Формальная семантика языка разрешения сущностей и слияния данных и ее применение для верификации потоков работ интеграции данных

© С. А. Ступников

Институт проблем информатики Федерального исследовательского центра «Информатика и управление» Российской академии наук
Москва
sstupnikov@ipiran.ru

Аннотация

В течение всего периода развития методов и средств интеграции возникали вопросы их верификации, т.е. формальной проверки на соответствие заданным требованиям. Можно выделить следующие уровни интеграции: интеграция моделей данных, сопоставление и интеграция схем данных и собственно интеграция данных. В работе рассматривается подход определению формальной семантики высокоуровневых программ интеграции данных в языке спецификаций, поддержанном средствами формального доказательства. Рассматриваемая семантика применяется для верификации потоков работ интеграции структурированных данных. Свойства программ, подлежащие проверке, представляются в виде выражений выбранного языка спецификаций. Затем, с использованием формальных средств доказательства, спецификация, выражающая семантику конкретного потока работ интеграции данных, проверяется на соответствие необходимым свойствам. Практической целью работы является определение оснований для формальной верификации свойств потоков работ при решении задач в различных средах интеграции данных.
Работа выполнена при поддержке РФФИ (гранты 14-07-00548, 15-29-06045, 16-07-01028).

1 Введение

Разработка методов и средств интеграции данных становится в настоящее время все более актуальной в связи со значительным ростом объемов и разнообразия данных. Различные по семантике и структурированности данные могут быть необходимы при решении одной задачи в науке или промышленности. Увеличение объемов данных требует применения масштабируемых платформ распределенной параллельной обработки данных, таких, как Apache Hadoop [4]. При этом все составляющие процесса интеграции данных должны быть реализованы в виде программ над соответствующей платформой.

Можно выделить следующие уровни интеграции (от более высокого к более низкому): интеграция моделей данных [18, 19], сопоставление и интеграция схем данных [29] и собственно интеграция данных. Завершение интеграции на более высоком уровне обычно является предусловием для интеграции на более низком уровне. Необходимо отметить, что в данной работе рассматривается интеграция *структурированных* данных, т.е. данных, конформных некоторой схеме, определяющей типы (структуры) данных.

На уровне моделей данных происходит сопоставление элементов моделей (языков), на уровне схем – сопоставление типов (структур) данных и их атрибутов, на уровне собственно данных определяется, каким именно образом следует преобразовывать исходные коллекции данных, элементы коллекций и их атрибуты. Идеальным описанием процесса интеграции данных можно считать совокупность декларативных правил (подобная программе на языке Datalog), такой подход называется *обмен данными* (data exchange) [11]. Однако, на практике, процесс интеграции данных представляет собой набор операций трансформации данных, представленных в SQL или подобном ему языке, причем важным является порядок исполнения операций. Обычно интеграция данных организуется в виде потоков работ. Как в случае обмена данными, так и в случае потоков работ, интеграция данных согласована с предыдущим этапом интеграции – сопоставлением схем.

Процессы интеграции данных являются достаточно важными и сложными, и в течение всего периода развития методов и средств интеграции возникали вопросы определения их формальной семантики и *верификации*, т.е. формальной проверки на соответствие заданным требованиям. В ИПИ ФИЦ ИУ РАН данным вопросам в последние годы уделялось существенное внимание. Разрабатывались методы и средства унификации моделей данных с доказательным сохранением семантики [17, 18, 19,

Труды XVIII Международной конференции DAMDID/RCDL'2016 «Аналитика и управление данными в областях с интенсивным использованием данных», Ершово, 11-14 октября 2016

20, 35, 34], методы и средства верификации композиционного проектирования информационных систем [33, 31] (относящиеся к уровню схем). Разрабатываются продвинутые среды интеграции данных и решения задач над неоднородными информационными ресурсами, базирующиеся на Hadoop [21, 22, 36]. Краткий обзор упомянутых и родственных им работ приведен в разделе 2. Настоящая работа нацелена на развитие методов определения формальной семантики и верификации на третьем уровне – уровне данных. При этом предполагается, что необходимое определение семантики и верификация на уровне моделей данных и схем уже проведена. Предполагается также, что процесс интеграции данных представляется в виде набора операций трансформации данных на SQL-подобном языке, а не в виде декларативной Datalog-подобной программы. При этом свойства процесса интеграции в целом не выражаются явно, определяются лишь правила преобразования данных в конкретных операциях.

