CRL: язык правил анализа и интерпретации произвольных таблиц

© А. О. Шигаров © В. В. Парамонов
Институт динамики систем и теории управления им. В.М. Матросова СО РАН, Иркутск
shigarov@icc.ru slv@icc.ru

Аннотация

В работе обсуждаются вопросы трансформации произвольных таблиц, представленных в формате табличного процессора, к структурированному виду. Такие таблицы изначально не содержат семантической информации о представленных в них данных. Однако только после восстановления семантические отношений можно привести информацию из произвольной таблицы к канонической форме, которая может быть загружена в базу данных стандартными ETL средствами. В работе предлагается специальный язык правил анализа и интерпретации таких таблиц, позволяющий разрабатывать простые программы для восстановления отсутствующих семантических отношений. При этом процесс анализа и интерпретации таблиц состоит в исполнении этих правил. Отдельные CRL программы могут разрабатываться для различных наборов таблиц, обладающих схожей компоновкой.

1 Введение

Многие таблицы, представленные в формате табличного процессора (например, Excel), являются примером неструктурированных табличных данных. Такие таблицы, в отличие от «реляционных», не могут быть загружены напрямую в базу данных. Они требуют предварительной трансформации к структурированному виду.

Интеграция неструктурированных табличных данных может рассматриваться как ETL процесс, включающий следующие этапы: извлечение данных из произвольных таблиц, трансформацию их к структурированному виду и загрузку в базу данных. Этап трансформации табличных данных требует восстановления семантических отношений, т.е. пар вида: «вхождение-метка», «метка-метка» и «метка-категория» (Рис. 1). В литературе, посвященной пониманию таблиц, эти задачи принято называть анализом и интерпретацией таблиц [1, 5, 6, 14, 15].

В данной работе предлагается реализация этапа трансформации на основе исполнения правил анализа и интерпретации произвольных таблиц. Рассматривается оригинальный язык представления таких правил — CRL (Cells Rule Language).

Предлагаемый язык CRL реализует подход к анализу и интерпретации таблиц, описанный в работе [15]. Основная идея последнего заключается в том, что для различных классов таблиц могут быть разработаны отдельные наборы продукционных правил анализа и интерпретации.

Продукционные CRL правила позволяют восстанавливать отсутствующие семантические отношения по доступным фактам таким как: пространственное расположение ячеек, их стилевые характеристики и текстовое содержание.

CRL реализован в соответствии с требованиями свободной системы управления бизнес-правилами Drools [8] к предметно-ориентированным языкам. Это позволяет транслировать CRL правила в DRL (Drools Rule Language) [8] конструкции и исполнять их в системе Drools Expert [8].

Оставшаяся часть данной работы организована следующим образом. В разделе 2 обсуждаются известные методы анализа и интерпретации таблиц. Представление табличных данных в виде фактов при исполнении правил описывается в разделе 3. Ключевые конструкции и примеры использования языка CRL приводятся в разделе 4.

2 Родственные работы

Известные методы анализа и интерпретации таблиц можно разделить на две группы: предметно-ориентированные и предметно-независимые.

Предметно-ориентированные методы [3, 16, 17] основаны на использовании онтологий или баз знаний, описывающих предметные области. Такие методы позволяют сопоставить естественно-языковое содержание таблицы с понятиями некоторой предметной области.

Например, в проекте TANGO [16] метод понимания таблиц основан на использование библиотеки фреймов данных. Каждый фрейм позволяет сопоставить определенный тип данных с

Труды XVII Международной конференции DAMDID/RCDL'2015 «Аналитика и управление данными в областях с интенсивным использованием данных», Обнинск, 13-16 октября 2015

табличными атрибутами и данными, используя регулярные выражения, словари и открытые ресурсы (например, WordNet). Embley и др. [3] используют для анализа таблиц онтологии, включающие дополнительно наборы фреймов данных по аналогии с работой Tijerino и др. [16]. Фреймы позволяют связать содержание таблиц с объектами онтологии. Wang и др. [17] рассматривают проблему понимания веб таблиц как ассоциирование их содержания с понятиями, представленными в базе знаний PROBASE.

