Комбинированный метод верификации масштабных моделей данных

В.А. Семенов, С.В. Морозов, Д.В. Ильин
ИСП РАН, Москва, Россия
sem@ispras.ru, serg@ispras.ru, denis.ilyin@ispras.ru

Аннотация. Статья адресована актуальной проблеме верификации масштабных моделей данных, применяемых в различных индустриальных областях и специфицируемых на популярных универсальных объектно-ориентированных языках, таких как EXPRESS, UML/OCL. Основные преимущества языков информационного моделирования (высокая выразительность, декларативность, богатство синтаксических конструкций) отрицательно сказываются в процессе автоматической верификации спецификаций. Существующие методы верификации моделей вследствие высокой вычислительной сложности не могут использоваться для решения данной проблемы. В статье предлагается комбинированный метод верификации объектно-ориентированных моделей данных, основанный на последовательной редукции к нескольким постановкам математических задач: линейного программирования, удовлетворения ограничений (CSP), выполнимости булевых формул (SAT). Детально исследуется ключевая проблема определения необходимого количества экземпляров для генерации корректной коллекции объектов и ее редукция к задаче целочисленного линейного программирования. Работа поддержана РФФИ (грант 13-07-00390).

Ключевые слова: верификация моделей, объектно-ориентированное моделирование, UML/OCL, EXPRESS, логическое программирование в ограничениях, линейное целочисленное программирование, семантическая реконсиляция

1. Введение

В настоящее время значительную роль в процессе разработки сложных программных систем играет моделирование. Применение модельно-ориентированного подхода (MDA) [1] позволяет решить многие проблемы программной инженерии, в частности, проблемы мобильности, интероперабельности, масштабируемости программного обеспечения, и, в конечном итоге, повысить качество и сократить ресурсы, затрачиваемые на его разработку и тестирование. В рамках данного подхода модели программной системы являются базовыми спецификациями для всех последующих этапов разработки. В связи с этим особую важность при разработке на основе моделей (Model Driven Development) приобретает корректность используемых спецификаций. Выявление ошибок

проектирования на первоначальной стадии разработки позволяет сократить ее время, уменьшить риски и, как результат, снизить ее стоимость.

В работе [2] выделяются две различных постановки задачи проверки корректности концептуальных схем: внутренняя и внешняя проверки. Фундаментальным свойством корректности в контексте внутренней проверки является выполнимость (satisfiability) моделей, а именно, жизнеспособность (liveliness) их классов и ассоциаций, исполняемость (executability) операций, безызбыточность (nonredundancy) семантических ограничений. Задача в данной постановке может быть автоматически решена методами формальной верификации. С точки зрения внешней проверки наиболее важным фактором является применимость (usability) моделей к конкретным требованиям пользователя. Задача в данной постановке сводится к валидации полноты покрытия семантики заданной предметной области и не может быть решена только формальными методами.

Программные средства верификации и валидации для автоматической или автоматизированной проверки корректности концептуальных схем существуют [11, 17, 26, 28, 31, 34], но их высокая вычислительная сложность является значительным недостатком. Основные преимущества языков информационного моделирования (высокая выразительность, декларативность, богатство синтаксических конструкций) отрицательно сказываются в процессе автоматической верификации спецификаций. В связи с постоянно возрастающей сложностью используемых в программной инженерии моделей возникает необходимость разработки и применения эффективных методов для их верификации.

Настоящая работа адресована проблеме верификации масштабных моделей данных, применяемых в различных индустриальных областях и специфицируемых на популярных универсальных объектно-ориентированных языках (EXPRESS [3], UML/OCL [4, 5] и т.п.). Примерами таких моделей служат IFC (архитектура и строительство) [6], CIS/2 (строительство с использованием стальных конструкций) [7], STEP (машиностроение) [8], ISO 15926 (нефтегазодобывающая промышленность) [9]. Данная проблема является актуальной, поскольку масштабные индустриальные модели разрабатываются крупными консорциумами участников в течение продолжительного времени (см. пример в табл. 1) и могут содержать трудно обнаруживаемые визуально ошибки, такие как избыточные либо переопределенные ограничения. Подобные ошибки являются одной из причин периодического пересмотра утвержденных информационных стандартов и принятия поправок к ним.

	IFC 1.5.1	IFC 2.0	IFC 2x	IFC 2x2	IFC 2x3	IFC 4
Год принятия	1997	1999	2000	2003	2006	2013
Объектные типы	186	290	370	623	653	766
Типы, определяемые пользователем	95	157	228	312	327	391
Вычисляемые атрибуты	44	45	41	55	96	105
Процедуры и функции	25	27	25	37	38	42
Ограничения в объектных типах	107	168	196	271	340	638
Ограничения в определяемых пользователем типах	12	12	13	16	24	24
Ограничения уникальности	1	3	14	14	17	4
Глобальные ограничения	0	0	3	3	2	2

Табл. 1. Эволюция модели IFC

В разделе 2 проводится аналитический обзор существующих методов верификации моделей. В разделе 3 предлагается комбинированный метод верификации объектно-ориентированных моделей данных, основанный на последовательной редукции к нескольким постановкам математических задач: линейного программирования, удовлетворения ограничений (CSP), выполнимости булевых формул (SAT). В разделе 4 детально исследуется ключевая проблема определения необходимого количества экземпляров для генерации корректной коллекции объектов и ее редукция к задаче целочисленного линейного программирования. В разделе 5 демонстрируется пример применения предложенного комбинированного метода. В заключении приводятся выводы о перспективности применения метода для верификации масштабных индустриальных моделей данных и намечаются направления его дальнейшего исследования и развития.

2. Аналитический обзор существующих методов верификации моделей

Существуют многочисленные решения для верификации моделей, базирующиеся как на UML/OCL, так и на других языковых средствах (диаграммы «сущность-связь» (ER) [10], язык Alloy [11], языки объектно-ориентированных баз данных [12], дескрипционные логики [13]). Лежащие в их основе подходы основываются на редукции исходной задачи верификации модели к той или иной математической постановке и ее дальнейшему решению известными методами. Как правило, большинство существующих средств проверяют выполнимость исходной модели, т.е. возможность создания некоторого набора экземпляров ее классов, не нарушающего определенных в ней ограничений. В ряде из них осуществляются дополнительные проверки, например, локализация избыточных или зависимых ограничений. В данном разделе представлены краткие описания нескольких известных средств верификации, сгруппированные по используемым математическим методам. В последнем подразделе проводится сравнение данных решений.

2.1. Методы, основанные на системах линейных неравенств

Метод, основанный на трансформации концептуальных схем в системы линейных неравенств, был впервые предложен Ленцерини и Нобили [14] для верификации ER-диаграмм, которые включают сущности (классы), бинарные связи (ассоциации) и ограничения кардинальности. Система формируется по набору связей и ограничений кардинальности заданной ER-модели за полиномиальное время относительно размера исходной схемы. В работе было введено определение сильной выполнимости модели, которое гарантирует проверку корректности ограничений кардинальности и подразумевает возможность создания набора экземпляров, включающего как минимум один экземпляр каждой сущности и не нарушающего ни одного из определенных ограничений. Теоретически доказано, что ER-модель является сильно выполнимой тогда и только тогда, когда существует решение соответствующей системы неравенств. Система может быть решена методами линейного программирования. В работе [14] был также представлен алгоритм, позволяющий локализовать подмножество ограничений, нарушающих выполнимость ER-модели. Он основан на преобразовании исходной ER-диаграммы в ориентированный мультиграф и поиске в нем критических циклов.

Мараее и Балабан [15] развили оригинальный метод Ленцерини–Нобили для верификации диаграммы классов UML, включающей не только бинарные ассоциации с ограничениями кардинальности, но и отношения обобщения/специализации с соответствующими ограничениями ”complete”, “incomplete”, “disjoint”, “overlapping”. Алгоритм формирования системы неравенств также имеет полиномиальную сложность, но обладает рядом

ограничений. В частности, он неприменим к UML диаграммам с циклическими иерархиями классов при наличии ограничений обобщения/специализации. Прочие конструкции языка UML (n-арные ассоциации, агрегации, композиции, сложные ограничения кардинальности в виде комбинации интервалов, ограничения OCL) также не обрабатываются.

2.2. Методы, основанные на трансформации в реляционные логики

Наиболее известным средством верификации данного класса является язык Alloy [16]. Он использует логику предикатов первого порядка для описания моделей структурированных данных вместе с семантическими ограничениями и рассматривается как простое приложение, которое позволяет проектировать модели и проверять их корректность с помощью анализатора Alloy. Проектировщик описывает модель на языке Alloy, который имеет текстовую нотацию, затем анализатор переводит конструкции языка в логическую формулу в конъюнктивной нормальной форме и разрешает её с помощью решателя SAT (другими словами, находит такое состояние модели, при котором логическая формула возвращает истинное значение). Синтаксис языка Alloy совместим с UML, но оценка покрытия UML текстовой нотацией Alloy весьма затруднительна. Первая версия Alloy имела ограниченные объектно-ориентированные свойства. В более поздних версиях появилась поддержка наследования, полиморфизма, мульти-арных отношений, кванторов.

UML2Alloy [17] играет важную роль в создании моста между языками UML/OCL и Alloy. UML2Alloy — приложение для верификации диаграмм классов UML с помощью анализатора Alloy в контексте технологии MDD. Кроме диаграмм классов UML поддерживаются также инварианты OCL. Трансформация модели UML/OCL в конструкции языка Alloy осуществляется автоматически, в качестве исходных данных принимается файл в формате XMI, сгенерированный с помощью ArgoUML [18].

