств сущности, которые присущи самой сущности, но не являются связями с другими сущностями (мощности у множеств, меры связности у связных графов, рис. 3).

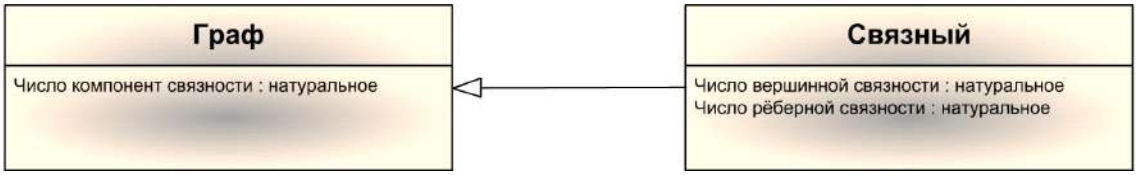

Рис. 3. Классы с атрибутами и отношение обобщения между ними

2. **Отношения между классами.** Мы используем почти всю номенклатуру отношений UML (рис. 4). Обобщение используется, чтобы показать иерархию: граф — связный граф. Агрегация используется, чтобы показать отношение целое — часть: множество — элемент. Композиция — отношение, похожее на агрегацию, отличие в том, что в отношении композиции без части целое бессмысленно (вершины — часть графа, но без вершин нет графа), а при агрегации целое имеет смысл без частного (элемент — часть множества, но без элемента множество все равно остается множеством). Зависимость используется для представления определений математических терминов: «формальный язык — это множество слов», поэтому определение формального языка можно выразить через понятия «множество» и «слово». Реализация — отношение, показывающее, что класс объектов является воплощением другого абстрактного понятия. К примеру, множество абстрактное понятие, а представление множества в виде битовой шкалы — реализация. Ассоциация — широкий класс отношений, в который фактически включены все остальные отношения, которые могут понадобиться.

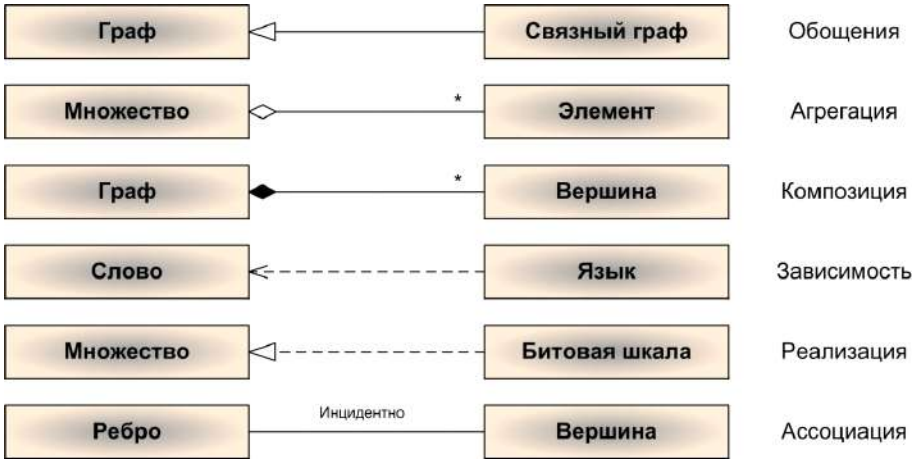

Рис. 4. Отношения между сущностями в онтологиях

3. **Действия.** Очевидно, что в онтологии нужна возможность показывать на диаграмме спецификации алгоритмов: что поступает на вход алгоритма, что выдается на выходе. Это можно сделать в UML несколькими способами. Первый способ, самый естественный, заключается в том, чтобы считать алгоритм единичным действием и показать, какие объекты поступают на вход и какие объекты появляются на выходе (рис. 5, сверху). Другими словами, объяснить алгоритм «на примере». Используя нотацию контактов (pins), которые появились в UML 2, при этом возможно дать имена параметрам алгоритма и включать эти имена в спецификацию входов и выходов алгоритма. К сожалению, при этом затруднительно показывать алгоритмы на тех же диаграммах, что и иерархию понятий, поскольку это нарушает синтаксические правила канонических диаграмм UML (на диаграммах классов допустимы классы, но не допустимы действия с контактами). Другой способ состоит в том, чтобы ввести специальный стереотип сущностей «activity», который допустим на диаграмме классов, и связывать такую сущность с классами входных и выходных объектов указанием потока данных (рис. 5, снизу). Этот способ более формальный, но менее наглядный.

Рис. 5. Два способа спецификации входов и выходов алгоритмов

4. **Комментарии, пакеты и рамки.** Очень часто при визуализации на диаграмме желательно разместить важную, но не формализованную информацию. Например, имя автора онтологии и дата создания, целевая аудитория, для которой составлена онтология и т.п. Для этого мы используем соответствующую графическую конструкцию UML — прямоугольник с загнутым уголком. Наглядность диаграмм можно повысить, если объединять однородные сущности в группы и давать им имена. Для этой цели мы используем группирующее средство UML — пакет — прямоугольник с ярлычком снаружи. Наконец,

для целей идентификации диаграмм в большой онтологии очень удобно пользоваться рамками (прямоугольник с ярлычком в левом верхнем углу), которые появились в версии UML 2. На рис. 6 приведён пример.