Перечисленные методы понимания таблиц [4, 17, 18] 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 [2, 4, 10-13], вместо применения внешних знаний о предметной области, основаны на анализе и интерпретации пространственной, стилевой и содержательной информацию из таблиц.

Например, метод Gatterbauer и др. [4] основан на анализе исключительно пространственной и стилевой информации в формате CSS2. При этом используется различные предположения о структуре и стиле нескольких наиболее общих типов HTML таблиц. Также Pivk и др. [12, 13] 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 3-х распространенных типов. Kim и др. [10] используют анализ пространственной, стилевой и естественно-языковой информации из веб таблиц, основываясь на встроенных правилах и регулярных выражениях для 5-и типов таблиц. В недавних работах Embley и Nagy [2, 11] приводится метод трансформации веб (HTML) таблиц 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 [2, 11] основан на нескольких достаточно общих встроенных в алгоритмы предположениях о структуре сводных таблиц.

Перечисленные предметно-независимые методы понимания таблиц основаны на использовании ограниченного набора предположений о структурах, стилях и содержании некоторых типов таблиц. Эти предположения встроены в предлагаемые ими алгоритмы и ограничивают классы таблиц, которые могут анализироваться и интерпретироваться данными методами с высокой точностью и полнотой.

По сравнению с известными методами, наш подход позволяет разделить предположения о таблицах на две части: базовые и программируемые. Базовые предположения встроены в алгоритмы и структуры данных, и общую модель таблиц. Они описаны в разделе 3. Программируемые предположения записываются на языке CRL или DRL. Они собираются в отдельные базы знаний, предназначенные для обработки различных классов (типов) таблиц.

Кроме того, язык CRL позволяет использовать как предметно-ориентированную информацию, представленную в виде формальных описаний категорий (доменов) на языке YAML, так и предметно-независимую о структуре, графическом оформлении, и содержании таблиц.

3 Представление табличных данных

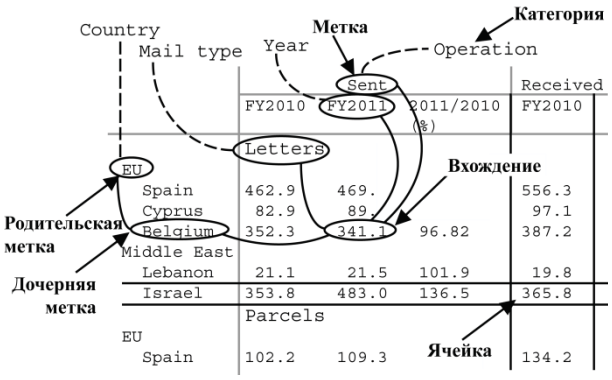


Рис. 1. Вхождения, метки, категории и их связи внутри таблицы.

При описании табличных данных мы опираемся терминологически на абстрактную модель таблицы, предлагаемую в работе [18], заимствуя термины: «вхождение», «метка» и «категория». Вхождение является значением данных, представленным в некоторой таблице. Каждое значение 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 (домену). 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. Рассматриваемые понятия иллюстрируются на Рис. 1.

В процессе исполнения CRL правил исходная и восстанавливаемая табличная информация должна быть представлена в виде фактов. В рабочей памяти системы исполнения CRL 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 таблицы. Каждый факт, представляющий ячейку, имеет следующие характеристики, доступные через соответствующие поля (выделенные здесь моноширинным шрифтом):

1. координаты: `cl` — левый и cr — правый столбец, `rt` — верхняя и `rb` — нижняя строка;
2. стилевые настройки `style`, основные из которых: `font` — шрифт, `horzAlignment` — горизонтальное и `vertAlignment` — вертикальное выравнивание, цвета `fgColor` — переднего и `bgColor` — заднего плана, типы и цвета четырех границ (`leftBorder` — левой, `topBorder` — верхней, `rightBorder` — верхней, и `bottomBorder` — верхней);
3. `text` — текстовое содержание, и `indent` — отступ (количество пробелов в начале текста);
4. упорядоченные наборы: `entries` — вхождений и `labels` — меток, порожденных из данной ячейки.

