ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2007. Т. 1. Вып. 2. С. 15–38

КОНСТРУИРОВАНИЕ КАНОНИЧЕСКИХ ИНФОРМАЦИОННЫХ МОДЕЛЕЙ ДЛЯ ИНТЕГРИРОВАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ*

В. Н. Захаров[1], Л. А. Калиниченко[2], И. А. Соколов[3], С. А. Ступников[4]

Аннотация: Рассматривается проблема унификации неоднородных моделей представления информации (баз данных, онтологий, сервисов, процессов) при проектировании распределенных информационных систем с разнородными информационными ресурсами. Особое внимание уделено верифицируемым методам отображения информационных моделей и синтеза расширяемых канонических информационных моделей. Предложена архитектура Унификатора информационных моделей и дан пример отображения конкретной информационной модели в каноническую. Дано сравнение предложенных методов с известными подходами в мировой практике.

Ключевые слова: информационная модель; семантика информационной модели; отображение моделей; коммутативность отображения модели; каноническая модель; синтез канонической модели; расширение канонической модели; принцип уточнения; верификация уточнения моделей; унификатор информационных моделей; метакомпиляция

1 Введение

Статья посвящена изучению, трансформации и унификации моделей представления информации в процессе проектирования и разработки интегрированных, интероперабельных информационных систем. Это важная проблема информатики как науки, изучающей естественные и искусственные информационные процессы. С нарастающей интенсивностью информатику рассматривают как естественную науку [1] (наряду с другими естественными науками), а не как науку, изучающую лишь сущности, искусственно созданные человеком. Информационные процессы являются частью природных явлений: примеры таковых общеизвестны в молекулярной биологии, квантовой физике, в экономике, в социальных процессах и пр. Информатика, таким образом, изучает как естественные, так и искусственные информационные процессы. Понимаемая широко (как computing [2]), информатика включает вычислительную науку (computer science), компьютерную инженерию и инженерию программ, информационные технологии (ИТ), информационную науку (information science), инженерию информационных систем. Трактовка информатики как естественной науки приводит к необходимости ее описания в терминах фундаментальных принципов, которые вскрывали бы глубинные структуры этой науки и показывали, как их можно применить в других областях. К основным категориям таких принципов (важных в контексте настоящей статьи) относятся [2]:

- ***категория вычислений***, рассматриваемых как последовательности трансформаций представлений информации в разнообразных информационных моделях;
- ***категория координации*** (например, кооперации сетевых агентов), рассматриваемой как интероперабельность (совместная работа) совокупности агентов в рамках конечных или бесконечных деятельностей (координируемых протоколами) для достижения некоторой общей цели. Задачи (работы), возникающие при координации, могут быть делегированы вычислительным процессам. При координации агенты обмениваются информацией, представленной в разнообразных информационных моделях;
- ***категория запоминания*** (хранение и извлечение информации), реализуемого системами хране-

*Работа выполнена при финансовой поддержке РФФИ (проект 06-07-08072-офи-а) и программы ОИТВС РАН «Фундаментальные основы информационных технологий и систем» (проект 1-10).

[1]Институт проблем информатики Российской академии наук, vzakharov@ipiran.ru
[2]Институт проблем информатики Российской академии наук, leonidk@synth.ipi.ac.ru
[3]Институт проблем информатики Российской академии наук, isokolov@ipiran.ru
[4]Институт проблем информатики Российской академии наук, ssa@ipi.ac.ru

ния информации, представляемой в той или иной информационной модели.

Собственно, эти категории являются основополагающими для описания информационных процессов, а модели представления информации (или информационные модели) составляют их базис. Настоящая работа выполнена при поддержке проекта РФФИ 06-07-08072-офи-а (проекты конкурсов РФФИ *офи* (ориентированные фундаментальные исследования) имеют целью дальнейшее продвижение тех ранее поддержанных в научных организациях фундаментальных исследований, в ходе работы над которыми исследователи обнаружили возможность использования результатов при создании новых технологий, материалов и услуг). Поэтому она скорее является прагматической, нежели теоретической или тем более философской. Вышесказанное позволяет, однако, определить место данных исследований и разработок в контексте информатики как науки об информационных процессах.

Известно (разработано) большое число моделей представления информации как природных, так и искусственно созданных. Природные информационные модели для изучения преобразуются в искусственные модели, реализуемые на компьютере. Поэтому в дальнейшем статья будет сосредоточена на искусственных моделях (соответствие естественных моделей искусственным можно проследить на примере биологических моделей [4, 3], астрономических моделей [6, 5] и др.; существенно, что одной природной модели, как правило, соответствует значительное число искусственных моделей, с той или иной степенью подробности отражающих ее семантику). Следует оговориться, что под информационными моделями понимаются языки для описания структуры информации, ее семантики, а также операций, определяющих характерные преобразования информации в такой модели. Модель ДНК служит примером подобного языка в молекулярной биологии. Он позволяет описывать общие генетические модели, представления конкретных структур (генов) и содержит специальные операции трансформации таких структур. Отображения модели ДНК в разнообразные искусственные компьютерные информационные модели активно изучались, например, в [4].

В различных областях науки (как и в различных областях деятельности людей вообще) наблюдается экспоненциальный рост объема накапливаемых экспериментальных (наблюдательных) данных, оформляемых в виде информационных ресурсов, каждый из которых представлен средствами определенной информационной модели. Настоящий период развития ИТ характеризуется взрывоподобным процессом создания информационных моделей. Это развитие происходит как в рамках конкретных инфраструктур (таких как архитектуры OMG, в частности архитектура CORBA, архитектуры, движимые моделями представления информации (MDA), архитектуры семантического Веба, сервис-ориентированные архитектуры, архитектуры электронных библиотек, архитектуры информационных грид систем), так и в стандартах конкретных информационных моделей — моделей данных (таких как, например, ODMG 2000, SQL 2003, UML, стеки XML и RDF моделей данных), моделей потоков работ (например, Staffware, COSA, InConcert, Eastman, FLOWer, Domino, Meteor, Mobile, MQSeries, Forte, Verve, Vis. WF, Changeng, IFlow, SAP/R3), языков процессной композиции сервисов (XPDL, BPEL4WS, BPML, XLANG, WSFL, WSCI), семантических моделей (включая онтологические модели и модели метаданных), моделей цифровых репозиториев данных и знаний в конкретных областях бизнеса, торговли, науки и многих других. Основу названных моделей составляют разнообразные понятия и парадигмы, не совместимые между собой. Этот процесс сопровождается другой тенденцией — накоплением использующих подобные модели информационных ресурсов, число которых экспоненциально растет. Такой рост вызывает все увеличивающуюся потребность совместного использования (интеграции) модельно неоднородных информационных компонентов и сервисов в различных применениях, а также их повторного использования и композиции для реализации интероперабельных информационных систем [7]. Указанные тенденции противоречивы: чем больше разнообразие применяемых моделей в различных ресурсах, тем более сложными становятся проблемы их интеграции и композиции. Эти тенденции не новы, но с течением времени разнообразие моделей (величина «Вавилонской башни» моделей) и их сложность растут вместе с ростом потребности достижения интеграции и композиции разномодельных компонентов и сервисов при решении задач. Предел возможности интеграции и композиции ресурсов близок к достижению: уже сейчас зачастую проще создать новый ресурс, чем найти и правильно применить существующий.

Масштабы конструирования интероперабельных систем или систем интеграции разнородных информационных ресурсов (далее для краткости И-систем) для решения задач в разнообразных областях государственного управления, производства, бизнеса, науки и культуры огромны и продолжают быстро расти. То, что и интероперабельные системы, и системы интеграции разнородных ресурсов объединены здесь под общим названи-

ем И-системы, не случайно. Интероперабельность означает совместную работу ресурсов при решении конкретной задачи. Для этого нужно, чтобы композиция ресурсов, выполняющих совместную работу, семантически реализовывала части именно этой задачи [8]. При интеграции неоднородных ресурсов нужно уметь семантически отождествлять объекты, представленные в различных информационных моделях, и семантически правильно отображать схемы интегрируемых ресурсов в глобальную схему. Поскольку в общем случае ресурсы неоднородны (представлены в различных моделях), при создании как интероперабельных систем, так и систем интеграции неоднородных ресурсов для однородного представления их семантики требуется приведение различных информационных моделей к унифицированному виду в рамках некоторой унифицирующей информационной модели, которая называется ***канонической***.

Вместе с тем применяемые методологические и технологические приемы создания И-систем не обладают необходимыми даже минимальными средствами унифицированного описания семантики разнородных компонентов и поэтому малоэффективны. Как уже было показано, проблемы конструирования И-систем порождаются различием моделей (синтаксиса и семантики языков) представления информационных ресурсов, подлежащих интеграции или композиции в составе И-системы. В такие модели входит и язык, на котором выражается спецификация требуемой (будущей) системы. Такие языки перекрывают широкий спектр средств моделирования информации, включая структуры (типы) данных и их семантику, алгоритмы вычислительных функций и методов, спецификации одновременных процессов (потоков работ).

Процессы интеграции или композиции ресурсов при конструировании И-систем оказываются неосуществимыми при попытке манипулирования разномодельными спецификациями компонентов. Единственный практический выход — приведение разномодельных спецификаций к общей, унифицированной модели. Такое семантическое преобразование разномодельных спецификаций к унифицированному виду в канонической модели нуждается в специальных методах и инструментальных средствах, которых нет в практике проектирования И-систем. Приходится полагаться на интуитивные приемы отождествления спецификаций требуемых ресурсов с заданными свойствами и проводить преобразования информационных моделей вручную, без точного учета семантики используемых моделей и подтверждения правильности выполняемых преобразований. Лишь для простейших моделей удается получать удовлетворительные результаты. Такая практика, особенно для объектных и процессных моделей, оборачивается дорогостоящим, длительным и малообоснованным процессом получения унифицирующих спецификаций вручную, либо приводит к практической неосуществимости интеграции (композиции) готовых ресурсов.

Следовательно, для эффективного конструирования И-систем требуются специальные методы и инструменты. Для унификации разнородных спецификаций прежде всего требуется умение сопоставлять спецификации различных ресурсов друг с другом так, чтобы можно было отвечать на вопрос, можно ли при реализации И-системы использовать спецификацию существующего ресурса вместо фрагмента спецификации И-системы. Для этого достаточно доказать, что рассматриваемые спецификации находятся в отношении уточнения. Говорят, что спецификация A уточняет спецификацию D, если A можно использовать вместо D так, что пользователь D не будет замечать этой замены. Средства доказательства факта уточнения спецификации некоторого компонента спецификацией другого компонента (реализуемые на основе теоретико-модельных нотаций и соответствующего инструментария) составляют фундамент предлагаемых методов конструирования унифицирующих (канонических) моделей представления информации в И-системах. Каноническая информационная модель служит в качестве общего языка, эсперанто, для адекватного выражения семантики разнообразных моделей представления информации, используемых в ресурсах И-системы.

Основной принцип синтеза канонической информационной модели для И-системы состоит в расширяемости ее ядра в разнородной среде, включающей различные информационные модели, используемые для представления ресурсов конкретной И-системы. Ядро канонической модели фиксируется. Для каждой конкретной информационной модели M_i среды определяется расширение ядра канонической модели так, чтобы оно вместе с ядром уточнялось бы моделью M_i. Такая уточняющая трансформация моделей должна быть доказуемо правильной. Каноническая модель среды синтезируется как объединение расширений, образованных для моделей M_i среды.

На протяжении длительного периода времени в лаборатории композиционных методов проектирования информационных систем ИПИ РАН разрабатывались методы синтеза канонических моделей для широкого спектра реальных информационных моделей: структурированных, объектных, сервисных, процессных, включая произвольные их комбинации [9–20]. При этом рассматриваются

полные спецификации моделей (языков), включая средства описания как информационных структур (типов данных), так и поведения (операций, функций и процессов) [14, 15, 21].

