Анализ и интерпретация произвольных таблиц на основе исполнения CRL-правил

А. О. Шигаров*, И. В. Бычков, В. В. Парамонов, П. В. Белых

Институт динамики систем и теории управления им. В.М. Матросова СО РАН, Иркутск, Россия

*Контактный e-mail: shigarov@icc.ru

Работа посвящена вопросам извлечения данных из произвольных полуструктурированных таблиц и их трансформации к структурированной форме, из которой они могут быть загружены в базу данных с помощью стандартных ETL-средств. Предложен формальный язык правил анализа и интерпретации таблиц, называемый CRL. Исполнение таких правил позволяет восстанавливать семантику таблицы, отсутствующую изначально, но необходимую для извлечения и трансформации табличной информации. Экспериментальные данные показывают применимость предлагаемого языка к задачам интеграции неструктурированных табличных данных.

Ключевые слова: интеграция неструктурированных табличных данных, анализ и интерпретация таблиц, извлечение информации из таблиц, трансформация таблиц.

Введение

Важность вопросов интеграции неструктурированных данных отмечается многими исследователями [1–4]. Промышленный стандарт OASIS UIMA [5] определяет основное свойство неструктурированной информации как отсутствие в ней явно заданной семантики (структуры). Поэтому в отличие от структурированной информации она не может интерпретироваться машинами. Можно говорить, что таблицы, не включающие явно семантику, являются неструктурированными табличными данными. В частности, многие таблицы, представленные в текстовых документах, электронных таблицах и вебстраницах, являются примерами такой информации. Они могут иметь сложную компоновку ячеек, не позволяющую напрямую загружать их содержание в базу данных. Предварительно табличные данные должны быть извлечены и структурированы.

Интеграция неструктурированных табличных данных может рассматриваться как ETL-процесс, включающий следующие этапы: извлечение данных из исходных произвольных таблиц, их трансформацию к структурированному виду и загрузку в целевую базу данных. В данной работе рассматриваются вопросы извлечения данных из произвольных таблиц (рис. 1), представленных в формате табличного процессора Excel, их трансформации к канонической (структурированной) форме (рис. 2), информация из которой затем может загружаться в базу данных с помощью известных ETL-средств.

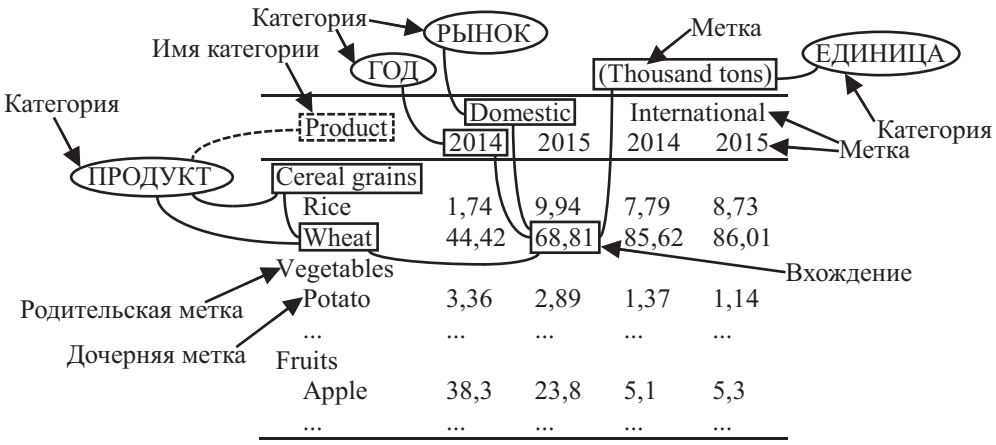


Рис. 1. Семантические отношения в произвольной таблице

ДАННЫЕ	ГОД	РЫНОК	ПРОДУКТ	ЕДИНИЦА
1,74	2014	Domestic	Cereal grains | Rice	Thousand tons
9,94	2015	Domestic	Cereal grains | Rice	Thousand tons
7,79	2014	International	Cereal grains | Rice	Thousand tons
8,73	2015	International	Cereal grains | Rice	Thousand tons
44,42	2014	Domestic	Cereal grains | Wheat	Thousand tons
68,81	2015	Domestic	Cereal grains | Wheat	Thousand tons
...	...	...	...	...
5,3	2015	International	Fruits | Apple	Thousand tons
...	...	...	...	...

Рис. 2. Каноническая форма таблицы, показанной на рис. 1

Предлагается реализация этих этапов на основе исполнения правил анализа и интерпретации произвольных таблиц. Схема предлагаемого процесса интеграции неструктурированных табличных данных показана на рис. 3.

Преобразование произвольной таблицы к канонической форме требует восстановления отсутствующих табличных семантических отношений, т. е. пар вида вхождение — метка, метка — метка и метка — категория (см. рис. 1). В литературе [6–9] задачу восстановления табличной семантики принято называть анализом и интерпретацией таблиц. В работах [10, 11] предложен метод восстановления табличной семантики на основе исполнения правил анализа и интерпретации таблиц.

Основная идея последнего заключается в том, что для различных типов таблиц, скомпонованных в соответствии с некоторыми требованиями, могут быть разработаны отдельные наборы продукционных правил. Их исполнение позволяет восстанавливать отсутствующие семантические отношения по доступным фактам, таким как пространственное расположение ячеек, их стилевые характеристики и текстовое содержание.

Настоящая работа развивает методологию, предложенную в предшествующих работах [10,11]. В ней предложен язык правил анализа и интерпретации таблиц, называемый CRL (Cells Rule Language). Он реализован в соответствии с требованиями свободной системы управления бизнес-правилами Drools [12] к предметно-ориентированным языкам. Это позволяет транслировать CRL-правила в конструкции нативного языка Drools — DRL [12] и исполнять их в данной системе.

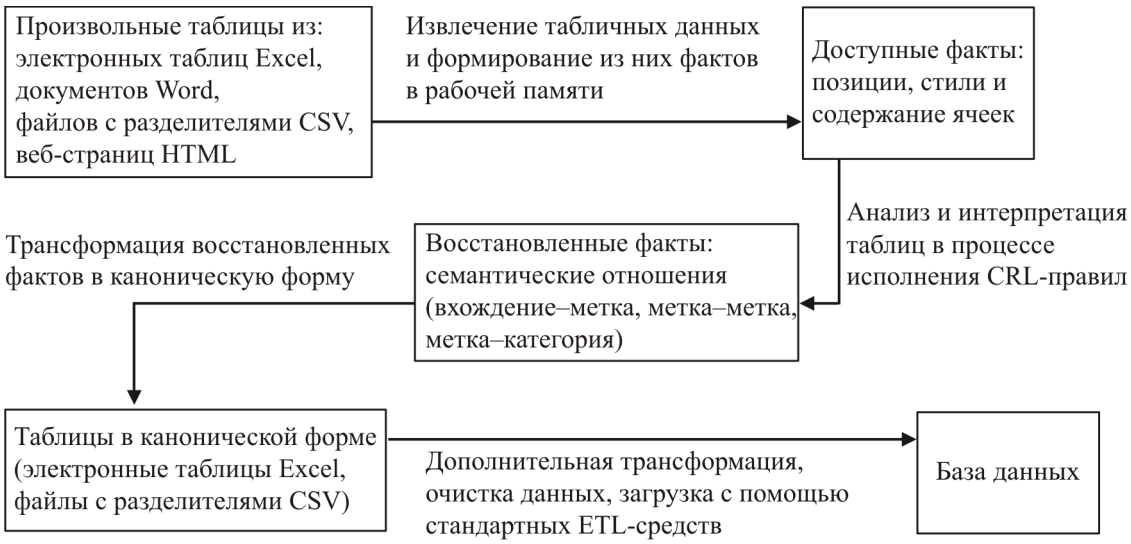


Рис. 3. Схема интеграции неструктурированных табличных данных с использованием исполнения правил анализа и интерпретации таблиц

Предлагаемая методология ориентирована в основном на задачи интеграции неструктурированной табличной информации, в особенности на случаи, когда для наполнения базы данных используются данные, представленные в большом количестве произвольных таблиц, имеющих схожие структурные, стилевые и содержательные особенности. Она реализована в представляемом в настоящей работе прототипе системы извлечения и трансформации данных из произвольных электронных таблиц. Экспериментальные результаты показывают применимость CRL-языка для разработки систем анализа и интерпретации произвольных таблиц.

Далее в работе рассматриваются: модель таблицы и структуры данных для представления табличной информации в виде фактов в процессе исполнения правил (разд. 1); конструкции языка CRL (разд. 2) и примеры CRL-правил (разд. 3); генерация канонической формы из восстановленной семантической информации таблицы (разд. 4); реализация и архитектура прототипа системы извлечения и трансформации табличных данных (разд. 5) и ее экспериментальная оценка (разд. 6); сравнение с известными методами анализа и интерпретации таблиц (разд. 7).