Известны многочисленные альтернативные решения для автоматической верификации UML моделей, основанные на трансформации диаграмм UML с OCL ограничениями в логики первого порядка и дальнейшей редукции к задаче SAT. Среди них особо отметим решение, предложенное университетом Бремена [19]. В отличие от UML2Alloy в нем поддерживается больше конструкций языка UML/OCL. Оно также позволяет не только решать задачу верификации UML/OCL модели быстрее, но и дополнительно обнаруживать зависимые инварианты (т.е. инвариант, который логически следует из другого).

Все решения, основанные на редукции к SAT, имеют один общий недостаток: ограниченную поддержку арифметических выражений (поддерживается только целочисленная арифметика без мультипликативных операций). При трансформации в логическую формулу арифметические выражения

преобразуются в логические, оперирующие с булевскими переменными. Исходная семантика при этом теряется. Кроме того, расширение области значений целочисленных переменных может привести к комбинаторному взрыву в размере генерируемой логической формулы.

2.3. Методы, использующие дескрипционные логики

Дескрипционные логики — семейство формализмов для представления знаний. Они позволяют выводить одни знания из других, представленных в базе знаний. В последнее время дескрипционные логики широко применяются в концепции семантической паутины (Semantic Web) для построения онтологий [20]. Верификация UML диаграмм с помощью данного формализма основана на их трансляции в ту или иную дескрипционную логику, а затем использовании одной из известных машин ДЛ-вывода для проверки выполнимости концептов и непротиворечивости утверждений. Любой дескрипционной логике свойственен компромисс между выразительностью и сложностью логического вывода. Выразительные логики, как правило, не могут применяться для вывода высокой сложности. С другой стороны, логики с эффективными процедурами вывода не обладают высокой выразительностью. По этой причине степень покрытия семантики языка UML дескрипционными логиками не является высокой. Ограничения OCL не поддерживаются ни одним из известных решений.

Среди существующих решений для верификации UML диаграмм отметим следующие. Первое разработано в университете Рима [21]. Оно транслирует диаграмму классов UML в дескрипционную логику ALCQI и использует ДЛ-машины FaCT [22] и RACER [23] для проверки выполнимости исходной модели. Из ограничений целостности поддерживаются только ограничения кардинальности атрибутов и ассоциаций, а также ограничения наследования ”disjoint” и ”overlapping”. В работе теоретически доказывается экспоненциальная сложность данной задачи. Второе решение, разработанное в университете Брюсселя [24], основано на использовании дескрипционной логики ALCQRIFO и машины вывода Loom [25]. Оно позволяет анализировать совместно диаграммы классов, последовательностей и состояний и находить такие нестыковки между ними, как отсутствие определений сущностей, атрибутов и операций, провисшие ассоциации, нарушение порядка выполнения и т.п. Конфликты подобного рода, как правило, возникают в процессе редактирования моделей UML, когда изменения, сделанные, например, в статической модели (диаграмме классов), забывают перенести на связанные с ней динамические модели. Используемая дескрипционная логика имеет высокую выразительность, однако в работе не приводятся теоретические и экспериментальные оценки сложности и не исследуется применимость данной логики к анализу масштабных UML моделей.

2.4. Методы, использующие логики высших порядков

Решения, основанные на использовании логик высшего порядка, проводят верификацию посредством доказательства теорем. Автоматическое доказательство не поддерживается, и требуется участие высококвалифицированного пользователя, что является существенным недостатком данных средств. Это объясняется фактом невозможности автоматически различить, где произошла ошибка: в формулировке теоремы или в выбранной стратегии доказательства. Рассмотрим два наиболее известных средства, основанных на данном формализме.

Prototype Verification System (PVS) [26] — среда верификации формальных спецификаций, интегрированная в Emacs. Она включает в себя язык спецификаций, основанный на классической типизированной логике высшего порядка, а также программу доказательства теорем, поддерживающую несколько процедур принятия решений. Существуют методики трансформации диаграмм классов (исключая отношения обобщения/специализации) и состояний UML, а также ограничений OCL в язык PVS [27].

Higher-order logic and object constraint language (HOL-OCL) [28] — приложение для верификации диаграмм классов UML, которая проводится посредством доказательства теорем с помощью универсального пакета Isabelle/HOL [29]. С помощью HOL-OCL возможно найти эквивалентные ограничения, но он не способен определить подмножество ограничений, нарушающих выполнимость исходной модели. HOL-OCL поддерживает верификацию диаграмм классов UML, за исключением стереотипов и ограничений кратности концов ассоциаций. С помощью HOL-OCL возможна также проверка выполнимости инвариантов OCL, при этом поддерживаются только основные типы данных OCL: целочисленный, действительный, строковый и логический. Обобщенные типы OCL, такие как OclVoid, OclModelElementType и OclType, не могут быть явно описаны в его внутреннем представлении.

2.5. Методы удовлетворения ограничений

Средства верификации данного класса редуцируют задачу верификации моделей к задаче удовлетворения ограничений (CSP) [30] и ее решению методами логического программирования. Задача формулируется как конечное множество переменных и множество определенных на них ограничений. Решение осуществляется в два этапа. На первом определяются кардинальные числа для множеств экземпляров каждого класса и ассоциации. На втором находятся допустимые значения атрибутов и ролей данных экземпляров. Области определения значений переменных, как правило, ограничиваются, что является существенным недостатком. Невозможность найти решение при текущих заданных границах может вовсе не означать его отсутствия в случае установки границ другим способом.

Наиболее известным приложением, относящимся к данному классу средств верификации моделей является UMLtoCSP [31, 32]. Данное приложение принимает модель UML в формате ArgoUML XMI и текстовый файл с ограничениями OCL, на основе исходных данных формулирует задачу удовлетворения ограничений (CSP), а затем использует основанный на расширении языка Prolog решатель ограничений ECLiPSe [33] для нахождения ее решения. При этом проверяется выполнимость модели, а также наличие избыточных ограничений. Весь процесс полностью автоматизирован и не требует вмешательства пользователя. UMLtoCSP поддерживает основные элементы диаграммы классов UML за исключением зависимостей, агрегаций и стереотипов, а также инварианты, пред- и постусловия OCL.

2.6. Средства валидации

В качестве примера рассмотрим одно из наиболее распространенных приложений валидации. UML-based Specification Environment (USE) [34] — приложение генерации тестовых последовательностей для моделей, описываемых на UML/OCL. Формализм, лежащий в основе USE, базируется на оригинальном языке ASSL (A Snapshot Sequence Language). На основе заданного пользователем исходного состояния моделируемой системы и набора параметров USE генерирует последовательность состояний системы, описываемых на языке ASSL. Затем, используя метод «обход с возвратами», оно проверяет корректность каждого из сгенерированных состояний и либо находит хотя бы одно корректное, либо делает вывод о некорректности модели. Из всех рассматриваемых в данном обзоре решений USE наиболее широко охватывает конструкции UML/OCL: поддерживаются диаграммы классов, последовательностей и активностей UML, инварианты, пред- и постусловия языка OCL. При проверке постусловий выполнение операций UML моделируется вручную. Пользователь должен соответствующим образом изменить популяцию объектов, а затем запустить тест, проверяющий, удовлетворяет ли модифицированная популяция данному постусловию. Приложение USE неприменимо для валидации масштабных моделей, поскольку лежащий в его основе формализм основан на переборе всех возможных состояний и для нахождения решения требуется полный обход пространства поиска.

2.7. Сравнение и критика

Обсудим возможности применения вышеперечисленных методов для решения рассматриваемой задачи верификации масштабных индустриально значимых моделей данных. Методы, основанные на редукции к системам линейных неравенств или к дескрипционным логикам, не могут полноценно использоваться для ее решения по причине недостаточного покрытия семантики языков информационного моделирования. Поддержка только ограничений кардинальности не позволит решить поставленную задачу в полном объеме. Подходы, основанные на доказательстве теорем или методах

валидации, требуют ручного вмешательства, что представляется проблематичным, принимая во внимание высокую размерность исходных данных.

Сравним оставшиеся два подхода по производительности. Для этого обратимся к работе [35]. В ней описаны тесты производительности с использованием решений на основе редукции к задачам SAT и CSP на различных наборах данных. Естественно, что длительность верификации всецело зависит от особенностей используемого метода и набора тестовых данных. Однако можно заметить следующее: ни одно из решений не показывает приемлемого результата при работе с масштабными моделями, содержащими порядка 1000 классов и 1000 ассоциаций, что соответствует реальным индустриальным моделям, применяемым на практике. Метод CSP не может завершиться за разумное время уже на моделях, содержащих несколько десятков классов. Метод SAT, как правило, работает быстрее CSP. Однако и он не справляется с моделями, состоящими из нескольких сотен классов. Таким образом, вышерассмотренные решения не могут использоваться для верификации масштабных индустриально значимых моделей данных.

3. Предлагаемый комбинированный метод верификации масштабных объектно-ориентированных моделей данных

3.1. Основные определения

В настоящей статье остановимся на рассмотрении диаграмм классов UML, ограничения в которых формулируются на языке OCL, а также спецификаций на языке EXPRESS, поскольку эти два языковых средства наиболее часто используются для моделирования данных в различных индустриальных областях. Они предоставляют широкий спектр конструкций для описания структур данных и семантических ограничений, накладываемых на них. Несмотря на некоторую разницу в используемой терминологии и наборе поддерживаемых языковых конструкций, легко выделить общую часть, характерную для большинства языков моделирования, перейдя на уровень метамодели. Основываясь на работах [36, 37, 38], введем следующие определения.