При преобразовании структурированных, слабоструктурированных и объектных моделей в каноническую разработанные методы сохраняют информацию и операции в соответствии с принципом уточнения [21]. Отображение процессов при синтезе их канонической модели требует сохранения семантики одновременного поведения (concurrency). В 2004 г. авторы, используя недавно обнаруженную возможность интерпретации процессных событий в формальных методах спецификации [22–25], создали метод конструирования доказательных уточнений процессных спецификаций и синтезировали расширяемую каноническую процессную модель для широкого класса процессных моделей (сотни таких моделей используются в мире только в различных коммерческих системах управления потоками работ), языков процессной композиции веб-сервисов [26]. Этот результат завершил многолетние исследования и позволил достичь совмещения двух практически важных требований — полноты охвата канонической моделью семантики разнообразных требующихся на практике моделей представления информации с доказательностью правильности представления в расширяемой канонической модели разнообразных практически используемых моделей. Эти результаты были опубликованы в [15]. Таким образом, решение многих актуальных практических задач стало более обоснованным (например, конструирование виртуальных организаций, трейдинг процессов или сервисов).

Однако, ввиду взрывоподобного расширения разнообразия информационных моделей, при использовании разработанных методов отображения спецификаций ресурсов в спецификации И-систем и синтеза канонических моделей И-систем и применении теории уточнения невозможно справляться вручную с такой «Вавилонской башней» информационных моделей. Поэтому представляется важным создание Конструктора унифицирующих информационных моделей (Унификатора моделей, для краткости) для автоматизации разработанных методов синтеза канонических моделей при проектировании И-систем. Унификатор позволяет доказательно приводить множество разнотипных информационных моделей ресурсов к каноническому, унифицированному представлению и реализует стратегически важный этап конструирования И-систем.

Методы конструирования расширяемой канонической информационной модели для И-систем, функции и архитектура Унификатора моделей и пример отображения конкретной информационной модели в каноническую в процессе ее расширения рассматриваются в последующих разделах статьи.

2 Метод конструирования унифицирующих информационных моделей

В данном разделе рассматривается подход к строгому определению информационных моделей и использованию их как формальных объектов в процессе синтеза канонической информационной модели. Каждая информационная модель определяется синтаксисом и семантикой двух языков — языка определения информации, описывающей состояние некоторой системы (информационного ресурса или И-системы), и языка оперирования такими состояниями (информацией, характеризующей состояние). При отображении некоторой информационной модели ресурса (исходной модели) в каноническую информационную модель (целевую) необходимо сохранение информации и операций. Для этого достаточно, чтобы целевая модель уточнялась исходной моделью[1]. Установление факта уточнения моделей требует проведения формального доказательства, реализация которого возможна в формальной информационной модели, которая в дальнейшем называется ***абстрактной информационной метамоделью***.

Понятие ***уточнения моделей данных*** определяется следующим образом. Тип данных в исходной модели t_s уточняет тип целевой модели t_t, если и только если они отображаются в две спецификации ***абстрактной метамодели данных*** так, что образ типа t_s в исходной модели является уточнением образа типа t_t в целевой модели.

Схема системы S_s уточняет схему S_t, если и только если для каждого типа t_s в S_s есть тип t_t в S_t (и S_t не содержит других типов) такой, что t_s является уточнением t_t. Модель данных M_s уточняет

[1]Исходная информационная модель M_s эквивалентна целевой информационной модели M_t, если M_t уточняется M_s и M_s уточняется M_t. Для целей создания И-систем достаточно рассматривать уточнение модели M_t моделью M_s. Установление факта эквивалентности моделей при использовании предлагаемого подхода возможно, но является избыточным. Вообще говоря, употребление эквивалентности моделей может определяться видом абстрактной метамодели и техникой построения отображения моделей. Так, в работе [11] при построении отображений моделей в денотационной семантике подтверждалась эквивалентность моделей при их отображении.

модель данных M_t, если и только если для каждой допустимой схемы S_s в M_s существует допустимая схема S_t в M_t такая, что S_s является уточнением S_t.

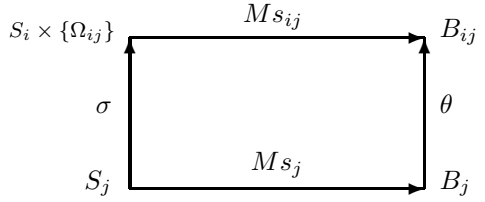
Основные принципы синтеза канонической информационной модели формулируются следующим образом.

Принцип расширения информационных моделей данных. Каноническая модель должна быть расширяемой. Расширение канонической модели происходит при рассмотрении каждой новой исходной модели: целевая модель расширяется путем добавления к ее языку определения информации новых (измененных) типов данных, которые могут быть уточнены определениями типов исходной модели.

Принцип коммутативного отображения информационных моделей. При отображении исходной модели в каноническую необходимо сохранение информации и операций. Это требование достигается, если отображение моделей данных является коммутативным.

Множество всех схем систем, которые могут быть выражены на языке определения информации модели M_i, обозначается S_i. Множество спецификаций абстрактной метамодели, прообразами которых при семантическом отображении являются схемы из S_i, обозначается B_i. Обозначим $Ms_i : S_i \to B_i$ семантическую функцию модели M_i. Отображение $f = \langle \sigma, \theta \rangle$ модели M_j в расширение M_{ij} модели M_i коммутативно, если выполняются следующие условия:

- диаграмма отображения схем языков определения информации является коммутативной:



- отображение θ является уточнением.

Здесь Ω_{ij} обозначает множество новых (модифицированных) типов расширения целевой модели, обеспечивающих конструирование необходимых уточнений типами исходной модели.

Аналогичную диаграмму следует дать для отображения средств языка манипулирования информацией. Однако, поскольку рассматриваемая в работе каноническая модель является объектной, основные операции манипулирования информацией определены в составе типов и классов языка определения. Это позволяет в данном рассмотрении ограничиться только диаграммой отображения схем.

Принцип синтеза унифицирующей канонической информационной модели. Синтез канонической модели есть процесс построения расширений ядра канонической модели, уточняемых различными информационными моделями ресурсов, включаемыми в среду И-системы, а также процесс слияния этих расширений с канонической моделью. В создаваемой согласно этому принципу унифицирующей канонической модели разнообразные исходные модели имеют однородное представление, уточняемое такими исходными моделями.

2.1 Абстрактная информационная метамодель

В настоящей работе в качестве абстрактной информационной метамодели используется Нотация абстрактных машин (Abstract Machine Notation, AMN). Язык AMN обеспечивает манипулирование теоретико-множественными спецификациями в логике первого порядка и доказательство уточнения спецификаций [27–29]. Специальные инструментальные средства (B-технология [27]) предоставляют возможность доказательства коммутативности диаграмм отображения моделей полуавтоматическим способом: теоремы, требуемые для доказательства уточнения моделей, генерируются средствами B-технологии автоматически, но их доказательство может быть интерактивным.

Язык AMN как теоретико-модельная нотация позволяет рассматривать состояния и операции системы интегрированно как спецификацию пространства состояний и поведения (определенного операциями на состояниях) абстрактных машин. Спецификация состояния абстрактной машины вводится переменными состояния вместе с инвариантами — ограничениями, которые должны всегда удовлетворяться. Операции определяются на основе расширения формализма охраняемых команд Дейкстры. Ключевым понятием AMN является уточнение, позволяющее соотносить спецификации систем различных уровней абстракции. Уточняющая спецификация может быть значительно более детальной, чем уточняемая спецификация. Конструируется уточняющая спецификация на основе алгоритмического уточнения и уточнения данных [28]. Уточнение формализуется в AMN путем формулировки ряда теорем специального вида, так называемых ***proof obligations***. Такие теоремы

формулируются автоматически при помощи инструментальных средств поддержки B-технологии (B-Toolkit [30], AntelierB [31]) на основании ***склеивающих инвариантов***, соотносящих состояния уточняемой и уточняющей систем. Теоремы могут быть доказаны при помощи инструментальных средств поддержки автоматического и(или) интерактивного доказательства.

2.2 Спецификация операций в AMN

Операции абстрактных машин основаны на обобщенных подстановках. Любая операция в AMN имеет следующий вид

$$r_1, \ldots, r_n \leftarrow op(p_1, \ldots, p_m) = S\,.$$

Здесь op — имя операции, $r_1, \ldots, r_n$ — выходные параметры операции, $p_1, \ldots, p_m$ — входные параметры операции, S — подстановка, определяющая действие операции на пространстве состояний.

Язык обобщенных подстановок (Generalized Substitution Language, GSL) позволяет описывать переходы между состояниями системы. Каждая обобщенная подстановка S определяет преобразователь предиката, связывающий некоторое постусловие R со своим ***слабейшим*** предусловием $[S]R$, что гарантирует сохранение R после выполнения операции. В этом случае говорят, что S ***устанавливает*** R. «Слабейшее» предусловие означает, что предикат «начального состояния», связанный с некоторым предикатом «заключительного состояния», должен разрешать максимально большое число состояний. В табл. 1 рассматриваются основные виды обобщенных подстановок и соответствующие им слабейшие предусловия. Здесь S, T, T_1, T_2 означают подстановки; x, y, t — переменные; E, F — выражения; G, G_1, G_2, P — предикаты; $P\{x \to E\}$ — предикат P, в котором все свободные вхождения переменной x заменены на E.

Таблица 1 Обобщенные подстановки и их семантика

Обобщенная подстановка S	$[S]P$
$x := E$	$P\{x \to E\}$
$skip$	P
$x := E \Vert y := F$	$[x, y := E, F]P$
$S[]T$	$[S]P \wedge [T]P$
SELECT G_1 THEN T_1 WHEN G_2 THEN T_2 END	$(G_1 \Rightarrow [T_1]P) \wedge$ $(G_2 \Rightarrow [T_2]P)$
PRE G THEN T END	$G \wedge [T]P$
ANY t WHERE G THEN T END	$\forall t \bullet (G \Rightarrow [T]P)$
$S\,;T$	$[S][T]P$
IF G THEN S ELSE T END	$(G \Rightarrow [S]P) \wedge$ $(\neg G \Rightarrow [T]P)$

2.3 Виды конструкций и структурные механизмы AMN

В AMN существует три вида конструкций:

(1) *абстрактная машина (abstract machine)*;

(2) *уточнение (refinement)*;

(3) *реализация (implementation)*.

Абстрактная машина может быть только уточняемой конструкцией, и при описании операций абстрактной машины не разрешается использовать последовательную и циклическую подстановки. Реализация может быть только уточняемой конструкцией, при описании операций реализации не разрешается использовать недетерминированные подстановки (SELECT, ANY), параллельную подстановку и предусловие. Реализации также не разрешается иметь собственных переменных. Уточнение является более универсальной конструкцией, так как может использоваться как в качестве уточняемой, так и в качестве уточняющей конструкции, язык описания операций уточнения не имеет таких ограничений как в абстрактной машине, так и в реализации. Поэтому конструкция уточнения является наиболее предпочтительной для однородного представления в AMN спецификаций канонической модели.

Уточнение выглядит следующим образом.

REFINEMENT r
REFINES m
SEES sm
INCLUDES im
SETS s
CONSTANTS c
PROPERTIES $P(s, c)$
VARIABLES x
INVARIANT $I(x)$
INITIALISATION S
OPERATIONS $O_1; \ldots; O_n$
END

Уточнение с именем r содержит переменные в разделе VARIABLES, которые и определяют состояние системы. Начальная инициализация переменных определяется подстановкой, определенной в разделе INITIALISATION. Изменять состояние системы могут только операции, определенные в разделе OPERATIONS. Состояние системы должно удовлетворять инварианту после инициализации; операции также должны сохранять инвариант. Раздел SETS содержит определение множеств, раздел CONSTANTS содержит имена констант, использующихся в уточнении, раздел PROPERTIES содержит предикат, описывающий свойства констант.

