электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 519.681

О трудноформализуемых задачах

Говоров М.И., *аспирант*
кафедра «Информационные системы и телекоммуникации»,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный руководитель: Выхованец В.С., д.т.н., профессор
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
deviatkov@bmstu.ru

«– Мы сами знаем, что она не имеет решения, — сказал Хунта, немедленно ощетиниваясь. – Мы хотим знать, как её решать.

– Как-то ты странно рассуждаешь, Кристо... Как же искать решение, когда его нет? Бессмыслица какая-то...

– Извини, Теодор, но это ты очень странно рассуждаешь. Бессмыслица – искать решение, если оно и так есть. Речь идёт о том, как поступать с задачей, которая решения не имеет ...»

Аркадий и Борис Стругацкие.
Понедельник начинается в субботу.

Существуют классы корректно поставленных задач – массовых проблем, для которых, тем не менее, доказано отсутствие каких-либо алгоритмов их решения [1]. Алгоритмически неразрешимой является, например, проблема остановки – по описанию алгоритма распознать с помощью другого алгоритма, остановится или нет анализируемый алгоритм на заданных входных данных. Другими неразрешимыми проблемами являются проблемы эквивалентности алгоритмов, тождества двух математических выражений, распознавания того, можно ли или нет из имеющихся автоматов, собрать заданный автомат, а также множество проблем, относящихся к топологии, теории групп и другим прикладным областям. Такие задачи будем называть неформализуемыми.

Алгоритмическая неразрешимость массовой проблемы не означает неразрешимости той или иной единичной задачи из данного класса. Та или иная конкретная задача может иметь решение, причем даже вполне очевидное, а для другой задачи может существовать простое доказательство отсутствия решения. Но в целом,

данный класс задач не имеет общего универсального алгоритма решения, применимого ко всем задачам. Для решения отдельных подклассов задач нужно разрабатывать свои алгоритмы; для некоторых отдельных задач требуется разработка уникальных методов решения.

Конкретизация массовой проблемы позволяет найти решение той или иной задачи, но каждый раз по-своему. Например, в общей постановке задача распознавания применимости алгоритма неразрешима, однако, если рассматривать конкретный алгоритм, то решение может быть найдено, и это решение будет вытекать из описания самого алгоритма. Однако, для другого алгоритма потребуется искать новое решение. Такие задачи будем называть трудноформализуемыми.

Таким образом, основной особенностью трудноформализуемых задач является наличие их общей формальной постановки (эффективно распознаваемых условий), которая, тем не менее, не имеет единого алгоритма решения. Иными словами, для каждого конкретного случая решения трудноформализуемой задачи (для каждого набора входных данных) приходится искать свой, присущий только этому случаю (этому набору данных) алгоритм решения. Примерами трудноформализуемых задач могут служить задачи аннотирования текстов, доказательства теорем, распознавания образов, принятия решений, классификации и кластеризации данных и т.п.

Решение трудноформализуемых задач требует значительных мыслительных усилий и привлечения большого объема знаний. Это вызвано, в основном, большим числом сущностей предметной области, которые требуются для описания решения таких задач, а также сложными взаимосвязями между ними.

Решение трудноформализуемх задач основано на изучении предметной области под углом зрения некоторой конкретной прикладной проблемы. Именно в процессе такого изучения и появляется понимание специфики решаемой задачи, а выражение этой специфики на языке программирования позволяет получить искомое решение.

Общим недостатком известных методологий и технологий программирования является практическое игнорирование того факта, что трудноформализуемую задачу не опишешь понятиями, встроенными в универсальные языки программирования [2]. Последнее приводит к явлению, известному как семантический (смысловой) разрыв [3]. Семантический разрыв возникает между содержательными представлениями о предметной области и ее формальной моделью на языке программирования. Проявление семантического разрыва, например, в области обратных методов программной инженерии, заключается в том, что результатом изучения программы является знание отдельного индивида, которое не отчуждается от индивида и легко им теряется [4].

Последствиями семантического разрыва являются высокая стоимость разработки программных средств, их низкая эффективность и надежность, объективная трудоемкость, интеллектуальная и технологическая сложность самого процесса программирования. Для сокращения семантического разрыва используется повышение уровня абстракций языков программирования в рамках объектно-ориентированного подхода [5]. Последнее снимает только часть проблем, не затрагивая существенным образом набор понятий и выразительные возможности самого языка программирования.

При объектно-ориентированном подходе знания о предметной области формализуются в виде множества объектов, характеризуемых поведением и состоянием, которые выражаются некоторым набором методов и свойств. Объекты состоят между собой в отношениях агрегации и объединяются в группы (классы), находящиеся в отношениях наследования. Однако выразительные способности объектно-ориентированных языков, в конечном итоге, так и остались привязанными к встроенным в эти языки примитивным языковым конструкциям.

В итоге, проектирование информационных систем до сих пор выполняется в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве разработчиков, их практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках получаемых результатов. Кроме того, в процессе жизненного цикла программное средство подлежит постоянному изменению в соответствии с изменяющимися потребностями пользователей и развитием их представлений о предметной области, что еще более усложняет разработку и сопровождение программ.