В процессе интеграции собственно данных обычно выделяют различные этапы: трансформация данных, разрешение сущностей (entity resolution) [23, 13] и слияние данных (data fusion) [6, 7, 10]. *Трансформация данных* подразумевает их преобразование из исходной схемы (схемы коллекции - источника данных) в целевую (единую интегрированную схему). Под *разрешением сущностей* обычно понимают выделение и связывание информации об одной и той же сущности реального мира из разных коллекций данных [23]. Под *слиянием сущностей* понимают комбинацию различных представлений одной и той же сущности реального мира в единое представление [28]. Этапы интеграции данных могут состоять из большого количества операций, и не обязательно идут в указанном порядке. Поэтому более правильно говорить, что процесс интеграции данных представляет собой поток работ, деятельности которого представляют собой отдельные операции трансформации структурированных (типизированных) данных, разрешения или слияния сущностей. Ряд обобщенных операций слияния данных выделен в работе [7].

Наряду с платформами распределенной параллельной обработки данных разрабатываются высокоуровневые языки программирования, которые могут быть использованы (или прямо предназначены) для трансформации данных, разрешения и слияния сущностей в рамках этих платформ. К таким языкам относятся, в частности, Jaql [5], Pig Latin [29], Highlevel Integration Language (HIL) [15]. Распределенное исполнение программ, написанных на таких языках, в среде Hadoop достигается путем компиляции высокоуровневых программ в программы на императивных языках (например, Java), которые, в свою очередь, исполняются с использованием средств поддержки в Hadoop вычислительной модели MapReduce [27].

Идея подхода, предлагаемого в данной статье, состоит в том, чтобы сообщить высокоуровневым программам интеграции данных семантику в некотором языке спецификаций, поддержанном средствами формального автоматического и/или интерактивного доказательства: для языка интеграции данных строится его отображение в язык спецификаций. Свойства программ, подлежащие проверке, представляются в виде выражений выбранного языка спецификаций. Затем, с использованием формальных средств доказательства, спецификация, выражающая семантику конкретного потока работ интеграции данных, проверяется на соответствие необходимым свойствам.

Для иллюстрации подхода в качестве языка интеграции данных выбран HIL – язык, разработанный компанией IBM, поставляемый в составе Hadoop-решения BigInsights [14], и используемый, например, в проектах интеграции финансовых и социальных данных [15]. HIL – это язык высокого уровня для описания сложных потоков обработки данных, включающих операции трансформации данных, разрешения сущностей и слияния данных. Данные при этом могут поступать из больших коллекций данных, представленных в различных схемах. Язык был применен, в частности, в проекте интеграции данных из социальных сетей [15], была продемонстрирована масштабируемость применения языка как по разнообразию схем данных, так и по объему данных. По сравнению с традиционными средствами построения процессов извлечения-трансформации-загрузки (ETL) данных, основанных на реляционной модели и языке SQL, язык HIL предлагает значительно более гибкие декларативные средства интеграции данных, нацеленные на разрешение сущностей и слияние данных.

В качестве языка спецификаций выбран язык Нотация абстрактных машин (AMN) [1], поддержанный средствами формального доказательства [2]. Выбор AMN мотивирован возможностями этого языка по спецификации операций трансформации данных и опытом автора по использованию AMN для определения формальной семантики. Для формализации семантики языка HIL недостаточно лишь средств верификации процессов (таких, как конечные автоматы или сети Петри) – необходимы средства верификации сложных операций (деятельностей, составляющих процессы). Вместо AMN возможно использование и других языков, предоставляющих аналогичные возможности, например, RAISE или Z.

Структура статьи выглядит следующим образом. В разделе 2 рассматриваются родственные работы по определению формальной семантики и верификации процесса интеграции данных на разных уровнях. В разделе 3 излагаются и иллюстрируются на примерах основные принципы семантического отображения языка HIL в язык AMN. В разделе 4 иллюстрируется на примере верификация свойств операций

разрешения сущностей и слияния данных. В заключении подведены итоги статьи и обозначены направления дальнейшей работы.

2 Родственные работы

Работы по определению семантики языков концептуального моделирования и моделей данных известны с конца 1990-ых гг. В работе [16] отображение типа связи модели ODMG'93 в AMN было использовано для верификации его отображения в каноническую модель данных. При отображении моделей данных должны сохраняться информация и семантика операций языка манипулирования данными.

Работы Lano, Bicarregui (например, [25]) посвящены формальному определению языка UML в языке RAL (Real-time Action Logic). Показано, как полученная семантика может использоваться для верификации преобразования UML-диаграмм (усиления инвариантов, рационализации ассоциаций, элиминации ассоциации много-ко-многим, транзитивности агрегации, преобразования интерфейса).

В работе [33] определена семантика объектной модели данных в языке AMN для верификации композиционного проектирования информационных систем: доказательства того, что композиция готовых программных компонентов может быть использована в качестве реализации абстрактной спецификации системы, подлежащей созданию.

В дальнейшем, язык AMN использовался для верификации отображения различных классов моделей данных: процессных [17], онтологических [20], массив-базированных [34], графовых [35].