Имя конструкции, которую уточняет r, содержится в разделе REFINES.

Разделы SEES и INCLUDES отвечают за композицию уточнения с другими конструкциями. Композиция SEES используется в случае, если нескольким конструкциям необходимо обеспечить возможность чтения значений переменных, множеств и констант некоторой машины. Композиция INCLUDES используется в ситуации, когда конструкции необходимо использовать другую конструкцию как свою подсистему. Переменные включаемой конструкции становятся переменными включающей конструкции; их изменение может производиться только при помощи операций включаемой конструкции. Инвариант включаемой конструкции становится частью инварианта включающей конструкции.

2.4 Формализация понятия уточнения в AMN

Рассмотрим, каким образом формализуется факт уточнения конструкции M конструкцией N в AMN. Заметим, что факт уточнения может быть установлен только в том случае, если конструкции M и N (табл. 2) ***согласованы***, т. е. удовлетворяют следующим требованиям.

- раздел REFINES конструкции N должен содержать имя M;
- для каждой операции конструкции M конструкция N должна содержать операцию с точно такой же сигнатурой;
- инвариант конструкции N должен содержать так называемый ***склеивающий*** инвариант (инвариант уточнения) R, задающий соотношение между состояниями уточняемой и уточняющей конструкций.

Таблица 2 Спецификации M и N

REFINEMENT M	REFINEMENT N
REFINES K	REFINES M
CONSTANTS c_M	CONSTANTS c_N
PROPERTIES P_M	PROPERTIES P_N
VARIABLES v	VARIABLES w
INVARIANT I_M	INVARIANT I_N
INITIALISATION $Init_M$	INITIALISATION $Init_N$
OPERATIONS	OPERATIONS
$y \leftarrow op(x) =$	$y \leftarrow op(x) =$
PRE $Pre_{op,M}$	PRE $Pre_{op,N}$
THEN	THEN
$Def_{op,M}$	$Def_{op,N}$
END	END
...	...
END	END

Определение 1. N **уточняет** M*, если верны следующие теоремы (proof obligations):*

- *Теорема* **непустоты объединенного состояния**. *Существует объединенное состояние M и N, удовлетворяющее инвариантам M и N:*

$$P_M \wedge P_N \Rightarrow \exists(v, w) \bullet (I_M \wedge I_N).$$

- *Теорема* **уточнения инициализации**. *Инициализация N уточняет инициализацию M:*

$$P_M \wedge P_N \Rightarrow [Init_N]\neg[Init_M]\neg I_N.$$

- *Теорема* **уточнения операций**. *Каждая из операций N уточняет соответствующую операцию M, т. е. при условии выполнения инварианта уточнения и предусловия уточняемой операции, выполняется предусловие уточняющей операции; и для каждого случая исполнения $Def_{op,N}$ существует исполнение $Def_{op,M}$ из соответствующего начального состояния (задаваемого инвариантом уточнения R), которое устанавливает точно такие же значения выходных параметров и сохраняет инвариант уточнения на постсостояниях:*

$$P_M \wedge P_N \wedge I_M \wedge I_N \wedge Pre_{op,M} \Rightarrow \\ Pre_{op,N} \wedge \\ [Def_{op,N}\{y \rightarrow y'\}]\neg[Def_{op,M}]\neg(I_N \wedge y' = y).$$

В целом, для верификации отображения информационной модели M_j в расширение M_i требуется определить AMN-семантику M_j и AMN-семантику расширенной M_i. После этого применяется B-технология, чтобы доказать коммутативность отображения моделей. Это ведет к доказательству того, что M_j является уточнением расширения M_i. Важно, что отображение информационных моделей, основанное на AMN и на технике уточнения, применимо как к структурированным, объектным, онтологическим, так и сервисным и процессным моделям.

3 Архитектура средств конструирования унифицирующих информационных моделей

Целью Унификатора моделей является унификация множества информационных моделей (называемых ***исходными***), совместно использующихся в некоторой И-системе. Унификация исходной

модели R есть приведение ее к канонической информационной модели C, т. е. создание такого расширения E канонической модели (которое может быть и пустым) и такого отображения M исходной модели в расширенную каноническую, что исходная модель ***уточняет*** расширенную каноническую модель. Уточнение моделей означает, что для любой допустимой спецификации r модели R ее образ $M(r)$ при отображении M уточняется спецификацией r. В результате унификации также должна быть получена возможность доказывать уточнение произвольной конкретной спецификацией r модели R ее образа $M(r)$. Верификация уточнения моделей осуществляется на наборе образцов спецификаций исходной модели.

Таким образом, для обеспечения деятельности по унификации информационных моделей необходимы следующие основные языки и формализмы:

- ядро канонической информационной модели;
- формализм, позволяющий описывать синтаксис информационных моделей и специфицировать трансляторы из одной модели в другую;
- формализм, поддерживающий верификацию уточнения.

В качестве ядра канонической информационной модели в настоящей работе рассматривается язык СИНТЕЗ [32], ориентированный на семантическую интероперабельность и композиционное проектирование информационных систем в широком диапазоне существующих неоднородных информационных компонентов.

Для формального описания синтаксиса и трансляторов моделей используются языки метакомпиляции SDF (Syntax Definition Formalism) и ASF (Algebraic Specification Formalism), обеспеченные инструментальной поддержкой Meta-Environment [33].

Для формализации семантики информационных моделей и верификации уточнения используется язык спецификаций AMN, основанный на логике предикатов первого порядка и теории множеств и поддержанный технологией и инструментальными средствами доказательства уточнения (B-технология [27]).

Деятельность по унификации исходной модели R, осуществляемая экспертом при поддержке Унификатора моделей, разбивается на следующие этапы (синтаксис и семантика ядра и существующих расширений канонической модели, а также онтологические аннотации конструкций канонической модели предполагаются определенными):

1. Формализация синтаксиса и семантики модели R.
2. Интеграция эталонных схем модели R и канонической информационной модели.
3. Создание необходимого расширения E канонической модели C.
4. Построение отображения модели R в расширенную каноническую модель.
5. Верификация уточнения моделью R расширенной канонической модели.

Следует отметить, что этап верификации уточнения моделей является достаточно трудоемким и технически сложным и потому может применяться при унификации моделей по требованию.

Рассмотрим более подробно указанные этапы.

1. ***Формализация синтаксиса и семантики исходной модели***. Входными данными этого этапа являются синтаксис и семантика модели R, изложенные в некоторых нормативных документах (описаниях соответствующих языков). Синтаксис моделей чаще всего представляется в каком-либо варианте формы Бэкуса–Наура. Семантика модели может быть как вербальной, представленной в виде комментария к синтаксису, так и частично формализованной (например, с использованием логики и теории множеств).

   (a) Определение синтаксиса трансформируется экспертом в язык SDF, и это новое определение считается формальным описанием синтаксиса.

   (b) Семантика модели формализуется экспертом в виде определяемого на языке ASF транслятора модели в язык AMN. Такая формализация необходима для последующей верификации уточнения моделей и может быть опущена, если верификация не является обязательной.

2. ***Интеграция эталонных схем исходной и канонической моделей***.

   Эталонной схемой информационной модели называется абстрактное описание, содержащее значимые связи между понятиями, соответствующими конструкциям модели. Понятия и связи, составляющие эталонную схему, могут быть аннотированы вербальными определениями.

   Целью интеграции эталонных схем является выделение релевантных конструкций исходной и канонической моделей. Входными данными этапа являются SDF-синтаксис исходной модели и ее вербальная семантика.

   (a) Автоматически выполняется построение заготовки эталонной схемы исходной модели

на основе ее формального синтаксиса. ***Заготовка эталонной схемы*** представляет собой совокупность наименований понятий, соответствующих конструкциям модели, и связей между ними.

(b) Заготовка эталонной схемы превращается экспертом в собственно ***эталонную схему*** модели. Эксперт аннотирует составляющие эталонной схемы вербальными определениями, взятыми из документов, описывающих вербальную семантику модели. Эксперт может уточнить автоматически порожденную заготовку, добавив или удалив новые элементы или связи между ними.

(c) На основе вербальных определений автоматически порождаются векторы дескрипторов элементов эталонной схемы, с использованием векторной модели производится поиск элементов эталонной схемы канонической модели, близких к элементам эталонной схемы исходной модели (эталонная схема канонической модели при этом предполагается построенной). Эксперт подтверждает либо отвергает найденные соответствия, а также добавляет соответствия, не найденные автоматически. Результатом интеграции является список соответствий конструкций исходной и канонической моделей. Одной конструкции канонической модели может соответствовать несколько конструкций исходной модели (и наоборот).

3. ***Создание расширения канонической модели***. К необходимости создания расширения эксперт приходит в процессе интеграции эталонных схем исходной и канонической моделей: обнаруживается, что средств ядра (или ядра с уже построенными расширениями) канонической модели недостаточно для выражения некоторых конструкций исходной модели. К этому выводу эксперт приходит, если:

   – между конструкциями исходной модели и конструкциями канонической модели алгоритмы интеграции не установили связей;

   – на основе анализа документов, описывающих синтаксис и семантику исходной и канонической модели сделано экспертное заключение о невыразимости конструкций исходной модели в канонической.

   При создании расширения для канонической модели предполагаются определенными:

   – документы, описывающие синтаксис и семантику;

   – формальный SDF-синтаксис;

   – AMN-семантика (т. е. вербальные правила и инструментальные средства отображения в язык AMN);

   – онтологическое описание.

   Для исходной модели необходимо обеспечить:

   – документы, описывающие синтаксис и семантику;

   – формальный SDF-синтаксис;

   – AMN-семантику;

   – эталонную схему модели;

   – установленные автоматически или с помощью эксперта и подтвержденные экспертом соответствия элементов исходной и канонической моделей;

   – неформальное определение отображения тех конструкций исходной модели в конструкции канонической, для которых установлено соответствие.

   Для создания расширения эксперту могут понадобиться все вышеперечисленные данные:

   (a) Создание расширения эксперт начинает с определения названия расширения и расширяемой версии канонической модели (т. е. выбирает расширяемую версию из списка).

   (b) Дальнейшая деятельность эксперта заключается в итеративном редактировании синтаксиса расширения и вербальной семантики (включая правила отображения AMN). При этом эксперт обращается к описаниям исходной и канонической моделей, к интерфейсу интеграции эталонных схем моделей. Одновременно эксперт осуществляет дополнение вербального описания отображения исходной модели в каноническую для недостающих конструкций.

   (c) После того как вербальное отображение исходной модели в каноническую полностью определено и зафиксированы абстрактный синтаксис расширения и вербальная семантика, для расширения создается эталонная схема (этапы 2a, 2b).

   (d) Далее уточняется картина интеграции эталонных схем исходной модели и расширенной канонической модели: устанавливаются и фиксируются соответствия между конструкциями исходной модели и конструкциями расширения.

4. ***Построение отображения исходной модели в расширенную каноническую модель***. Входными данными этапа являются список соответствий конструкций исходной и канонической моделей, SDF-синтаксис и вербальная семантика исходной и канонической моделей.

   (a) Эксперт разрабатывает неформальные правила отображения исходной модели в расширенную каноническую.

   (b) Автоматически генерируется ***заготовка ASF-транслятора*** исходной модели в расширенную каноническую. Заготовка строится на основе информации, содержащейся в списке соответствий элементов исходной и канонической моделей и SDF-синтаксисах моделей.

   (c) Эксперт формализует вербальные правила отображения, превращая автоматически порожденную заготовку транслятора в полноценный ASF-транслятор исходной модели в каноническую.

5. ***Верификация уточнения исходной моделью расширенной канонической модели***. Верификация уточнения осуществляется на наборе образцов спецификаций исходной модели. Желательно, чтобы образец являлся достаточно характерной спецификацией исходной модели (типовой конструкцией). Примерами могут служить образцы потоков работ (workflow patterns) [34], использованные для синтеза канонической процессной модели [15]. В случае, когда для модели затруднительно выделить образцы, верификация может быть проведена для используемых при конструировании И-системы конкретных схем в исходной модели.

