Rule-Based Canonicalization of Arbitrary Tables in Spreadsheets

Alexey O. Shigarov(✉), Viacheslav V. Paramonov, Polina V. Belykh, and Alexander I. Bondarev

Matrosov Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia
`shigarov@icc.ru`
`http://cells.icc.ru`

Abstract. Arbitrary tables presented in spreadsheets can be an important data source in business intelligence. However, many of them have complex layouts that hinder the process of extracting, transforming, and loading their data in a database. The paper is devoted to the issues of rule-based data transformation from arbitrary tables presented in spreadsheets to a structured canonical form that can be loaded into a database by regular ETL-tools. We propose a system for canonicalization of arbitrary tables presented in spreadsheets as an implementation of our methodology for rule-based table analysis and interpretation. It enables the execution of rules expressed in our specialized rule language called CRL to recover implicit relationships in a table. Our experimental results show that particular CRL-programs can be developed for different sets of tables with similar features to automate table canonicalization with high accuracy.

Keywords: Unstructured data integration · Table understanding · Table analysis and interpretation · Spreadsheet data transformation

1 Introduction

A large number of data is presented as arbitrary tables in spreadsheets. Many of them are unstructured data [1]. They lack explicit semantics required to be interpreted by computer programs. At the same time, arbitrary tables can be a valuable data source for business intelligence. To be accessible for data analysis and visualization their data need to be extracted, transformed, and loaded into databases. However, if they have complex layouts of cells then the use of familiar ETL-tools is often not enough to automatically populate a database with their data.

The paper presents the rule-based data transformation from arbitrary tables presented in spreadsheets (Fig. 1) to the canonical form (Fig. 2) that can be loaded into a database by standard ETL-tools (Fig. 3). It consists in table analysis and interpretation for recovering cell-role pairs where a role is defined as an

DOI: 10.1007/978-3-319-46254-7_7

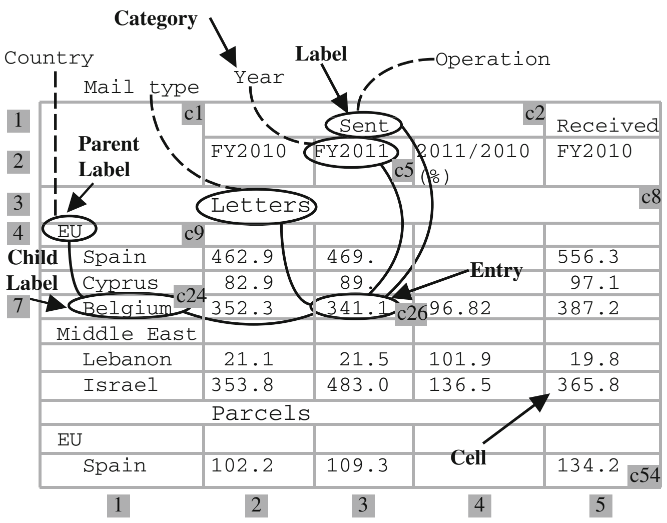


Fig. 1. An arbitrary table and its relationships

entry, label, or category, as well as entry-label, label-label, and label-category pairs.

Many methods for table analysis and interpretation were suggested in recent years. Some of them use extraction ontologies [11] or data frames [26] to bind natural language content of a table with their concepts. In a like manner Wang et al. [27] consider table understanding as associating a table with concepts in the general purpose knowledge taxonomy. Govindaraju et al. [14] combine table understanding and NLP techniques to extract relations from both text and tables. Astrakhantsev et al. [3] propose a method for data extraction from tables based on modeling user behavior. There are several methods [4,12] dealing with web-tables that fit in well with the relational form. The domain-independent methods [6,7,10,13,17–22] are based on the analysis and interpretation of spatial, style and textual information from tables instead of using external knowledge. These methods are generally designed for a few of widespread types of arbitrary tables.