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

Метка (`label`) — структура, представляющая метки, включает следующие поля: `value` — строковое значение; если метка сгенерирована из ячейки, то `cell` — ссылка на неё; `category` — ссылка на категорию, которой принадлежит данная метка; в случае наличия иерархии меток, `parent` — соответствующая родительская метка, `children` — набор её дочерних меток.

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

Представленные структуры реализованы как Java классы в соответствии с соглашениями JavaBeans [8]. Это позволяет использовать их для представления табличных данных как фактов в любой системе исполнения правил, реализующей спецификацию «JSR 94: Java Rule Engine API» [10]. При этом каждый факт является экземпляром одного из этих классов.

4 CRL правила

CRL правила являются продукционными. Их левая часть содержит условия, накладываемые на известные факты, а правая — следствия, изменяющие состав таблицы и восстанавливающие семантические отношения её элементов.

В данной работе рассматриваются только ключевые конструкции CRL правил. Полная спецификация этого языка, определенная как набор отображений из CRL предложений в DRL конструкции, доступна по адресу: http://cells.icc.ru/pub/crl.

4.1 Условия

Условия записываются в левой части правил и позволяют выбирать из рабочей памяти ячейки, вхождения, метки и категории с заданными ограничениями. Условие обязательно включает одно из следующих ключевых слов: **`cell`**, **`entry`**, **`label`** или **`category`**, определяющих тип фактов соответственно: ячейки, вхождения, метки или категории, за которым 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, накладываемые на запрашиваемые факты. Они отделяются от обязательной части условия знаком двоеточия ‘:’. Любое ограничение является выражением, вычисление которого приводит к значению булевского типа данных («истина» или «ложь»). Ограничения следуют синтаксису языка выражения MVEL. Они отделяются друг от друга с помощью запятой ‘,’, которая интерпретируется как логическая конъюнкция. Синтаксис условий представлен ниже:

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

В демонстрируемых здесь и далее CRL конструкциях и примерах CRL правил, ключевые слова выделены полужирным шрифтом, а имена переменных курсивом.

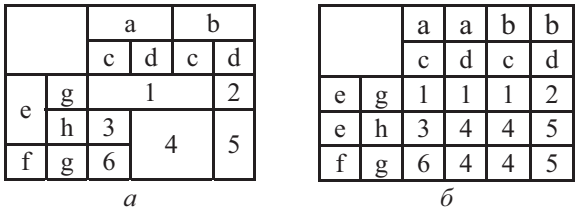
4.2 Следствия

Разделение ячеек. В основном, разделять объединенную ячейку необходимо в том случае, когда она содержит вхождения, имеющие одинаковое значение, но при этом связанные с разными метками. Восстановление таких связей на основе анализа декомпозиции таблицы может быть упрощено благодаря предварительному разделению подобных ячеек.

Данная операция позволяет разделить любую объединенную ячейку `$cell`, состоящую из *n* плиток, на *n* ячеек, каждая из которых полностью повторяет её содержание и стилевые характеристики:

```
split $cell
```

Координаты формируемой *i* ячейки вычисляются по *i* плитке исходной ячейки. Все ячейки, полученные в результате такого разделения, вносятся в рабочую память, в то время как, исходная объединенная ячейка удаляется из неё.



а *б*

Рис. 2. Объединенные (а) и разделенные (б) ячейки.

Следующее CRL правило позволяет разделить все непустые объединенные ячейки (Рис. 2, *а*). В результате таблицы будут иметь вид, показанный на Рис. 2, *б*:

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

Объединение ячеек. Некоторые таблицы могут содержать ошибочно разделённые ячейки. Серия пустых ячеек часто может неявно повторять содержание некоторой непустой ячейки. В таких случаях, можно восстановить объединённые ячейки для унификации структуры таблиц и расширения класса таблиц, описываемого одним набором CRL правил.