   Идея верификации уточнения образцом S исходной модели его образа CS при отображении в каноническую модель состоит в следующем: образец S и его образ CS отображаются в язык AMN, и уточнение доказывается средствами B-технологии для соответствующих AMN-спецификаций. Уточнение AMN-спецификаций будет свидетельствовать об уточнении образцом S его образа CS.

   (a) Экспертом выделяются образцы исходной модели, строятся их спецификации S_i и проверяются на соответствие формальному синтаксису.

   (b) С использованием транслятора исходной модели в AMN, полученного на этапе 1b, автоматически порождаются AMN-спецификации S_i^{AMN} образцов.

   (c) С использованием полученного на этапе 4c транслятора исходной модели в каноническую автоматически порождаются канонические спецификации CS_i образцов.

   (d) С использованием транслятора канонической модели в AMN [18], автоматически порождаются канонические AMN-спецификации CS_i^{AMN} образцов.

   (e) Экспертом с использованием средств B-технологии автоматически и/или интерактивно доказывается уточнение спецификациями S_i^{AMN} спецификаций CS_i^{AMN}. Текст доказательств фиксируется.

Рассмотрим архитектуру Унификатора моделей (рис. 1) и то, каким образом компоненты архитектуры поддерживают различные этапы деятельности эксперта по унификации моделей. На рисунке пунктирные стрелки обозначают доступ из одного компонента к интерфейсу другого компонента.

Унификатор состоит из следующих крупных компонентов (групп компонентов):

- Meta-Environment (для определения и проверки корректности синтаксиса моделей и трансляторов моделей);
- B-Toolkit (поддерживающий язык AMN и средства доказательства уточнения спецификаций);
- репозиторий метаинформации;
- менеджер моделей.

Meta-Environment и B-Toolkit представляют собой самостоятельные продукты. Meta-Environment поддерживает следующие этапы унификации моделей: формализация синтаксиса и семантики исходной модели (этап 1), формализация синтаксиса расширения канонической модели (этап 3b), формализация вербальных правил отображения исходной модели в каноническую и построение транслятора моделей на основе автоматически сгенерированной заготовки (этап 4c), проверка образцов исходной модели на соответствие формальному синтаксису (этап 5a). B-Toolkit поддерживает этап 5e автоматического и интерактивного доказательства уточнения спецификаций.

Репозиторий метаинформации представляет собой объектно-реляционную базу данных и предназначен для реализации ***реестра моделей*** и хранения спецификаций. Реестр моделей (или просто ***реестр***) содержит ***регистрационные карты*** моделей, расширений канонической модели, образцов. В регистрационных картах сохраняется вся информация, порождаемая в ходе унификации моделей (в том числе и информация, порождаемая в ходе взаимодействия эксперта с компонентами Meta-Environment и B-Toolkit).

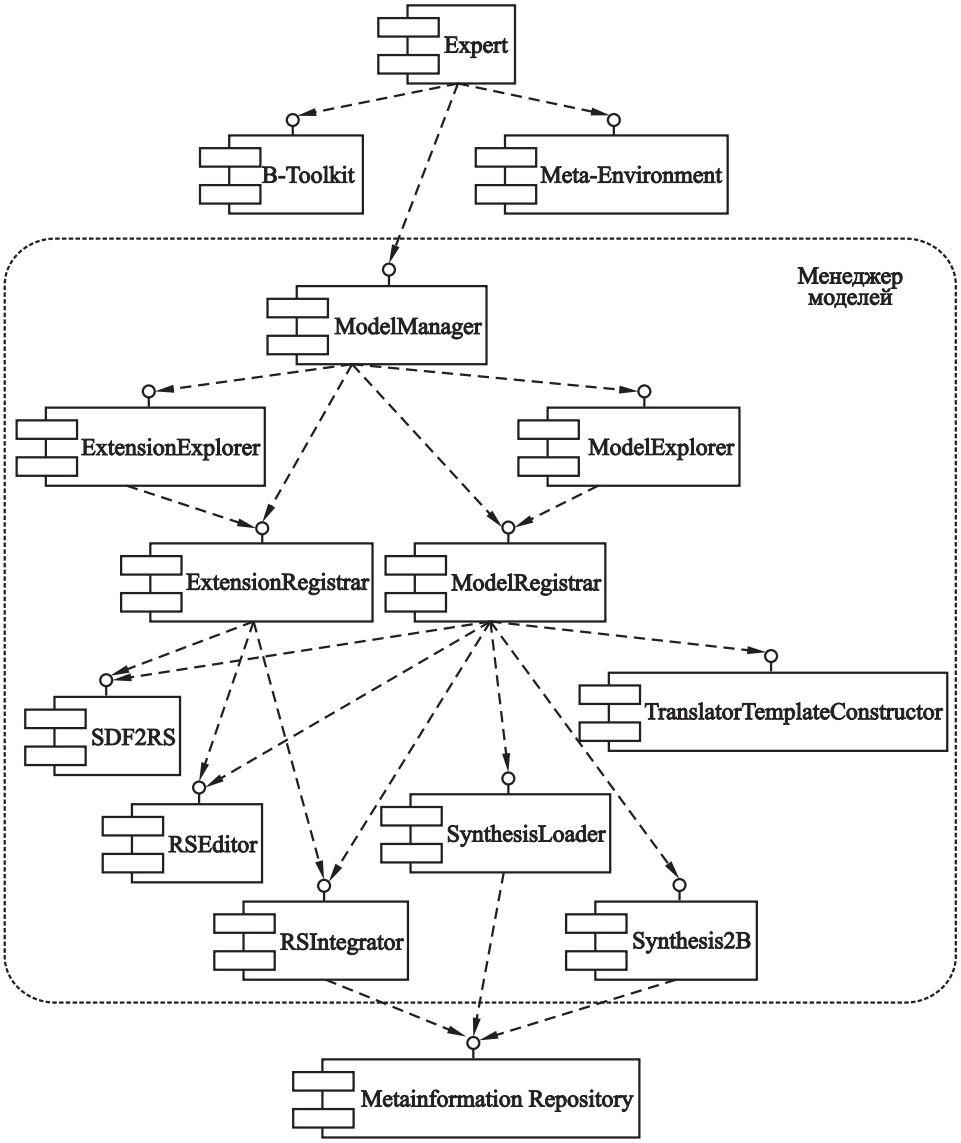


Рис. 1 Архитектура Унификатора моделей

В качестве примера приведем список данных, составляющих регистрационную карту информационной модели:

- название модели (краткое, полное);
- синтаксис модели:
  - название документа, описывающего синтаксис (стандарт, заявка и т. д.);
  - дата документа;
  - ссылка на документ (URI);
  - формальный SDF-синтаксис модели;
- семантика модели:
  - название документа, описывающего семантику;
  - дата документа;
  - ссылка на документ (URL);
  - вербальные правила отображения модели в AMN (текстовый файл);
  - спецификация ASF-транслятора модели в AMN;
- спецификация заготовки эталонной схемы модели;
- спецификация эталонной схемы модели;
- список регистрационных карт примеров модели;
- ссылка на расширение канонической модели, с которым интегрируется модель;

– соответствие элементов исходной и канонической моделей;
– вербальные правила отображения модели в каноническую;
– спецификация ASF-транслятора модели в каноническую.

Все остальные компоненты Унификатора объединены в группу, называемую ***менеджер моделей***.

Компонент ***ModelManager*** предоставляет графический интерфейс, позволяющий эксперту подключиться к конкретному репозиторию метаинформации, а также вызвать компоненты, которые обеспечивают:

– поиск информационной модели в реестре;
– регистрацию новой модели в реестре;
– просмотр расширений, зарегистрированных в реестре;
– регистрацию нового расширения канонической модели в реестре.

Компонент ***ModelExplorer*** предоставляет графический интерфейс, позволяющий производить поиск модели в реестре по полному или краткому названию модели, названиям или ссылкам на документы, описывающие синтаксис или семантику модели.

Компонент ***ExtensionExplorer*** предоставляет графический интерфейс, позволяющий просматривать названия и вербальные описания зарегистрированных расширений канонической модели.

Компонент ***ModelRegistrar*** предоставляет графический интерфейс, позволяющий просматривать и редактировать регистрационные карты моделей и их образцов, содержащихся в реестре.

Компонент ***ExtensionRegistrar*** предоставляет графический интерфейс, позволяющий просматривать и редактировать регистрационные карты расширений канонической модели, содержащихся в реестре.

Компонент ***SDF2RS*** (SDF to Reference Schema) обеспечивает автоматическое построение заготовки эталонной схемы исходной модели (этап 2a) или расширения (этап 3c) на основе их формального синтаксиса.

Компонент ***RSEditor*** (Reference Schema Editor) предоставляет графический интерфейс, позволяющий превратить заготовку эталонной схемы модели (этап 2b) или расширения (этап 3c) в собственно эталонную схему.

Компонент ***RSIntegrator*** (Reference Schema Integrator) обеспечивает поддержку интеграции эталонных схем исходной модели и канонической модели (этап 2c) или расширения канонической модели (этап 3d).

Компонент ***SynthesisLoader*** обеспечивает помещение эталонных схем моделей (этапы 2b, 3c) и спецификаций образцов моделей в репозиторий метаинформации (этап 5c).

Компонент ***TranslatorTemplateConstructor*** обеспечивает автоматическую генерацию заготовки ASF-транслятора исходной модели в каноническую (этап 4b).

Компонент ***Synthesis2B*** обеспечивает автоматическое отображение спецификаций канонической модели в язык AMN (этап 5d).

4 Пример отображения исходной информационной модели в каноническую

В этом разделе рассматривается пример отображения в каноническую модель фрагмента модели OWL (Web Ontology Language, [35]) — языка семантической разметки для публикации и совместного использования онтологий в Веб.

4.1 Формализация синтаксиса OWL

Синтаксис и семантика OWL описаны в документе [36]. Заметим, что в этом документе описан не XML-синтаксис, а **абстрактный** синтаксис, независимый от XML представления.

Синтаксис представлен в версии расширенной формы Бэкуса–Наура, правила которой имеют следующий вид:

```
axiom ::= ... |
 'ObjectProperty(' propertyID
 ['Deprecated'] { annotation }
 {'super(' propertyID ')'}
 ['inverseOf(' propertyID ')']
 ['Symmetric']
 [ 'Functional' | 'InverseFunctional' |
 'Functional' 'InverseFunctional' |
 'Transitive' ]
 { 'domain(' classID ')' }
 { 'range(' classID ')' } ')'
```

Здесь символ `::=` разделяет голову и тело правила, вертикальная черта (|) означает альтернативу, опциональные части правила заключаются в квадратные скобки [ ], повторяющиеся части заключаются в фигурные скобки { }.

Первым этапом унификации модели является преобразование исходного синтаксиса в формальный SDF-синтаксис:

```
module unifier/owl/OWL-Syntax

sorts
 Axiom ObjectPropertyAxiom ...

context-free syntax
 ObjectPropertyAxiom -> Axiom

 "ObjectProperty" "(" PropertyID
 ("Deprecated")? Annotation*
 SuperProperty* InverseOf?
 ("Symmetric")? PropertyKind?
 ObjectPropertyDomain*
 ObjectPropertyRange* ")"
 -> ObjectPropertyAxiom
```

Как можно видеть, формальный синтаксис OWL образует SDF-модуль ***OWL-Syntax***, включающий раздел сортов (куда попадают все нетерминальные символы синтаксиса) и раздел контекстно-свободного синтаксиса (включающий правила). Разделителем в правилах SDF является символ `->`, голова правила располагается справа от разделителя, тело — слева. Опциональные части правил помечаются знаком вопроса `?`, повторяющиеся части — знаком `*`.