1. Представление табличных данных

При описании табличной семантики мы опираемся терминологически на абстрактную модель таблицы, предлагаемую в работе [13], заимствуя термины “вхождение”, “метка” и “категория”. Вхождение является значением данных, представленным в некоторой таблице. Каждое вхождение описывается одной или несколькими метками, которые могут быть представлены заголовками таблицы, сносками, названием таблицы, а также информацией, содержащейся в контексте. Каждая метка принадлежит только одной категории. Метки могут организовывать иерархию в виде ориентированного дерева. При этом метки, организованные в одном дереве, должны принадлежать одной категории. Каждое вхождение может быть ассоциировано только с одной меткой в каждой категории. Примеры рассматриваемых понятий и отношений между ними проиллюстрированы на рис. 1.

Предлагаемая модель таблицы имеет два уровня: структурный и семантический, определяющие исходные и восстанавливаемые факты соответственно. Первый из них определяет исходные факты, описывающие структуру, стили и содержание ячеек таблицы: $\langle C, S_r, S_c, T\rangle$, где C — набор ячеек, S_r — набор строк, S_c — набор столбцов и T — набор общих характеристик таблицы. Второй служит для представления восстанавливаемых фактов — семантических отношений: $\langle E, L, D, R_{EL}, R_{LL}, R_{LD}\rangle$, где E — набор вхождений, L — набор меток, D — набор категорий, R_{EL} — набор пар вида вхождение — метка, R_{LL} — набор пар метка — метка и R_{LD} — набор пар метка — категория.

В процессе исполнения CRL-правил исходная и восстанавливаемая табличная информация должна быть представлена в виде фактов. В рабочей памяти системы исполнения правил могут быть представлены факты одного из четырех типов: ячейки, вхождения, метки и категории.

Ячейка (`cell`) — основная структура, предназначенная для представления исходных табличных данных, извлеченных из некоторой ячейки. Каждый факт, представляющий ячейку, имеет следующие характеристики, доступные через соответствующие поля (выделенные здесь моноширинным шрифтом).

- Координаты в пространстве строк и столбцов: `cl` — левый и `cr` — правый столбец, `rt` — верхняя и `rb` — нижняя строка. Следует отметить, что ячейка может располагаться на нескольких смежных строках и/или столбцах, при этом всегда имея прямоугольную форму. Кроме того, любые две ячейки не могут пересекаться.
- Стилевые настройки вложены в отдельную структуру `style`. Основные из них: `font` — содержит все типичные шрифтовые характеристики (`font.name` — имя, `font.height` — размер, `font.color` — цвет и др.); тип выравнивания содержания: `horzAlignment` — горизонтального и `vertAlignment` — вертикального; цвета `fgColor` — переднего и `bgColor` — заднего плана; структуры, описывающие типы (`thin`, `medium`, `dashed` и др.) и цвета четырех границ ячейки: `leftBorder` — левой, `topBorder` — верхней, `rightBorder` — правой и `bottomBorder` — нижней.
- Текстовое содержание ячейки доступно через поле `text`. Несмотря на то что ячейка может содержать не только текст, но и изображения, RTF или формулу, в данной работе эти случаи не рассматриваются.
- Дополнительные характеристики: `cellType` — тип данных, установленный в электронной таблице Excel (числовой, денежный, текстовый и др.); `indent` — отступ (количество пробелов в начале текста); `width` — ширина и `height` — высота ячейки; `mark` — слово, маркирующее ячейку; `table` — ссылка на таблицу, содержащую данную ячейку; упорядоченные наборы `entries` — вхождений и `labels` — меток, порожденных из данной ячейки.

Вхождение (`entry`) — структура, служащая для представления значений данных, она состоит из следующих полей: `value` — строковое значение; `cell` — ссылка на ячейку его происхождения; `labels` — набор ассоциированных с ним меток.

Метка (`label`) — структура, представляющая метку, включает следующие поля: `value` — строковое значение; `cell` — ссылка на ячейку, из которой сгенерирована данная метка; `category` — ссылка на категорию этой метки; `parent` — ссылка на родительскую метку, а `children` — набор дочерних меток для данной метки в случае наличия иерархии.

Категория (`category`) обеспечивает представление фактов, описывающих категории с помощью следующих полей: `name` — имя категории и `labels` — набор принадлежащих ей меток.

2. CRL-правила

Продукционные CRL-правила служат для анализа и интерпретации таблиц. Их левая часть содержит условия, накладываемые на известные факты, а правая — действия (следствия), изменяющие состав таблицы и восстанавливающие семантические отношения ее элементов. В данной работе рассматриваются основные конструкции CRL-правил. Полная спецификация этого языка является набором DSL-отображений CRL в DRL-конструкции. Она доступна по адресу http://cells.icc.ru/pub/crl/crl.dsl.

Условия

Условия записываются в левой части правил и позволяют запрашивать размещенные в рабочей памяти ячейки, вхождения, метки и категории с заданными ограничениями:

```
cell $cell : constraints
entry $entry : constraints
label $label : constraints
category $category : constraints
```

Условие состоит из трех частей: ключевого слова, идентификатора переменной и списка ограничений. Его запись начинается с одного из следующих ключевых слов: `cell`, `entry`, `label` или `category`, указывающего тип фактов соответственно ячейки, вхождения, метки или категории. За ним следует имя переменной, предваряемое стартовым символом '$'. Созданная переменная может использоваться в других условиях и следствиях как ссылка на выбранные факты. После этого могут быть перечислены ограничения (`constraints`), накладываемые на запрашиваемые факты. Они отделяются от обязательной части условия знаком двоеточия ':'. Любое ограничение является выражением, вычисление которого приводит к значению булева типа данных ("истина" или "ложь"). Ограничения следуют DRL-синтаксису. Они отделяются друг от друга с помощью запятой ',', которая интерпретируется как логическая конъюнкция. Условие без списка ограничений означает запрос всех доступных фактов указанного типа.

Разделение ячеек

Данное следствие позволяет разделить объединенную ячейку `$cell`, состоящую из n плиток, на n ячеек, каждая из которых полностью повторяет ее содержание и стилевые характеристики:

```
split $cell
```

Координаты формируемой i-ячейки вычисляются по i-плитке исходной ячейки. Например, если объединенная ячейка имеет координаты $(\texttt{cl}, \texttt{rt}, \texttt{cr}, \texttt{rb})=(1, 1, 2, 2)$, то она разделяется на четыре ячейки с координатами $(1, 1, 1, 1)$, $(2, 1, 2, 1)$, $(1, 2, 1, 2)$ и $(2, 2, 2, 2)$. Все ячейки, полученные в результате такого разделения, вносятся в рабочую память, в то время как исходная объединенная ячейка удаляется из нее.

В основном разделять объединенные ячейки необходимо в том случае, когда они содержат вхождения. Ячейку, состоящую из n-плиток и содержащую m-вхождений, можно рассматривать как контейнер $n \times m$ повторяющихся вхождений. Ожидается, что разделение такой ячейки позволяет избежать сложности ассоциирования ее вхождений с метками из одной категории на основе анализа структуры таблицы.

Объединение ячеек

Противоположное предыдущему действие состоит в объединении двух соседних ячеек `$cell1` и `$cell2` с одной общей стороной:

```
merge $cell1 -> $cell2
```

В результате для адресата `$cell2` вычисляются новые координаты с помощью аргумента `$cell1` так, чтобы охватить обе ячейки. Остальные характеристики объединенной ячейки `$cell2` не меняются, но ячейка `$cell1` удаляется из модели обрабатываемой таблицы и рабочей памяти.

Данное действие может производиться для объединения ячеек с общим содержанием, например, когда две соседние ячейки содержат две разные части одной метки. Кроме того, серия пустых ячеек часто может неявно повторять содержание некоторой непустой ячейки. В таких случаях можно восстановить объединенные ячейки для унификации структуры таблиц и расширения класса таблиц, описываемого одним набором CRL-правил.

Маркировка ячеек

Данное действие привязывает к ячейке `$cell` некоторую отметку `@mark` — слово с обязательным стартовым символом @:

```
set mark @mark -> $cell
```

В последующих правилах такая отметка может использоваться для запроса промаркированных ею ячеек без необходимости повторять список ограничений. Маркировка ячеек позволяет упростить правила, сделать их запись более ясной и выразительной в тех случаях, когда возможно разделить ячейки обрабатываемой таблицы на некоторые наборы и интерпретировать их далее различными цепочками правил. Обычная практика состоит в том, чтобы сопоставить общие отметки тем ячейкам, которые выполняют одинаковую функцию или принадлежат одному региону внутри таблицы.

Порождение вхождений и меток

Действия, представленные ниже, обеспечивают создание вхождений и меток, привязывая их к некоторой ячейке `$cell`. В следующей форме их значения задаются аргументами — строковыми выражениями `entry_value` и `label_value` соответственно, которые обычно являются результатом обработки содержания ячейки адресата `$cell`:

```
new entry entry_value -> $cell
new label label_value -> $cell
```

В том случае, когда значение порождаемого вхождения или метки эквивалентно содержанию ячейки `$cell`, может использоваться сокращенная безаргументная форма:

```
new entry $cell
new label $cell
```

В результате созданные вхождения и метки добавляются в обрабатываемую таблицу и рабочую память системы исполнения правил. При этом они и ячейка их происхождения содержат взаимные ссылки друг на друга. Последнее обеспечивает доступ через ячейку к ее вхождениям и меткам и наоборот.