The existing methods mostly deal with web-tables presented in HTML format. There are a few papers that are centered on tables presented in spreadsheets [2,5–8,10,16,19]. Erwig et al. [2,5] consider the issues of detecting errors in spreadsheets. Cunha et al. [8] focus on data normalization in spreadsheets. Hung et al. [15,16] propose the rule-based transformation of spreadsheet data into structured form using their original spreadsheet-like formula language called TranSheet. Chen and Cafarella [6,7] present a domain-independent method for extracting relational data from spreadsheets. Their method is designed for a widespread type of tables with a region of numeric values and two accompanied hierarchical regions of headings on the top and the left. The recent papers of

DATA	OPERA-TION	YEAR	MAIL TYPE	COUNTRY
462.9	Sent	2010	Letters	EU | Spain
82.9	Sent	2010	Letters	EU | Cyprus
...	...	...	...	...
12.3	Sent	2010	Parcels	Middle East | Lebanon
469.4	Sent	2011	Letters	EU | Spain
89.7	Sent	2011	Letters	EU | Cyprus
341.1	Sent	2011	Letters	EU | Belgium
21.5	Sent	2011	Letters	Middle East | Lebanon
...	...	...	...	...
556.3	Received	2010	Letters	EU | Spain
...	...	...	...	...
11.3	Received	2011	Parcels	Middle East | Lebanon

Fig. 2. A fragment of the table in the canonical form

Nagy, Embley, Seth and Krishnamoorthy [9,10,19,23] are devoted to the data transformation from web-tables converted into CSV files to a relational database.

We propose a system for canonicalization of arbitrary tables presented in spreadsheets as an implementation of our methodology for rule-based table analysis and interpretation [25]. It enables the execution of rules expressed in our specialized rule language called CRL [24] to recover missing relationships describing table semantics. CRL-rules map explicit features (layout, style, and text of cells) of an arbitrary table to its implicit semantic relationships. In contrast with the existing methods, our system is a tool for unstructured data integration. Instead of fixing one or more types of tables with typical structures, we propose to develop different declarative CRL-programs for particular sets of tables with similar features that can be both typical and specific. Our system supports relative cell addressing compared to the TranSheet [15,16]. In addition, we use the fixed canonical form instead of specifying a target schema.

2 Recovering Relationships in a Table

We use the terminology of the Wang's table model [28]: entries (data values), labels (headers), and categories (domains), Fig. 1. Our table model supports the following basic principles: (1) each cell can contain one or more entry, label and category values simultaneously, (2) an entry can be associated with only one label in each category, (3) each label must be associated with only one category, and (4) a category can be hierarchical. It is discussed more detail in [24,25].

Cells, entries, labels, and categories of a table are facts in the rule matching process. CRL-rules map explicit features of a table to its implicit relationships. The left hand side of a rule defines conditions using to query facts inserted into the working memory of a rule engine. Its right hand side contains actions that typically serve to generate new facts or to modify the existing facts.

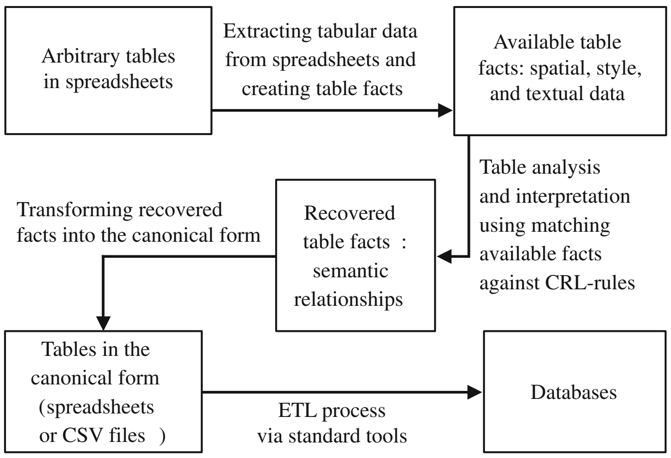

Arbitrary tables in spreadsheets
Extracting tabular data from spreadsheets and creating table facts
Available table facts: spatial, style, and textual data
Table analysis and interpretation using matching available facts against CRL-rules
Recovered table facts : semantic relationships
Transforming recovered facts into the canonical form
Tables in the canonical form (spreadsheets or CSV files)
ETL process via standard tools
Databases