При формализации в синтаксисе могут появиться новые нетерминальные символы, детализирующие синтаксис и помогающие избавиться от недетерминизма при программной обработке. Так, аксиома объектного свойства (***ObjectProperty***) в SDF-синтаксисе OWL была выделена в отдельный сорт, и для нее было образовано отдельное правило.

4.2 Формализация семантики OWL в языке AMN

Формализация семантики заключается в построении ASF-транслятора, отображающего спецификации OWL в спецификации AMN. В данном разделе будут кратко проиллюстрированы принципы отображения OWL в AMN на примере онтологии вин:

```
Class(Wine
 restriction(hasColor cardinality(1))
 restriction(hasFlavor cardinality(1))
 restriction(hasSugar cardinality(1)))
Class(IceWine
 restriction(hasFlavor
  allValuesFrom(oneOf(Strong Moderate))))
Class(DessertWine
Wine
restriction(hasSugar
 allValuesFrom(oneOf(OffDry Sweet))))
EquivalentClasses(IceWine
 intersectionOf(DessertWine
  restriction(hasColor value(White))))

ObjectProperty(hasColor
 domain(Wine) range(WineColor))
ObjectProperty(hasFlavor
 domain(Wine) range(WineFlavor))
ObjectProperty(hasSugar
 domain(Wine) range(WineSugar))

EnumeratedClass(WineColor White Rose Red)
EnumeratedClass(WineFlavor
 Delicate Moderate Strong)
EnumeratedClass(WineSugar
 Sweet OffDry Dry)
```

Основной класс онтологии — *Wine*, подклассами его являются *IceWine* и *DessertWine*. Вино характеризуется цветом (свойство *hasColor*), вкусом (*hasFlavor*) и содержанием сахара *hasSugar*. *IceWine* — это белое десертное вино с сильным или средним вкусом.

Образом при отображении данной онтологии в AMN является конструкция *Wines*:

REFINEMENT $Wines$
SETS Ind;
$WineColor = \{White, Rose, Red\}$;
$WineFlavor = \{Delicate, Moderate, Strong\}$;
$WineSugar = \{Sweet, OffDry, Dry\}$
VARIABLES
$Wine, IceWine, DessertWine,$
$hasColor, hasFlavor, hasSugar$
INVARIANT
$Wine \in POW(Ind) \wedge IceWine \in POW(Ind) \wedge$
$DessertWine \in POW(Ind) \wedge$
$DessertWine \subseteq Wine \wedge$
$hasColor \in Wine \leftrightarrow WineColor \wedge$
$\forall wine.(wine \in Wine \Rightarrow$
$\quad \text{card}\,(hasColor[\{wine\}]) = 1) \wedge$
$hasFlavor \in Wine \leftrightarrow WineFlavor \wedge$
$\forall wine.(wine \in Wine \Rightarrow$
$\quad \text{card}\,(hasFlavor[\{wine\}]) = 1) \wedge$
$hasSugar \in Wine \leftrightarrow WineSugar \wedge$
$\forall wine.(wine \in Wine \Rightarrow$
$\quad \text{card}\,(hasSugar[\{wine\}]) = 1) \wedge$
$\text{ran}\,(IceWine \lhd hasFlavor) =$
$\quad \{Strong, Moderate\} \wedge$
$\text{ran}\,(DessertWine \lhd hasSugar) =$
$\quad \{OffDry, Sweet\} \wedge$
$IceWine = DessertWine \cap$
$\{x | x \in \text{dom}\,(hasColor) \wedge hasColor(x) = White\}$
OPERATIONS
$include\_Wine(ind) = \ldots$
$include\_IceWine(ind) = \ldots$
$include\_DessertWine(ind) = \ldots$

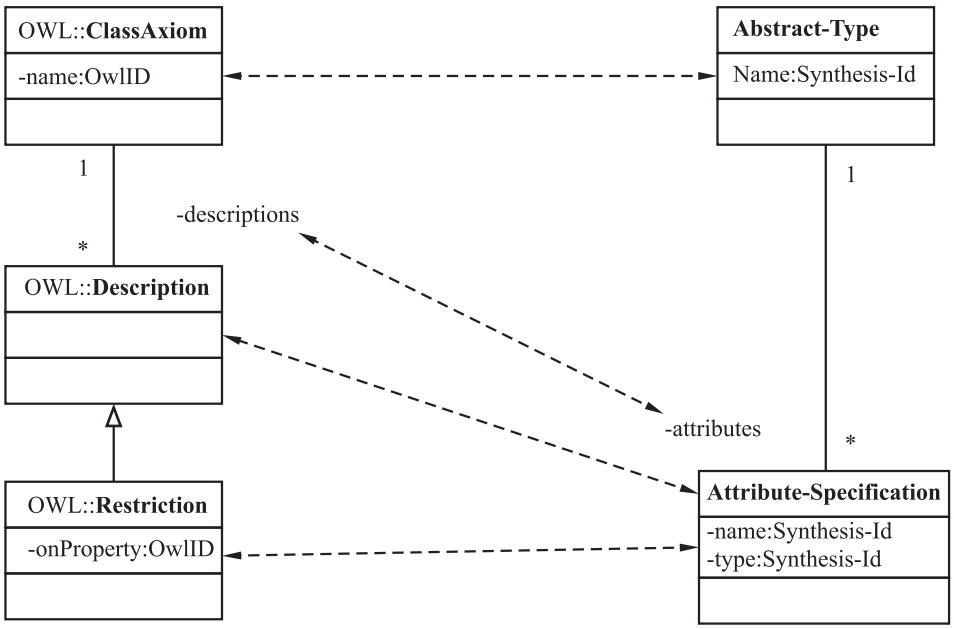


Рис. 2 Интеграция эталонных схем OWL и канонической модели

$set\_hasColor(ind, val) = \ldots$
$set\_hasSugar(ind, val) = \ldots$
$set\_hasFlavor(ind, val) =$
PRE
$ind \in Wine \wedge val \in WineFlavor \wedge$
$ind \in IceWine \Rightarrow val \in \{Strong, Moderate\}$
THEN
$hasFlavor(ind) := val$
END

Множество всех индивидуализированных объектов (individuals) онтологии представляется в AMN множеством Ind, перечислимые классы (например, *WineColor*) представляются отдельными множествами. Классы (например, *Wine*) представляются одноименными переменными, типизированными в инварианте как подмножества Ind.

Объектные свойства (например, *hasColor*) представляются одноименными переменными, типизированными в инварианте как отношения между множествами, представляющими области определения и области значения свойств. Различные ограничения на классы и свойства (ограничения значений и кардинальности свойств, аксиомы эквивалентности классов и т. д.) представляются соответствующими частями инварианта. Каждому классу также ставится в соответствие операция, пополняющая класс новым элементом, а каждому свойству — операция, изменяющая значение этого свойства у некоторого индивидуализированного объекта. В примере рассмотрено тело лишь одной операции, соответствующей свойству *hasFlavor*. Операция изменяет значение свойства *hasFlavor* у индивидуализированного объекта *ind* на значение *val*, но только в том случае, когда выполняется предусловие операции, гарантирующее выполнение инварианта после выполнения операции.

4.3 Интеграция эталонных схем OWL и канонической модели

В данном разделе на небольших примерах будет продемонстрирована интеграция эталонных схем модели OWL и канонической модели (рис. 2), построенных на основе формального SDF-синтаксиса моделей.

На левой стороне рисунка в виде UML-диаграммы изображены типы, атрибуты и связи эталонной схемы модели OWL. Представлены связи двух видов: отношение обобщения — специализации (например, тип *Restriction* является подтипом *Description*) и ассоциации различных кардинальностей (например, с аксиомой класса *ClassAxiom* могут быть связаны описания — *Description*).

На правой стороне рисунка изображены типы эталонной схемы канонической модели.

В эталонных схемах типы, атрибуты и связи онтологически аннотируются экспертом на основании описаний моделей. Аннотации используются при установлении соответствий между конструк-

циями исходной и канонической моделей, происходящем автоматизированным образом с участием эксперта.

Полный список соответствий конструкций OWL и канонической модели (использующийся для построения транслятора OWL в каноническую модель) для рассмотренных на рис. 2 подмножеств моделей приведен в табл. 3. На рисунке соответствия обозначены пунктирными стрелками.

Таблица 3 Соответствия конструкций OWL и канонической модели

Конструкция OWL	Конструкция канонической модели
OwlID	Synthesis-Id
ClassAxiom	Abstract-Type
ClassAxiom.name	Abstract-Type.name
ClassAxiom.descriptions	Abstract-Type.attributes
Description	Attribute-Specification
Restriction	Attribute-Specification
Restriction.onProperty	Attribute-Specification.name

4.4 Создание расширения канонической модели

В процессе интеграции эталонных схем OWL и канонической модели было обнаружено, что атрибуту *kind* типа *ObjectPropertyAxiom*, определяющему вид объектного свойства (транзитивное, функциональное и т. д.), а также понятию *ObjectPropertyKind* не соответствуют никакие конструкции канонической модели.

В канонической модели понятию объектного свойства (*ObjectPropertyAxiom*) соответствует понятие метакласса ассоциаций (*Association-Metaclass*). Было принято решение о расширении канонической модели новыми метатипами ассоциаций — *Transitive*, *Functional* и т. д. В качестве примера рассмотрим спецификацию метатипа *Transitive*:

```
{ Transitive;
  in: association, metatype;
  instance_section: {
    domain: type;
    range: type;
    transitivity: {
      in: predicate, invariant;
      {{ all a/this.domain.inst,
            b/this.domain.inst,
            c/this.domain.inst(
          b/this.range.inst &
          in([a,b], this) &
          in([b,c], this) ->
            in([a,c], this)) }}
} } }
```

Транзитивность ассоциаций выражается инвариантом *transitivity* и состоит в следующем: если a состоит в ассоциации с b и b состоит в ассоциации с c, то a состоит в ассоциации с c (a, b, c — значения некоторого типа данных). Так же выражается и вербальная семантика транзитивной ассоциации.

Семантику транзитивной ассоциации в AMN рассмотрим на следующем примере. Пусть тип *Region* содержит атрибут *subRegionOf*, принадлежащий метатипу ассоциаций *SubRegionOf* (который является подтипом метатипа транзитивных ассоциаций *Transitive*):

```
{ Region; in: type;
  subRegionOf: {set;
    type_of_element: Region;};
}
{ SubRegionOf; in: association, metatype;
  superclass: Transitive;
  instance_section: {
    domain: Region;
    range: Region;
}}
```

Транзитивность ассоциации означает, что если регион A является подрегионом B и регион B является подрегионом C, то A является подрегионом C. При отображении в AMN данная семантика будет выражаться следующим инвариантом:

$$\forall a, b, c\,(a \in ext\_Region \wedge b \in ext\_Region \wedge c \in ext\_Region \wedge a \in subRegionOf(b) \wedge b \in subRegionOf(c) \Rightarrow a \in subRegionOf(c))$$

Здесь *subRegionOf* — переменная, представляющая в AMN соответствующий атрибут, *ext_Region* — множество, представляющее все допустимые экземпляры типа *Region*.

Подобным же образом осуществляется расширение семантики канонической модели для других видов ассоциаций.

Список соответствий конструкций OWL и канонической модели расширяется следующим соответствием:

Конструкция OWL	Конструкция канонической модели
ObjectPropertyAxiom.kind	Association-Metaclass.superclass

Это соответствие означает, что вид объектного свойства OWL выражается в канонической модели отношением суперкласса между метатипом ассоциаций, выражающим объектное свойство (в примере — *SubRegionOf*), и некоторым встроенным метатипом ассоциаций (в примере — *Transitive*).