Категоризация меток

Язык CRL предусматривает два способа категоризации меток. В первом из них метка `$label` ассоциируется с выбранной категорией `$category`, доступной в рабочей памяти:

```
set category $category -> $label
```

Во втором аргумент является строковым выражением `category_name`, указывающим имя категории:

```
set category category_name -> $label
```

В последнем случае прежде всего выполняется поиск категории по заданному имени `category_name` в модели обрабатываемой таблицы. Если такая категория существует, то она ассоциируется с меткой `$label`. В противном случае в модели создается новая категория с именем `category_name`, ассоциируемая затем с адресатом `$label`.

Ассоциирование меток

Две метки могут быть ассоциированы друг с другом как родительский — `$label1` и дочерний — `$label2` узел:

```
set parent label $label1 -> $label2
```

Предполагается, что все метки, связанные между собой такими отношениями в некоторое ориентированное дерево, в случае их категоризации должны принадлежать одной иерархической категории. Обратное приводит к аварийному завершению интерпретации таблицы.

Кроме того, такие связи формируют составные значения меток следующим образом. Если в дереве меток существует путь $l_1,\ldots,l_n$, где l_1 — его корень, то составное значение метки l_n включает значения всех узлов в этом пути в порядке их вложенности, начиная с корня: $l_1|l_2|\ldots|l_n$. Полученные составные значения могут использоваться при генерации выходной канонической формы.

Группировка меток

Рассматриваемое действие добавляет две метки `$label1` и `$label2` в одну группу — некоторый набор меток:

```
group $label1 -> $label2
```

Когда одна из меток (аргумент или адресат) уже принадлежит некоторой группе, вторая также добавляется в нее. Если обе ячейки принадлежат двум разным группам, то обе группы объединяются в одну.

В том случае, когда сгруппированная метка ассоциируется с некоторой категорией, предполагается, что все остальные метки из той же самой группы также должны принадлежать данной категории. После этого все некатегоризированные метки этой группы также ассоциируются с данной категорией. Попытка связать метки из одной группы с разными категориями приводит к аварийному завершению интерпретации таблицы.

В некоторых таблицах можно установить факт того, что несколько меток принадлежат одной категории, используя только пространственные и стилевые характеристики ячеек, но в то же время не определяя, какая именно это категория. Например, в сводных таблицах, являющихся результатом многомерного анализа данных, часто каждая строка внутри «шапки», каждый столбец внутри “боковика” формируется метками отдельной категории. В таких случаях можно определить, что метки, порожденные из ячеек одной строки (столбца), принадлежат одной категории.

Группы, метки которых не сопоставлены категориям, рассматриваются как анонимные категории. По завершении исполнения правил для каждой из них создается отдельная категория с автоматически сгенерированным именем, с которой ассоциируются все ее метки.

Ассоциирование вхождений

Данное действие связывает вхождение `$entry` с меткой `$label`:

```
add label $label -> $entry
```

При этом выполняется проверка принципа, в соответствии с которым вхождение может быть связано только с одной меткой в каждой категории. Нарушение этого базового предположения останавливает дальнейший процесс интерпретации таблиц. С другой стороны, это предполагает, что каждая метка, сопоставленная вхождению, должна принадлежать некоторой категории. Поэтому, если метка `$label` не связана с категорией, то ассоциирование вхождения `$entry` откладывается. Вместо этого данная метка становится кандидатом, попытка ассоциировать вхождение `$entry` с которым выполняется только после ее категоризации.

Также ассоциирование вхождения `$entry` и метки со значением, заданным строковым выражением `label_value` из выбранной категории `$category`, может выполняться в следующей форме:

```
add label label_value
from $category -> $entry
```

В этом случае выполняется поиск метки по представленному значению `label_value` среди меток категории `$category`. Когда такой метки нет, она создается внутри данной категории. Затем найденная или созданная метка связывается с вхождением `$entry`.

Еще один способ ассоциировать вхождение `$entry` и метку со специфицированным значением `label_value` состоит в использовании имени категории, заданного строковым выражением `category_name`, вместо переменной, ссылающейся на нее, как показано ниже:

```
add label label_value
from category_name -> $entry
```

Обработка такого следствия начинается с проверки: существует ли в модели таблицы категория с заданным именем `category_name`. Если нет, то создается пустая категория с таким именем. Далее найденная или созданная категория используется в данной операции, как описано выше: для поиска или создания метки со значением `label_value`.

Следует отметить, что две последние формы позволяют создавать метки независимо от ячеек, определяя их непосредственно в CRL-правилах. Эта возможность является

важной, поскольку часть меток не может быть сгенерирована из содержания ячеек обрабатываемой таблицы. Например, для некоторого набора таблиц исключительно из контекста может быть известно, что представленные в них данные характеризуются некоторой единицей измерения, временем или местом. В таком случае целесообразно включить такую информацию напрямую в CRL-правила.

Вспомогательные действия

Дополнительно к описанным действиям язык CRL предоставляет также ряд вспомогательных, таких как изменение значения ячейки, вхождения или метки, обновление фактов в рабочей памяти и вывод отладочной информации в командную строку.

3. Применение CRL-правил

Предлагаемая модель и CRL-правила позволяют работать с распространенными и редкими особенностями таблиц. Некоторые примеры применения CRL-правил к таким особенностям представлены в данном разделе. Кроме того, дополнительные примеры правил доступны по адресу http://cells.icc.ru/pub/crl.

Объединенные ячейки

На рис. 4 показаны четыре таблицы, которые можно отнести к одному типу. Эти таблицы могут иметь непустые объединенные ячейки (рис. 4, *а*). Разделение этих ячеек может быть выполнено следующим образом:

```
when
    cell $c : cl!=cr || rt!=rb, !blank
then
    split $c
```

В результате таблица из рис. 4, *а* приводится к виду, показанному на рис. 4, *б*.

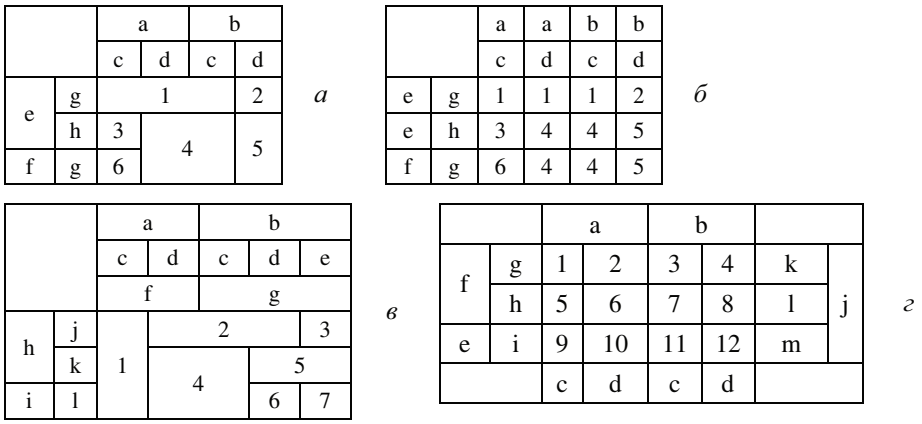


Рис. 4. Сводные таблицы: для удобства вхождения представлены числами, а метки — латинскими символами; объединенные (*а*) и разделенные (*б*) ячейки; регионы меток могут располагаться сверху, слева, снизу и справа от региона вхождений, при этом пустые угловые "критические" ячейки указывают их позиции (*а–г*)

Критические ячейки

Для таблиц, представленных на рис. 4, *а–в*, можно предположить, что пустая ячейка, расположенная в верхнем левом углу, определяет границы между двумя регионами меток и одним регионом вхождений. В работе [14] подобные ячейки принято называть "критическими". Полагаясь на сделанное предположение, можно записать следующее правило, порождающее вхождения:

```
when
  cell $corner : cl==1, rt==1, blank
  cell $c : cl>$corner.cr, rt>$corner.rb
then
  new entry $c
```

Строчные и столбцовые метки

При обработке сводных таблиц, подобных показанным на рис. 4, *а–в*, метки можно разделить на строчные и столбцовые в зависимости от того, как они адресуют вхождения: по строкам или столбцам соответственно. Следующий пример демонстрирует порождение столбцовых меток в таких таблицах и маркировку их ячеек словом `@ColumnHeading`:

```
when
  cell $corner : cl==1, rt==1, blank
  cell $c : cl>$corner.cr, rb<=$corner.rb
then
  set mark @ColumnHeading -> $c
  new label $c
```

Группы меток в сводных таблицах

Для таких таблиц (рис. 4) можно предположить, что две столбцовые метки, порожденные из ячеек одной строки, принадлежат одной категории, а в противном случае — двум разным категориям. Используя это предположение, можно сгруппировать столбцовые метки для ассоциирования их с анонимными категориями:

```
when
  cell@ColumnHeading $c1
  cell@ColumnHeading $c2 : rt==$c1.rt
then
  group $c1.label -> $c2.label
```

Аналогично можно считать, что строчные метки, происходящие из ячеек одного столбца, должны быть ассоциированы с одной категорией. Например, метки таблицы, показанной на рис. 4, *в*, могут быть сгруппированы в пять категорий следующим образом: {a, b}, {c, d, e}, {f, g}, {g, h}, {h, i}, и {j, k, l}.