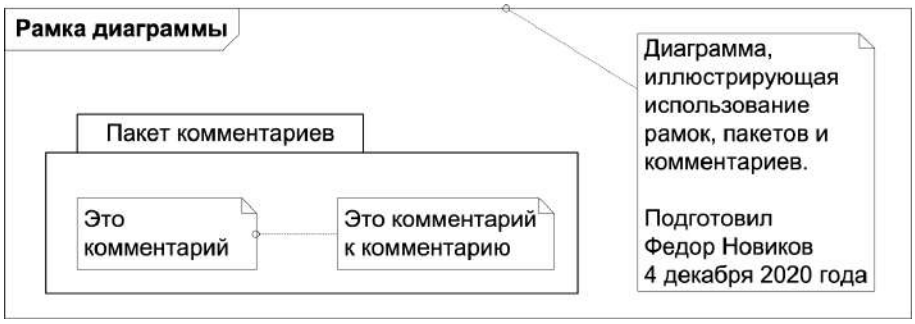

Рис. 6. Использование комментариев, пакетов и рамок

Казалось бы, можно на этом закончить. Но на практике возникают определенные проблемы. А именно, если все сущности и отношения одной онтологии пытаться располагать на одной диаграмме, то может быть слишком много стрелочек и фигур, диаграмма окажется перегруженной.

6. РЕШЕНИЕ ПРОБЛЕМЫ ПЕРЕГРУЖЕННОСТИ ДИАГРАММЫ

Чтобы понять, в чем проблема, рассмотрим некоторые неудачные схемы (рис. 7).

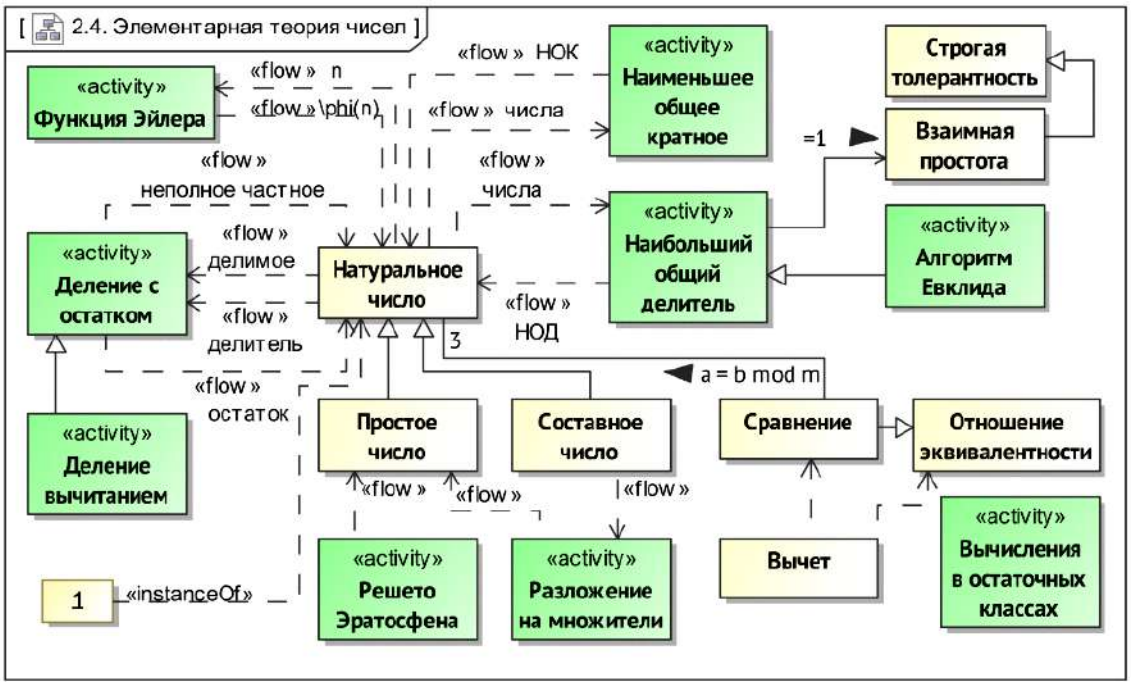

Рис. 7. Пример диаграммы из онтологии дискретной математики версии 2019 года

Сразу бросается в глаза основная проблема — на диаграмме слишком много разнотипных стрелочек, не видно, откуда и куда они идут. Нарушено практическое правило:

7 ± 3 сущности на одной диаграмме. Напрашивается очевидное решение: распределить информацию по нескольким диаграммам.

Заметим, например, что каждая стрелка обобщения показывает один факт: простое число — специализация натурального числа, алгоритм Евклида — специализация алгоритма нахождения НОД. Эти отношения не зависят друг от друга, и их можно показать на одной диаграмме, или на разных диаграммах, как удобнее. В таких случаях распределение информации на несколько диаграмм не вызывает затруднений.

Кроме этого, есть менее очевидная проблема, когда при разнесении стрелок по разным диаграммам теряется смысл онтологии в целом. Рассмотрим, например, следующую диаграмму (рис. 8).