4.5 Построение отображения OWL в расширенную каноническую модель

Структура ASF-транслятора моделей в общем случае выглядит следующим образом:

```
module <имя модуля>
<список импортируемых модулей>
hiddens
  variables
   <список переменных>
exports
  context-free syntax
   <список сигнатур функций трансляции>
equations
<список правил трансляции>
```

Автоматически генерируются следующие элементы транслятора (тем самым образуя заготовку, подлежащую доработке экспертом):

- имя транслятора — `unifier/owl2synthesis/owl-translator`;
- список импортируемых (используемых) модулей:

```
imports unifier/owl/OWL-Syntax
imports unifier/synthesis/
  Synthesis-Syntax
```

- список переменных:

```
"ObjectPropertyAxiom"[0-9\']* ->
  ObjectPropertyAxiom
"Attribute-Specification"[0-9\']* ->
  Attribute-Specification
"Attribute-Specification*"[0-9\']* ->
  Attribute-Specification*
```

Данные определения означают, что в правилах трансляции могут использоваться переменные соответствующих сортов (например, переменная *Attribute-Specification2* сорта *Attribute-Specification*).

- список сигнатур функций трансляции:

```
t-Module-Def(Ontology) -> Module-Def
t-Type-Specification-List(Directive*)
-> {Type-Specification ","}*
t-Class-Declarator-List(Directive*)
-> {Class-Declarator ","}*
```

В данном примере функция *t-Module-Def* преобразует онтологию OWL в модуль канонической модели и порождена из соответствия элементов *Ontology* и *Module-Def*, функция *t-Type-Specification-List* преобразует список директив в список типов и порождена из соответствия связей *Ontology.directives* и *Module-Def.type*, функция *t-Class-Declarator-List* преобразует список директив в список классов и порождена из соответствия связей *Ontology.directives* и *Module-Def.class_specification*.

- список заготовок правил трансляции:

```
[Module-Def]
Synthesis-Id := t-Module-Name(OwlID),
Type-Specification* :=
t-Type-Specification-List(Directive*),
Class-Declarator* :=
t-Class-Declarator-List( Directive* )
====>
t-Module-Def(Ontology(OwlID Directive*))
=
{ Synthesis-Id; in: module;
  type: Type-Specification*;
  class_specification:
   Class-Declarator*;
}
```

Приведенное правило описывает действие функции *t-Module-Def* и порождено на основе анализа элемента *t-Module-Def*, его атрибутов и ассоциаций (имя модуля соответствует имени онтологии OWL, секции типов и секции классов модуля соответствуют директивы). В правиле использованы рекурсивные вызовы функций преобразования списка директив в списки типов и классов.

Списки импортируемых модулей, переменных, сигнатур функций трансляции, правил трансляции дорабатываются (расширяются, изменяются) экспертом. По ASF-описанию транслятора при помощи средств среды Meta-Environment автоматически генерируется программный код транслятора на языке C. На этапе верификации уточнения моделей 5c код транслятора автоматически компилируется, и полученный C-транслятор используется для порождения канонических спецификаций образцов исходной модели. Вообще, одной из основных целей создания транслятора является его использование для конструирования И-систем: транслятор применяется для отображения схемы источника, описанной в исходной модели, в каноническую модель.

4.6 Верификация уточнения моделью OWL расширенной канонической модели

В данном подразделе рассматривается верификация отображения OWL в каноническую модель на примере схемы OWL, а именно — онтологии вин, рассмотренной в подразделе 4.2. Для данной онтологии уже рассмотрены спецификация на

OWL и семантика в AMN. Осталось рассмотреть отображение онтологии в каноническую модель, отображение канонической спецификации в AMN и верификацию уточнения AMN-спецификаций.

Каноническая спецификация, полученная применением транслятора OWL в каноническую модель (рассмотренного в предыдущем подразделе) к винной онтологии, выглядит следующим образом:

```
{ Wines; in: module, ontology;
type:
{ Wine; in: type, owl;
  hasColor: WineColor;
   metaslot
    in: HasColor;
    min_card: 1;
    max_card: 1
   end
  ...
},
{ DessertWine; in: type, owl;
  supertype: Wine;
  restriction_hasSugar: {
    in: predicate, invariant;
    {{ all w/DessertWine (
        in(w.hasSugar,
           {OffDry, Sweet})) }}
  };
},
{ IceWine; in: type, owl;
  restriction_hasFlavor: {
    in: predicate, invariant;
    {{ all w/IceWine (
        in(w.hasFlavor,
           {Strong, Moderate})) }}
  };
},
{ HasColor;
  in: association, metatype, owl;
  instance_section: {
    domain: Wine;
    range: WineColor;
} },
...
{ WineColor;
  { enum; enum_list: {White, Rose, Red} }
},
 ...
class_specification:
{ wine; in: class, owl;
  instance_section: Wine;
},
{ dessertWine; in: class, owl;
  superclass: wine;
  instance_section: Wine;
},
{ iceWine; in: class, owl;
  instance_section: {
    in: type, owl;
    supertype: DessertWine;
    equivalentClasses: {
      in: predicate, invariant;
      {{ iceWine = intersect(dessertWine,
           { w | in(w, wine) &
              w.hasFlavor = White })
      }}
    };
  };
}; }
```

Перечислимые классы OWL (например, *WineColor*) представляются в канонической модели одноименными перечислимыми типами. Классы (например, *Wine*) представляются одноименными типами (*Wine*) и классами (*wine*) канонической модели. Объектные свойства (например, *hasColor*) представляются атрибутами типов и метатипами ассоциаций (*HasColor*). Ограничения значений свойств представляются инвариантами типов (например, *DessertWine.restriction_hasSugar*). Ограничения, выражаемые отдельными аксиомами (например, аксиомы эквивалентности классов), представляются инвариантами классов канонической модели (например, *iceWine.equivalentClasses*).

Образом при отображении данной онтологии в AMN является конструкция *Wines*:

REFINEMENT $Wines$
SETS $AVAL$,
$WineColor = \{White, Rose, Red\}$,
$WineFlavor = \{Delicate, Moderate, Strong\}$,
$WineSugar = \{Sweet, OffDry, Dry\}$
CONSTANTS
$Obj, ext\_Wine, ext\_DessertWine, ext\_IceWine$
PROPERTIES
$Obj \in POW(AVAL) \wedge$
$ext\_Wine \in POW(Obj) \wedge$
$ext\_DessertWine \in POW(Obj) \wedge$
$ext\_IceWine \in POW(Obj) \wedge$
$ext\_DessertWine \subseteq ext\_Wine$
$\wedge\, ext\_IceWine \subseteq ext\_DessertWine$
VARIABLES
$wine, iceWine, dessertWine,$
$hasColor, hasFlavor, hasSugar$
INVARIANT
$wine \in POW(ext\_Wine) \wedge$
$iceWine \in POW(ext\_IceWine) \wedge$
$dessertWine \in POW(ext\_DessertWine) \wedge$
$dessertWine \subseteq wine \wedge$
$hasColor \in ext\_Wine \rightarrow WineColor \wedge$
$hasFlavor \in ext\_Wine \rightarrow WineFlavor \wedge$
$hasSugar \in ext\_Wine \rightarrow WineSugar \wedge$
$\forall w(w \in ext\_IceWine \Rightarrow$
$hasFlavor(w) \in \{Strong, Moderate\}) \wedge$
$\forall w(w \in ext\_DessertWine \Rightarrow$

Таблица 4 Число теорем

Теоремы	Число теорем	Число автоматически доказанных теорем
Теорема непустоты объединенного состояния	1	0
Теоремы уточнения операции $include\_Wine$	17	5
Теоремы уточнения операции $include\_DessertWine$	20	5
Теоремы уточнения операции $include\_IceWine$	21	5
Теоремы уточнения операции $set\_hasColor$	11	1
Теоремы уточнения операции $set\_hasSugar$	11	1
Теоремы уточнения операции $set\_hasFlavor$	11	1
Общее число теорем	92	18

$hasSugar(w) \in \{OffDry, Sweet\}) \wedge$
$iceWine = dessertWine \cap$
$\{w | w \in wine \wedge hasColor(w) = White\}$
OPERATIONS
$include\_Wine(obj) = \ldots$
$include\_IceWine(obj) = \ldots$
$include\_DessertWine(obj) = \ldots$
$set\_hasColor(obj, val) = \ldots$
$set\_hasSugar(obj, val) = \ldots$
$set\_hasFlavor(obj, val) =$
PRE
$obj \in wine \wedge val \in WineFlavor \wedge$
$obj \in iceWine \Rightarrow val \in \{Strong, Moderate\}$
THEN
$hasFlavor(obj) := val$
END

Множество всех значений абстрактных типов данных представляется в AMN множеством $AVAL$, множество всех значений объектных типов данных — его подмножеством Obj. Перечислимые типы (например, *WineColor*) представляются отдельными множествами. Типы представляются множествами всех своих допустимых значений — экстенсионалами (например, тип $Wine$ представляется экстенсионалом ext_Wine). Экстенсионалы объектных типов типизируются в секции PROPERTIES как подмножества Obj. Отношение тип–подтип представляется отношением множество–подмножество на экстенсионалах.

Классы (например, *wine*) представляются одноименными переменными, типизированными в инварианте как подмножества экстенсионалов типов своих экземпляров. Отношение класс–подкласс представляется отношением множество–подмножество на соответствующих переменных.

Атрибуты типов (например, *hasColor*) представляются одноименными переменными, типизированными в инварианте как функции, определенные на экстенсионале типа.

Различные ограничения на классы и свойства (ограничения значений и кардинальности свойств, аксиомы эквивалентности классов и т. д.) представляются соответствующими частями инварианта.

Каждому классу также ставится в соответствие операция, пополняющая класс новым элементом, а каждому атрибуту — операция, изменяющая значение этого свойства у некоторого объекта. В примере рассмотрено тело лишь одной операции, соответствующей атрибуту *hasFlavor*. Операция изменяет значение свойства *hasFlavor* у объекта *obj* на значение *val*, но только в том случае, когда выполняется предусловие операции, гарантирующее выполнение инварианта после выполнения операции.

Спецификации AMN, отвечающие онтологии вин OWL и ее образу при отображении в каноническую модель, были введены в инструментальное средство автоматизации доказательства уточнения AMN (B-Toolkit 5.1.4). Далее автоматически были сформулированы 92 теоремы, в совокупности утверждающие факт уточнения спецификаций. Большое количество теорем объясняется тем, что сложные теоремы автоматически подразделяются инструментальным средством на более простые, которые можно доказывать независимо. Например, теорема уточнения операции *set_hasFlavor* была разделена на 11 теорем. С использованием автоматических средств доказательства было доказано 18 теорем, остальные были доказаны интерактивно. В табл. 4 указано общее число теорем и число автоматически доказанных теорем.

5 Состояние проблемы

Представляемая работа является новой, она не имеет аналогов в мировой практике. Наиболее близкие исследования активно ведутся в последние десять лет Microsoft Research и рядом университетов Европы и США с целью создания унифицированных средств манипулирования метаданными, главным образом в базах данных (спецификации интерфейсов лишь упоминаются). Эти средства сводятся к отображению схемы баз данных из одной модели

данных в *соответствующую* схему базы данных в другой модели данных [37]. Такие преобразования необходимы при решении ряда практических задач, включая обмен данными, интеграцию данных, формирование хранилищ данных, семантическую обработку запросов к базам данных.

Основным оператором отображения схем является *ModelGen* [38], параметрами которого служат исходная модель данных M_1, целевая модель данных M_2, исходная схема базы данных S_1, выраженная в M_1. Результатом *ModelGen* является целевая схема S_2, выраженная в M_2. Существенно, что *ModelGen* должен быть универсальным оператором, применимым к любым моделям данных в рассматриваемой среде.

Основная идея подхода *ModelGen* заключается в использовании метамодели — набора конструкций, используемых для определения моделей данных, являющихся экземплярами метамодели. Средством определения конструкций этого набора являются унифицированные, не зависящие от моделей данных метаконструкции, классифицированные в [39] Халлом и Кингом в ряд категорий:

- лексем (наборов атомарных значений);
- абстрактов (сущностей, типов и т. д.);
- агрегатов (конструкций, основанных на подмножестве декартова произведения — таблиц, связей и пр.);
- функций (атрибутов, свойств);
- иерархий и пр.