Выполнение данной операции состоит в объединении двух соседних ячеек *$cell1* и *$cell2*, с одной общей стороной, в одну:

```
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.

Маркировка ячеек. Данная операция привязывает к ячейке *$cell* некоторую отметку *@mark* — слово со стартовым символом ‘@’.

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

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. Например, если таблица состоит из таких регионов как «шапка», «боковик» и «тело», тогда ячейкам в зависимости от их расположения в этих регионах могут быть сопоставлены следующие отметки соответственно: `@head`, `@stub` и `@body`. Использование этих отметок в левых частях последующих правил позволяет заменить наборы ограничений, необходимые для определения соответствующих регионов ячеек.

Например, в таблицах подобных Рис. 2, ячейки целесообразно разделить на три части: данные (‘1’,...,‘6’), заголовки столбцов (‘a’,...,‘d’) и строк (‘e’,...,‘h’), в зависимости от их расположения относительно пустой ячейки в верхнем левом углу. Так, следующее CRL правило сопоставляет каждой ячейки *$c*, расположенной под пустым верхним левым углом *$corner*, отметку — слово `@RowHeading`, описывающее роль этой ячейки, как заголовок строки:

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

Создание вхождений и меток. Эти операции обеспечивают создание вхождений и меток на основе некоторой ячейки *$cell*. При этом в следующей форме значения вхождения и метки задаются аргументами — строковыми выражениями *entry_value* и *label_value* соответственно:

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

В том случае, когда значение создаваемого 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 взаимный доступ между ними в дальнейшем процессе исполнения правил.

Далее показаны примеры создания вхождений и меток из текстового содержания ячеек. Следующее CRL правило, демонстрирует создание вхождений. Часто, вхождения являются числами. В условии правила, показанного ниже, выбираются все ячейки, текст которых удовлетворяет регулярному выражению «\d+» (одна или несколько цифр). Для этого используется DRL оператор `matches`. В правой части для каждой из них порождается вхождение:

```
when cell $c : text matches "\\d+"
then new entry $c
```

C1	C2	C3
a = 1	b = 2	c = 3
d = 4	e = 5	f = 6
g = 7	h = 8	i = 9

а

	а a	б b
в с	1	2
г d	3	4

б

Рис. 3. Ячейки ниже первой строки содержат текст вида «ключ=значение», где «ключ» является меткой, а «значение» вхождением — (*а*); в двуязычной таблице заголовки дублируются: кириллические и латинские символы — (*б*).

В некоторых таблицах ячейка может одновременно содержать вхождение и метку. Например, в таблице на Рис. 3, *а*, каждая ячейка, расположенная ниже первой строки, содержит текст вида «ключ=значение». При этом предполагается, что «ключ» является меткой, а «значение» — связанным с ней вхождением. Правило для создания меток из части «ключ» и вхождений из части «значение» текста таких ячеек показано ниже:

```
when
  cell $cell : rt>1, $t : text
```

```
then
  new label left($t,'=') -> $cell
  new entry right($t,'=') -> $cell
```

Другим примером являются таблицы (Рис. 3, *б*), где информация дублируется на двух языках. Предполагая, что ячейка содержит два слова, одно на русском, а другое на английском, и оба являются метками, то можно использовать следующее CRL правило для создания соответствующих меток:

```
when
  cell $c : cl==1 || rt==1, !blank,
    $t : text
then
  new label extract($t, "[а-я]+") -> $c
  new label extract($t, "[a-z]+") -> $c
```

В правиле CRL функция `extract` извлекает из текста *$t* ячейки *$c* его часть, удовлетворяющую регулярному выражению, переданному во втором аргументе.

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*, определяющим имя категории:

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

Здесь, прежде всего, выполняется поиск данной категории по заданному имени в модели обрабатываемой таблицы. Если категория с таким именем существует, то она связывается с меткой *$label*. В противном случае, в модели создается новая категория с именем *category_name*, после чего она ассоциируется с адресатом *$label*.

A / B	a1	a2
b1	1	2
b2	3	4
b3	5	6

а

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

б

Рис. 4. Ячейка в верхнем левом углу содержит имена двух категорий: ‘A’ для меток, формируемых из ячеек шапки (‘a1’, ‘a2’), и ‘B’ для меток из боковика (‘b1’, ‘b2’, ‘b3’) — (*а*); иерархия меток, формируемая отступами в тексте ячеек боковика — (*б*).

Далее рассматривается пример связывания меток с некоторой категорией по её имени. В некоторых таблицах ячейки могут содержать имена категорий, например Рис. 4, *а*. Для таблиц подобных показанной на Рис. 4, *а*, следующее правило может быть использовано, чтобы сопоставить метки, формируемые из ячеек шапки, и категорию, сгенерированную из текста ячейки в левом верхнем углу:

```
when
  cell $corner : cl==1, rt==1, $t : text
  label $label : cell.cl > $corner.cr
then set category token($t, 0) -> $label
```

Здесь, CRL функция `token` возвращает первое слово из текста *$t*, взятого из ячейки в верхнем левом углу *$corner*. Результат используется в качестве имени категории, ассоциируемой с меткой *$label*.

Ассоциирование меток. Две метки могут быть ассоциированы друг с другом как родительский *$label1* и дочерний узел *$label2*, формируя иерархию (дерево) внутри некоторой категории:

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

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

Кроме того, такие связи формируют составные значения меток следующим образом. Если в дереве меток существует путь: *$label1*,...,*$labeln*, где *$label1* — его корень, то составное значение метки *$labeln* включает значения всех узлов в этом пути в порядке их вложенности, начиная с корня. Восстановление составных значений может быть полезным, в тех случаях, когда имеющее смысл значение одной метки требует дополнительного прочтения нескольких других меток.

Пример отношений между метками представлен на Рис. 4, *б*. Предполагается, что уровни вложенности заголовков строк в боковике, формируемые отступами от левого края в тексте, отражаются в иерархию меток. В таблице на Рис. 4, *б* каждый уровень вложенности добавляет отступ равный четырем пробелам. Следующее CRL правило на основе сделанных предположений строит деревья соответствующих меток:

```
when
  cell $c1 : cl==1, $l1 : label
  cell $c2 : cl==1, rt>$c1.rt,
    indent==$c1.indent+4, $l2 : label
  no cells : cl==1, rt>$c1.rt,
    rt<$c2.rt, indent==$c1.indent
then set parent label $l1 -> $l2
```

Группировка меток. В некоторых таблицах, можно восстановить, что нескольких меток принадлежат одной категории, используя только пространственные и стилевые характеристики ячеек, но, в тоже 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.

Описанная возможность реализуются в CRL с помощью группировки меток. Две метки *$label1* и *$label2* могут быть отнесены к одной группе следующим образом:

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

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

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

Те 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 её метки.

Например, предполагая, что в таблицах на Рис. 2, метки, сгенерированные из заголовков строк и расположенные в одном столбце, составляют одну категорию, можно записать следующее правило:

```
when
  cell@RowHeading $c1 : $l1 : label
  cell@RowHeading $c2 : c1==$c1.c1,
    cr==$c1.cr, $l2 : label
then group $l1 -> $l2
```

В результате будут созданы группы меток, определяющие анонимные категории, например, для таблиц на Рис. 2: {‘e’, ‘f’} и {‘g’, ‘h’}. Аналогичное правило можно записать для меток, порожденных из заголовков столбцов и при этом расположенных в одной строке.

Ассоциирование вхождений. Данная операция связывает вхождение *$entry* с меткой *$label*:

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

При этом выполняется проверка базового условия: вхождение может быть связано только с одной меткой в каждой категории. Нарушение этого предположения останавливает дальнейший процесс интерпретации таблиц.

С другой стороны, 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, а попытка ассоциировать вхождение *$entry* с ней, выполняется только после её 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 *$category*. Когда такой метки нет, то она создаётся внутри данной категории. Затем, найденная или созданная метка связывается с вхождением *$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 неё, 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 создаётся 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.

Далее приводятся примеры ассоциирования вхождений. Создание метки независимо от ячеек показано в следующем CRL правиле: каждое вхождение ассоциируется с заданной меткой “tons” из категории “unit”:

```
when entry $e
then add label "tons" from "unit" -> $e
```

	a	b
c	1	2*
d	3	4**
* u ** v		

а

	α 阿爾法	公測
γ 伽馬	1 一	2 二
δ 三角洲	3 三	4 四

б

Рис. 5. Таблица содержит сноски (‘u’ и ‘v’) — (*а*); каждая непустая ячейка содержит либо две метки, либо два вхождения — (*б*).

В таблице, показанной на Рис. 5, *а*, вхождения ‘2’ и ‘4’ связаны со сносками ‘u’ и ‘v’ соответственно через ссылки ‘*’ и ‘**’. Такие сноски могут интерпретироваться как метки из специальной категории. Следующее CRL правило может быть использовано для подобных таблиц, чтобы ассоциировать вхождения с метками, созданными из таких сносок, использую ссылки, представленные последовательностями символов ‘*’:

```
when
  cell $footer : rb==table.numOfRows,
    $fn : text
  entry $e : cell.text matches ".+\\*+",
    $ref : extract(cell.text, "\\*+")
then
  add label between($fn, $ref, '\n')
    from "Footnote" -> $e
```

В левой части этого правила, мы запрашиваем «подвал» таблицы `$footer`, самую нижнюю ячейку, с текстом `$fn`, и вхождения `$e`, связанные ячейки которых содержат текст, удовлетворяющий регулярному выражению (`.+\*+`). При этом ссылка на сноску (`\*+`) запоминается в переменной `$ref`. В правой части, CRL функция извлекает подстроку из текста сносок `$fn`, между вхождением `$ref` ссылки на сноску и символом перевода на новую строку (`\n`).

В некоторых таблицах ячейки могут содержать по нескольку меток или вхождений (Рис. 5, *б*). Как это было отмечено в разделе 3, каждый факт, представляющий ячейку, имеет два упорядоченных набора: один для хранения ссылок на метки, другой для ссылок на вхождения. Это позволяет получить доступ к меткам и вхождениям по их индексам в этих наборах. Например, следующее правило демонстрирует ассоциирование вхождений с метками по их индексам для двуязычных таблиц, подобных Рис. 5, *б*:

```
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]
```

В данном правиле предполагается, что в предшествующем процессе исполнения правил метки и вхождения на одном языке были созданы в ячейках первыми и соответственно имеют индекс 0, а на другом сгенерированы вторыми и их индекс 1.

Вспомогательные следствия. Дополнительно к описанным следствиям язык CRL предоставляет также ряд вспомогательных операций, таких как: изменение текстового содержания ячейки, значения вхождения или метки; обновление фактов в рабочей памяти; и вывода отладочной информации в командную строку.

5 Заключение

Таблицы от некоторого поставщика (издателя) часто следуют набору заданных требований. В результате они могут обладать достаточной схожестью структуры, стиля и содержания, чтобы разработать для их анализа и интерпретации отдельный набор продукционных правил. Во многих случаях, это позволяет формировать предположения о таблицах в виде наборов таких правил (баз знаний), вместо того чтобы встраивать их в алгоритмы обработки.

В работе [15] экспериментально показано, что правила анализа и интерпретации таблиц могут разрабатываться на языке DRL и исполнятся в системе Drools Expert. Данный язык имеет общее назначение для представления продукционных правил, но для правил анализа и интерпретации таблиц достаточно использовать только часть его возможностей. Исследование этих возможностей привело к созданию предметно-ориентированного языка CRL, основанного на DRL конструкциях. Язык CRL скрывает несущественные для рассматриваемых задач детали DRL конструкций и позволяет сфокусироваться на реализации логики правил анализа и интерпретации таблиц. При этом CRL правила могут транслироваться в формат DRL.

Предлагаемый нами подход ориентирован на использование в процессе интеграции неструктурированных табличных данных. Язык CRL может применяться для разработки программного обеспечения ETL на уровни трансформации табличной информации, содержащейся в электронных таблицах, документах текстового процессора, веб-страницах.

Дальнейшее исследование и развитие предлагаемого подхода требует разработки, и внедрения в языки правил функций обработки строк, очистки табличных данных, и анализа расположения ячеек.

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

Литература

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

[2] Embley D.W., Nagy G., Seth S. Transforming Web Tables to a Relational Database // In Proc. of the 22nd Int. Conf. on Pattern Recognition. Stockholm, Sweden. 2014.

[3] Embley D.W., Tao C., Liddle S.W. Automating the Extraction of Data from HTML Tables with Unknown Structure // Data & Knowledge Engineering. 2005. Vol. 54, No 1. pp. 3-28.

[4] Gatterbauer W., Bohunsky P., Herzog M., Krüpl B., Pollak B. Towards Domain-Independent Information Extraction from Web Tables // In Proc. of the 16th Int. Conf. on World Wide Web. New York, US. 2007. pp. 71-80.

[5] Hurst M. The Interpretation of Tables in Texts. PhD Thesis. UK, University of Edinburgh. 2000.

[6] Hurst M. Layout and language: Challenges for table understanding on the web. In Proc. of the first Int. Workshop on Web Document Analysis. 2001. pp. 27-30.

[7] JavaBeans Specification 1.01 Final Release, http://www.oracle.com/technetwork/java/javase/tech/spec-136004.html

[8] JBoss Drools, http://www.drools.org

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

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

[11] Nagy G., Embley D.W., Seth S. End-to-End Conversion of HTML Tables for Populating a Relational Database // In Proc. of the 11th IAPR Int. Workshop on Document Analysis Systems. IEEE. 2014. pp. 222-226.

[12] Pivk A., Cimiano P., Sure Y., Gams M., Rajkovic V., Studer R. Transforming Arbitrary Tables into Logical Form with TARTAR // Data & Knowledge Engineering. 2007. Vol. 60, No 3. pp. 567-595.

[13] Pivk A., Cimianob P., Sure Y. From Tables to Frames // Web Semantics: Science, Services and Agents on the World Wide Web. 2005. Vol. 3, No 2-3. pp. 132-146.

[14] e Silva A., Jorge A., Torgo L. Design of an end-to-end method to extract information from tables // International Journal on Document Analysis and Recognition. 2006. Vol. 8, No 2. pp. 144-171.

[15] Shigarov A. Table Understanding Using a Rule Engine // Expert Systems with Applications. 2015. Vol. 42, No 2. pp. 929-937.

[16] Tijerino Y.A., Embley D.W., Lonsdale D.W., Ding Y., Nagy G. Towards Ontology Generation from Tables // World Wide Web: Internet and Web Information Systems. 2005. Vol. 8, No 3. pp. 261-285.

[17] Wang J., Wang H., Wang Z., Zhu K.Q. Understanding Tables on the Web // In Proc. of the 31st Int. Conf. on Conceptual Modeling. Springer-Verlag. Florence, Italy. 2012. pp. 141-155.

[18] Wang X. Tabular Abstraction, Editing, and Formatting. PhD Thesis. University of Waterloo, Waterloo, Ontario, Canada. 1996.

CRL: A Rule Language for Analysis and Interpretation of Arbitrary Tables

Alexey O. Shigarov, Viacheslav V. Paramonov

The paper discusses issues of the transformation of information from arbitrary tables presented in spreadsheets into the structured form. These tables contain no relationships describing their semantics. However, only after the semantic relationships are recovered, the information from an arbitrary table can be loaded into a database by standard ETL tools. We suggest the CRL rule language for table analysis and interpretation. It allows developing a simple program to recover the missing semantic relationships. Particular sets of the rules can be developed for different types of tables to provide the transformation step in unstructured tabular data integration.