Определение 1. *Объектно-ориентированной моделью* называется система $S = \langle T, \prec, \triangleright, R \rangle$, где:

- $T = T_D \cup C$ — множество простых типов данных T_D и объектных типов модели $C = C_C \cup C_A$, представимых абстрактными и конкретными классами C_A и C_C соответственно;
- $\prec$ — частичный порядок на C, отражающий отношения обобщения/специализации между классами и используемый в качестве основы для встраиваемого механизма полиморфизма. Отношение $\prec$ обладает свойством транзитивности $\forall c_1, c_2, c_3 \in C, c_1 \prec c_2 \wedge c_2 \prec c_3 \Rightarrow c_1 \prec c_3$;
- $\triangleright$ — отношения ассоциации, композиции и агрегации (связи), устанавливаемые между классами модели;
- $R = R_C \cup R_M \cup R_U \cup R_F \cup R_L \cup R_G$ — множество семантических ограничений модели, представимое ограничениями кардинальности R_C, кратности связей R_M, уникальности связей R_U, уникальности атрибутов классов R_F, а также локальными R_L и глобальными R_G правилами. Локальные и глобальные правила представляются произвольными логическими выражениями, определяемыми на отдельных типах данных либо их совокупности соответственно.

Определение 2. *Невырожденной* будем называть модель S, такую что $C_C \in S \mid \exists c \in C_C$.

Введем следующие обозначения: $o_i(c)$ — экземпляр класса $c \in C$ (объект), $O = \{o_1(c_1),...,o_{m_1}(c_1),...,o_1(c_n),...,o_{m_n}(c_n)\}$ — конечное множество (коллекция) объектов модели, O_c — подмножество ее объектов, являющихся экземплярами класса $c \in C$ и формирующих его простой экстент, $\overline{O_c}$ — расширенный экстент, объединяющий экземпляры класса и всех его специализаций, а также $\{a_i\}_c, i = \overline{1,n}$ — конечное множество атрибутов класса $c \in C$, $\{a_i\}_{o(c)}, i = \overline{1,n}$ — конечное множество значений атрибутов объекта $o(c)$.

Определение 3. *Семантически корректной* будем называть коллекцию объектов O, удовлетворяющих всем ограничениям модели R.

Фундаментальным свойством корректности объектно-ориентированной модели является ее выполнимость, т.е. возможность создания набора

экземпляров классов модели, не нарушающих определенных в ней ограничений. В работах [32, 35] даются определения слабой и сильной выполнимости модели. Переформулируем эти определения с использованием введенных обозначений.

Определение 4. Невырожденная модель S называется *слабо выполнимой* в том и только том случае, если существует семантически корректная коллекция O, такая что $\exists c \in C \mid o(c) \in O$.

Определение 5. Невырожденная модель S называется *сильно выполнимой* в том и только том случае, если существует семантически корректная коллекция O, такая что $\forall c \in C_C \exists o(c) \in O$.

Если определение слабой выполнимости не вызывает нареканий, то определение сильной выполнимости может привести к ложному утверждению о семантической некорректности модели из-за ограничений кардинальности вида «из подмножества классов $C_1 \subset C$ в корректную коллекцию O могут попасть объекты только заданного количества классов m, такого что $m < |C_1|$». Рассмотрим конкретные примеры, приведенные на рис. 1. Пусть для абстрактного класса *A* определено ограничение кардинальности $r_A \in R_C$ в виде интервала $I_{r_A} = [1,1]$. Тогда невозможно создать семантически корректную коллекцию объектов, в которой бы присутствовали экземпляры обоих специализаций данного класса *C1* и *C2* (рис. 1а). Аналогичным образом, пусть для класса *B* определено ограничение кардинальности $r_B \in R_C$ в виде интервала $I_{r_B} = [1,1]$. Тогда невозможно создать семантически корректную коллекцию объектов, в которой бы одновременно присутствовали экземпляры классов *D1* и *D2*, связанные с классом *B* взаимно исключающими отношениями агрегации (рис. 1б). Очевидно, что в обоих примерах нарушается вышеприведенное определение сильной выполнимости. Тем не менее, обе модели являются корректными.

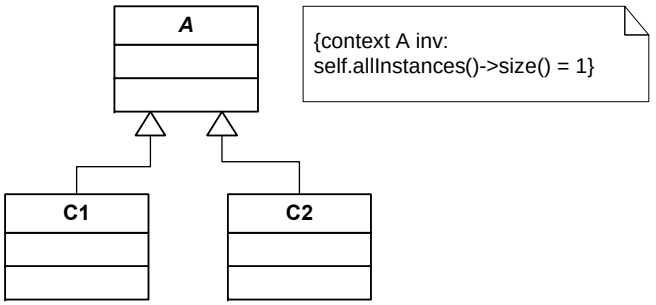

Рис. 1а. Отношения наследования, не допускающие наличия экземпляров одного из классов в коллекции.

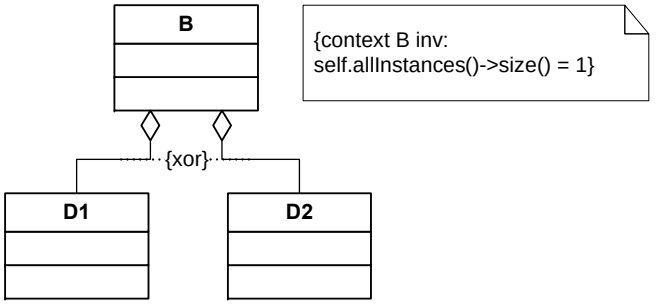

Рис. 1б. Отношения агрегации, не допускающие наличия экземпляров одного из классов в коллекции.

Дадим альтернативное определение сильной выполнимости, позволяющее избежать подобного ложного суждения о некорректности.

Определение 6. Невырожденная модель называется *сильно выполнимой* в том и только том случае, если существует множество семантически корректных коллекций $\{O\}$, такое что $\forall c \in C_C \exists o(c) \in O \mid O \subset \{O\}$.

3.2. Основные компоненты предлагаемого метода

Каждый из рассмотренных методов решения задачи верификации моделей работает по следующему сценарию: исходная задача редуцируется к той или иной математической постановке (к задаче линейного программирования, логического программирования, выполнимости булевых формул и т.п.), затем применяется известный метод решения поставленной математической задачи. При редукции каждому элементу верифицируемой модели ставится в соответствие одна или несколько переменных. Поиск решения осуществляется в области многомерного пространства, покрывающей все возможные

состояния модели. Однако размерность редуцируемой задачи становится настолько высокой, что с ней не справляются даже хорошо зарекомендовавшие себя подходы.

Анализ известных объектно-ориентированных моделей, применяемых в различных прикладных областях, показывает, что в них можно выделить группы сильно и слабо связанных между собой семантических ограничений. К первой группе можно отнести ограничения кардинальности, кратности связей и уникальности, определяющие в совокупности размер корректной популяции. Ко второй группе относятся локальные и глобальные правила, каждое из которых по отдельности влияет только на отдельный класс или небольшую группу взаимосвязанных классов.

В связи с этим для решения задачи верификации моделей предлагается использовать комбинированный метод, позволяющий сначала разрешить сильно связанные ограничения и определить необходимый размер семантически корректной коллекции объектов, а затем изолированно обрабатывать локальные и глобальные правила, определенные в модели, и установить корректные значения атрибутов объектов и ассоциаций между ними, удовлетворяющих каждому ограничению в отдельности. При этом формируются альтернативные наборы частных решений в виде набора элементарных операций инициализации атрибутов и ассоциаций. На заключительном этапе предполагается комбинировать полученные частные решения и согласовывать их между собой на основе оригинального метода семантической реконсиляции [39]. Данный метод подразумевает выявление зависимостей между операциями в параллельных транзакциях (применительно к рассматриваемой задаче это транзакции формирования данных для вышеупомянутых частных решений) и применение логического вывода для их семантически корректного слияния. В том случае, если в результате применения метода реконсиляции была сгенерирована коллекция данных, можно говорить о выполнимости модели и непротиворечивости в ней системы семантических ограничений. В случае отсутствия решения метод позволит определить несовместные операции и локализовать подмножество элементов исходной модели, в котором, возможно, допущена ошибка.

Таким образом, исходная задача верификации объектно-ориентированных моделей редуцируется к нескольким математическим постановкам: на этапе определения размера коллекции объектов — к задаче линейного программирования, на этапе формирования альтернативных частных решений — к задаче CSP, а на этапе их реконсиляции — к задаче выполнимости булевых формул (SAT). Методы решения данных математических задач хорошо известны, и для них имеется большое количество реализаций.

На наш взгляд, использование предлагаемого комбинированного метода позволит избежать проблем производительности, свойственных известным подходам. На начальном этапе размерность задачи будет относительно высокой. Однако известные методы редукции к системам линейных

неравенств и дальнейшему решению задачи линейного целочисленного программирования имеют наилучшую среди остальных полиномиальную оценку сложности [14, 15, 40]. На остальных этапах, когда область поиска решения будет локализована путем нахождения необходимого количества объектов, ограничения будут рассматриваться по отдельности и охватывать относительно небольшие группы объектов, что позволит серьезно сократить размерность задачи. С другой стороны, комбинированный метод позволит объединить достоинства отдельных методов, в частности обрабатывать подавляющее большинство известных языковых конструкций и достичь максимально возможной степени покрытия семантики языков информационного моделирования.