Заданные категории

Значения некоторой категории могут задаваться в виде наборов естественно-языковых и регулярных выражений в формате YAML [15]. Например, категория, названная "Год"

и ограниченная значениями (1995,. . . , 2015), может быть описана с помощью трех регулярных ограничений следующим образом:

```
# category YEAR
name: Год
description: некоторый год с 1995 по 2015
constraints:
- "199[5-9]"
- "200[1-9]"
- "201[0-5]"
```

Например, метки, являющиеся значениями (1995,. . . , 2015), могут быть ассоциированы с описанной категорией “Год”:

```
when
  category $c : name=="Year"
  label $l : $c.canHaveLabel(value)
then
  set category $c -> $l
```

Другой пример демонстрирует описание категории “КодСтраны”, которая содержит двухбуквенные коды стран:

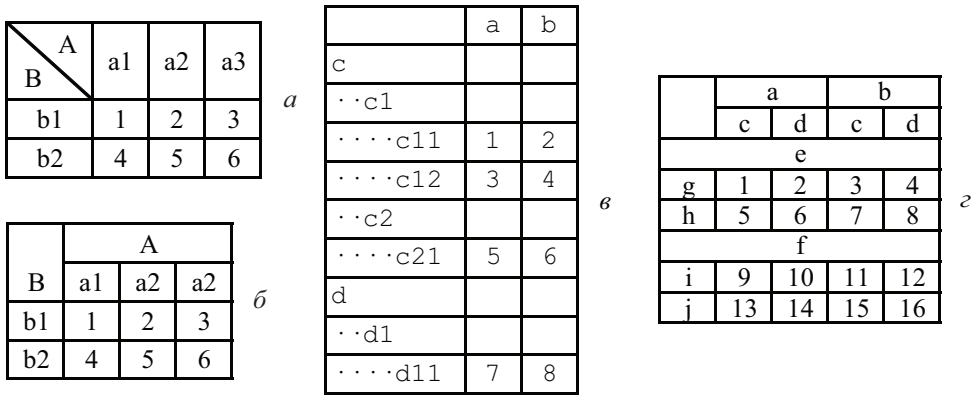
```
# category COUNTRY_CODE
name: КодСтраны
description: двухбуквенный код страны в стандарте ISO 3166
labels:
- AD
- AE
- ..
- ZW
```

Следующее правило может использоваться, чтобы ассоциировать метки, представляющие коды стран, с категорией “КодСтраны”:

```
when
  category $c : name=="CountryCode"
  label $l : $c.hasLabel(value)
then
  set category $c -> $l
```

Имена категорий внутри таблицы

Некоторые таблицы содержат имена категорий (например, A и B на рис. 5, *а* и *б*), которые могут использоваться для категоризации их меток. Например, в случае таблиц с компоновкой, показанной на рис. 5, *а*, можно предположить, что ячейка в верхнем левом углу содержит два имени категорий. Первое из них адресует столбцовые, а второе — строчные метки. Следующее правило создает категорию из первого слова, которое возвращает функция `token` из текста, содержащегося в ячейке в верхнем левом углу `$corner`, и ассоциирует с ней столбцовые метки:



B \ A	a1	a2	a3
b1	1	2	3
b2	4	5	6

а

B	A		
	a1	a2	a2
b1	1	2	3
b2	4	5	6

б

	a	b
c		
··c1		
····c11	1	2
····c12	3	4
··c2		
····c21	5	6
d		
··d1		
····d11	7	8

в

	a		b	
	c	d	c	d
e				
g	1	2	3	4
h	5	6	7	8
f				
i	9	10	11	12
j	13	14	15	16

г

Рис. 5. Заголовки A и B являются именами категорий: A описывает столбцовые (a1, a2, a3), а B — строчные (b1, b2) метки (*а–б*); отступы (·) от левого края в тексте строчных меток {c,. . . , c21, d,. . . , d11} (*в*); перерезы e и f пересекают регион вхождений по столбцам (*г*)

```
when
  cell $corner : cl == 1, rt == 1
  label $l : cell.rb <= $corner.rb
then
  set category token($corner, 0) -> $l
```

Иерархии строчных меток

Одной из распространенных практик является использование в тексте таблиц отступов от левого края, чтобы показать существующие иерархические отношения между ее строчными метками (рис. 5, *в*). Например, в следующем правиле, используя предположение о том, что каждому уровню вложенности строчных меток соответствует дополнительный отступ (`indent`), равный двум пробелам, мы восстанавливаем отношения между родительскими (`$c1.label`) и дочерними (`$c1.label`) узлами:

```
when
  cell $c1 : cl==1, $l1 : label
  cell $c2 : cl==1, rt>$c1.rt, indent==$c1.indent+2, $l2 : label
  no cells : cl==1, rt>$c1.rt, rt<$c2.rt, indent==$c1.indent
then
  set parent label $c1.label -> $c2.label
```

Перерезы

Таблица может иметь специальные объединенные ячейки, называемые “перерезами”, которые пересекают ее тело через все столбцы (рис. 5, *г*). Предполагается, что такой переrез содержит метку, адресующую вхождения в расположенном непосредственно под ним регионе. Например, метка ‘e’ адресует вхождения {1,. . . , 8} на рис. 5, *г*. В таком случае можно записать следующее правило для обнаружения и маркировки перерезов:

```
when
  cell $c : cl==1, cr==table.numOfCols
then
  set mark @CutInHead -> $c
```

Многозначные ячейки

Одна ячейка может содержать несколько значений (вхождений и меток) одновременно (рис. 6). Например, в двуязычной таблице, показанной на рис. 6, *а*, каждая ячейка содержит либо две метки, либо два вхождения на двух языках. Предполагая, что в таких таблицах первое слово в ячейке записано на одном языке, а второе дублирует его на другом, можно записать следующее правило генерации меток из ячеек, расположенных в самой верхней строке или самом левом столбце:

```
when
  cell $c : cl==1 || rt==1, !blank
then
  new label token($c, 0) -> $c
  new label token($c, 1) -> $c
```

В данном примере (рис. 6, *а*) мы предполагаем, что вхождения и метки могут быть связаны друг с другом, только если они записаны на одном языке. Кроме того, предполагается, что вначале сгенерированы вхождения и метки для одного языка, а затем для второго. Правило, реализующее эти предположения, показано ниже:

```
when
  cell $c1 : containsLabel()
  cell $c2 : containsEntry(), cl==$c1.cl || rt==$c1.rt
then
  add label $c1.label[0] -> $c2.entry[0]
  add label $c1.label[1] -> $c2.entry[1]
```

Для таблиц типа, который показан на рис. 6, *б*, где любая ячейка `$c`, расположенная ниже первой строки, содержит текст вида “ключ=значение”, следующее правило порождает метку из его части “ключ” и вхождение из части “значение”, возвращаемые функциями `left` и `right` соответственно:

```
when
  cell $c : rt>1
then
  new label left($c, '=') -> $c
  new entry right($c, '=') -> $c
```

	α 阿爾法	β 公測
γ 伽馬	1 一	2 二
δ 三角洲	3 三	4 四

а

C1	C2
a = 1	b = 2
c = 3	d = 4
e = 7	f = 8
g = 10	h = 11

б

Рис. 6. Многозначные ячейки: двуязычные таблицы с дублированием вхождений и меток на двух языках (*а*); ячейки, содержащие текст вида “ключ=значение”, где метка “ключ” связана с вхождением “значение” (*б*)

Следующее правило связывает порожденные описанным способом метки и вхождения:

```
when
  cell $c : rt>1
then
  add label $c.label -> $c.entry
```

Сноски

В зависимости от целевого представления могут использоваться разные интерпретации табличных сносок. В следующем примере (рис. 7, *а*) предполагается, что сноски 'x' и 'y', связанные с вхождениями 1 и 2 через ссылки * и **, соответственно являются метками. Представленное ниже правило порождает метки из сносок `$notes`, содержащихся в "подвале" таблицы `$footer`, ассоциирует их с категорией, названной `footnotes`, и связывает их с вхождениями, используя соответствующие ссылки `$ref`:

```
when
  cell $footer : onLastRow, $notes : text
  entry $e : cell.text matches ".+\\*+", $ref : extract(cell.text, "\\*+")
then
  add label between($notes, $ref, '\n') from "footnotes" -> $e
```

В другом примере (рис. 7, *б*) сноски 'u' и 'v' считаются частями меток 'b' и 'f' соответственно. Представленное ниже правило порождает метки, добавляя в их значения текст соответствующих сносок, заключенный в круглые скобки (например, b (u) и f (v)):

```
when
  cell $footer : onLastRow, $fn : text
  cell $c : text matches ".+\\*+",
          $val : left(text, '*'), $ref : substr(text, '*')
then
  new label format("%s (%s)", $val, between($fn, $ref, '\n')) -> $c
```

Чередование регионов вхождений и меток

В таблицах, демонстрируемых на рис. 8, используется чередование регионов вхождений и меток. При этом регионы меток выделены разными цветами: (l1, l2, l3) — "зеленым", (c1, l4, l5, l6) — "синим" и (c2, l7, l8, l9) — "оранжевым". Предполагается, что каждый

	a		b	
	c	d	c	d
e	1*	2**	3	4
f	5	6	7	8
g	9	10	11	12
* x ** y				

а

	a		b*	
	c	d	c	d
e	1	2	3	4
f**	5	6	7	8
g	9	10	11	12
* u ** v				

б

Рис. 7. Сноски в таблицах

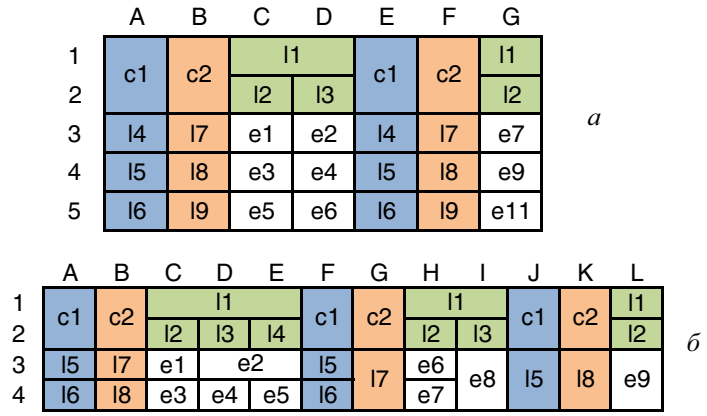


Рис. 8. Таблицы с чередованием регионов вхождений и меток; регионы меток выделены цветом

цвет соответствует отдельной категории. В данном примере цвета могут использоваться, чтобы восстановить вхождения, метки и категории.