Были разработаны обобщенные методы и средства унификации моделей (приведения их к единой канонической модели данных) [18, 19]. Верификация отображений моделей при этом также основывалась на представлении семантики моделей в языке AMN.

Много работ известно в области верификации *трансформаций моделей* [9]. Эти работы проводятся в рамках Движимой моделями инженерии (MDE). Под моделями в MDE понимаются как модели данных (языки), так и концептуальные схемы. Трансформация моделей – это реализация их отображения, набор правил, в совокупности определяющих, каким образом сущности исходной модели должны быть преобразованы в сущности целевой модели. Верификации подлежат такие свойства трансформаций, как *завершаемость* (процесс трансформации успешно завершается для любой правильно определенной исходной модели), *определенность* (уникальность целевой модели для заданной исходной модели и трансформации), *синтаксическая корректность* трансформации по отношению к языку трансформаций, *сохранение семантики исполнения* (трансформация исполняется в соответствии с ее спецификацией). Для формального доказательства свойства трансформаций представляются в формальных языках, различных вариантах логики первого порядка (например, Calculus of Inductive Constructions). В зависимости от мощности используемого языка, для доказательства свойств могут быть использованы автоматические средства (SAT-решатели, средства проверки моделей) или автоматизированный логический вывод.

Известны также различные подходы к определению формальной семантики трансформации данных, например, в работе [24] рассматривается формальная семантика диаграмм потоков данных в языке VDM (Vienna Development Method). В работе [32] для представления семантики процессов извлечения-трансформации-загрузки (ETL) данных используется язык LDL (Logic-Based Data Language). Целью этих работ является обеспечение перехода от концептуальной модели процессов трансформации данных к логической. В системе Clio [12] реализован подход обмена данными, когда трансформация данных представлена в виде Datalog-подобной программы с логической семантикой.

Особенность данной работы состоит в том, что формальная семантика в языке AMN сообщается высокоуровневому языку разрешения сущностей и интеграции (HIL), семантика применяется для верификации потоков работ интеграции данных, выраженных на HIL. Основные черты подхода выделены в следующих разделах.

3 Формальная семантика языка разрешения сущностей и интеграции HIL

Подход к определению формальной семантики языка HIL демонстрируется в данном разделе на примере интеграции финансовых данных, публикуемых Комиссией по ценным бумагам и биржам (SEC) США. Пример рассматривался в работах, посвященных системе интеграции Midas [8] и языку HIL [15]. Целью рассматриваемого потока работ интеграции данных (являющегося частью общей задачи интеграции финансовых данных) является формирование коллекции сущностей *Person,* представляющей информацию о лицах, управляющих ведущими компаниями США (рис. 1). Исходными для потока являются две коллекции: *InsiderReportPerson* (для краткости, *IRP*) и *JobChange*. В первой коллекции содержатся данные, извлеченные из внутренних отчетов компаний о заработной плате сотрудников. Во второй коллекции содержатся данные, извлеченные из отчетов о принятии на работу или смене позиций сотрудников компаний.

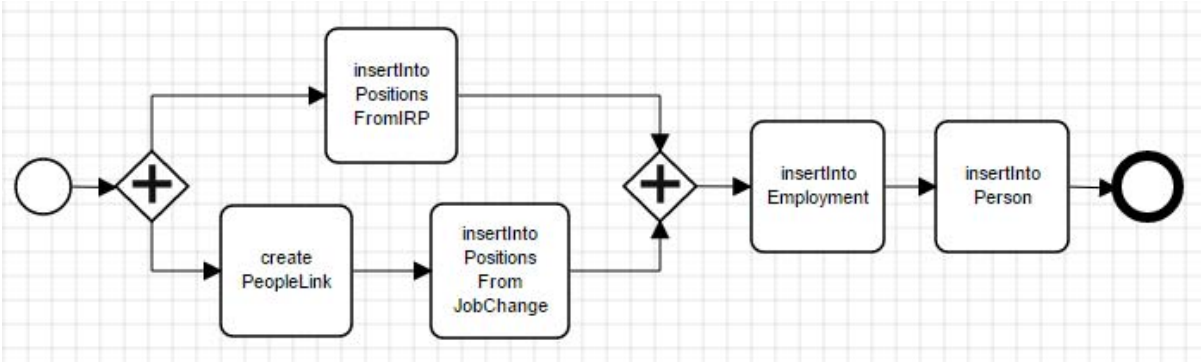

Рисунок 1 Поток работ интеграции данных об управляющих компаниями лицах

Само извлечение информации из неструктурированных данных не является предметом настоящей работы. Поток включает пять операций:

- *insertIntoPositionsFromIRP* (извлечение данных о позициях управляющих лиц из коллекции *IRP* и включение их в промежуточную коллекцию *Positions*);
- *createPeopleLink* (создание коллекции *PeopleLink* связей между управляющими лицами, упомянутыми во внутренних отчетах, и записями о смене позиций);
- *insertIntoPositionsFromJobChange* (извлечение данных о позициях управляющих лиц из коллекции *JobChange* и включение их в промежуточную коллекцию *Positions*);
- *insertIntoEmployment* (включение данных о найме сотрудников компаниями в промежуточную коллекцию *Employment*);
- *insertIntoPerson* (включение данных об управляющих лицах и их местах работы в коллекцию *Person*).