4. Определение размера корректной коллекции объектов

Ключевой проблемой, возникающей при генерации корректной коллекции объектов O, является определение допустимого количества экземпляров каждого класса из множества C, которое бы не нарушало семантические ограничения модели. Отметим, что упомянутые выше ограничения кардинальности, кратности и уникальности связей, уникальности атрибутов классов, а также отношения обобщения/специализации явным или неявным образом должны приниматься во внимание при решении данной задачи. Рассмотрим более подробно каждый из введенных видов ограничений и отношений в модели.

4.1. Ограничения кардинальности

Ограничения кардинальности задаются на классах модели в виде одного или нескольких интервалов и определяют допустимое число экземпляров соответствующих объектных типов. Например, в языке EXPRESS кардинальность объектных экстентов для некоторого класса c может быть определена явным образом с использованием логических выражений на основе функции `SIZEOF(c)`. Для класса UML ограничение кардинальности может быть специфицировано в виде инварианта OCL, в выражении которого участвует следующая последовательность операций: `self.allInstances()->size()`.

```
SCHEMA PersonOrganization;

  TYPE Label = STRING(255);
  END_TYPE;

  TYPE ActorRole = Label;
  END_TYPE;
```

```
  ENTITY Organization;
      Id          : INTEGER;
      Name        : Label;
      Description : OPTIONAL STRING;
      Roles       : LIST [1:?] OF UNIQUE ActorRole;
    INVERSE
      Engages     : SET [0:?] OF Person FOR EngagedIn;
    UNIQUE
      UR1 : Id;
    WHERE
      WR1 : SIZEOF( QUERY( temp <* Engages | 'Director'
in temp.Roles ) ) = 1;
  END_ENTITY;

  ENTITY Person;
      Id           : INTEGER;
      FamilyName   : OPTIONAL Label;
      GivenName    : OPTIONAL Label;
      MiddleNames  : OPTIONAL LIST [1:?] OF Label;
      PrefixTitles : OPTIONAL LIST [1:?] OF Label;
      SuffixTitles : OPTIONAL LIST [1:?] OF Label;
      Roles        : LIST [0:?] OF UNIQUE ActorRole;
      EngagedIn    : SET [0:?] OF Organization;
    UNIQUE
      UR1 : Id;
    WHERE
      WR1 : EXISTS(FamilyName) OR EXISTS(GivenName);
  END_ENTITY;

RULE DirectorRule FOR (Person);
  WHERE
    WR1 : SIZEOF( QUERY(temp <*Person | 'Director' in
temp.Roles AND
          SIZEOF( temp.EngagedIn ) <> 1 ) = 0;
END_RULE;

END_SCHEMA;
```

Рис 2. Представление схемы `PersonOrganization` *на языке EXPRESS*

Пусть $r \in R_C$ задает кардинальность соответствующего класса $c \in C$ в виде интервала I_r, тогда допустимое число экземпляров этого объектного типа должно лежать в данном интервале или:

$$low(I_r) \le \left|\overline{O_c}\right| \le high(I_r) \quad (1),$$

где функции $low(I_r)$ и $high(I_r)$ возвращают значения нижней и верхней границы интервала I_r соответственно.

Заметим, что переход от семантических ограничений кардинальности в языках информационного моделирования к линейным неравенствам носит необходимый, но не всегда достаточный характер. Рассмотрим это на примере фрагмента информационной схемы `PersonOrganization`, специфицированной на языке EXPRESS (см. рис. 2). Предположим, что каждую организацию возглавляет директор, который не может совмещать эту позицию с деятельностью в других организациях. В рамках информационной схемы это ограничение специфицировано с помощью глобального правила `DirectorRule`. Вместе с локальным правилом `WR1` в классе `Organization` оно приводит к очевидному условию для кардинальности: $\left|\overline{O_{PERSON}}\right| \ge \left|\overline{O_{ORGANIZATION}}\right|$.

Однако, чтобы удовлетворить исходные семантические ограничения, требуется не только создать необходимое количество объектов классов `Person` и `Organization`, но и задать правильным образом значения их атрибутов и ассоциаций. В рамках предложенной вычислительной стратегии это осуществляется на следующих этапах верификации, где размер генерируемой коллекции также может быть уточнен путем добавления или удаления некоторого количества объектов, необходимого для достижения ее корректного состояния.

Существенно, что если для класса c непосредственно определено n отношений специализации, $c \prec c_i, c_i \in C, i = \overline{1,n}$, то имеет место следующее равенство:

$$\left|\overline{O_c}\right| = \sum_{i=1}^{n} \left|\overline{O_{c_i}}\right| + \left|O_c\right| \quad (2),$$

и ограничение кардинальности неявно налагает дополнительные условия и для специализированных классов. В случае отсутствия специализаций у класса условие (2) трансформируется в простое и очевидное равенство: $\left|\overline{O_c}\right| = \left|O_c\right|$ (простой и расширенный экстенты в данном случае совпадают).

4.2. Множественное наследование

Остановимся более подробно на общей модели наследования классов, предусматриваемой языками информационного моделирования. Возьмем за основу схему наследования, применяемую в языке EXPRESS, как наиболее

общую. Все возможные случаи наследования в UML также покрываются данной схемой.

Итак, в языке EXPRESS схема наследования задается при определении суперкласса с помощью конструкции SUPERTYPE и выражения ограничения с использованием следующих спецификаторов:

— ONEOF устанавливает ограничение взаимоисключения для экземпляров подклассов, что соответствует ограничению disjoint в языке UML. Ни для одного из классов, входящих в список ONEOF, не может быть создан экземпляр, принадлежащий одновременно и любому другому классу из данного списка;

— ANDOR допускает любую комбинацию для конструируемых экземпляров подклассов, что соответствует ограничению overlapping в языке UML. Это означает, что экземпляр суперкласса может быть одновременно экземпляром более чем одной из его специализаций;

— AND определяет обязательные комбинации для экземпляров подклассов. Каждый экземпляр суперкласса в этом случае всегда формируется как составной объект и является одновременно экземпляром каждой из групп, разделенных в спецификации ограничения с помощью AND. Заметим, что непосредственный аналог данного ограничения в UML отсутствует, хотя его можно эмулировать путем комбинации ограничения overlapping и спецификаций перекрываемых классов как абстрактных.

Языком допускаются сложные вложенные выражения ограничений суперклассов. Однако в большинстве случаев используется простое наследование на основе конструкции ONEOF и сложное наследование на основе AND/ANDOR для групп, каждая из которых задается взаимоисключающим образом с помощью вложенной конструкции ONEOF. В случае простого наследования действует условие (2), приведенное в предыдущем разделе, где n — количество специализаций, указанное в конструкции ONEOF. Исследуем варианты множественного наследования и алгебраические условия для кардинальности, индуцируемые произвольными выражениями для ограничений суперкласса.

Введем оператор комбинации классов & для определения нового составного класса, экземпляры которого формируются как комбинации объектов перечисленных классов-операндов. Для оператора & имеют место следующие тождества:

```
A&A ≡ A
A&B ≡ B&A
A&(B&C) ≡ (A&B)&C = A&B&C
```

Первое тождество означает, что компонентами составного объекта могут являться лишь объекты различных классов. Второе и третье тождества

определяют свойства коммутативности и ассоциативности для оператора &, означая, что при составлении сложного объекта порядок классов не важен.

Определим оператор объединения классов как оператор задания множества перечисленных классов-операндов. Запись [A,B,C] означает задание множества классов A,B,C, а запись [] — пустое множество. Имеют место следующие тождества для введенного оператора объединения:

```
[A,B] ≡ [B,A]
[A,A,B] ≡ [A,B]
[A,[B,C]] ≡ [A,B,C]
[A]&[B,C] ≡ [A&B,A&C]
```

Отметим, что оператор комбинации классов аналогичен спецификатору AND, а оператор объединения классов — спецификатору ONEOF. Используя вышеописанный формализм, можно переписать выражения ограничений суперклассов в терминах множеств. Список ONEOF приводится к множеству, содержащему соответствующий набор классов. Выражение со спецификатором AND может быть заменено оператором комбинации классов & с соответствующими операндами. Наконец, спецификатор ANDOR задает множество, состоящее из классов-операндов по отдельности и их комбинации. Правила и типовые случаи приведения выражений для ограничений суперклассов даются ниже:

```
ONEOF(A,B,...)──> [A,B,...]
A AND B —> [A&B]
A ANDOR B —> [A,B,A&B]
A AND ONEOF(B1,B2) —> A&[B1,B2] = [A&B1,A&B2]
ONEOF(A1,A2) AND ONEOF(B1,B2) ──>
[A1,A2]&[B1,B2] =[A1&B1,A1&B2,A2&B1,A2&B2]
```

Применяя данные правила рекурсивно, можно «раскрыть» выражение ограничения суперкласса и получить требуемое представление для всех допустимых комбинаций классов, не содержащее ни одного терма ONEOF, ANDOR и AND. Тогда условие для кардинальности приобретает вид:

$$\left|\overline{O_c}\right| = \sum_{i=1}^{n} \left|O_{c_i}\right| + \left|O_c\right| \qquad (3),$$

где n — общее количество специализаций суперкласса, порождаемых всеми его простыми и составными подклассами рекурсивно. В отличие от равенства (2) здесь суммируются простые экстенты специализаций, поскольку составные объекты принадлежат одновременно расширенным экстентам нескольких специализаций и требуется исключить их повторный подсчет. Однако при этом следует рекурсивно учесть все возможные специализации суперкласса, включая не только его непосредственные, но и порождаемые ими.