В следующем правиле выбираются ячейки, выделенные синим цветом, они маркируются словом `@blue` и используются для порождения меток:

```
when
  cell $c : style.bgColor=="#4f81bd"
then
  new label $c
  set mark @blue -> $c
```

Другое правило демонстрирует, как цветовые характеристики ячеек могут использоваться, чтобы восстановить отношения между вхождениями и метками: вхождения ассоциируются с метками, порожденными в ячейках, выделенных синим цветом:

```
when
  label $l: cell.mark=="@blue", $c1 : cell
  entry $e : cell.(rt>=$c1.rt&&rt<=$c1.rb) ||
        (rb>=$c1.rt&&rb<=$c1.rb), cl>$c1.cr), $c2 : cell
  no cells@blue : cl>$c1.cr, cr<$c2.cl
then
  add label $l -> $e
```

4. Генерация канонической формы

Восстановленные семантические отношения обеспечивают генерацию канонической формы таблицы, устроенную как таблицы реляционной базы данных, где столбцы являются полями, первая строка содержит имена полей, а каждая последующая строка является записью. Предлагаемая каноническая форма обязательно включает поле `DATA`, содержащее все восстановленные вхождения. Остальные поля соответствуют восстановленным категориям. Каждое из них содержит метки, принадлежащие соответствующей категории. Каждая запись представляет восстановленные отношения между вхождением и метками.

Для примера в результате анализа и интерпретации таблиц, показанных на рис. 9, *а*, могут быть восстановлены следующие отношения:

а

	a		b	
	c	d	c	d
e	1	2	3	4
f	5	6	7	8

б

A / B	a1		a2	
	a11	a12	a21	a22
b1	1	2	3	4
b2	5	6	7	8

в

DATA	C1	C2	C3
1	a	c	e
2	a	d	e
3	b	c	e
4	b	d	e
5	a	c	f
6	a	d	f
7	b	c	f
8	b	d	f

г

DATA	A	B
1	a1 | a11	b1
2	a1 | a12	b1
3	a2 | a21	b1
4	a2 | a22	b1
5	a1 | a11	b2
6	a1 | a12	b2
7	a2 | a21	b2
8	a2 | a22	b2

Рис. 9. Исходные таблицы (*а*, *в*) и их канонические формы (*б*, *г*)

```
<entries>={1,2,3,4,5,6,7,8}
<labels>={a,b,c,d,e,f}
<groups>={{a,b},{c,d},{e,f}}

<entry_label_pairs>=
{(1,a),(1,c),(1,e),(2,a),(2,d),(2,e),
(3,b),(3,c),(3,e),(4,b),(4,d),(4,e),
(5,a),(5,c),(5,f),(6,a),(6,d),(6,f),
(7,b),(7,c),(7,f),(8,b),(8,d),(8,f)}
```

В данном примере предполагается, что каждая восстановленная группа меток является анонимной категорией. Используя эти группы, генерируются три категории, которые ассоциируются с метками следующим образом:

```
<categories>={C1,C2,C3}

<label_category_pairs>={(a,C1),(b,C1),
(c,C2),(d,C2),(e,C3),(f,C3)}
```

В другом примере (рис. 9, *в*) предполагается, что A и B являются именами категорий, а столбцовые метки образуют иерархию:

```
<entries>={1,2,3,4,5,6,7,8}
<labels>={a1,a11,a12,a2,a21,a22,b1,b2}
<categories>={A,B}

<entry_label_pairs>={(1,a11),(1,b1),
(2,a12),(2,b1),(3,a21),(3,b1),(4,a22),
(4,b1),(5,a11),(5,b2),(6,a12),(6,b2),
(7,a21),(7,b2),(8,a22),(8,b2)}
```

```
<label_label_pairs>={(a11,a1),(a12,a1),
(a21,a2),(a22,a2)}

<label_category_pairs>={(a1,A),(a11,A),
(a12,A),(a2,A),(a21,A),(a22,A),(b1,B),
(b2,B)}
```

Восстановленная семантика позволяет трансформировать исходную таблицу в каноническую форму. Например, на рис. 9 таблица *а* приводится к канонической форме *б*, а таблица *в* — к форме *г*.

5. Реализация

На основе предлагаемого метода и языка правил CRL разработан прототип системы для извлечения и трансформации табличных данных из произвольных электронных таблиц Excel в структурированную форму. Его архитектура приведена на рис. 10.

Разбор и загрузка данных из таблиц Excel производится с помощью свободной системы Apache POI [16]. В результате формируются объекты, представляющие таблицу,

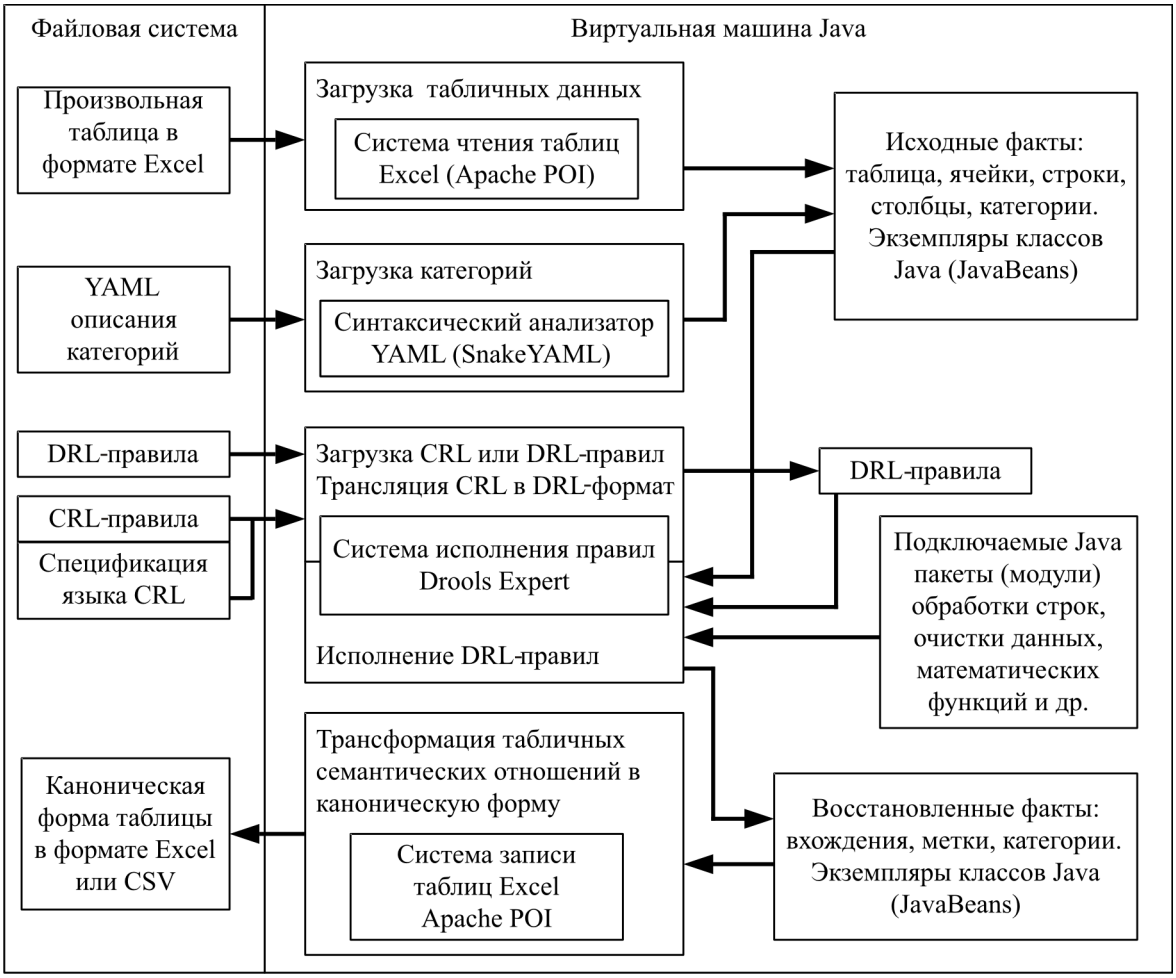


Рис. 10. Архитектура прототипа системы извлечения и трансформации табличных данных

ее ячейки, строки и столбцы. Они добавляются как факты в рабочую память системы исполнения правил. Опционально одна или несколько категорий, описанных с помощью языка YAML [15], могут загружаться в систему, также формируя соответствующие факты в рабочей памяти Drools. При этом синтаксический разбор YAML-описаний выполняется в свободной системе SnakeYAML [17].