Среди рассматриваемых операций *createPeopleLink* является характерным примером разрешения сущностей, а операции *insertIntoEmployment* и *insertIntoPositionsFromJobChange* – слияния сущностей.

Следует отметить, что в языке HIL отдельным операциям трансформации, разрешения или слияния сущностей не присваиваются имена, операции выражаются в виде правил в SQL-подобном синтаксисе. Приведенные выше имена присвоены соответствующим правилам для удобства рассуждений.

Также нужно отметить, что порядок исполнения операций в языке HIL напрямую не фиксируется [15]. Этот порядок определяется при компиляции с тем ограничением, что все промежуточные коллекции, используемые операциями, должны быть материализованы, т.е. исполнены операции, осуществляющие включение данных в эти коллекции. Поток работ, изображенный на рис. 1 в графическом представлении языка BPMN, отражает неявную семантику ограничений на последовательность исполнения операций в рассматриваемом примере. Ромб со знаком «+» означает разделение потока на параллельные ветви и слияние параллельных ветвей. Например, операция *insertIntoEmployment* должна быть исполнена после операций *insertIntoPositionsFromIRP, createPeopleLink, insertIntoPositionsFromJobChange,* поскольку она использует коллекции *PeopleLink* и *Positions,* формируемые соответствующими операциями.

Семантика основных конструкций языка HIL будет определена ниже в языке AMN, основанном на теории множеств и типизированном логике предикатов первого порядка. Спецификации AMN называются *абстрактными машинами* и сочетают в себе пространства состояний и поведения машины, определенного операциями на состояниях. Ниже перечисляются и иллюстрируются на примерах основные принципы определения семантики HIL.

1. Семантика программ на языке HIL. Каждая программа на HIL представляется в AMN отдельной конструкцией вида REFINEMENT [BAMN] (такого рода конструкции допускают наиболее широкие возможности определения спецификаций). В частности, упомянутый выше пример программы представляется конструкцией *PersonFusion*:

```
REFINEMENT PersonFusion
```

2. Семантика операторов объявления типов сущностей и типов индексов. Типы сущностей и типы индексов представляются в AMN переменными абстрактных машин в разделе VARIABLES, и типизируются в разделе INVARIANT. Например, *тип сущностей* IRP, определяемый в HIL следующим образом (атрибут *name* отвечает имени персоны, *cik* – уникальному идентификатору персоны в документации SEC, *bdate* – дате рождения, *company* – имени компании, *companyCIK* – идентификатору компании, *title* – занимаемой должности, флаги *isOfficer* и *isDirector* – является ли персона служащим или управляющим лицом компании):

```
declare IRP: set [ name: string, cik: int, bdate: string;
  company: string, companyCIK: int, title: string,
  isOfficer: Boolean, isDirector: Boolean ];
```

представляется в конструкции *PersonFusion* следующим образом:

```
VARIABLES IRP, …
INVARIANT
 IRP: POW(struct(
   name: STRING_TYPE, cik: INT,
   bdate: STRING_TYPE, company: STRING_TYPE,
   title: STRING_TYPE, isOfficer: BOOL,
   isDirector: BOOL
   )) & …
```

Для типа *IRP* в разделе переменных объявляется одноименная переменная, которая типизируется в инварианте как подмножество (POW) множества всех структур (struct) – кортежей. Кортежи состоят из атрибутов, соответствующих атрибутам типа *IRP*. Между встроенными типами HIL и AMN установлено взаимооднозначное соответствие, используемое при типизации атрибутов структур в AMN. Например, тип *string* языка HIL соответствует типу STRING_TYPE в AMN, тип Boolean – типу BOOL и т.д.

Тип индекса (представляющего собой отображение из множества экземпляров типа ключа в экземпляры типа значения) *Positions*, определяемый в HIL следующим образом (ключ задается парой ⟨идентификатор персоны *cik,* имя компании *company*⟩, а значение – множеством должностей *set[title]*):

```
declare Positions:
fmap [cik: int, company: string] to set [title: string];
```

также представляется в конструкции *PersonFusion* одноименной переменной, типизируемой в инварианте:

```
VARIABLES … Positions, …
INVARIANT
Positions: struct(cik: INT, company: STRING_TYPE) +->
  POW(struct(title: STRING_TYPE))
```

Для представления отображения используется конструктор частичной функции AMN, обозначаемый символами «+->». Типы ключа и значения, как и в случае с типом сущностей *IRP*, также представляются при помощи конструкторов структур и множества подмножеств.