Аналогичные условия могут быть получены для всех суперклассов объектно-ориентированной модели. Отметим, что формированию вектора

кардинальности должны предшествовать анализ отношений множественного наследования и идентификация всех составных классов в соответствии с описанными выше правилами раскрытия выражений для ограничений суперклассов.

4.3. Ограничения кратности и уникальности связей

Ограничения кратности связей R_M определяются на отношениях $\triangleright$ таким образом, что с каждым $r \in R_M$ ассоциировано два интервала I_r^f и I_r^b, задающих допустимые значения кратностей прямого и обратного отношений соответственно. Ограничения уникальности $r_U(c_1 \triangleright c_2) \in R_U$ определяются непосредственно на отношениях ассоциации между классами и исключают повторное вхождение одних и тех же объектов. Отношения композиции и агрегации автоматически предполагают наличие ограничений уникальности. Выразим данные семантические ограничения в виде алгебраических условий.

Для обязательных ассоциаций с неуникальными элементами должны выполняться следующие условия:

$$\begin{cases} low(I_r{}^f) \geq 1 \wedge \left|\overline{O_{c_1}}\right| \geq 1 \rightarrow \left|\overline{O_{c_2}}\right| \geq 1 \\ low(I_r{}^b) \geq 1 \wedge \left|\overline{O_{c_2}}\right| \geq 1 \rightarrow \left|\overline{O_{c_1}}\right| \geq 1 \end{cases} \quad (4),$$

поскольку задание нижней ненулевой границы кратности предполагает существование, по крайней мере, одного экземпляра ассоциируемого класса, как для прямого, так и для обратного отношения ассоциации.

Для обязательных ассоциаций с уникальными элементами должна выполняться следующая система условий:

$$\begin{cases} \left|\overline{O_{c_1}}\right| \geq 1 \rightarrow \left|\overline{O_{c_2}}\right| \geq low(I_r^f) \\ \left|\overline{O_{c_2}}\right| \geq 1 \rightarrow \left|\overline{O_{c_1}}\right| \geq low(I_r^b) \\ low(I_r^f)\left|\overline{O_{c_1}}\right| \leq high(I_r^b)\left|\overline{O_{c_2}}\right| \\ low(I_r^b)\left|\overline{O_{c_2}}\right| \leq high(I_r^f)\left|\overline{O_{c_1}}\right| \end{cases} \quad (5).$$

Первые два условия требуют наличие минимального количества экземпляров ассоциируемых классов, определяемого нижними границами кратностей соответствующих отношений. Следующие два условия позволяют исключить случаи невозможного разрешения ассоциаций в силу недостатка или избытка объектов соответствующих классов.

Выделим важные частные случаи формализации алгебраических условий по заданным ограничениям кратности. Для однонаправленной ассоциации

кратность обратного отношения можно считать равной $I_r^b = [0, \infty]$, и в этом случае последние два условия вырождаются. В случае задания обязательной ассоциации «один к одному» ($I_r^f = [1,1]$, $I_r^b = [1,1]$) система сводится к простому равенству $\left|\overline{O_{c_1}}\right| = \left|\overline{O_{c_2}}\right|$. Если два класса $c_1, c_2 \in C$ связаны отношением композиции, то кратность обратного отношения равна $I_r^b = [1,1]$ и система приобретает вид $low(I_r^f)\left|\overline{O_{c_1}}\right| \leq \left|\overline{O_{c_2}}\right| \leq high(I_r^f)\left|\overline{O_{c_1}}\right|$.

4.4. Ограничения уникальности атрибутов

Ограничения уникальности атрибутов $r \in R_F$ определяются для отдельных классов модели $c \in C$ путем задания группы атрибутов $\{c.a_i\}, i = \overline{1,n}$, комбинации значений которых не могут повторяться в семантически корректной коллекции данных. Заметим, что задание ограничений уникальности на атрибутах, имеющих конечную область определения значений, может налагать дополнительные условия на допустимое количество объектов соответствующего класса. К таковым, в частности, относятся атрибуты, определенные на перечислимых, логических типах или же на целых числах с дополнительными ограничениями области допустимых значений. Пусть в классе $c \in C$ задано ограничение уникальности $r \in R_F$ для группы атрибутов $\{c.a_i\}, i = \overline{1,n}$, определенных на конечных множествах $t_i \in T_D, i = \overline{1,n}$, тогда должно выполняться:

$$\left|\overline{O_c}\right| \leq \left|t_1\right|\left|t_2\right| \ldots \left|t_n\right| \quad (6),$$

где $\left|t_i\right|$ — мощность соответствующего домена.

Обсудим более экзотические, но возможные варианты задания ограничения уникальности на коллекциях. Здесь интерес представляют случаи, когда тип элементов коллекции имеет ограниченное множество значений. К ним относятся рассмотренные выше перечислимые типы, логические типы, целочисленные типы с дополнительными ограничениями допустимой области значений, а также классы, участвующие в типизированных ассоциациях. Для краткости ограничимся наиболее часто применяемыми в языках информационного моделирования вариантами коллекций, определенных на упорядоченных и неупорядоченных множествах и мультимножествах.

Итак, пусть в классе $c \in C$ определено ограничение уникальности $r \in R_F$ для атрибута, являющегося коллекцией с кратностью, заданной интервалом

I_r, и типом элементов $t_j \in T_D$, являющимся конечным множеством значений. Пусть $k = \left| t_j \right|$ — мощность соответствующего домена (заметим, что если коллекция является ассоциацией с классом $c_j \in C$, то $k = \left| \overline{O_{c_j}} \right|$), $n = low(I_r)$, $m = high(I_r)$. Тогда данное семантическое ограничение приводит к следующим условиям кардинальности.

Для множества это — $\left| \overline{O_c} \right| \le \sum_{i=n}^{m} C_k^i$, где суммирование ведется по всем возможным сочетаниям значений из имеющегося набора размером k. Это довольно слабое условие. Для мультимножества аналогичное условие кардинальности приобретает вид $\left| \overline{O_c} \right| \le \sum_{i=n}^{m} C_{k+i-1}^{k-1}$, поскольку элементы в мультимножествах могут повторяться и это приводит к увеличению верхней оценки для кардинальности $\left| \overline{O_c} \right|$. Для упорядоченного мультимножества условие слабее, чем для множества, но сильнее, чем для мультимножества: $\left| \overline{O_c} \right| \le \sum_{i=n}^{m} k^i$. Наконец, для упорядоченного множества условие определяется общим количеством возможных размещений и имеет вид $\left| \overline{O_c} \right| \le \sum_{i=n}^{m} A_k^i$.

Следующая таблица объединяет полученные результаты для различных типов коллекций. Из полученных оценок можно выделить два случая, дающих линейные ограничения: первый, когда заранее известно, что $k = 1$ (например, это следует из другого ограничения модели), второй, когда ассоциация не является коллекцией, т.е. $n = m = 1$ (см. табл. 2). Дополнительный случай, когда вышеприведенные соотношения трансформируются в линейные неравенства, соответствует коллекциям необъектных типов. Тогда $k = \left| t_j \right|$ не является переменной задачи и комбинаторные выражения вырождаются в константы.

Тип коллекции	Условие кардинальности	Оценка
Мультимножество	$\left\|\overline{O_c}\right\| \le \sum_{i=n}^{m} C_{k+i-1}^{k-1}$	$k = 1 \to \left\|\overline{O_c}\right\| \le m - n + 1$ $n = m = 1 \to \left\|\overline{O_c}\right\| \le k$
Упорядоченное мультимножество	$\left\|\overline{O_c}\right\| \le \sum_{i=n}^{m} k^i$	$k = 1 \to \left\|\overline{O_c}\right\| \le m - n + 1$ $n = m = 1 \to \left\|\overline{O_c}\right\| \le k$
Множество	$\left\|\overline{O_c}\right\| \le \sum_{i=n}^{m} C_k^i$	$k = 1 \to \left\|\overline{O_c}\right\| \le 1$ $n = m = 1 \to \left\|\overline{O_c}\right\| \le k$
Упорядоченное множество	$\left\|\overline{O_c}\right\| \le \sum_{i=n}^{m} A_k^i$	$k = 1 \to \left\|\overline{O_c}\right\| \le 1$ $n = m = 1 \to \left\|\overline{O_c}\right\| \le k$

Табл. 2. Условия кардинальности, порождаемые уникальными коллекциями

4.5. Редукция к задаче линейного программирования

Для заданной объектно-ориентированной модели $S = \langle T, \prec, \triangleright, R \rangle$ поставим задачу определения мощностей объектных экстентов $\left| O_c \right|$ для всех конкретных классов $c \in C_C$, при которых коллекция объектов O будет являться семантически корректной. Решение задачи в данной постановке соответствует поиску необходимого количества объектов для генерации коллекции с целью проверки сильной выполнимости верифицируемой модели согласно определению 5. В предположении, что семантические ограничения исходной модели допускают одновременное наличие экземпляров каждого конкретного класса, сведем задачу определения мощностей к традиционной постановке линейного целочисленного программирования [40, 41].