Структуры данных, представляющие ячейки, вхождения, метки и категории, реализованы как Java-классы в соответствии с соглашениями спецификации JavaBeans [18]. При этом каждый факт является экземпляром одного из этих классов. Это позволяет использовать их для представления табличных данных как фактов в любой системе исполнения правил, реализующей спецификацию "JSR 94: Java Rule Engine API" [19]. Следовательно, сами правила анализа и интерпретации таблиц могут записываться на таких широко распространенных языках правил, как DRL [12], JESS [20] и RuleML [21], или представляться с помощью таблиц принятия решений, как, например, OpenRules [22].

В прототипе в качестве системы исполнения правил используется свободное программное обеспечение Drools Expert. При этом правила могут быть выражены либо на встроенном языке DRL, либо на специальном языке CRL. Последний из них реализован в соответствии с требованиями свободной системы управления бизнес-правилами Drools к предметно-ориентированным языкам. Система Drools обеспечивает трансляцию CRL-правил в DRL-конструкции по предоставленной спецификации CRL-языка (http://cells.icc.ru/pub/crl/crl.dsl).

В правилах могут использоваться различные дополнительные Java-классы из подключаемых пакетов, обеспечивающие в том числе обработку строк и естественного языка, очистку данных, математические функции. Предполагается, что они являются доступными при исполнении правил.

Факты, загруженные в рабочую память, сопоставляются скомпилированным правилам. В результате создаются новые факты, описывающие семантику таблицы: вхождения, метки, категории и отношения между ними. Они трансформируются в каноническую форму таблицы, которая затем записывается в Excel-файл с помощью системы Apache POI.

6. Экспериментальная оценка

В предлагаемой экспериментальной оценке в качестве текстовых таблиц используется коллекция данных, собранная в рамках проекта TANGO (http://tango.byu.edu) по развитию методов анализа и интерпретации таблиц для генерации онтологий. Она содержит 200 таблиц, импортированных из 10 источников (сайтов со статистической информацией) и доступна по адресу http://tango.byu.edu/data.

Подробное описание этой коллекции, наборы CRL-правил, разработанные для оценки, а также полученные результаты (канонические формы таблиц) доступны по адресу http://cells.icc.ru/pub/crl.

В результате проведенного анализа сделан ряд предположений о табличной структуре, стиле и содержании, которые выполняются для большинства таблиц из коллекции TANGO.

1. Таблица всегда состоит из четырех регионов ячеек, разделенных двумя воображаемыми линиями, как показано на рис. 11, *а*: (1) — верхний левый содержит ячейки с именами категорий; (2) — верхний правый включает ячейки меток, описывающие

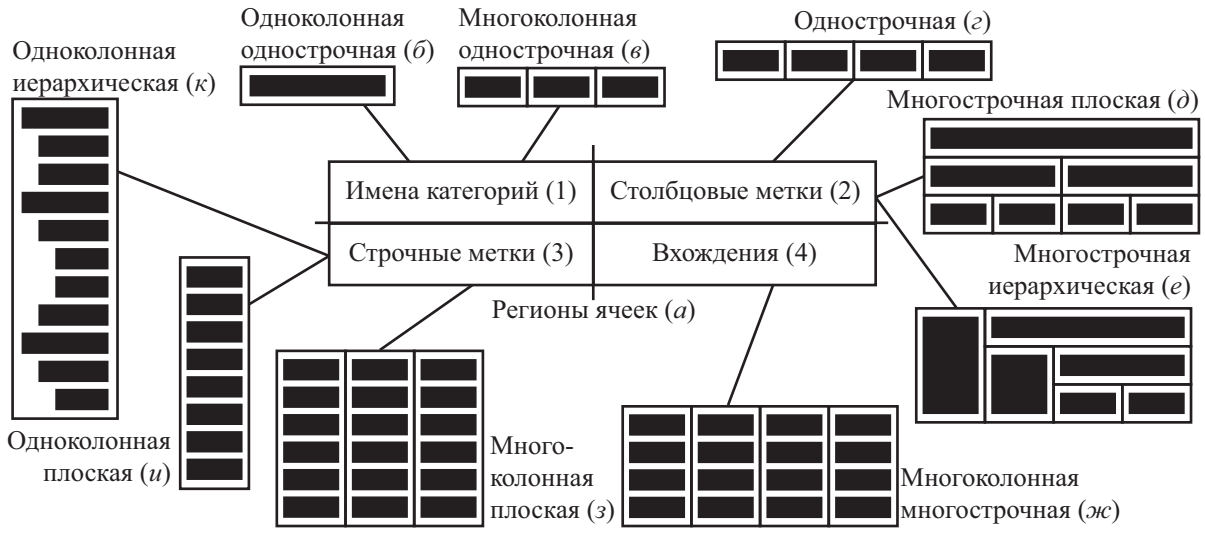


Рис. 11. Регионы ячеек ($а$) и виды их компоновок ($б$–$к$) в таблицах из коллекции TANGO

вхождения по столбцам; (3) — нижний левый включает ячейки меток, описывающие вхождения по строкам; (4) — правый нижний содержит ячейки вхождений.

2. Каждый регион может иметь одну из соответствующих ему компоновок, как показано на рис. 11. В табл. 1 приводятся данные по распределению этих компоновок среди таблиц TANGO.
3. Одна ячейка может содержать только одно вхождение, одну метку или одно имя категории одновременно. Несмотря на то что в некоторых из представленных таблиц содержание одной ячейки правильнее было бы интерпретировать как несколько меток, мы ограничиваемся данным предположением для упрощения эксперимента. Также для упрощения ссылки на сноски, содержащиеся в ячейках, рассматриваются как часть соответствующей метки или вхождения.
4. Каждая непустая ячейка из региона (1) задает имя категории, а все метки, происходящие из того же самого столбца, принадлежат этой категории.
5. Столбцовые метки (2) могут образовывать иерархию. В таблице она может быть представлена таким образом, что вложенный заголовок находится непосредственно под охватывающим заголовком. При этом заголовок верхнего уровня охватывает, по крайней мере, два вложенных заголовка (рис. 11, $д$ и $е$).

Т а б л и ц а 1. Использование различных компоновок в таблицах из коллекции TANGO

Регион	Вид компоновки	Рис. 11	Количество случаев, %
Ячейки имен категорий (1)	Одноколонная однострочная	*б*	94.5
	Многоколонная однострочная	*в*	5.5
Ячейки столбцовых меток (2)	Однострочная плоская	*г*	65.5
	Многострочная плоская	*д*	26
	Многострочная иерархическая	*е*	8.5
Ячейки строчных меток (3)	Многоколонная плоская	*з*	5.5
	Одноколонная плоская	*и*	47
	Одноколонная иерархическая	*к*	47.5
Ячейки вхождений (4)	Многоколонная многострочная	*ж*	100

6. Строчные метки (3), происходящие из самого левого столбца (боковика), могут образовывать иерархию (рис. 11, *к*). В таблице она может быть представлена отступами заголовков от левого края, так что каждый уровень вложенности добавляет два пробела. Также признаками иерархии могут быть символы дефиса '-' в начале вложенного заголовка и выделение полужирным шрифтом заголовков первого уровня.
7. Вхождения (4) являются числами или специальными обозначениями, например '#', 'F', '...', 'x', 'NA', указывающими на отсутствие, ненадежность, конфиденциальность и другие свойства.

Все используемые предположения формально записаны в виде левых частей (условий) правил из набора CRL-200. Они определяют один тип таблиц, названный нами TANGO-200.

В рассматриваемом эксперименте оцениваются точность и полнота каноникализации таблиц.

Полнота — исходная таблица считается обработанной полностью, если вхождения, метки, вхождение — метка, метка — метка, содержащиеся в ней, также включены в полученную каноническую форму.

Точность — исходная таблица считается обработанной точно, если все вхождения, метки, связи вида вхождение — метка, связи вида метка — метка, включенные в полученную каноническую форму, также содержатся в ней самой.

Оценка производится следующим образом. Два эксперта независимо друг от друга сравнивают исходные таблицы и их канонические формы, полученные в результате исполнения CRL-правил, давая заключение о полноте и точности каноникализации. В тех случаях, когда их решения не совпадают, окончательное заключение принимается совместно уже тремя экспертами.

Полученные оценки представлены в табл. 2. В результате применения набора CRL-200 к коллекции TANGO полнота составляет 87 %, а точность — 90 %. При этом 33 таблицы обрабатываются с ошибками восстановления семантической информации, снижающими точность и полноту каноникализации. В основном (28 из 33 таблиц) ошибки происходят при восстановлении связей вида метка — метка, содержащихся в регионах строчных меток (3) с одноколонной иерархической компоновкой (рис. 11, *к*).