3. Семантика операций трансформации, разрешения и слияния сущностей. Каждому правилу языка HIL ставится в соответствие операция абстрактной машины AMN. Например, правилу создания коллекции *Empoloyment* в HIL:

```
insert into Employment …
select … from … where … ;
```

соответствует операция *insertIntoEmployment* конструкции *PeopleFusion*:

```
OPERATIONS
  insertIntoEmployment =
  SELECT … THEN … END;
```

3.1. Семантика включения данных в коллекции с типами сущностей. Примером операции такого рода является создание коллекции *PeopleLink* в HIL*:*

```
create PeopleLink as
select [cik: p.cik, docID: j.docID, span: j.span ]
from IRP p, JobChange j
match using
        rule1: normName(p.name) = normName(j.name)
check if not(null(j.bdate)) and not(null(p.bdate))
then j.bdate = p.bdate;
```

Сущности формируемой коллекции определяются в секции *select* и включают атрибуты *cik* (идентификатор персоны)*, docID* (номер документа, где встречается упоминание о смене должности персоны), *span* (место в документе, где встречается упоминание о смене должности). Данные извлекаются (секция *from*) из коллекций *IRP* и *JobChange* (происходит соединение коллекций). В секции *match* происходит предварительный отсев кортежей, полученных в результате соединения коллекций: имена персон *name* должны в обеих коллекциях совпадать с точностью до нормализации (функция *normName*). В секции *check* происходит дополнительная проверка отобранных в *match* кортежей: в случае, если в обеих коллекциях имеются данные о дате рождения персоны, даты должны совпадать. В AMN формирование коллекции *PeopleLink* осуществляется в операции *createPeopleLink:*

```
createPeopleLink =
SELECT …
THEN
PeopleLink :=
{ rr | #(pp, jj).( pp: IRP & jj: JobChange &
        rr = rec(cik: pp'cik, docID: jj'docID, span: jj'span) &
        normName(pp'name) = normName(jj'name) &
        (jj'bdate /= null_string & pp'bdate /= null_string =>
              jj'bdate = pp'bdate)  )
};
…
END
```

Переменной *PeopleLink* присваивается множество, образованное при помощи конструкции выделения множества *{rr | F(rr)}* (здесь *rr* – имя переменной, отвечающей элементу множества, *F(rr)* – формула, которая должны обращаться в истину на элементах множества). Переменная *rr* типизируется типом записи *rec*, обладающим необходимыми для типа коллекции атрибутами. Для каждой из соединяемых коллекций заводится по переменной (*pp* и *jj*), переменные связываются квантором всеобщности # и типизируются как принадлежащие соответствующим коллекциям: *pp: IRP & jj: JobChange*. Условия из секций *match* и *check* представляются соответствующими условиями на переменные *pp* и *jj*. Необходимые функции (например, *normName*) определяются как константы в разделе ABSTRACT_CONSTANTS конструкции *PersonFusion* и типизируются в разделе *PROPERTIES*:

```
ABSTRACT_CONSTANTS normName, …
PROPERTIES
  normName: STRING_TYPE --> STRING_TYPE
```

Функция *normName* типизируется тотальной функцией из строк в строки.

3.2. Семантика включения данных в коллекции с типами индексов. Рассмотрим отличительные особенности представления в AMN операций включения данных в коллекции с типами индексов на примере операции *insertIntoPositionsFromIRP*:

```
insert into Positions ! [id: i.cik, company: i.company ]
select [ title: normTitle(i.title) ]
from  IRP i;
```

Знак «!» после идентификатора пополняемой коллекции *Positions* означает, что пара атрибутов *cik* и *company*, следующих после него, будет использоваться в коллекции в качестве ключа. В AMN такое пополнение коллекции *Positions* осуществляется в операции *insertIntoPositionsFromIRP*:

```
insertIntoPositionsFromIRP =
SELECT …
THEN
   Positions := Positions \/
   { rr | #ii.(ii: IRP &
         rr = rec(cik: ii'cik, company: ii'company) |->
         { rr1 | #ii1.(ii1: IRP & ii1'cik = ii'cik &
               ii1'company = ii'company &
               rr1 = rec(title: normTitle(ii1'title))) }
       )
   };
   …
END;
```

Множество *Positions* пополняется при помощи операции объединения множеств «\/». Пополняющее множество, как и в правиле 3.1, образуется при помощи конструкции выделения множеств по переменной *rr*. Отличие состоит в том, что переменная *rr* типизируется типом пары. Первым элементом пары является запись *rec(cik: ii'cik, company: ii'company),* соответствующая типу ключа коллекции Positions. Вторым элементом пары является множество названий должностей *title*, извлеченных из коллекции *IRP* с условием совпадения атрибутов ключа. Элементы пары соединяются знаком «|->».