Fig. 3. Unstructured ETL for arbitrary tables in spreadsheets using CRL-rules

Input facts are formed from Excel cells of an arbitrary table. Each cell fact corresponds to an Excel cell of a table with slight modifications. It specifies four positions (`cl` — left column, `rt` — top row, `cr` — right column, and `rb` — bottom row), style characteristics (`style`), including a font (`style.font`), horizontal (`style.horzAlignment`) and vertical (`style.vertAlignment`) alignment, text rotation (`style.rotation`), background (`style.bgColor`) and foreground (`style.fgColor`) colors, border types and colors (`style.leftBorder`, `style.topBorder`, `style.rightBorder`, and `style.bottomBorder`), a primitive data type (`type`), an indent (`indent`), and its textual content (`text`).

For example, the table shown in Fig. 1 consists of 54 cells. Thus, the 54 cell facts `<cells>={c1,c2,...,c54}` are inserted into the working memory of a rule engine, including:

```
c1=(cl=1,rt=1,cr=1,rb=2,text=null)
c2=(cl=2,rt=1,cr=4,rb=1,text="Sent")
c5=(cl=3,rt=2,cr=3,rb=2,text="FY2010")
c8=(cl=1,rt=3,cr=5,rb=3,style.font.bold=true, text="Letters")
c9=(cl=1,rt=4,cr=1,rb=4,text="EU")
c24=(cl=1,rt=7,cl=1,rb=7,indent=4,text="Belgium")
c26=(cl=3,rt=7,cl=3,rb=7,type=Type.NUMERIC,text="341.1")
```

Moreover, an input can include category facts. A category can be specified as a set of its labels and constraints (regular expressions defining ranges of permissible labels) in YAML[1] format. Our YAML schema defines the following fields: a category name, list of label values, and list of constraints. Note that the current version of our system supports specifying only plain categories. In the

[1] http://yaml.org.

example below, we specify the category `YEAR` with two labels `2010` and `2011` and two constraints on admissible label values `FY2000,...,FY2016`:

```
# category YEAR
name: Year
labels:
- 2010
- 2011
constraints:
- "FY200[0-9]"
- "FY201[0-6]"
```

For the table shown in Fig. 1 we can define and insert into the working memory four category facts:

```
d1=(name="OPERATION", labels={"Sent","Received"})
d2=(name="YEAR", constraints={"FY200[0-9]", "FY201[0-6]"})
d3=(name="MAIL_TYPE", labels={"Letters","Parcels"})
d4=(name="COUNTRY", labels={"Afghanistan",...,"Zimbabwe"})
```

All inserted facts are accessible through condition elements of a rule. A condition element enables to query facts of one of four types (`cell`, `entry`, `label`, or `category`) that satisfy constraints represented as Java-expressions. For example, the condition element `<cell $c : cl == 1>` queries all cells located in the leftmost column (e.g. `c1`, `c8`, `c9`, `c24`), the other `<cell $c : type==Type.NUMERIC>` returns all cells with the `NUMERIC` type (e.g. `c26`).

Typically, CRL-rules can be separated into two groups designed for two functionally different stages: the first for recovering cell-role pairs where a role is defined as an entry, label, or category, and second for recovering entry-label, label-label, and label-category pairs.