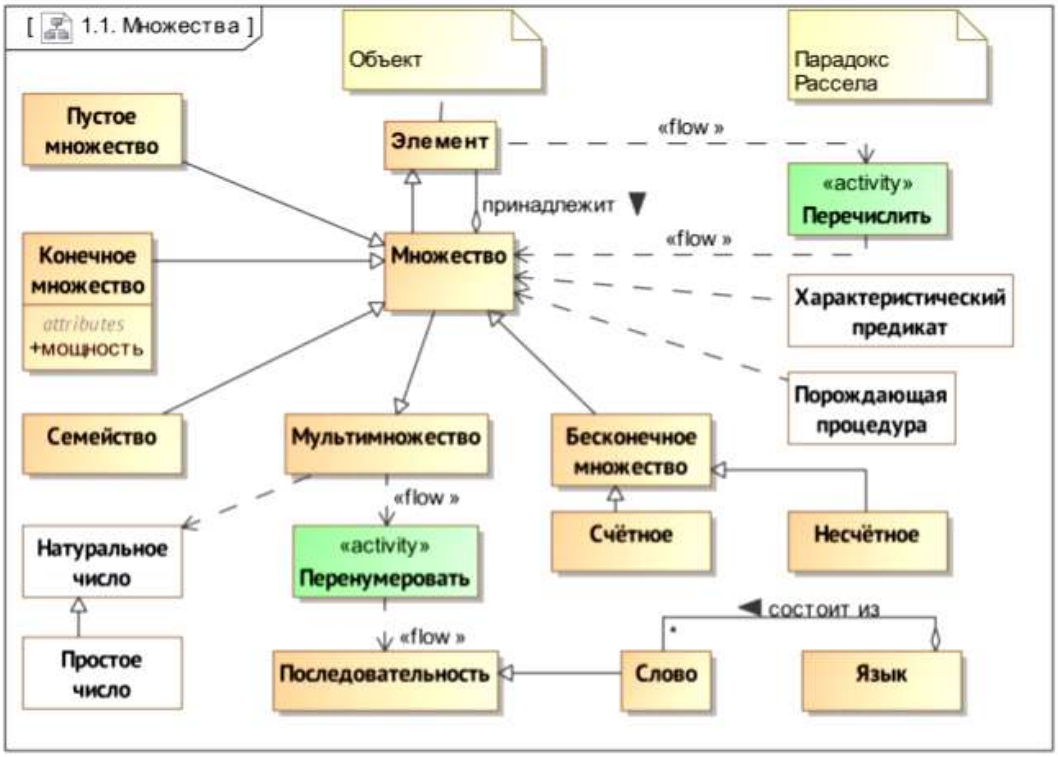

Рис. 8. Другой пример диаграммы из онтологии дискретной математики версии 2019 года

На этой диаграмме прилежный студент может усмотреть важный факт: множество можно задать перечислением элементов, характеристическим предикатом или порождающей процедурой. Если ради экономии места при распределении сущностей по разным диаграммам все три способа задания множеств не окажутся на одной диаграмме, то студент (особенно нерадивый студент) может утратить самое важное — ощущение полноты своих знаний (то есть студент не осознает, что перечень способов задания множеств является исчерпывающим). Таким образом, количественное ограничение 7 ± 3 сущности на одной диаграмме не должно применяться слепо и бездумно.

Наше решение состоит в том, чтобы распределять информацию по диаграммам не только по количеству сущностей, но по качественным признакам на три части, или, лучше сказать, на три грани. Первая грань отражает отношения наследования, агрегации, композиции понятий, вторая грань отражает зависимости между понятиями, третья —

алгоритмические аспекты. Для примера, разобьем нашу исходную диаграмму на рис. 8 на три диаграммы, соответствующие трём граням: рис. 9, 10, 11.