4. Семантика порядка исполнения операций. Для отслеживания фактов исполнения операций в конструкции *PersonFusion* заводится переменная *state*, отвечающая состоянию потока работ:

```
VARIABLES … state, …
INVARIANT
  state: struct(PeopleLinkCreated: BOOL,
       PositionsFromIRPInserted: BOOL,
       PositionsFromJobChangeInserted: BOOL,
       EmploymentInserted: BOOL,
       PersonInserted: BOOL
    ) & …
```

Переменная типизируется в инварианте типом структуры, атрибуты которого отражают завершение каждой из операций потока работ. Предусловием исполнения каждой из операций является завершение всех других необходимых операций. Также, завершающим действием каждой операции является установка флага завершения операции переменной *state* в значение FALSE. Для операции *insertIntoEmployment* соответствующие предусловия и присваивания выглядят в AMN следующим образом:

```
OPERATIONS
…
insertIntoEmployment =
SELECT state'PositionsFromIRPInserted = TRUE &
  state'PositionsFromJobChangeInserted = TRUE
THEN
  …
  state'EmploymentInserted := TRUE
END;
```

Тело операции образовано конструкцией SELECT [BAMN]. Это означает, что действия, указанные после ключевого слова THEN исполняются только в том случае, если предикат после ключевого слова SELECT обращается в истину. Операция *insertIntoEmployment* может быть исполнена только тогда, когда обе операции *insertIntoPositionsFromIRP* и *insertIntoPositionsFromJobChange* исполнены (т.е. флаги *state'PositionsFromIRPInserted* и *state'PositionsFromJobChangeInserted* имеют значение TRUE). После завершения операции флаг *state'EmploymentInserted* также должен быть установлен в значение TRUE.

Начальная инициализация переменной state производится в разделе INITIALISATION, флаги завершения всех операций устанавливаются в значение FALSE:

```
INITIALISATION
   state := rec(PeopleLinkCreated: FALSE,
      PositionsFromIRPInserted: FALSE,
      PositionsFromJobChangeInserted: FALSE,
      EmploymentInserted: FALSE,
      PersonInserted: FALSE)
  || …
```

4 Применение формальной семантики для языка разрешения сущностей и слияния данных для верификации потоков работ интеграции данных

Формулы, отражающие свойства потока работ интеграции данных, подлежащие верификации, добавляются в инвариант конструкции AMN, отражающей семантику потока работ интеграции данных (в рассматриваемом примере такой конструкцией является *PersonFusion*). Затем семантические спецификации помещаются в программное средство Atelier B [2], где осуществляется проверка синтаксиса спецификаций и корректности типизации переменных и термов.

Затем осуществляется автоматическая генерация теорем, отражающих сохранение инварианта при выполнении операций спецификации. Теоремы доказываются с использованием автоматических и интерактивных средств доказательства Atelier B.

В качестве примера свойства потока работ интеграции данных рассмотрим одно из возможных свойств сохранения информации при извлечении данных из исходных коллекций *IRP* и *JobChange* и формировании коллекции *Person*. Свойство формулируется в виде формулы логики предикатов и добавляется в инвариант конструкции *PersonFusion*:

```
INVARIANT … &
 (state'PersonInserted = TRUE =>
  !(nn, cc, tt).(
   (#(irp).(irp: IRP & nn = irp'name &
            cc = irp'company & tt = irp'title) or
   #(jc).(jc: JobChange & nn = jc'name &
          cc = jc'company & tt = jc'appointedAs)) =>
   #(pers).(pers: Person & pers'name = normName(nn) &
         #(ee).(ee: pers'emp & ee'company = cc &
           #(pos).(pos: ee'positions &
                    pos'title = normTitle(tt)) ) ) ) ) & …
```

В формуле утверждается, что по завершении потока работ выполняется следующее свойство: для любых наборов, состоящих из имени персоны, названия компании и должности, встречающихся совместно в записях коллекции *IRP* либо в записях коллекции *JobChange*, в коллекции *Person* найдется запись с такими же значениями имени персоны (с точностью до нормализации), названия компании и должности. Таким образом, из исходных коллекций извлекается вся возможная информация о местах работы и должностях персон.

Спецификация конструкции *PersonFusion* была помещена в средство Atelier B, была осуществлена синтаксическая проверка и проверка типов спецификации. Для каждой из операций спецификации были автоматически сгенерированы по три теоремы, в совокупности утверждающие сохранение инварианта при завершении операций.