The first group includes actions for generating entry and label facts from cell facts. Usually, a value of a created fact is a text of a cell, but it can also be a part or modification of the text. The cell becomes the provenance for the generated entry or label. For example, we can create the following label facts from the cells `c2`, `c5`, `c8`, `c9`, and `c24` of the table shown in Fig. 1 respectively:

```
l1=(value="Sent", cell=c2), l2=(value="2011", cell=c5)
l3=(value="Letters", cell=c8), l4=(value="EU", cell=c9)
l5=(value="Belgium", cell=c24)
```

as well as the entry fact `e1=(value="341.1", cell=c26)` from the cell `c26`. Moreover, to simplify and generalize recovering relationships, rules of this stage often contain actions for splitting or merging cells, removing rows and columns, and modifying style and text content of cells.

The second group uses actions for associating entries, labels, and categories. A label can be associated with a category using either the corresponding category fact or specified name of the category. In the latter case, we look for the category by its name. If it exists, then the label is associated with it. Otherwise, we try to create it before categorizing. In our example (Fig. 1) we need to associate the labels with the categories as follows:

```
l1=(value="Sent", cell=c2, category=d1)
l2=(value="2011", cell=c5, category=d2)
l3=(value="Letters", cell=c8, category=d3)
l4=(value="EU", cell=c9, category=d4)
l5=(value="Belgium", cell=c24, category=d4)
```

Furthermore, two labels can be associated as parent and child. As the result, labels form trees to support hierarchical categories and compound values. All labels connected in a tree must be associated with the same category. In the case (Fig. 1) we relate the labels `l4` and `l5` as parent-child:

```
l4=(value="EU", cell=c9, category=d4, children={l5,...})
l5=(value="Belgium", cell=c24, category=d4, parent=l4)
```

It means that the compound value of the label `l5` is `"EU|Belgium"`.

In some cases, we can define that several labels relate to the same category, without knowing what the category is. For example, we may know that labels, which are located in the same row, relate to the same category. Instead of categorizing labels can be grouped in pairs. Each group of labels can be considered as an anonymous category. If one or more labels of a group are associated with a category then the remaining labels from the group also must be associated with the category. If no labels in a group are categorized then we create a category with the automatically generated name and associate all labels of this group with it.

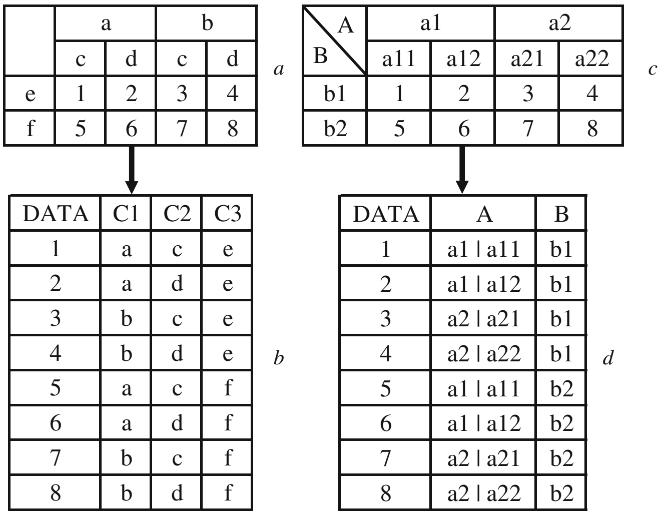
An entry can be associated with a label using either the corresponding label fact or a specified value of the label from a designated category. In the latter case, we try to find this category among all accessible categories. If this category does not exist then we create it. After that, we look for the label with the specified value in the founded or created category. When there is no label, we create it, using this value. At last, the entry is associated with the founded or created label. Note that, this allows to generate labels independently of cells. In the example (Fig. 1) the entry `e1` is associated with the labels `l1`, `l2`, `l3`, `l4`, and `l5` i. e.

```
e1=(value="341.1", cell=c26, labels={l1,l2,l3,l4,l5})
```

As is mentioned above, any entry can be associated with only one label from each category. Moreover, this means that the label must belong to a category. If the added label is uncategorized then it is not associated with the entry at that moment. At first, it becomes a candidate that may be associated automatically with the entry only after it is categorized. Finally, all labels, which are not categorized or grouped in rule firing, are associated with a default category after that.