Самым эффективным методом сокращения семантического разрыва видится приближение выразительных средств языков программирования к специализированным предметным языкам, свойственным описываемой предметной области. В этом случае понятия языка программирования совпадают или очень близки понятиям, используемым для описания стоящей прикладной задачи. Однако ни одна из известных технологий программирования эффективными средствами для описания прикладных понятий не располагает.

Суть предлагаемого подхода заключается в том, что предлагается разработать технологию программирования, которая предназначена для решения трудноформализуемых задач и базируется на следующих основных положениях.

Для решения каждой трудноформализуемой задачи и в процессе ее решения будем создавать свой, присущий только этой задаче специализированный предметный язык, отражающий понятийную структуру рассматриваемой проблемной области. Для этого

выявленные в процессе анализа понятия включим во множество понятий создаваемого языка, а способы выражения понятий положим в его синтаксис [6].

Описание понятийной структуры проблемной области и синтаксиса проблемного языка выполним на протоязыке, имеющем фиксированный синтаксис и семантику, которые минимально достаточны для определения синтаксиса и описания семантики любого проблемного языка [7]. Семантическую роль протоязыка ограничим реализацией аксиомы, позволяющей вводить первичные семантические категории, необходимые для описания семантики произвольных языковых конструкций.

Описание семантики осуществим методом математической индукции, заключающимся в использовании семантических категорий, которые определяются по мере необходимости, в процессе описания проблемного языка и средствами этого языка [8]. Базу индукции, или первичные семантические категории, объявим с помощью аксиомы и представим формулами некоторой внешней теории, например, пояснениями на естественном языке, или с помощью команд некоторого моделирующего устройства. На содержательном уровне это выглядит как использование создаваемого проблемного языка для описания своей семантики. Последнее выполнимо благодаря возможности пополнения множества базовых семантических категорий и наличия механизма определения новых семантических категорий на основе имеющихся.

Для задания понятийной структуры предусмотрим в протоязыке выразительные средства для описания способов абстрагирования понятий, а именно: типизации-конкретизации, обобщения-специализации, агрегации-декомпозиции и ассоциации-индивидуализации [9]. Выделенные абстракции рассматриваются как умственные операции, необходимые и достаточные для мысленного выделения и превращения в отдельные понятия тех представлений, которые накоплены относительно формализуемой проблемной области.

Так как понятийная структура проблемной области может быть общей для нескольких задач, решение которых может преследовать различные цели, в том числе и исключающие друг друга, при описании семантики предусмотрим возможность задания одной или нескольких прагматик, являющихся, по своей сути, различными контекстными интерпретациями одной и той же формы выражения понятия [10].

Так как известные методы грамматического разбора не могут быть использованы в разрабатываемой технологии по причине отсутствия явных ограничений на формы выражения понятий, то для реализации лексического, синтаксического и семантического анализа текстов программ применим метод разнесенного грамматического разбора [11]. Разнесенный грамматический разбор позволяет эффективно выполнить анализ текста,

порожденного контекстными и неоднозначными грамматиками.

Список литературы

1. Кузнецов О. П., Адельсон-Вельский Г. М. Дискретная математика для инженера. М.: Энергоатомиздат, 1988.
2. Непейвода Н. Н., Скопин И. Н. Основания программирования. М.: РХД, 2003.
3. Майерс Г. Архитектура современных ЭВМ: В 2-х книгах. Кн. 1. М.: Мир, 1985.
4. Формальные спецификации в технологиях обратной инженерии и верификации программ / Бурдонов И.Б., Демаков А.В., Косачев А.С. и др. // Труды Института системного программирования. 1999. № 1. С. 31-43.
5. Александреску А. Современное проектирование на C++: Обобщенное программирование и прикладные шаблоны проектирования. М.: Вильямс, 2004.
6. Выхованец В.С. Прикладной понятийный анализ // Труды Международной конференции «Когнитивный анализ и управление развитием ситуаций». М.: Институт проблем управления, 2009. С. 62-65.
7. Выхованец В.С. Контекстная технология программирования // Труды IV-й Международной конференции «Параллельные вычисления и задачи управления». М.: Институт проблем управления, 2008. С. 1288-1327.
8. Выхованец В.С. Описание семантики контекстно-свободных языков методом математической индукции // Научно-техническая информация. Сер. 2: Информационные процессы и системы. 2008. № 7. С. 6-14.
9. Выхованец В.С. Исчисление понятий // Труды VII-й Международной конференции «Когнитивный анализ и управление развитием ситуаций» (CASC'2007, Москва). М.: Институт проблем управления, 2007. С. 31-35.
10. Выхованец В.С., Иосенкин В.Я. Понятийный анализ и контекстная технология программирования // Проблемы управления. 2005. № 4. С. 2-11.
11. Выхованец В.С. Разнесённый грамматический разбор // Проблемы управления. 2006. № 3. С. 32-43.