5 Заключение

В статье рассмотрены основные принципы представления формальной семантики языка HIL, предназначенного для описания сложных потоков работ интеграции данных, включающих операции трансформации, разрешения и слияния сущностей. Рассмотренная семантика применяется для формальной верификации свойств потоков работ интеграции данных. Следующим шагом в работе будет реализация семантического отображения языка HIL в виде трансформации на высокоуровневом языке, например, ATL [3] или QVT. Это позволит использовать формальную верификацию свойств потоков работ при решении задач в различных средах интеграции [21, 22, 36].

Литература

[1] Abrial J.-R. The B-Book: Assigning Programs to Meanings. Cambridge: Cambridge University Press, 1996.

[2] Atelier B, the industrial tool to efficiently deploy the B Method. http://www.atelierb.eu/

[3] ATL - a model transformation technology. 2016. - https://eclipse.org/atl/

[4] Apache Hadoop Project. 2016. - http://hadoop.apache.org/

[5] Kevin S. Beyer, Vuk Ercegovac, Rainer Gemulla, Andrey Balmin, Mohamed Eltabakh, Carl-Christian Kanne, Fatma Ozcan, Eugene J. Shekita. Jaql: A Scripting Language for Large Scale Semistructured Data Analysis. VLDB 2011.

[6] Bleiholder J., Naumann F. Data fusion // ACM Computing Surveys (CSUR), 2009. Vol. 41. Iss. 1. Article No. 1. doi: 10.1145/1456650.1456651.

[7] Bleiholder J. Data fusion and conflict resolution in integrated information systems. — Potsdam: Hasso-Plattner-Institut, 2010. D.Sc. Diss. 184 p.

[8] D. Burdick, M. A. Hernández, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. R. Stanoi, S. Vaithyanathan, and S. Das. Extracting, Linking and Integrating Data from Public Sources: A Financial Case Study. IEEE Data Eng. Bull., 34(3):60–67, 2011.

[9] Daniel Calegari, Nora Szasz. Verification of Model Transformations: A Survey of the State-of-the-Art. Electronic Notes in Theoretical Computer Science. 292: 5–25, 2013.

[10] Luna Dong X., Naumann F. Data fusion — resolving data conflicts in integration // Proc. VLDB Endowment, 2009. Vol. 2. Iss. 2. P. 1654–1655.

[11] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: semantics and query answering. Theoretical Computer Science, 336(1):89–124, 2005.

[12] Ronald Fagin, Laura M. Haas, Mauricio Hernández, Renée J. Miller, Lucian Popa, Yannis Velegrakis. Clio: Schema Mapping Creation and Data Exchange. Conceptual Modeling: Foundations and Applications, LNCS 5600:198-236. 2009.

[13] Getoor L., Machanavajjhala A. Entity resolution for big data // KDD'13: 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining Proceedings, 2013. P. 1527–1527.

[14] IBM InfoSphere BigInsights Information Center. http://goo.gl/Bd8SE9

[15] Hernandez M., Koutrika G., Krishnamurthy R., Popa L., Wisnesky R. HIL:A high-level scripting language for entity integration // EDBT'13: 16th

Conference (International) on Extending Database Technology Proceedings, 2013. P. 549–560.

[16] Kalinichenko L.A. Method for Data Models Integration in the Common Paradigm. Proc. of the First East-European Symposium on Advances in Databases and Information Systems ADBIS'97. -- St.-Petersburg: Nevsky Dialect, 1997. -- V. 1: Regular Papers. -- P. 275--284.

[17] Kalinichenko L.A., Stupnikov S.A., Zemtsov N.A. Extensible Canonical Process Model Synthesis Applying Formal Interpretation // Avances in Databases and Information Systems: Proceedings of the East European Conference. - Berlin-Heidelberg: Springer-Verlag, 2005. - P. 183-198.

[18] Kalinichenko L.A., Stupnikov S.A. Constructing of Mappings of Heterogeneous Information Models into the Canonical Models of Integrated Information Systems // Advances in Databases and Information Systems: Proc. of the 12th East-European Conference. - Pori: Tampere University of Technology, 2008. - P. 106-122.

[19] Kalinichenko L.A., Stupnikov S.A. Heterogeneous information model unification as a pre-requisite to resource schema mapping // A. D'Atri and D. Saccà (eds.), Information Systems: People, Organizations, Institutions, and Technologies (Proc. of the V Conference of the Italian Chapter of Association for Information Systems itAIS). – Berlin-Heidelberg: Springer Physica Verlag, 2010. – P. 373-380.

[20] Kalinichenko L.A., Stupnikov S.A. OWL as Yet Another Data Model to be Integrated // Advances in Databases and Information Systems: Proc. II of the 15th East-European Conference. - Vienna: Austrian Computer Society, 2011. -- P. 178-189.