Во-первых, на точность и полноту влияют противоречия, содержащиеся в таблицах. Например, если в одном случае полужирный шрифт используется для подчеркивания иерархии между метками, то в другом — только для того, чтобы выделить агрегированные данные внутри таблицы. Во-вторых, фактором, снижающим эффективность, является то, что в некоторых таблицах для части семантических отношений не использовались структурные или типографические приемы выделения. В-третьих, некоторые таблицы содержат неаккуратную структуру или спутанные признаки иерархии меток.

Таблица 2. Экспериментальные результаты по двум типам таблиц

Набор данных	TANGO-200	TANGO-SUB
Всего таблиц	200	105
Всего ячеек	22 757	10 893
Всего правил	16	13
Полнота, %	87	95
Точность, %	89	95

Хотя в данном эксперименте не производится оценка восстановления имен категорий, однако можно отметить, что в результате восстанавливаются 96 имен категорий, из которых 6 являются ошибками.

Дополнительно экспериментальная оценка проведена для специального подтипа таблиц, названного TANGO-SUB, представляющего подмножество таблиц типа TANGO-200. Тип TANGO-SUB отличается от общего типа TANGO-200 только тем, что компоновка региона строчных меток (3) может являться либо одноколонной плоской (рис. 11, *д*), либо многоколонной плоской (рис. 11, *д*). Среди таблиц коллекции TANGO к типу TANGO-SUB можно отнести 105 таблиц. Для анализа и интерпретации таких таблиц составлен отдельный набор правил CRL-SUB. От набора CRL-200 он отличается только тем, что из него исключены три правила, предназначенные для восстановления отношений метка — метка в случае одноколонной иерархической компоновки (рис. 11, *д*) региона строчных меток. Полученные для этого случая оценки показаны в табл. 2. Представленное сужение типа до TANGO-SUB приводит к упрощению набора правил и росту полноты и точности каноникализации таблиц по сравнению с общим типом TANGO-200.

7. Сравнение с родственными работами

Известные методы анализа и интерпретации таблиц можно разделить на две группы: предметно-ориентированные [23–26] и предметно-независимые [29 – 38]. Предметно-ориентированные методы основаны на использовании предметных онтологий и таксономий общего назначения, описывающих различные понятия и факты. Они позволяют сопоставить естественно-языковое содержание таблицы с понятиями из таких онтологий или таксономий.

В одной из первых работ в данном направлении [23] предлагается метод интерпретации и каноникализации таблиц, основанный на использовании предметной онтологии, определяющей подъязык строительных спецификаций. В проекте TANGO [24] метод понимания таблиц основан на использовании фреймов данных, каждый из которых позволяет сопоставить определенный тип данных с табличными атрибутами и значениями, используя регулярные выражения, словари и открытые ресурсы, такие как WordNet [27]. В [25] предлагается встраивать такие фреймы данных в специальные онтологии, предназначенные для извлечения информации из таблиц. При этом предполагается, что фреймы позволяют связать содержание таблиц с объектами онтологии. Wang и др. [26] рассматривают проблему понимания веб-таблиц как ассоциирование их содержания с понятиями, представленными в таксономии общего назначения ProBase [28], покрывающей многие мировые понятия и факты.

Перечисленные методы [23–26] преимущественно используют предметные знания о естественно-языковом содержании таблиц. На практике этого не всегда достаточно, для более точного и полного извлечения информации из таблицы часто также требуется анализ пространственной и стилевой информации.

Предметно-независимые методы [29 – 38] вместо применения внешних знаний о предметной области используют анализ и интерпретацию пространственной, стилевой и содержательной информации из таблиц.

Метод Gatterbauer и др. [29] основан на анализе исключительно пространственной и стилевой информации в формате CSS2. При этом используются различные предположения о структуре и стиле нескольких наиболее общих типов HTML-таблиц. Также

Pivk и др. [30–32] предлагают методологию и систему TARTAR для автоматизации трансформации HTML-таблиц в структурированную форму (семантические фреймы). Методология TARTAR основана на эвристиках о структуре и содержании таблиц трех распространенных типов. Kim и др. [33] используют анализ пространственной, стилевой и естественно-языковой информации из веб-таблиц, основываясь на встроенных правилах и регулярных выражениях для пяти типов таблиц. В недавних работах Embley и Nagy [34,35] приводится метод трансформации веб-таблиц (HTML) в реляционную базу данных. Используя исключительно анализ структуры таблицы, они группируют атрибуты в категории без привязки к доменам реляционной базы данных или понятиям предметной области. Метод, построенный в [34, 35], основан на нескольких достаточно общих встроенных в алгоритмы предположениях о структуре сводных таблиц.

Упомянутые методы в основном ориентированы на задачи извлечения информации из таблиц в формате HTML, содержащихся в веб-страницах. В то же время большое количество неструктурированной табличной информации представлено в электронных таблицах. Chen и Cafarella [36 – 38] представляют предметно-независимый метод для извлечения реляционных данных из электронных таблиц как часть системы интеграции таких данных — Senbazuru. Их метод работает с широко распространенным типом таблиц, когда иерархические атрибуты располагаются слева и сверху от численных данных. Некоторые вопросы трансформации данных из электронных таблиц рассматриваются в работах [39, 40], посвященных проблеме обнаружения ошибок в табличных данных, и работе [41], фокусирующейся на нормализации данных.

Перечисленные предметно-независимые методы [29 – 38] основаны на использовании ограниченного набора предположений о структурах, стилях и содержании отдельных типов таблиц. Эти предположения встроены в предлагаемые ими алгоритмы и ограничивают классы эффективно обрабатываемых таблиц.

В отличие от них в работах [42, 43] предлагается трансформация данных из электронных таблиц в структурированную форму, основанная на правилах. При этом трансформационные правила разрабатываются на специализированном языке TranSheet. Они описывают отображения между исходной структурой таблиц и целевой схемой данных. Предлагаемый ими язык подходит для выполнения таких трансформаций во многих случаях. Однако описываемые ими отображения не допускают изменений в расположении исходных таблиц. Используя абсолютную адресацию электронных таблиц, они строго определяют соответствия между их ячейками и элементами целевой схемы.

Основные отличия предлагаемой нами методологии от известных аналогов состоят в следующем.

- Как и перечисленные методы, мы также используем предположения о структуре, стиле и содержании таблиц, но в отличие от них предлагаемый подход позволяет разделить предположения о таблицах на две части: базовые и специальные. Базовые предположения встроены в алгоритмы и структуры данных. Они выражены в предлагаемой модели таблицы, описанной в разд. 1. Специальные предположения программируются на языках CRL и DRL. Они собираются в отдельные наборы, предназначенные для обработки различных типов таблиц. В качестве фактов может использоваться как предметно-независимая информация о структуре, графическом оформлении и содержании таблиц, так и предметно-ориентированная, представленная в виде формальных описаний категорий на языке YAML.
- В работе показано, что CRL-правила могут описывать различные особенности таблиц, которые игнорируются большинством известных методов: смешанное распо-

ложение меток и вхождений (например, перерезы или чередование столбцов меток и вхождений); нечисловые значения данных; многозначные ячейки (когда несколько вхождений и меток располагаются одновременно в одной ячейке); иерархии меток, построенные с помощью отступов, шрифтов или знаков; сноски (примечания).

- По сравнению с языком TranSheet [42, 43] CRL-правила могут допускать изменчивость расположения ячеек в рамках одного типа таблиц за счет использования их пространственных, стилевых и содержательных характеристик без фиксации их абсолютных адресов.

Заключение

В данной работе предложен предметно-ориентированный язык CRL для представления правил анализа и интерпретации таблиц, основанный на конструкциях языка общего назначения DRL. Язык CRL скрывает несущественные для рассматриваемых задач детали DRL-конструкций и позволяет сфокусироваться на реализации логики правил анализа и интерпретации таблиц.

Представленная экспериментальная оценка показывает, что CRL-правила могут использоваться для преобразования произвольных таблиц к канонической форме с высокой полнотой и точностью. При этом один набор CRL-правил может описывать широкий тип (класс) таблиц. Также показано, что сужение типов таблиц может позволить упростить CRL-правила и привести к росту полноты и точности каноникализации таблиц в рамках суженного типа.

Наблюдения показывают, что произвольные таблицы могут содержать “грязные” данные. Дальнейшая работа может быть посвящена интеграции методов анализа и интерпретации таблиц с техниками очистки данных, а также исследованию и развитию специализированных алгоритмов обработки широко распространенных табличных особенностей, таких, как иерархии строчных меток в самом левом столбце.

Благодарности. Работа выполнена при финансовой поддержке РФФИ (грант № 15-37-20042) и Совета по грантам Президента РФ (стипендия СП-3387.2013.5).

Список литературы / References

[1] **Ferrucci, D., Lally, A.** UIMA: an architectural approach to unstructured information processing in the corporate research environment // Natural Language Engineering. 2004. Vol. 10, No. 3-4. P. 327–348

[2] **Feldman, R., Sanger, J.** The text mining handbook: Advanced approaches in analyzing unstructured data. Cambridge University Press, 2006. 423 p.