Сформируем вектор неизвестных переменных $\{x_i\} \in \aleph$, $i = \overline{1,k}$, $k = \left| C_C \right|$, каждая из которых ассоциирована с соответствующим конкретным классом модели $c_i \in C_C$ и определяет мощность его простого экстента $x_i = \left| O_{c_i} \right|$. Рассмотрим задачу минимизации $\min f(x)$ с целевой функцией

$f(x) = \sum_{i=1}^{k} x_i$, что будет соответствовать поиску простейшей репрезентативной коллекции данных. Составим общую систему линейных неравенств путем включения в нее соотношений (1) для всех ограничений кардинальности, определенных для классов модели, условий (5) — для ассоциаций, агрегаций и композиций с уникальными элементами, неравенств (6) — для ограничений уникальности атрибутов, заданных на конечных доменах. Равенства (2) для простых или (3) для сложных отношений специализации выражаются в переменных x_i с учетом условия, что мощности простых экстентов абстрактных классов равны нулю, и подставляются в соотношения (1), (5), (6). Поскольку в рассматриваемой постановке предполагается, что $\left|\overline{O_{c_i}}\right| \geq 1, i = \overline{1,n}, n = |C|$, то условия (4) вырождаются и не включаются в общую систему задачи линейного программирования, а условия (5) приобретают вид линейных неравенств. К системе добавляются дополнительные неравенства:

$$x_i \geq 1, i = \overline{1,k} \quad (7),$$

формирующие необходимые условия наличия хотя бы одного экземпляра каждого конкретного класса в генерируемой коллекции. Что касается условий кардинальности, порождаемых уникальными коллекциями (см. табл. 2), то они включаются в систему в тех вышеупомянутых случаях, когда являются линейными неравенствами. Более слабые ограничения в остальных случаях предлагается проверять как постусловия.

Сформированная задача может быть решена известными методами, такими как симплекс-метод или метод угловых точек [42]. Найденное решение может быть взято за основу для генерации объектной коллекции с целью верификации сильной выполнимости модели. Факт отсутствия решения служит основанием для заключения о невыполнимости модели и наличии в ней избыточных или переопределенных ограничений. К сожалению, данный метод не позволяет локализовать множество данных ограничений. Для данных целей необходимо применять другие методы [14].

Заметим, что для проверки слабой выполнимости согласно определению 4 достаточно наличия одного объекта заданного класса $c_j \in C_C$. Однако формирование семантически корректного набора данных может потребовать задание и вспомогательных объектов в дополнение к основному, поскольку ограничения кардинальности носят нетривиальный характер. Тогда задача определения мощностей объектных экстентов сводится к аналогичной постановке линейного целочисленного программирования, где условия (7) трансформируются в следующие:

$$\begin{cases} x_j \geq 1 \\ x_i \geq 0 \end{cases}, i = \overline{1,k}, i \neq j \quad (8).$$

Для подтверждения сильной выполнимости модели согласно определению 6 (в случаях, когда семантические ограничения модели не допускают одновременное наличие экземпляров каждого конкретного класса) необходимо и достаточно сформировать множество из $k = |C_C|$ коллекций, каждая из которых включает хотя бы один объект класса $c_i \in C_C, i = \overline{1,k}$ и служит для проверки слабой выполнимости.

4.6. Вычислительный эксперимент

С целью апробации разработанного метода определения размера корректной коллекции был проведен вычислительный эксперимент. В качестве исходной модели взята последняя редакция IFC 4. Ее формальный статический анализ обнаружил 766 классов (из них 123 абстрактных и 643 конкретных), 207 отношений специализации, определенных в суперклассах данной модели, 63 бинарных ассоциации «один к одному», 1 обязательная бинарная ассоциация «многие ко многим» с уникальными элементами и невырожденными границами, 4 ограничения уникальности на символьных строках переменной длины. IFC использует простую модель наследования, где все специализации связаны соотношением `ONEOF` в рамках суперкласса. Таким образом, анализ сложной иерархии не требуется и параметры задачи линейного программирования могут быть сформированы за линейное время. Заметим, что количество ассоциаций в модели IFC значительно больше, но подавляющее большинство из них имеет вырожденные границы и не участвует в формировании задачи.

Вектор неизвестных задачи имеет размер 643, что соответствует числу конкретных классов. В систему включаются 63 равенства к которым сводится система условий вида (5) для ассоциаций «один к одному» и 2 неравенства для ассоциации «многие ко многим». Ограничения уникальности определяются на символьных строках переменной длины с максимальной границей 255, что соответствует случаю уникального упорядоченного мультимножества. Поскольку алфавит в IFC ограничивается печатными символами ASCII (то есть мощность домена $k = 95$), соответствующие условия кардинальности приобретают вид: $\left|\overline{O_c}\right| \leq \sum_{i=1}^{255} 95^i$. С учетом того, что решается задача нахождения минимальной репрезентативной коллекции данных, подобными условиями можно пренебречь. В систему также дополнительно включаются 643 неравенства вида (7) для каждой из переменных задачи. Проведенный вычислительный эксперимент показал, что для относительно сложной модели

данных метод демонстрирует производительность, приемлемую для практического использования.

5. Пример применения комбинированного метода

Рассмотрим применение предложенного комбинированного метода на примере простейшей модели. Сборник научных трудов может включать от 10 до 20 статей. Каждая статья имеет не более двух авторов и должна быть одобрена тремя рецензентами. Авторами статей могут быть как студенты (только в соавторстве со своим научным руководителем), так и научные сотрудники, а рецензентами — только научные сотрудники. Каждый автор и каждый рецензент участвует в написании либо, соответственно, обзоре только одной статьи. При этом научные сотрудники не имеют право рецензировать собственные статьи. Научные сотрудники могут руководить несколькими студентами, студент обязан иметь одного научного руководителя. Соответствующая UML модель с OCL ограничениями представлена на рис. 3.

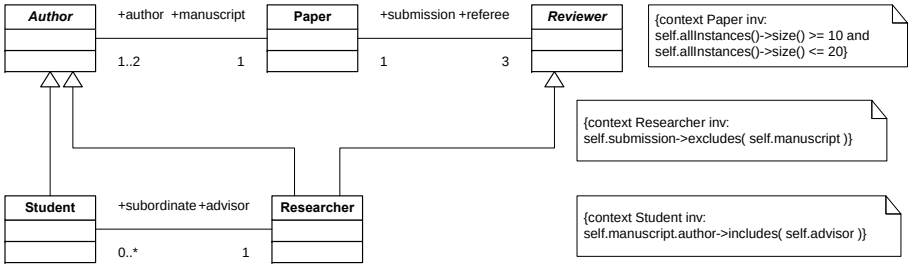

Рис. 3. Пример: UML модель «сборник научных трудов»

Множество $C = \{Paper, Author, Reviwer, Researcher, Student\}$ формирует систему классов модели, при этом $C_C = \{Paper, Researcher, Student\}$, а $C_A = \{Author, Reviwer\}$. Отношения специализации и ассоциации устанавливаются следующим образом: $Author \prec Student$, $Author \prec Researcher$, $Reviewer \prec Researcher$, $Paper \triangleright Author$, $Paper \triangleright Reviewer$, $Student \triangleright Researcher$. Введем следующие переменные: $x_1 = |O_{PAPER}|$, $x_2 = |O_{RESEARCHER}|$, $x_3 = |O_{STUDENT}|$. Отношения специализации, установленные в данной модели, приводят к следующим равенствам: $|\overline{O_{AUTHOR}}| = x_2 + x_3$, $|\overline{O_{REVIEWER}}| = x_2$. Тогда задача определения размера коллекции сводится к следующей математической постановке:

$$f(x) = x_1 + x_2 + x_3 \to \min \quad (9)$$

$$x_1 \geq 10 \quad (10)$$

$$x_1 \leq 20 \quad (11)$$

$$x_2 \geq 3 \quad (12)$$

$$x_2 = 3x_1 \quad (13)$$

$$x_1 \leq x_2 + x_3 \quad (14)$$

$$x_2 + x_3 \leq 2x_1 \quad (15)$$

$$x_3 \geq 1 \quad (16)$$

Здесь неравенства (10) и (11) индуцированы явным ограничением кардинальности класса `Paper`, специфицированным на OCL, неравенство (12) и равенство (13) — ассоциативным отношением $Paper \triangleright Reviewer$, неравенства (14) и (15) — ассоциативным отношением $Paper \triangleright Author$. Дополнительное неравенство (16) требует наличия хотя бы одного экземпляра класса `Student` в коллекции, генерируемой с целью проверки сильной выполнимости исходной модели.

Заметим, что данная задача не имеет решения, что сигнализирует о невыполнимости модели и наличии в ней переопределенных ограничений. В самом деле, согласно установленному ассоциативному отношению $Paper \triangleright Reviewer$ рецензентов (в данном случае научных сотрудников) должно быть в три раза больше, чем статей. При этом, поскольку каждый научный сотрудник еще является автором, он также должен участвовать и в написании хотя бы одной статьи, что определяется кардинальностью роли manuscript $I_r = [1,1]$ ассоциативного отношения $Paper \triangleright Author$. Но согласно установленной кардинальности $I_r = [1,2]$ роли `author` в данном отношении, общее количество авторов (включая как студентов, так и научных сотрудников) не может превышать количество статей больше, чем в два раза. Таким образом, одно из ограничений кардинальности, определяемое ассоциативными отношениями в модели, будет нарушено в любом случае.