[21] Leonid Kalinichenko, Sergey Stupnikov, Alexey Vovchenko and Dmitry Kovalev. Rule-based Multi-dialect Infrastructure for Conceptual Problem Solving over Heterogeneous Distributed Information Resources // New Trends in Databases and Information Systems. Selected Papers of the 17th European Conference on Advances in Databases and Information Systems and Associated Satellite Events. – Springer, 2013. Advances in Intelligent Systems and Computing, V. 241. – P. 61-68.

[22] Kalinichenko L. A., Stupnikov S. A., Vovchenko A. E., Kovalev D. Y. Conceptual Modeling of Multi-Dialect Workflows. Информатика и ее применения, 2014. 8(4):110-124.

[23] Kopcke H., Thor A., Rahm E. Evaluation of entity resolution approaches on real-world match problems // Proc. VLDB Endowment, 2010. Vol. 3. Iss. 1-2. P. 484–493.

[24] P. G. Larsen, N. Plat, H. Toetenel. A Formal Semantics of Data Flow Diagrams. Formal Aspects of Computing 3:1993.

[25] K. Lano, J. Bicarregui, A. Evans. Structured Axiomatic Semantics for UML Models // Rigorous Object-Oriented Methods: Proc. of the Conference. – <http://www.bcs.org/upload/pdf/ewic_ro00_paper5.pdf. 2000>.

[26] K. Lano, S. Kolahdouz-Rahimi, T. Clark. Language-Independent Model Transformation Verification. Verification of Model Transformations: Proceedings of the Third International Workshop on Verification of Model Transformations. CEUR Workshop Proceedings 1325:36-45, 2014.

[27] Donald Miner. MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems. O'Reilly Media, 2012.

[28] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving inconsistencies at schema-, tuple-, and value-level. IEEE Data Engineering Bulletin, 29(2):21–31, 2006.

[29] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data Processing. In SIGMOD, p. 1099–1110, 2008.

[30] Schema Matching and Mapping. Bellahsene, Zohra, Bonifati, Angela, Rahm, Erhard (Eds.). Springer, 2011.

[31] Stupnikov S.A., Kalinichenko L.A., Bressan S. Interactive discovery and composition of complex Web services // Advances in Databases and Information Systems: Proc. of the 10th East European Conference. LNCS 4152. - Berlin-Heidelberg: Springer-Verlag, 2006. - P. 216-231.

[32] Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., Skiadopoulos, S.: A generic and customizable framework for the design of ETL scenarios. Information Systems 30(7), 492–525 (2005).

[33] Ступников С. А. Моделирование композиционных уточняющих спецификаций: Дис. канд. техн. наук.: 05.13.17 (Теоретические основы информатики) / ИПИ РАН, диссертационный совет Д.002.073.01. – М., 2006. – 195 с.

[34] Ступников С. А. Унификация модели данных, основанной на многомерных массивах, при интеграции неоднородных информационных ресурсов. Труды 14-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2012. – Переславль-Залесский: Университет города Переславля, 2012. С. 67-77. Одновременная электронная публикация в CEUR Workshop Proceedings, Vol. 934, P. 42-52. http://ceur-ws.org/Vol-934/

[35] С. А. Ступников. Отображение графовой модели данных в каноническую объектно-фреймовую

информационную модель при создании систем интеграции неоднородных информационных ресурсов // Труды 15-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2013. – Ярославль: Ярославский государственный университет им. П. Г. Демидова, 2013. С. 193-202. CEUR Workshop Proceedings 1108:85-94. http://ceur-ws.org/Vol-1108

[36] Ступников С. А., Вовченко А. Е. Комбинированная виртуально-материализованная среда интеграции больших неоднородных коллекций данных // Электронные библиотеки: перспективные методы и технологии, электронные коллекции (RCDL 2014): Тр. 16-й Всеросс. науч. конф. – Дубна: ОИЯИ, 2014. С. 339-348. CEUR Workshop Proceedings 1297:201-210. http://ceur-ws.org/Vol-1297/

Formal Semantics and Verification of Entity Resolution and Data Fusion Operations

Sergey Stupnikov

During all the period of development of methods and tools for data integration the issues of their formal verification were arising. Three levels of integration can be distinguished: data model integration, schema matching and integration, and proper data integration. This work proposes an approach for definition of formal semantics for high-level data integration programs. The semantics is defined using a specification language supported by formal provers. The semantics is applied for verification of structured data integration workflows. Workflow properties to be verified are presented as expressions of the specification language chosen. After that a semantic specification of the data integration workflow is verified w.r.t. required properties. A practical aim of the work is to define a basis for formal verification of data integration workflows during problem solving in various integration environments.