Каждая модель определяется подмножеством таких конструкций и соответствующих им метаконструкций. Отображение схемы из одной модели данных в другую определяется в терминах трансформации метаконструкций. ***Супермоделью*** называется модель данных, содержащая конструкции, соответствующие всем метаконструкциям, известным системе. В системе [40] используется около дюжины таких конструкций. Каждая модель данных является ***специализацией*** супермодели, так что схема в любой модели данных является схемой в супермодели. Отображение схем выполняется как процесс элементарных шагов элиминации конструкций, отсутствующих в целевой модели, и, возможно, введения новых конструкций.

Процесс отображения включает следующие шаги:

1. Трансляцию исходной схемы в супермодель.
2. Трансляцию результата в целевую схему, выполняемую в рамках супермодели.
3. Трансляцию целевой схемы в целевую модель.

При этом шаги 1 и 3 объявляются трудоемкими и прямолинейными, поскольку каждая модель данных (исходная или целевая) ***поглощается*** (is subsumed) супермоделью. Единственная трансформация, согласно [38], кодированная авторами, соответствует шагу 2.

Инструментарий включает набор элементарных правил над метаконструкциями, выраженных на языке LDL (например, элиминация n-арных связей, элиминация атрибутов связей, элиминация связей «многие ко многим», замена связей ссылками, элиминация обобщений). Элиминация некоторой конструкции включает замену ее другой конструкцией. Композиция таких правил позволяет определять сложные трансформации. Унифицированные правила предполагают их специализацию в конкретной ситуации добавлением условий, определяющих, к каким концептам правила являются применимыми. Инструментарий позволяет верифицировать, принадлежат ли генерируемые схемы целевой модели, а также обнаруживать избыточность в последовательности элементарных преобразований.

В дальнейшем планируется введение макрооператоров [41–43] для шага 2:

- *Match* — возвращает отображение одной схемы в другую;
- *Merge* — для двух схем и отображения одной из них в другую возвращает интегрированную схему;
- *Diff* — для схемы и заданного для нее отображения возвращает остаток схемы, не входящий в отображение;
- *Compose* — возвращает композицию двух отображений схем.

Инструментарий поддерживает пользователей трех категорий:

(1) проектировщиков схем в заданных моделях, использующих *ModelGen* для отображения полученных схем;

(2) инженеров моделей, определяющих новые модели, применяя имеющиеся метаконструкции;

(3) инженеров метамодели, добавляющих новые метаконструкции к метамодели и определяющих правила преобразования для них (тем самым расширяется набор моделей, поддерживаемых системой).

Подход *ModelGen* использует четырехуровневый реляционный словарь [44]:

- уровень метасупермодели, определяющий структуру метаконструкций;

– уровень супермодели, на котором хранятся схемы, подлежащие трансляции;

– уровень моделей данных, на котором определяются конструкции всех моделей данных среды, причем каждая конструкция связана с соответствующей ей метаконструкцией;

– уровень схем, на котором расположены все схемы системы.

Основные операции над словарем заключаются в определении новых моделей данных, основанных на имеющихся конструкциях, и схем для уже определенной модели. После определения модели данных инструментарий автоматически создает структуры, необходимые для работы со схемами.

Развитием инструментария *ModelGen* является инструментарий MIDST [40], который дополнительно к отображению схем позволяет реализовать трансформацию на уровне данных (если задана база данных D_1 в схеме S_1, генерируется соответствующая база данных D_2 в схеме S_2).

Отдельно, без связи с инструментарием *ModelGen* или MIDST, рассматривается вопрос трансляции схем, который можно отнести к шагам 1 и 3 выше. Вопросы трансляции схем из исходной модели данных (например, объектной, реляционной, XML-ориентированной) в целевую рассматриваются в работах [45], где упоминается прототип интерактивной генерации реляционных схем из объектно-ориентированных, который встроен в Microsoft Visual Studio 2005. В процессе трансляции схем используются правила элиминации конструкций исходной схемы, которые отсутствуют в целевой модели данных.

Неопределенными остаются термины ***соответствия, специализации, поглощения*** во фразах типа:

– отображение схемы базы данных из одной модели данных в ***соответствующую*** схему базы данных в другой модели данных;

– каждая модель данных является ***специализацией*** супермодели, так что схема в любой модели данных является схемой в супермодели;

– генерируется *соответствующая* база данных D_2 в схеме S_2;

– каждая модель данных (исходная или целевая) ***поглощается*** супермоделью.

Заметна недостаточность теоретических обоснований и необходимого минимума формальных определений.

Одной из попыток формального обоснования рассматриваемых отображений схем можно считать работу [46]. В ней используется категорный подход, даны определения, на основе которых можно было бы попытаться определить точный смысл свойств отображений схем и моделей (таких как соответствие, специализация, поглощение). Однако совершенно не ясно, каким образом можно было бы доказывать выполнимость этих свойств. Более того, авторы признают ограниченность предлагаемых формализмов: «Мы не рассматриваем проблемы отображения одной модели данных (например, реляционной) в другую модель (например, объектно-ориентированную). Представленный формализм применим к должным образом упрощенной XML Schema».

Авторы отмечают, что хотя алгебраическая и теоретико-модельная семантика средств отображения схем и рассматривалась в [46, 47], она остается во многом не исследованной областью.

Интересны в этой связи рассуждения в работе [37]. Необходимо обеспечить, чтобы для каждого экземпляра исходной схемы существовал экземпляр целевой схемы, из которого можно было бы реконструировать экземпляр исходной схемы. Доминирование [47] соответствует отображению экземпляров исходной схемы в целевую, которое должно быть тотальной инъектирующей функцией.

Другими словами, образуется биекция между множеством исходных экземпляров и подмножеством целевых экземпляров данных. Это считается общим требованием правильности отображения исходной модели в целевую (что, конечно же, неточно, например в случае отображения моделей и схем для интеграции баз данных, поскольку дезориентирует пользователя целевой модели — он может ожидать появления экземпляров, которые никогда не появятся из-за семантики исходной модели). Утверждается, что универсальная характеризация отображения моделей данных (или схем баз данных) является сложным делом: например, при трансляции схем исходная и целевая модели могут обладать различными выразительными способностями, так что при интуитивной трансляции не удастся образовать ни доминирующей, ни эквивалентной схемы, что может привести к потере информации, представленной в рамках исходной схемы. Как выход из существующего положения предлагается применять реверсивные отображения, которые могут стать критерием правильности [41].

В работе [48] предлагается использовать расширяемую модель, а специализация и *refinement* могут применяться для расширения модели, в которой метаконструкции организованы в виде иерархии наследования (решетки) предопределенных понятий. Этот подход рассматривается как альтернатива подходу, основанному на метаконструкциях супермодели.

В работах Бернстайна макрооператоры над схемами баз данных (*Match, Merge, Diff, Compose*) рассматриваются чисто синтаксически: схемы и их отображения трактуются как графовые структуры, семантика схем, определяемая соответствующими языками определения данных [41], при этом отбрасывается. Авторы осознают этот недостаток и в будущем предполагают провести дополнительные исследования.

В практическом плане достижением считается разработанная в результате многолетних исследований Алмаден Центра фирмы ИБМ система CLIO [49], которая позволяет осуществлять отображение схем ресурсных баз данных в целевые схемы, выраженные в ограниченном подмножестве структурированных информационных моделей.

Кратко сравнение рассматриваемого в настоящей работе подхода (Синтез) и подхода *ModelGen,* развивавшихся независимо, заключается в следующем.

Выразительные способности супер- и канонической моделей. Цели отображения моделей данных и схем различны: в Синтезе оно применяется, прежде всего, для синтеза канонической информационной модели и интеграции информационных ресурсов. Поэтому в Синтезе рассматриваются разнообразные виды информационных ресурсов, не только базы данных.

Супермодель *ModelGen* можно сравнивать с канонической моделью лишь условно. Каноническая модель Синтеза является рабочим языком спецификации посредников как приложений. Супермодель не является законченным языком и служит в качестве вспомогательного средства отображения схем.

Набор метаконструкций ядра канонической модели (языка Синтез [32]) не ограничивается структурированными моделями данных, подобно *ModelGen*. Он значительно шире и включает абстрактные типы данных (АТД) — объектные и необъектные, фреймы, которые совместно с типизированными значениями приводят к образованию гибридной строго типизированной и слабоструктурированной модели, функции как составляющие абстрактных типов (методы) и автономные функции (специфицируемые своими сигнатурами, пред- и постусловиями), классы как множества объектов определенного типа, отношения подтипа, подкласса, связи между экземплярами АТД, метаклассы, инварианты АТД и классов, выражаемые средствами языка логических формул Синтеза, процессы. Конструкции супермодели легко включаются в состав конструкций канонической модели.

Отличие способов расширения моделей. Расширение канонической модели и расширение супермодели качественно различаются. Расширение канонической модели — это семантический процесс введения в модель новых образцов параметризованных замкнутых логических формул, выражающих в целевой модели зависимости данных, характерные для исходной модели, параметризованных родовых типов данных, представляющих новые типы данных, отсутствующие в ядре канонической модели, метафреймов, аннотирующих дополнительные свойства конструкций ядра в расширенной модели. В *ModelGen* процесс расширения супермодели — это, в основном, механический процесс введения в нее новых метаконструкций.

Сохранение информации при отображении моделей. Модели информационных ресурсов в Синтезе считаются определяемыми соответствующими им языками со всеми деталями этих языков — их синтаксисом и семантикой (для баз данных это два языка — язык определения данных и язык манипулирования данными). *ModelGen* абстрагируется от этого, извлекая из языка определения данных только спецификации структур данных. Исключается детальное рассмотрение семантики языков, ограничений целостности баз данных, функций. Это неизбежно приводит к потере информации при отображении моделей данных (например, при отображении объектных моделей в реляционные).

Метод Синтеза основан на формальном определении семантики схем в исходной и целевой модели для доказательства факта уточнения результата отображения схемы в целевую модель схемой исходной модели. Отношение уточнения является строго определенным, и его использование в методе для доказательства коммутативности диаграмм отображения спецификаций исходной модели данных в целевую позволяет сохранить информацию и операции и исключить использование плохо определенных понятий (соответствия, специализации, поглощения схем).

Различие архитектур. Инфраструктура Унификатора моделей разработана с учетом указанных отличий Синтеза и *ModelGen*. Процесс отображения из исходной модели данных в ядро канонической модели включает определение необходимого расширения ядра на основе сопоставления конструкций исходной и целевой моделей, доказательство правильности отображения и построение транслятора из исходной модели в целевую средствами метакомпиляции. Таким образом, получается готовый продукт преобразования схем. Сопоставление

конструкций исходной и целевой моделей и расширение ядра являются интерактивным творческим процессом, в котором помощь эксперту оказывают онтологические спецификации, аннотирующие конструкции исходной и целевой моделей, позволяющие устанавливать отношения близости конструкций.

Метод Синтеза позволяет отображать произвольные модели данных, в частности, для поддержки обмена данными. Это можно выполнять непосредственно из модели в модель или через посредство канонической модели. Целесообразность осуществления отображения моделей в рамках канонической модели отображением в нее исходной и целевой моделей (подобно отображению моделей и схем в супермодель) требует специального анализа.

Помимо универсальных средств отображения моделей данных и схем баз данных широко известны попытки разработать общие языки информационных моделей, спецификаций. Достаточно упомянуть UML, различные архитектуры разработки открытых систем (например, The Open Applications Group Integration Specification (OAGIS), модели архитектур программного обеспечения, определяющие компоненты программ, их свойства и связи между ними (Acme, Aesop, SADL)), выявление и описание типовых конструкций разнородных процессных моделей (проект, выполненный недавно в Голландии группой Ван дер Аальста), архитектуры открытых систем OMG, движимые моделями (MDA). Недостатком указанных попыток является стремление создать общий язык как единое целое, безотносительно к другим языкам спецификаций ресурсов.