Задача разрешима, если снять требование обязательности каждого автора участвовать в написании одной статьи, т.е. определить кардинальность роли `manuscript` как $I_r = [0,1]$. Тогда условие (15) вырождается и будет найдено следующее решение: $x_1 = 10$, $x_2 = 30$, $x_3 = 1$. Таким образом, для проверки сильной выполнимости модели необходимо сгенерировать коллекцию, содержащую 10 экземпляров класса `Paper`, 30 — класса `Researcher` и 1 — класса `Student`. Однако вывод о сильной выполнимости модели можно будет сделать только после корректной установки всех

ассоциативных отношений между сгенерированными экземплярами с учетом ограничений, что научный сотрудник не может рецензировать собственные статьи, а студенческая статья может быть выпущена только в соавторстве с научным руководителем.

Рассмотрим пример установки ассоциативных отношений в небольшом подмножестве сгенерированных объектов: $p_1 \in Paper$, $s_1 \in Student$, $r_1, r_2, r_3, r_4 \in Researcher$. Пусть при установке ассоциаций применяется следующий метод: ассоциированные экземпляры последовательно выбираются из множества допустимых объектов с их дальнейшим исключением из данного множества как уже задействованных. Тогда, учитывая кардинальности ассоциаций, соответствующие отношения могут быть расставлены следующим образом: (1) $s_1 \rhd r_1$, (2) $p_1 \rhd s_1$, (3) $p_1 \rhd (author) r_1$, (4) $p_1 \rhd (referee) r_1$, (5) $p_1 \rhd (referee) r_2$, (6) $p_1 \rhd (referee) r_3$. Анализ ограничения в контексте класса `Researcher` устанавливает следующие операции как взаимоисключающие: $(3) \oplus (4)$, а в контексте класса `Student` — выявляет следующее логическое соотношение: $(2) \rightarrow (1) \wedge (3)$. Тогда один из возможных результатов реконсиляции выглядит следующим образом: (1) $s_1 \rhd r_1$, (2) $p_1 \rhd s_1$, (3) $p_1 \rhd (author) r_1$, (5) $p_1 \rhd (referee) r_2$, (6) $p_1 \rhd (referee) r_3$, (7) $p_1 \rhd (referee) r_4$, где операция (7) добавлена для установки нового отношения межу статьей и рецензентом взамен исключенного (4).

6. Заключение

Таким образом, рассмотрена проблема верификации масштабных моделей данных и предложен комбинированный метод для ее решения. Он основан на последовательной редукции к нескольким постановкам известных математических задач: линейного программирования, удовлетворения ограничений (CSP), выполнимости булевых формул (SAT). Проведенные эксперименты демонстрируют перспективность комбинированной вычислительной стратегии и эффективность предложенного метода для верификации масштабных моделей данных. В настоящее время он позволяет успешно разрешить следующие виды ограничений: ограничения кардинальности объектных экстентов, кратности и уникальности связей, уникальности атрибутов объектов. Однако для успешного разрешения ограничений, задаваемых произвольными выражениями, необходимо провести дальнейшие исследования и реализовать остальные его компоненты, связанные с поиском частных решений, удовлетворяющих отдельным ограничениям, и их семантической реконсиляции.

Литература

[1] Model Driven Architecture: The Architecture of Choice for a Changing World. Executive Overview, October 2013. http://www.omg.org/mda/executive_overview.htm

[2] Calafat A.Q. Validation of UML Conceptual Schemas with OCL Constraints and Operators. PhD Thesis, adviced by Dr. E. Teniente, Universitat Politecnica de Catalunya, Barcelona, 2009.

[3] ISO 10303-11:2004. Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.

[4] Unified Modeling Language (UML), V2.4.1, Release Date: August 2011. http://www.omg.org/spec/UML/2.4.1

[5] Object Constraint Language (OCL), V2.3.1, Release Date: January 2012. http://www.omg.org/spec/OCL/2.3.1

[6] IFC4 Release Summary, March 2013. http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4-release/ifc4-release-summary

[7] Khemlani L. The CIS/2 Format: Another AEC Interoperability Standard. // AECbytes Newsletter, July 27, 2005. http://www.aecbytes.com/buildingthefuture/2005/CIS2format.html

[8] ISO 10303-1:1994. Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.

[9] ISO 15926-1:2004. Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 1: Overview and fundamental principles.

[10] Чен П. Модель «сущность-связь» — шаг к единому представлению о данных. // СУБД, № 3, 1995. http://www.osp.ru/dbms/1995/03/271.htm

[11] Alloy 4: a language & tool for relational models, 2012. http://alloy.mit.edu/alloy

[12] Formica A. Finite Satisfiability of Integrity Constraints in Object-Oriented Database Schemas. // IEEE Transactions on Knowledge and Data Engineering, 14(1), 2002, pp. 123–139.

[13] Baader F., et al. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

[14] Lenzerini M., Nobili P. On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata. // Proceedings of the 13th VLDB Conference, 1987, pp. 147–154.

[15] Maraee A., Balaban M. Efficient Reasoning about Finite Satisfiability of UML Class Diagrams with Constrained Generalization Sets. // LNCS 4530, Proceedings of European Conference on Model Driven Architecture — Foundations and Applications, 2007, pp. 17–31.

[16] Jackson D. Alloy: a lightweight object modelling notation. // ACM Transactions on Software Engineering and Methodology, 11(2), 2002, pp. 266–290.

[17] Anastasakis K., Bordbar B., Georg G., Ray I. UML2Alloy: A challenging model transformation. // LNCS 4735, Model Driven Engineering Languages and Systems, 2007, pp. 436–450.

[18] Ramirez A., Vanpeperstraete P., Rueckert A., Odutola K., Bennett J., Tolke L., van der Wulp W. ArgoUML User Manual: A tutorial and reference description, 2011. http://argouml.tigris.org/ documentation/manual-0.34

[19] Soeken M., Wille R., Kuhlmann M., Gogolla M., Drechsler R. Verifying UML/OCL Models Using Boolean Satisfiability. // Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010, pp. 1341–1344.
[20] Turhan A.Y. Description Logic Reasoning for Semantic Web Ontologies. // WIMS'11 Proceedings of the International Conference on Web Intelligence, Mining and Semantics, Article 6, 2011.
[21] Berardi D., Calvanese D., de Giacomo G. Reasoning on UML Class Diagrams. // Artificial Intelligence, 168(1–2), 2005, pp. 70–118.
[22] The FaCT System, April 2003. http://www.cs.man.ac.uk/~horrocks/FaCT
[23] Haarslev V., Möller R. RACER system description. // LNAI 2083, Automated Reasoning, 2001, pp. 701–705.
[24] van der Straeten R., Mens T., Simmonds J., Jonckers V. Using Description Logic to Maintain Consistency between UML Models. // LNCS 2863, The Unified Modeling Language. Modeling Languages and Applications, 2003, pp. 326–340.
[25] Loom Project Home Page, July 2007. http://www.isi.edu/isd/LOOM
[26] PVS Specification and Verification System, January 2013. http://pvs.csl.sri.com
[27] Kyas M., Fecher H., de Boer F.S., Jacob J., Hooman J., van der Zwaag M., Arons T., Kugler H. Formalizing UML Models and OCL Constraints in PVS. // Electronic Notes in Theoretical Computer Science, 115, 2005, pp. 39–47.
[28] Brucker A.D., Wolff B. The HOL-OCL book. Technical Report 525, ETH Zurich, 2006.
[29] Nipkow T., Paulson L.C., Wenzel M. Isabelle/HOL — A Proof Assistant for Higher-Order Logic. // LNCS 2283, 2002.
[30] Tsang E. Foundations of constraint satisfaction. Academic Press Limited, London & San-Diego, 1993.
[31] Cabot J., Clariso R., Riera D. UMLtoCSP: a tool for the formal verification of UML/OCL models using constraint programming. // Proceedings of the 22nd ACM/IEEE International Conference on Automated Software Engineering (ASE '07), 2007, pp. 547–548.
[32] Cabot J., Clariso R., Riera D. Verification of UML/OCL Class Diagrams using Constraint Programming. // Proceedings of the IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08), 2008, pp. 73–80.
[33] Apt K.R., Wallace M.G. Constraint Logic Programming using ECLiPSe. Cambridge University Press, 2007.
[34] Gogolla M., Bohling J., Richters M. Validating UML and OCL models in USE by automatic snapshot generation. // Software & System Modeling, 4(4), 2005, pp. 386–398.
[35] Shaikh A., Wiil U.K., Memon N. Evaluation of Tools and Slicing Techniques for Efficient Verification of UML/OCL Class Diagrams. // Advances in Software Engineering, 2011, Article ID 370198, 18 p.p.
[36] Breu R., Hinkel U., Hofmann C., Klein C., Paech B., Rumpe B., Thurner V. Towards a Formalization of the Unified Modeling Language. // LNCS 1241, ECOOP'97 — Object-Oriented Programming, 1997, pp. 344–366.
[37] Richters M., Gogolla M. On Formalizing the UML Object Constraint Language OCL. // LNCS 1507, Conceptual Modeling — ER'98, 1998, pp. 449–464.
[38] Семенов В.А., Морозов С.В., Тарлапан О.А. Инкрементальная верификация объектно-ориентированных данных на основе спецификации ограничений. // Труды Института системного программирования / под ред. В.П. Иванникова, т.8, ч.2, 2004, с. 21–52.