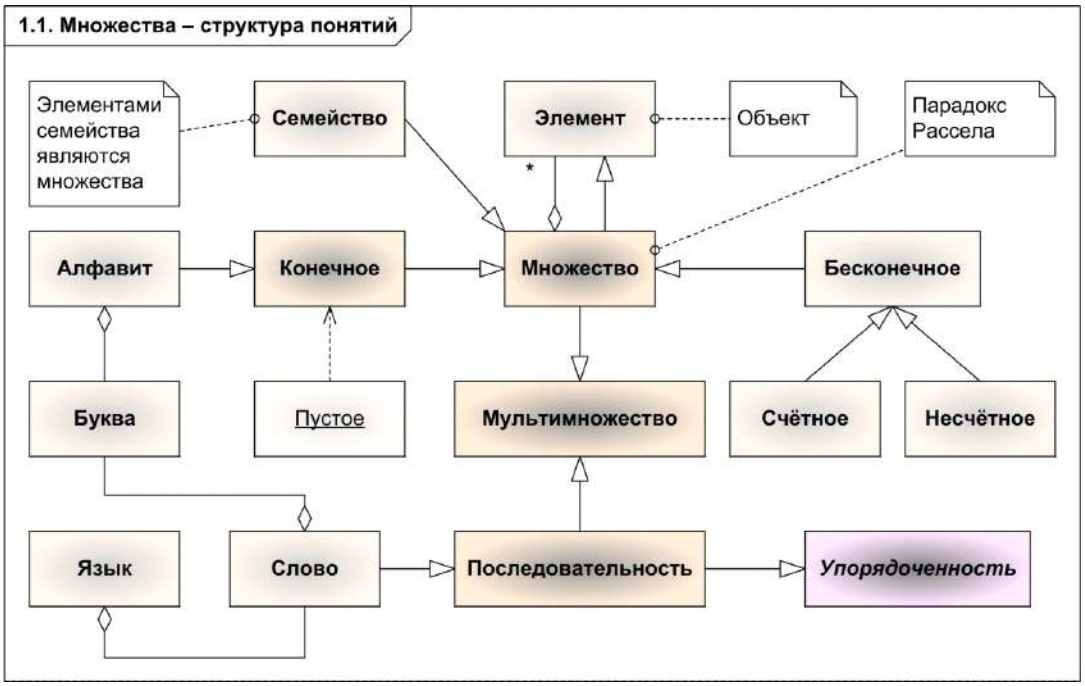

Рис. 9. Понятийная грань онтологии множеств

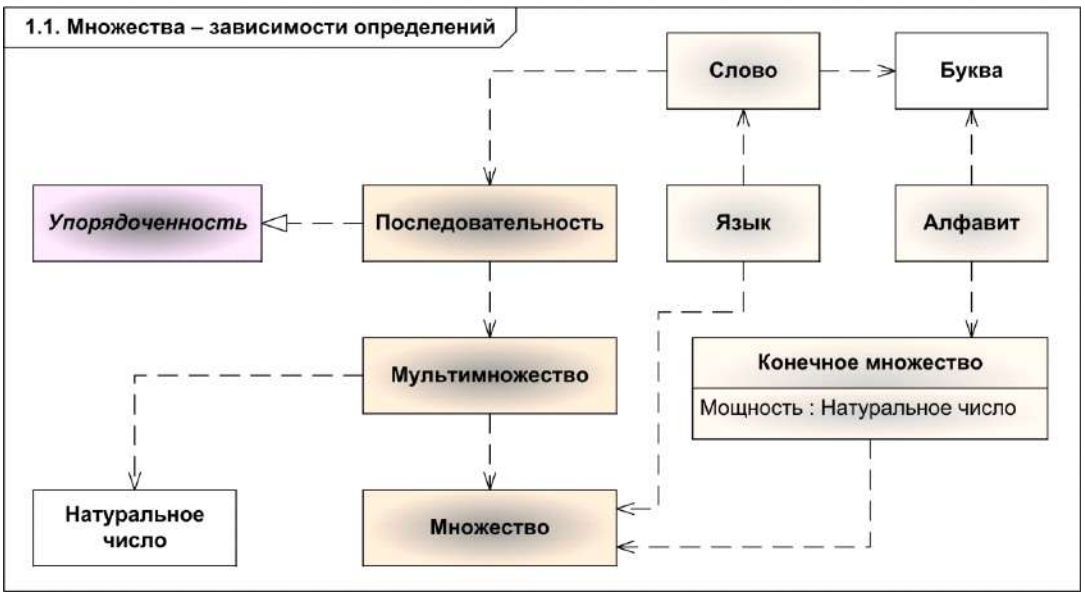

Рис. 10. Грань зависимостей понятий в онтологии множеств

Так мы добились двух вещей.

Во-первых, невооруженным глазом видно, что диаграммы перестали быть перегруженными стрелочками и легко читаются. Даже первая диаграмма доступна для восприятия, хотя на ней 17 фигур (рис. 9).

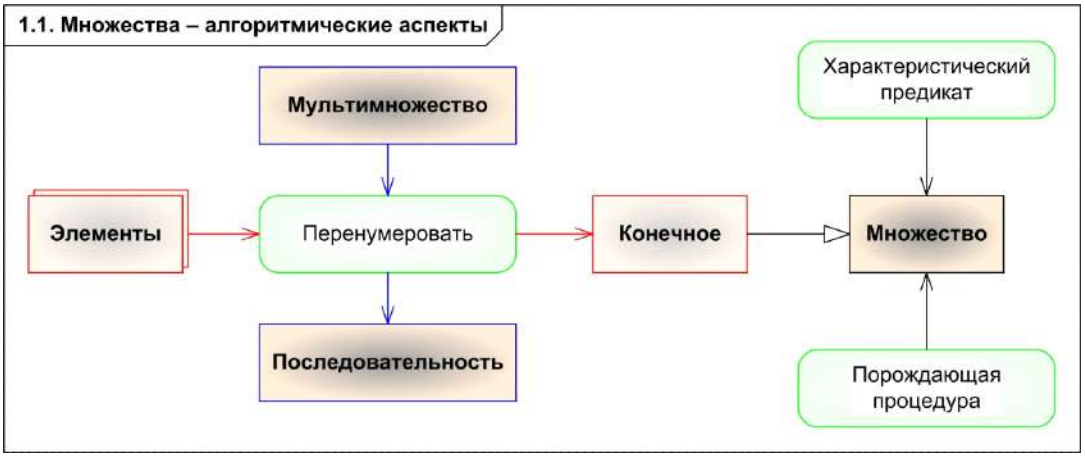

Рис. 11. Алгоритмическая грань онтологии множеств

Во-вторых, онтология теперь разделена на диаграммы по смыслу: первая диаграмма с наследованием, агрегацией и композицией показывает, какие понятия существуют, какая у этих понятий иерархия. Вторая диаграмма с помощью зависимостей показывает, что означают эти понятия: алфавит — это конечное множество букв, язык — это множество слов, слово — это последовательность букв. Третья диаграмма показывает, какие есть алгоритмы, как одни понятия могут переходить в другие: множество можно образовать порождающей процедурой, если возможно перенумеровать элементы множества, то оно конечно.

Если обобщить: получается, что каждая диаграмма имеет свою роль — первая знакомит с понятиями, вторая говорит, что эти понятия означают, третья показывает, как возможно из объектов одного класса построить другие объекты этого класса или же объекты других классов.