Для известных работ характерны отсутствие (недостаточность) точных спецификаций семантики языков, отсутствие верификации отображений информационных моделей (например, в MDA), отсутствие понятия синтеза *расширяемой* унифицирующей (канонической) модели. Разработанные авторами методы отличаются от известных тем, что во главу угла ставят расширяемую каноническую модель, конструируемую модульно путем расширения ее ядра, имеют точно определенные формальные основания, широко применяют принцип уточнения при унифицирующих преобразованиях моделей.

6 Заключение

Рост объема накапливаемой информации (информационных ресурсов) в различных областях деятельности человека сопровождается взрывоподобным процессом создания разнообразных информационных моделей (языков) представления информации и манипулирования ею. Чем больше разнообразие применяемых моделей в различных ресурсах, тем более сложными становятся процессы их интеграции и композиции при создании информационных систем. Сложность семантики информационных моделей делает такие процессы интеграции неосуществимыми при попытке манипулирования разномодельными спецификациями ресурсов. Единственным практическим выходом является приведение разномодельных спецификаций к общей, унифицированной модели, называемой канонической.

На протяжении длительного периода времени в лаборатории композиционных методов проектирования информационных систем ИПИ РАН разрабатывались методы синтеза канонических моделей для широкого спектра реальных информационных моделей: структурированных, объектных, сервисных, процессных, включая произвольные их комбинации. При этом рассматривались полные спецификации моделей (языков), включая средства описания как информационных структур (типов данных), так и поведения (операций, функций и процессов). При отображении структурированных, слабоструктурированных, объектных, процессных моделей в каноническую модель разработанные методы сохраняют информацию и операции в соответствии с принципом уточнения. В работе показано, как строить такие семантические отображения формально с целью верификации правильности достижения уточнения расширенной канонической модели исходной информационной моделью. В результате достигнута полнота охвата канонической моделью семантики разнообразных требующихся на практике видов моделей представления информации с возможностью доказательства правильности представления в расширяемой канонической модели неоднородных практически используемых моделей.

Разнообразные методы, на которые опирается данная работа, разрабатывались авторами на протяжении нескольких десятилетий. Эти методы более подробно рассмотрены в других работах авторов.

Вместе с тем в реальных системах ввиду большого разнообразия используемых информационных моделей применение разработанных методов вручную становится неэффективным. В работе предложена архитектура и функции компонентов Унификатора моделей, позволяющего доказательно приводить множество разнотипных информационных моделей ресурсов к каноническому представлению, опираясь на разработанные ранее методы. На

примере отображения фрагментов онтологического языка OWL, применяемого в Веб, продемонстрирован процесс доказательного отображения этой информационной модели в каноническую, ядро которой составляет гибридный объектно-фреймовый язык Синтез. Унификатор моделей находится в состоянии реализации.

Литература

1. *Denning P.* Computing is a Natural Science // Communications of the ACM, 2007. Vol. 50, № 7.
2. *Denning P.* Great principles of computing // Communications of the ACM, 2003. Vol. 46. № 11.
3. *Kroger P.* Molecular biology data: Database overview, modelling issues, and perspectives. — Munich: Institute for Informatics, Munich University. 2001.
4. *Bry F., Kroger P.* A computational biology database digest: Data, data analysis, and data management // J. of Distributed and Parallel Databases, 2003. Vol. 13. № 1. P. 7–42.
5. Data model for observation, Version 0.23. // IVOA DM WG Internal Draft, 2004.
6. *Cambresy L., Derriere S., Padovani P., Martinez A. P., and Richard A.* Ontology of astronomical object types, Version 1.0. // IVOA Technical Note http://www.ivoa.net/ Documents/ cover/ AstrObjectOntology-20061031.html, 2006.
7. *Kalinichenko L. A., Briukhov D. O., Martynov D. O., Skvortsov N. A., Stupnikov S. A.* Mediation framework for enterprise information system infrastructures // 9th Conference (International) on Enterprise Information Systems (ICEIS), 2007.
8. *Калиниченко Л. А.* Методология организации решения задач над множественными распределенными неоднородными источниками информации // Международная конференция «Современные информационные технологии и ИТ-образование». — М.: МГУ, 2005. С. 20–37.
9. *Kalinichenko L. A.* Data model transformation method based on axiomatic data model extension // 4th Conference (International) on Very Large Data Bases, 1978.
10. *Калиниченко Л. А.* Методы и средства интеграции неоднородных баз данных. — М.: Наука, 1983.
11. *Kalinichenko L. A.* Methods and tools for equivalent data model mapping construction // Proc. EDBT'90 Conference. Springer-Verlag, 1990. P. 92–119.
12. *Kalinichenko L. A., Skvortsov N. A.* Extensible ontological modeling framework for subject mediation // 4th Russian Scientific Conference "DIGITAL LIBRARIES: Advanced Methods and Technologies, Digital Collections." — Dubna, 2002.
13. *Kalinichenko L. A.* Canonical model development techniques aimed at semantic interoperability in the heterogeneous world of information modeling // Knowledge and model driven information systems engineering for networked organizations: Proc. of the CAiSE INTEROP Workshop. — Riga: Riga Technical University, 2004. P. 101–116.
14. *Kalinichenko L. A., Stupnikov S. A., Zemtsov N. A.* Extensible canonical process model synthesis applying formal interpretation // East-European Conference ADBIS'05. Springer, 2005.
15. *Калиниченко Л. А., Ступников С. А., Земцов Н. А.* Синтез канонических моделей для интеграции неоднородных источников информации. — М.: ИПИ РАН, 2005. 87 с.
16. *Калиниченко Л. А.* Синтез канонических моделей, предназначенных для достижения семантической интероперабельности неоднородных источников информации // Системы и средства информатики: Спец. вып. «Формальные методы и модели в композиционных инфраструктурах распределенных информационных систем». — М.: ИПИ РАН, 2005.
17. *Ступников С. А.* Формальная семантика ядра канонической объектной информационной модели // Системы и средства информатики: Спец. вып. «Формальные методы и модели в композиционных инфраструктурах распределенных информационных систем». — М.: ИПИ РАН, 2005.
18. *Ступников С. А.* Отображение спецификаций, выраженных средствами ядра канонической модели, в Нотацию Абстрактных Машин // Системы и средства информатики: Спец. вып. «Формальные методы и модели в композиционных инфраструктурах распределенных информационных систем». — М.: ИПИ РАН, 2005.
19. *Ступников С. А.* Автоматизация верификации уточнения при композиционном проектировании информационных систем и посредников // Системы и средства информатики: Спец. вып. «Формальные методы и модели в композиционных инфраструктурах распределенных информационных систем». — М.: ИПИ РАН, 2005.
20. *Ступников С. А., Брюхов Д. О.* Представление UML и OCL в канонической информационной модели // Системы и средства информатики: Спец. вып. «Формальные методы и модели в композиционных инфраструктурах распределенных информационных систем». — М.: ИПИ РАН, 2005.
21. *Kalinichenko L. A.* Method for data models integration in the common paradigm // Advances in Databases and Information Systems: Proc. of the 1st East-European Conference. — St. Petersburg: Nevsky Dialekt, 1997. P. 275–284.
22. *Butler M.* csp2B: A practical approach to combining CSP and B // Formal Aspects of Computing, 2000. Vol. 12.
23. *Treharne H., Schneider S.* How to Drive a B machine // Formal Specification and Development in Z and B: First International Conference of Z and B Users, 2000.
24. *Stupnikov S. A., Kalinichenko L. A., Dong J. S.* Applying CSP-like workflow process specifications for their refinement in AMN by pre-existing workflows // Advances in

Databases and Information Systems: Proc. of the 6th East-European Conference. — Bratislava: Slovak University of Technology, 2002. P. 206–215.

25. *Abrial J.-R.* B#: Toward a synthesis between Z and B // ZB'2003 — Formal Specification and Development in Z and B: International Conference of B and Z Users, 2003. P. 168–177.
26. *Stupnikov S. A., Kalinichenko L. A., Bressan S.* Interactive discovery and composition of complex Web services // East-European Conference on ADBIS'06. Springer, 2006.
27. *Abrial J.-R.* B-Technology. Technical overview. B-Core (UK) Ltd., 1993.
28. *Abrial J.-R.* The B-Book: Assigning programs to meanings. — Cambridge: Cambridge University Press, 1996.
29. *Cansell D., Mery D.* Foudations of the B Method // Computing and Informatics, 2003. Vol. 22, No. 3–4. P. 221–256.
30. The B-Toolkit. http://www.b-core.com/ONLINEDOC/BToolkit.html.
31. Atelier B: The industrial tool to efficiently deploy the B method. http://www.atelierb.societe.com/index_uk.html.
32. *Kalinichenko L. A., Stupnikov S. A., Martynov D. O.* SYNTHESIS: A language for canonical information modeling and mediator definition for problem solving in heterogeneous information resource environments. — M.: IPI RAS, 2007. 171 p.
33. *Van den Brand M. G. J., van Deursen A., Heering J., et al.* The ASF + SDF meta-environment: A component-based language development environment // Compiler Construction 2001 / Ed. by R. Wilhelm. Springer, 2001. P. 365–370.
34. *Van der Aalst W. M. P., ter Hofstede A. H. M., Kiepuszewski B., Barros A. P.* Workflow patterns // Distributed and Parallel Databases, 2003. Vol. 14. № 3. P. 5–51.
35. OWL Web ontology language reference. W3C Recommendation. http://www.w3.org/TR/owl-ref/, 2004.
36. *Patel-Schneider P. F., Hayes P., Horrocks I.* OWL Web ontology language semantics and abstract syntax // W3C Recommendation. http://www.w3.org/TR/owl-semantics/, 2004.
37. *Atzeni P.* Schema and data translation: A personal perspective // 11th East European Conference ADBIS 2007. Springer, 2007.
38. *Atzeni P., Cappellari P., Bernstein P.* ModelGen: Model independent schema translation // 21st International Conference on Data Engineering, 2005.
39. *Hull R., King R.* Semantic database modeling: Survey, applications and research issues // ACM Computing Surveys, 1987. Vol. 19. № 3.
40. *Atzeni P., Cappellari P., Gianforme G.* MIDST: Model independent schema and data translation // SIGMOD 2007 Conference.
41. *Bernstein P.* Applying model management to classical meta data problems // 2003 CIDR Conference.
42. *Melnik S., Rahm E., Bernstein P.* Rondo: A programming platform for generic model management // SIGMOD 2003 Conference.
43. *Melnik S., Bernstein P., Halevy A., Rahm E.* Supporting executable mappings in model management // SIGMOD 2005 Conference.
44. *Atzeni P., Cappellari P., Bernstein P.* A multilevel dictionary for model management // ER 2005 Conference. Springer-Verlag, 2005.
45. *Bernstein P., Melnik S., Mork P.* Interactive schema translation with instance-level mappings // 31st VLDB Conference, 2005.
46. *Alagic S., Bernstein P.* A model theory for generic schema management // DBPL, 2001.
47. *Miller R., Ioannidis Y., Ramakrishnan R.* Schema equivalence in heterogeneous systems: Bridging theory and practice // Information Systems, 1994. Vol. 19. № 1.
48. *Barsalou T., Gangopadhyay D.* M(dm): An open framework for interoperation of multimodel multidatabase systems // ICDE 1992. — Los Alamitos: IEEE Computer Society Press, 1992.
49. *Haas L. M., Hernandez M. A., Ho H., Popa L., and Roth M.* Clio, 2005. Grows up: From research prototype to industrial tool // Proc. of the ACM SIGMOD Conference, 2005. Baltimore, Maryland, USA.