[39] Семенов В.А., Ерошкин С.Г., Караулов А.А., Энкович И.В. Семантическая реконсиляция прикладных данных на основе моделей. // Труды Института системного программирования / под ред. В.П. Иванникова, т.13, ч.2, 2007, с. 141–164.
[40] Vanderbei R.J. Linear Programming. Foundations and Extensions. Third edition. Princeton University, 2008.
[41] Шевченко В.Н., Золотых Н.Ю. Линейное и целочисленное линейное программирование. — Нижний Новгород: издательство Нижегородского госуниверситета им. Н.И. Лобачевского, 2004.
[42] Карманов В.Г. Математическое программирование: учебное пособие. 5-е издание. — М.: Физматлит, 2004.

A combined method for verification of large-scale data models

V.A. Semenov, S.V. Morozov, D.V. Ilyin
ISP RAS, Moscow, Russia
sem@ispras.ru, serg@ispras.ru, denis.ilyin@ispras.ru

Annotation. The paper is addressed to the actual problem of verification of large-scale data models applied in various industrial areas and specified using popular general-purpose object-oriented languages, such as EXPRESS, UML/OCL. Main benefits of information modeling languages (high expressiveness, declarative nature, advanced set of syntactic units) negatively affect the process of automatic verification of the specifications. The known approaches are based on reduction of the original complex problem to some well-known mathematical statement and its solution by existing methods. The performed analytical survey of the existing methods for model verification demonstrates that they cannot be used for solving the problem because of their high computational complexity. A combined method for verification of large-scale data models is proposed in the paper. The method is based on sequential reduction to the several mathematical problem statements: linear programming, constraint satisfaction problem (CSP), Boolean satisfiability (SAT). Usage of the combined method allows to avoid efficiency issues peculiar to the known approaches. At the first step the polynomial complexity methods of integer linear programming are applied to the original large-scale problem and localize the search region for solution by detection of the necessary amount of objects. At the next steps constraints imposed onto relatively small groups of objects can be considered individually, which allows to reduce significantly dimension of the problem. The key problem of estimation of the number of instances intended for generation of correct object collection and its reduction to an integer linear programming problem is investigated in detail. The performed experiments demonstrate prospectivity of the combined computational strategy and efficiency of the proposed method for verification of large-scale data models. The work is supported by RFBR (grant 13-07-00390).

Keywords: model verification, object-oriented modeling, UML/OCL, EXPRESS, constraint logic programming, linear integer programming, semantic reconciliation

References

[1] Model Driven Architecture: The Architecture of Choice for a Changing World. Executive Overview, October 2013. http://www.omg.org/mda/executive_overview.htm
[2] Calafat A.Q. Validation of UML Conceptual Schemas with OCL Constraints and Operators. PhD Thesis, adviced by Dr. E. Teniente, Universitat Politecnica de Catalunya, Barcelona, 2009.
[3] ISO 10303-11:2004. Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.
[4] Unified Modeling Language (UML), V2.4.1, Release Date: August 2011. http://www.omg.org/spec/UML/2.4.1

[5] Object Constraint Language (OCL), V2.3.1, Release Date: January 2012. http://www.omg.org/spec/OCL/2.3.1
[6] IFC4 Release Summary, March 2013. http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4-release/ifc4-release-summary
[7] Khemlani L. The CIS/2 Format: Another AEC Interoperability Standard. *AECbytes Newsletter*, July 27, 2005. http://www.aecbytes.com/buildingthefuture/2005/CIS2format.html
[8] ISO 10303-1:1994. Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.
[9] ISO 15926-1:2004. Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 1: Overview and fundamental principles.
[10] Chen P. The entity-relationship model — toward a unified view of data. *ACM Transactions on Database Systems*, vol. 1, no. 1, 1976, pp. 9–36.
[11] Alloy 4: a language & tool for relational models, 2012. http://alloy.mit.edu/alloy
[12] Formica A. Finite Satisfiability of Integrity Constraints in Object-Oriented Database Schemas. *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 1, 2002, pp. 123–139.
[13] Baader F., et al. The Description Logic Handbook: Theory, Implementation and Applications. *Cambridge University Press*, 2003.
[14] Lenzerini M., Nobili P. On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata. *Proceedings of the 13th VLDB Conference*, 1987, pp. 147–154.
[15] Maraee A., Balaban M. Efficient Reasoning about Finite Satisfiability of UML Class Diagrams with Constrained Generalization Sets. *LNCS 4530, Proceedings of European Conference on Model Driven Architecture — Foundations and Applications*, 2007, pp. 17–31.
[16] Jackson D. Alloy: a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 2, 2002, pp. 266–290.
[17] Anastasakis K., Bordbar B., Georg G., Ray I. UML2Alloy: A challenging model transformation. *LNCS 4735, Model Driven Engineering Languages and Systems*, 2007, pp. 436–450.
[18] Ramirez A., Vanpeperstraete P., Rueckert A., Odutola K., Bennett J., Tolke L., van der Wulp W. ArgoUML User Manual: A tutorial and reference description, 2011. http://argouml.tigris.org/ documentation/manual-0.34
[19] Soeken M., Wille R., Kuhlmann M., Gogolla M., Drechsler R. Verifying UML/OCL Models Using Boolean Satisfiability. *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2010, pp. 1341–1344.
[20] Turhan A.Y. Description Logic Reasoning for Semantic Web Ontologies. *WIMS'11 Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, Article 6, 2011.
[21] Berardi D., Calvanese D., de Giacomo G. Reasoning on UML Class Diagrams. *Artificial Intelligence*, vol. 168, no 1–2, 2005, pp. 70–118.
[22] The FaCT System, April 2003. http://www.cs.man.ac.uk/~horrocks/FaCT
[23] Haarslev V., Möller R. RACER system description. *LNAI 2083, Automated Reasoning*, 2001, pp. 701–705.
[24] van der Straeten R., Mens T., Simmonds J., Jonckers V. Using Description Logic to Maintain Consistency between UML Models. *LNCS 2863, The Unified Modeling Language. Modeling Languages and Applications*, 2003, pp. 326–340.
[25] Loom Project Home Page, July 2007. http://www.isi.edu/isd/LOOM

[26] PVS Specification and Verification System, January 2013. http://pvs.csl.sri.com
[27] Kyas M., Fecher H., de Boer F.S., Jacob J., Hooman J., van der Zwaag M., Arons T., Kugler H. Formalizing UML Models and OCL Constraints in PVS. *Electronic Notes in Theoretical Computer Science*, vol. 115, 2005, pp. 39–47.
[28] Brucker A.D., Wolff B. The HOL-OCL book. *Technical Report 525, ETH Zurich*, 2006.
[29] Nipkow T., Paulson L.C., Wenzel M. Isabelle/HOL — A Proof Assistant for Higher-Order Logic. *LNCS 2283*, 2002.
[30] Tsang E. Foundations of constraint satisfaction. Academic Press Limited, London & San-Diego, 1993.
[31] Cabot J., Clariso R., Riera D. UMLtoCSP: a tool for the formal verification of UML/OCL models using constraint programming. *Proceedings of the 22nd ACM/IEEE International Conference on Automated Software Engineering (ASE '07)*, 2007, pp. 547–548.
[32] Cabot J., Clariso R., Riera D. Verification of UML/OCL Class Diagrams using Constraint Programming. *Proceedings of the IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08)*, 2008, pp. 73–80.
[33] Apt K.R., Wallace M.G. Constraint Logic Programming using ECLiPSe. *Cambridge University Press*, 2007.
[34] Gogolla M., Bohling J., Richters M. Validating UML and OCL models in USE by automatic snapshot generation. *Software & System Modeling*, vol. 4, no. 4, 2005, pp. 386–398.
[35] Shaikh A., Wiil U.K., Memon N. Evaluation of Tools and Slicing Techniques for Efficient Verification of UML/OCL Class Diagrams. *Advances in Software Engineering*, 2011, Article ID 370198, 18 p.p.
[36] Breu R., Hinkel U., Hofmann C., Klein C., Paech B., Rumpe B., Thurner V. Towards a Formalization of the Unified Modeling Language. *LNCS 1241, ECOOP'97 — Object-Oriented Programming*, 1997, pp. 344–366.
[37] Richters M., Gogolla M. On Formalizing the UML Object Constraint Language OCL. *LNCS 1507, Conceptual Modeling — ER'98*, 1998, pp. 449–464.
[38] Semenov V.A., Morozov S.V., Tarlapan O.A. Inkremental'naya verifikatsiya ob'ektno-orientirovannyh dannyh na osnove spetsifikatsii ogranichenij [Incremental verification of object-oriented data based on specification of constraints]. *Trudy ISP RAN* [*The Proceedings of ISP RAS*], vol. 8, no. 2, 2004, pp. 21–52 (in Russian).
[39] Semenov V.A., Eroshkin S.G., Karaulov A.A., Enkovich I.V. Semanticheskaya rekonsilyatsiya prikladnyh dannyh na osnove modelej [Model-based semantic reconciliation of applied data]. *Trudy ISP RAN* [*The Proceedings of ISP RAS*], vol. 13, no. 2, 2007, pp. 141–164 (in Russian).
[40] Vanderbei R.J. Linear Programming. Foundations and Extensions. Third edition. *Princeton University*, 2008.
[41] Shevchenko V.N., Zolotyh N.Yu. Linejnoe i tselochislennoe linejnoe programmirovanie [Linear and integer linear programming]. *Nizhniy Novgorod, N.I. Lobachevskiy State University Publ.*, 2004 (in Russian).
[42] Karmanov V.G. Matematicheskoe programmirovanie: uchebnoe posobie. 5-e izdanie [Mathematical programming: a tutorial. Fifth edition]. *Moscow, Fizmatlit Publ.*, 2004 (in Russian).