Есть, разумеется, очевидный минус у такого подхода — теперь вместо одной диаграммы у нас три, пусть и меньшего размера. Поэтому так стоит делать только в объемных онтологиях с большим количеством зависимостей и алгоритмов.

7. СОВЕТЫ ПО СОЗДАНИЮ ОНТОЛОГИЙ С ПРИМЕРАМИ

Рассмотрим некоторые хорошие примеры онтологий, разберемся, чем они хороши. «Хороши» в данном случае означают «наглядны» (понятно, что это субъективно). На основе этого выведем советы по созданию хороших онтологий.

1. Иерархию лучше подчеркивать пространственным расположением понятий (рис. 12). Обобщающее понятие должно быть ближе к центру, каждый следующий наследник распространяется в одном направлении от него.

В данном примере сразу видна иерархия: обобщающее понятие «характеристическая функция» расположена в центре, дальше во все стороны от нее отходят наследники. Каждый следующий наследник расположен в том же направлении, что и предыдущий, только дальше от центра.

2. Если есть много похожих понятий, либо алгоритмов, выполняющих похожие задачи, их стоит объединить в блок (рис. 13).

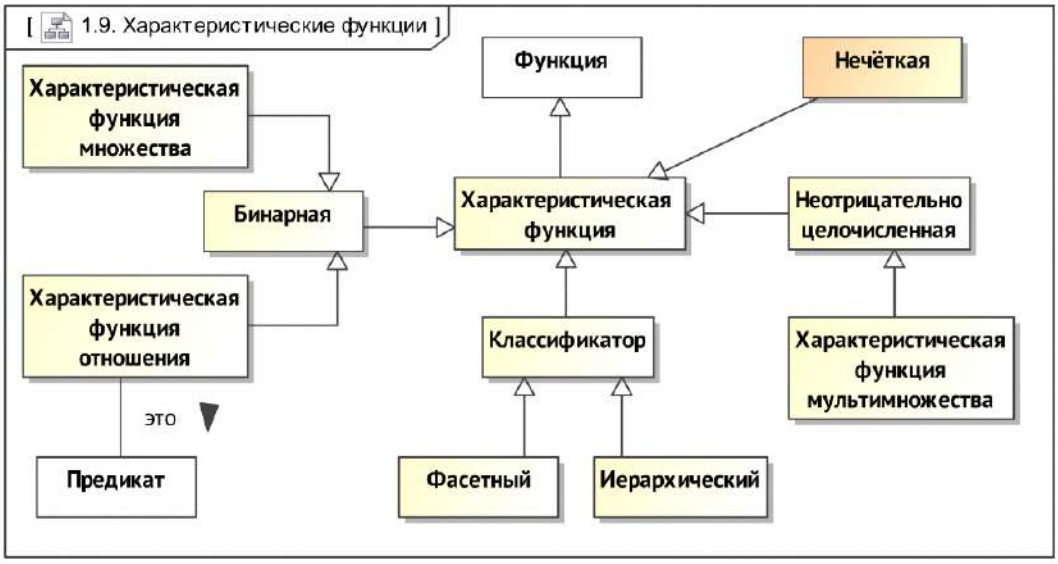

Рис. 12. Самое важное понятие в центре диаграммы

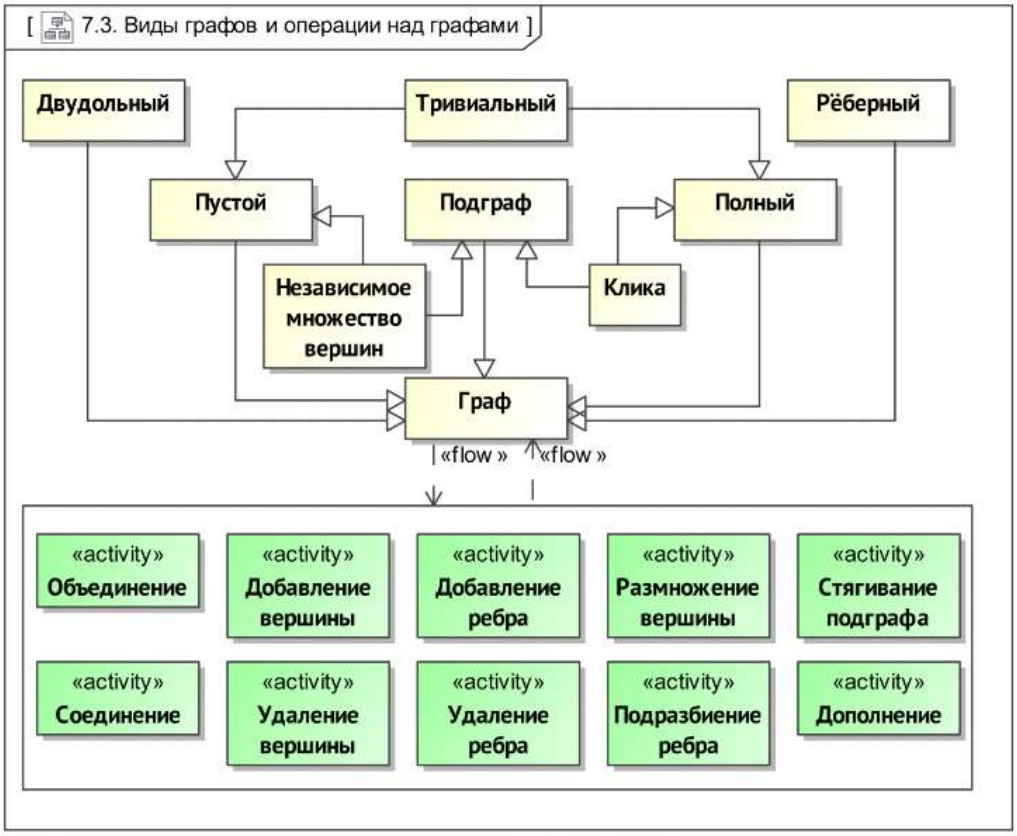

Рис. 13. Операции с графами собраны в безымянный пакет

На данной диаграмме все операции с графом выделены в отдельный блок — это позволяет писать две стрелки «flow» вместо двадцати.