[3] **Inmon, W.H., Nesavich, A.** Tapping into unstructured data: Integrating unstructured data and textual analytics into business intelligence. Prentice Hall PTR, 2007.

[4] **Doan, A., Naughton, J.F., Ramakrishnan, R., Baid, A., Chai, X. Chen, F., Chen, T., Chu, E., DeRose, P., Gao, B., Gokhale, C., Huang, J., Shen, W., Vuong, B.-Q.** Information Extraction Challenges in Managing Unstructured Data // ACM SIGMOD Record. 2009. Vol. 37, No. 4. P. 14–20.

[5] Unstructured Information Management Architecture (UIMA) Version 1.0, OASIS. 2009. Available at: http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html

[6] **Hurst, M.** The interpretation of tables in texts: PhD thesis. UK, University of Edinburgh, 2000.

[7] **Hurst, M.** Layout and language: Challenges for table understanding on the web // Proc. of the 1st Intern. Workshop on Web Document Analysis. Seattle, WA, USA, 2001. P. 27–30.

[8] **Embley, D., Hurst, M., Lopresti, D., Nagy, G.** Table-processing paradigms: a research survey // Intern. J. on Document Analysis and Recognition. 2006. Vol. 8, No. 2. P. 66–86.

[9] **e Silva, A.C., Jorge, A.M., Torgo, L.** Design of an end-to-end method to extract information from tables // Intern. J. on Document Analysis and Recognition. 2006. Vol. 8, No. 2. P. 144–171.

[10] **Шигаров А.О.** Восстановление логической структуры таблиц из неструктурированных текстов на основе логического вывода // Вычисл. технологии. 2014. Т. 19, № 1. С. 87–99. **Shigarov, A.O.** Recovering the logical structure of tables from unstructured texts based on logical inference // Comput. Technologies. 2014. Vol. 19, No. 1. P. 87–99. (in Russ.)

[11] **Shigarov, A.** Table understanding using a rule engine // Expert Systems with Applications. 2015. Vol. 42, No. 2. P. 929–937.

[12] Drools Expert. Available at: http://www.drools.org

[13] **Wang, X.** Tabular abstraction, editing, and formatting: PhD thesis. Waterloo, Ontario, Canada, University of Waterloo, 1996.

[14] **Nagy, G.** Learning the characteristics of critical cells from web tables // Proc. of the 21st Intern. Conf. on Pattern Recognition. Tsukuba: IEEE Comp. Soc., 2012. P. 1554–1557.

[15] YAML. Available at: http://yaml.org

[16] Apache POI. Available at: https://poi.apache.org

[17] SnakeYAML. Available at: http://www.snakeyaml.org

[18] JavaBeans Specification 1.01 Final Release. 1997. Available at: http://www.oracle.com/technetwork/java/javase/tech/spec-136004.html

[19] JSR 94: Java Rule Engine API. Available at: https://jcp.org/en/jsr/detail?id=94

[20] JESS. Available at: http://www.jessrules.com

[21] RuleML. Available at: http://ruleml.org

[22] OpenRules. Available at: http://openrules.com/ruleengine.htm

[23] **Douglas, S., Hurst, M., Quinn, D.** Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text // Proc. of the 4th Annual Symposium on Document Analysis and Information Retrieval. Las Vegas, NV, US, 1995. P. 535–546.

[24] **Tijerino, Y., Embley, D., Lonsdale, D., Ding, Y., Nagy, G.** Towards ontology generation from tables // World Wide Web: Internet and Web Information Systems. 2005. Vol. 8, No. 3. P. 261–285.

[25] **Embley, D., Tao, C., Liddle, S.** Automating the extraction of data from HTML tables with unknown structure // Data & Knowledge Engineering. 2005. Vol. 54, No. 1. P. 3–28.

[26] **Wang, J., Wang, H., Wang, Z., Zhu, K.Q.** Understanding Tables on the Web // Proc. of the 31st Intern. Conf. on Conceptual Modeling. Florence, Italy: Springer-Verlag, 2012. P. 141–155.

[27] WordNet. Available at: http://wordnet.princeton.edu

[28] ProBase. Available at: http://research.microsoft.com/en-us/projects/probase

[29] **Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., Pollak, B.** Towards domain-independent information extraction from Web tables // Proc. of the 16th Intern. Conf. on World Wide Web. New York, US, 2007. P. 71–80.

[30] **Pivk, A., Cimiano, P., Sure, Y.** From tables to frames // Web Semantics. Science: Services and Agents on the World Wide Web. 2005. Vol. 3, No. 2-3. P. 132–146.

[31] **Pivk, A.** Thesis: Automatic ontology generation from Web tabular structures // AI Communications. 2006. Vol. 19, No. 1. P. 83–85.

[32] **Pivk, A., Cimiano, P., Sure, Y., Gams, M, Rajkovič, V., Studer, R.** Transforming arbitrary tables into logical form with TARTAR // Data & Knowledge Engineering. 2007. Vol. 60, No. 3. P. 567–595.

[33] **Kim, Y.-S., Lee, K.-H.** Extracting Logical Structures from HTML Tables // Computer Standards & Interfaces. 2008. Vol. 30, No. 5. P. 296–308.

[34] **Embley, D., Seth, S., Nagy, G.** Transforming Web tables to a relational database // Proc. of the 22nd Intern. Conf. on Pattern Recognition. Washington, DC, USA: IEEE Comp. Soc., 2014. P. 2781–2786.

[35] **Nagy, G., Embley, D., Seth, S.** End-to-End Conversion of HTML Tables for Populating a Relational Database // Proc. of the 11th IAPR Intern. Workshop on Document Analysis Systems. Tours-Loire Valley, France: IEEE Comp. Soc., 2014. P. 222–226.

[36] **Chen, Z., Cafarella, M.** Automatic spreadsheet data extraction // Third Intern. Workshop on Semantic Search Over the Web (SSW), 2013. Riva del Garda, Italy: ACM. 2013. Vol. 54, No. 2. P. 72–79.

[37] **Chen, Z., Cafarella, M., Chen, J., Prevo, D., Zhuang, J.** Senbazuru: A prototype spreadsheet database management system // Proc. of the Intern. J. on Very Large Data Bases Endowment. 2013. Vol. 6, No. 12. P. 1202–1205.

[38] **Chen, Z., Cafarella, M.** Integrating Spreadsheet Data via Accurate and Low-effort Extraction // Proc. of the 20th ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining. New York, USA: ACM, 2014. P. 1126–1135.

[39] **Abraham, R., Erwig, M.** UCheck: A spreadsheet type checker for end users // J. of Visual Languages & Computing. 2007. Vol. 18, No. 1. P. 71–95.

[40] **Chambers, C., Erwig, M.** Automatic detection of dimension errors in spreadsheets // J. of Visual Languages & Computing. 2009. Vol. 20, No. 4. P. 269–283.

[41] **Cunha, J., Saraiva, J., Visser, J.** From Spreadsheets to Relational Databases and Back // Proc. of the ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation. Savannah, GA, USA: ACM, 2009. P. 179–188.

[42] **Hung, V., Benatallah, B., Saint-Paul, R.** Spreadsheet-based Complex Data Transformation // Proc. of the 20th ACM Intern. Conf. on Information and Knowledge Management. Glasgow, Scotland, UK: ACM, 2011. P. 1749–1754.

[43] **Hung, V.** Spreadsheet-based complex data transformation: PhD thesis. Sydney, Australia, School of Computer Science and Engineering, University of New South Wales, 2011.

Поступила в редакцию 7 сентября 2015 г.,
с доработки — 9 октября 2015 г.

Table analysis and interpretation based on execution of CRL rules

Shigarov, Alexey O.*, Bychkov, Igor V., Paramonov, Viacheslav V., Belykh, Polina V.

Matrosov Institute for System Dynamics and Control Theory of SB RAS, 664033, Irkutsk, Russia

*Corresponding author: Shigarov, Alexey O., e-mail: shigarov@icc.ru

Often, arbitrary tagged tables presented in spreadsheets, word documents, and web pages are a source of important information that needs to be loaded into a relational database. However, many of them have a complex structure that does not allow populating databases with their information directly. The paper is devoted to the issues of the rule-based data extraction and transformation from arbitrary tagged tables into structured (canonical) form that provides loading data into a database by standard ETL tools. We suggest a novel rule language called CRL for table analysis and interpretation. It enables developing simple declarative programs to recover table semantics. Our methodology for rule-based table analysis and interpretation is mainly oriented on the tasks of unstructured tabular data integration. We expect it to be useful when a large number of arbitrary tagged tables appertaining to a few types are necessary to transform into structured form. The methodology is implemented in our prototype of system for Excel spreadsheet unstructured tabular data extraction and transformation. Our methodology and tools can be used to develop software for populating relational databases with tabular information contained in spreadsheets, word documents, and web pages. The experimental evaluation presented in the paper shows the effectiveness of applying CRL rules for table analysis and interpretation.

Keywords: unstructured tabular data integration, table analysis and interpretation, infromation extraction from tables, unstructured ETL, table understanding, spreadsheet data extraction.

Acknowledgements. This research was partly supported by RFBR (grant No. 15-37-20042) and the Council for grants of the President of the Russian Federation (grant No. SP-3387.2013.5).

Received 7 September 2015
Received in revised form 9 October 2015