3. Алгоритмы целесообразно располагать на диаграмме между входными и выходными данными. Так лучше видно, что именно делает данный алгоритм — не приходится глазами долго искать, откуда идет выходная стрелка, к какому понятию выходит исходящая (рис. 14).

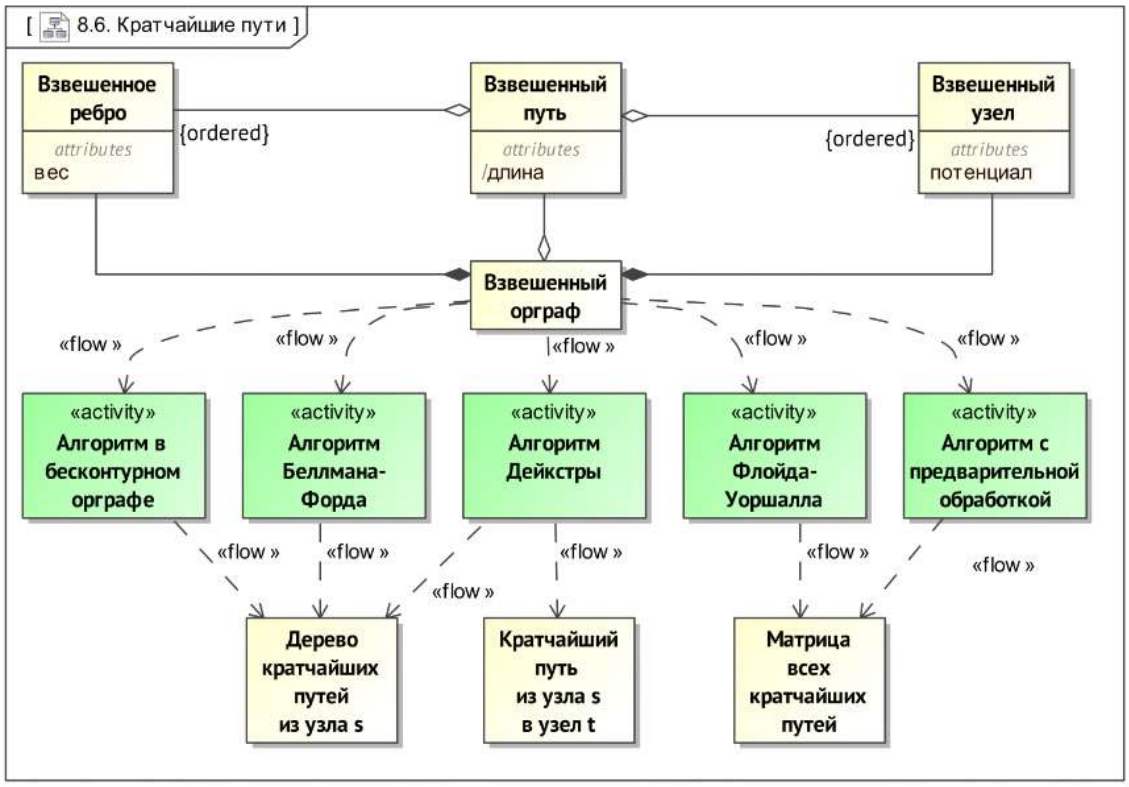

Рис. 14. Входы и выходы алгоритмов поиска кратчайших путей

На данной диаграмме присутствует много алгоритмов, но при этом из-за оптимального пространственного расположения понятий сразу видно, что им идет на вход, что на выход.

4. Последний и общий факт: онтологии должны быть консистентными. Другими словами, понятия, имеющие схожее расположение на диаграмме, должны быть схожи по смыслу, и наоборот, — схожие по смыслу понятия должны располагаться на диаграмме похожим образом. Для разных случаев это может обозначать совершенно разное, поэтому невозможно придумать всеобъемлющий пример. Рассмотрим такой (рис. 15).

Все виды представлений графа расположены в линию под понятием «граф». Обходы графа тоже расположены в линию. При этом и в том, и в другом случае ничего больше на этой линии нет. То есть похожие по смыслу объекты расположены похожим образом, и наоборот.

8. ЗАКЛЮЧЕНИЕ. ЧЕГО МЫ ДОБИЛИСЬ?

Мы сформулировали требования к онтологиям дискретной математики, которые предлагается использовать в процессе обучения. Далее мы провели обзор современных языков описания онтологий, из которых мы выбрали UML как наиболее подходящий для нашей

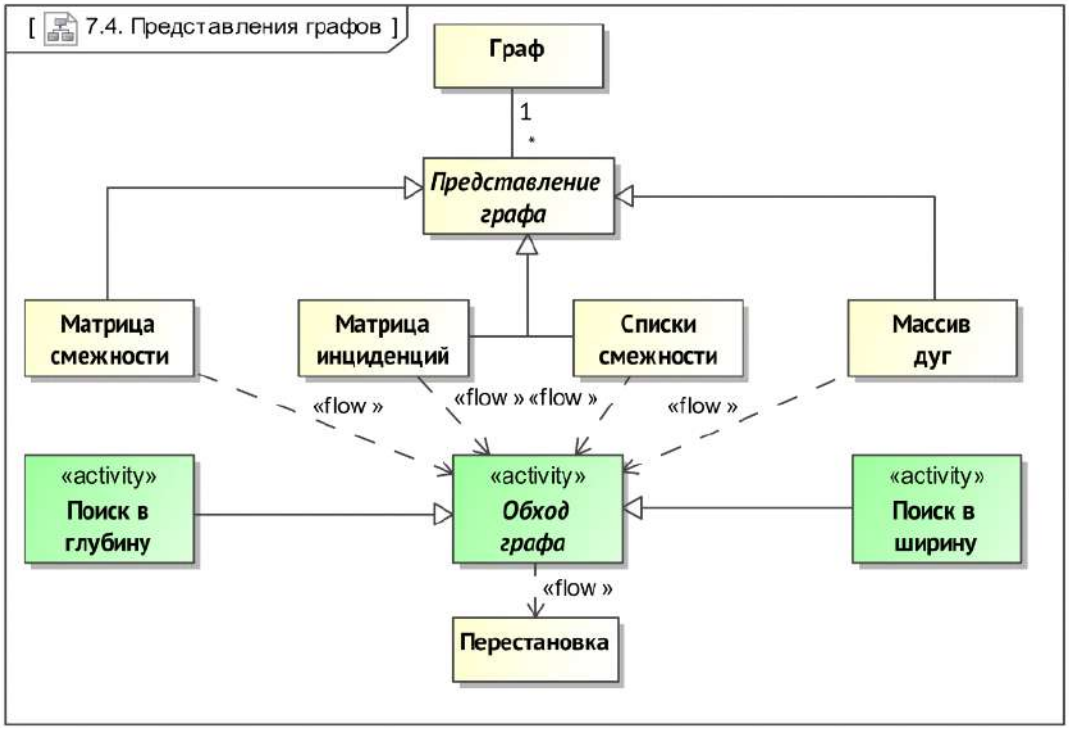

Рис. 15. Общая диаграмма о представлении графов

задачи. После ряда модификаций, основной из которых является разделение онтологии на три грани, на основе языка UML предложен способ описания онтологий, наиболее подходящий для формализации дискретной математики. Данный способ является достаточно выразительным, чтобы отражать наиболее важные аспекты дискретной математики (в отличие от таксономий, например), при этом является более строгим и формальным, чем другие аналоги (например RDF, RDFS), при этом он намного нагляднее, чем более формальные альтернативы (например OWL). В конце статьи разобраны наиболее удачные примеры применения нашего метода описания онтологий к дискретной математике и сформулированы советы по созданию таких диаграмм.

Список литературы

1. *Hofweber T.* Logic and Ontology // The Stanford Encyclopedia of Philosophy (Summer 2020 Edition), Edward N. Zalta (ed.), URL: https://plato.stanford.edu/archives/sum2020/entries/logic-ontology/ (дата обращения: 04.02.2021).
2. *Gruber T. R.* The role of common ontology in achieving sharable, reusable knowledge bases // Kr. 1991. Т. 91. С. 601–602.
3. *Лапшин В. А.* Онтологии в информационных системах. М.: Научный мир. 2010. 247 с. [Электронный ресурс] URL: http://isdwiki.rsuh.ru/moodle/pluginfile.php/128/course/section/36/bookLapshin.pdf (дата обращения: 04.02.2021).
4. Методы инженерии знаний. [Электронный ресурс]: Управление знаниями. URL: https://sites.google.com/site/upravlenieznaniami/inzeneria-znanij/sredstva-inzenerii-znanij (дата обращения: 04.02.2021).
5. The Taxonomy Database. [Электронный ресурс]: National Center for Biotechnology Information. URL: https://www.ncbi.nlm.nih.gov/taxonomy/ (дата обращения: 04.02.2021).

6. *Decker S. et al.* The semantic web: The roles of XML and RDF // IEEE Internet computing. 2000. Т. 4, № 5. С. 63–73.
7. *McBride B.* The resource description framework (RDF) and its vocabulary description language RDFS // Handbook on ontologies. Springer, Berlin, Heidelberg, 2004. С. 51–65.
8. *McGuinness D. L. et al.* OWL web ontology language overview // W3C recommendation. 2004. Т. 10, № 10. С. 2004.
9. Cranefield S., Purvis M. K. UML as an ontology modeling language // Intelligent Information Integration. 1999.
10. *Новиков Ф. А.* Дискретная математика: Учебник для вузов. 3-е изд. Стандарт третьего поколения. СПб.: Питер, 2017. 496 с.
11. *Новиков Ф. А.* Дискретная математика для программистов. [Электронный ресурс]: Профессиональная разработка программных систем. URL: http://uml3.ru/index.html (дата обращения: 04.02.2021).
12. *Schinz I. et al.* The Rhapsody UML verification environment // Proceedings of the Second International Conference on Software Engineering and Formal Methods, 2004. SEFM 2004. IEEE, 2004. С. 174–183.

Поступила в редакцию 26.12.2020, окончательный вариант — 04.02.2021.

Молотков Иван Игоревич, студент Академического университета им. Ж. И. Алфёрова, ✉ molotkov.ivan.igorevich@gmail.com

Новиков Федор Александрович, доктор технических наук, профессор СПбГЭТУ «ЛЭТИ» им. В. И. Ульянова (Ленина), ✉ fedornovikov51@gmail.com

Computer tools in education, 2021
№ 1: 68–84
http://cte.eltech.ru
doi:10.32603/2071-2340-2021-1-69-85

Ontology of Discrete Mathematics in Education

Molotkov I. I.[1], Student, ✉ molotkov.ivan.igorevich@gmail.com
Novikov F. A.[2], PhD, Professor, ✉ fedornovikov51@gmail.com

[1]Saint Petersburg National Research Academic University of the Russian Academy of Sciences,
8, building 3, let. A, Khlopina st., 194021, Saint Petersburg, Russia
[2]Saint Petersburg Electrotechnical University,
5, building 3, st. Professora Popova, 197376, Saint Petersburg, Russia

Abstract

Currently, ontologies are widely used in computer science for the formalized representation of knowledge about various subject areas. Special formal languages for describing ontologies have been developed and are successfully used, which allow describing ontologies in a form that is accessible for use by both humans and computers.Among the various options for using ontologies, a special place is occupied by the use of ontologies in education, since the systematization and ordering of knowledge, being the main competitive advantage of the ontological approach, is at the same time one of the main goals of the educational process. The article proposes original methods of constructing ontologies for use in the educational process of higher education. The central idea is the construction of

faceted, in other words, multifaceted ontologies, in which different aspects of the same subject area are described by conceptually similar, but syntactically different means. This approach provides a more accurate and semantically adequate description while maintaining the known brevity and clarity of designations. As a language for describing ontologies, it is proposed to use the unified modeling language UML 2, which has proven itself in formalization in many cases. The presentation is based on the example of constructing an ontology of discrete mathematics, and the ontology diagrams given in the article are introduced into the educational processes of the Academic and Polytechnic Universities of St. Petersburg.

Keywords: *Ontology, educational process, knowledge formalization, knowledge systematization, discrete mathematics, teaching, UML, RDF, RDFS, OWL.*

Citation: I. I. Molotkov and F. A. Novikov, "Ontology of Discrete Mathematics in Education," *Computer tools in education*, no. 1, pp. 68–84, 2021 (in Russian); doi: 10.32603/2071-2340-2021-1-69-85

References

1. T. Hofweber, "Logic and Ontology," E. N. Zalta ed., in *The Stanford Encyclopedia of Philosophy*, Sum. 2020. [Online]. Available: https://plato.stanford.edu/archives/sum2020/entries/logic-ontology/
2. T. R. Gruber, "The role of common ontology in achieving sharable, reusable knowledge bases," in *Proc. of the Second International Conference Principles of Knowledge Representation and Reasoning*, vol. 91, 1991, pp. 601–602.
3. V. A. Lapshin, "Ontologii v informatsionnykh sistemakh" [Ontologies in information systems], Moscow: Nauchnyi mir, 2010.[Online] (in Russian). Available: http://isdwiki.rsuh.ru/moodle/pluginfile.php/128/course/section/36/bookLapshin.pdf
4. "Metody inzhenerii znanii" [Knowledge Engineering Techniques], in *Upravlenie znaniyami*. [Online] (in Russian). Available: https://sites.google.com/site/upravlenieznaniami/inzeneria-znanij/sredstva-inzenerii-znanij
5. National Center for Biotechnology Information, "The Taxonomy Database," in *www.ncbi.nlm.nih.gov*. [Online]. Available: https://www.ncbi.nlm.nih.gov/taxonomy/
6. S. Decker et al., "The semantic web: The roles of XML and RDF," *IEEE Internet computing*, vol. 4, no. 5, pp. 63–73, 2000. doi: 10.1109/4236.877487
7. B. McBride, *The resource description framework (RDF) and its vocabulary description language RDFS*, Berlin, Heidelberg: Springer, pp. 51–65, 2004.
8. D. L. McGuinness et al., "OWL web ontology language overview, W3C recommendation," in *W3C*, 10 feb. 2004. [Online]. Available: https://www.w3.org/TR/2004/REC-owl-features-20040210
9. S. Cranefield and M. K. Purvis, *UML as an ontology modeling language*, Dunedin, New Zealand: Uneversity of Otago, 1999.
10. F. A. Novikov, *Discrete Mathematics: A Textbook for High Schools*, St. Petersburg: Piter, 2017 (in Russian).
11. F. A. Novikov, "Discrete mathematics for programmers," in *Professional'naya razrabotka programmnykh sistem*. [Online] (in Russian). Available: http://uml3.ru/index.html
12. I. Schinz et al., "The Rhapsody UML verification environment," in *Proc. of the Second International Conference on Software Engineering and Formal Methods, SEFM, 2004 – IEEE*, 2004. pp. 174–183.

Received 26-12-2020, the final version — 04-02-2021.

Ivan Molotkov, Student of Saint Petersburg National Research Academic University of the Russian Academy of Sciences, ✉ molotkov.ivan.igorevich@gmail.com

Fyodor Novikov, PhD, Professor of ETU "LETI", ✉ fedornovikov51@gmail.com