3 Generating a Canonical Table

Recovered relationships of a table enable generating its canonical form. We can consider the canonical form as a relational table. Its topmost row contains field (attribute) names. Each of the remaining rows is a record (tuple). It obligatorily includes the field named DATA that contains entries. Each recovered category becomes a separate field that contains its labels. Any record presents recovered relationships between an entry and one label in each category. Usually each record is unique within a canonical table.



a

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

c

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

b

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

d

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

Fig. 4. Two arbitrary tables (*a*, *c*) and their canonical forms (*b*, *d*)

For example, as a result of analysis and interpretation of the table shown in Fig. 4, *a* the following relationships can be recovered:

```
<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)}
```

This example supposes that each group is an anonymous category. Thus, we generate three categories and associate them with the grouped labels as follows:

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

Another example is the source table demonstrated in Fig. 4, *c* where we assume that 'A' and 'B' are category names, and the boxhead headings constitute hierarchy.

```
<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)}
```

4 Implementation

We implement the proposed system for table canonicalization as shown in Fig. 5. First, we parse tabular data of a source spreadsheet in Excel format via "Apache POI"[2] library and generate cell facts from them. Optionally, categories specified in YAML format can also be loaded and presented as facts, using SnakeYAML[3] library for parsing YAML.

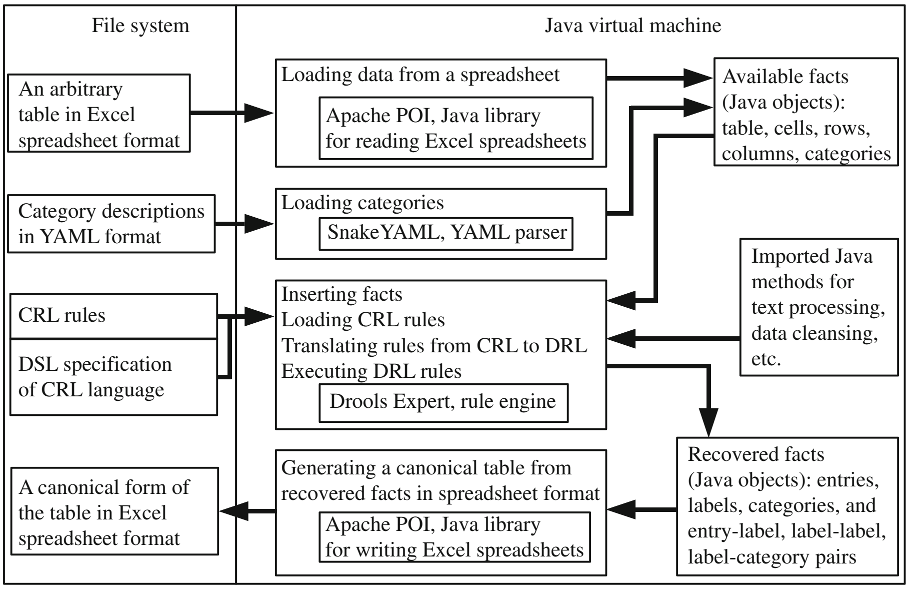

File system
Java virtual machine
An arbitrary table in Excel spreadsheet format
Category descriptions in YAML format
CRL rules
DSL specification of CRL language
A canonical form of the table in Excel spreadsheet format
Loading data from a spreadsheet
Apache POI, Java library for reading Excel spreadsheets
Loading categories
SnakeYAML, YAML parser
Inserting facts
Loading CRL rules
Translating rules from CRL to DRL
Executing DRL rules
Drools Expert, rule engine
Generating a canonical table from recovered facts in spreadsheet format
Apache POI, Java library for writing Excel spreadsheets
Available facts (Java objects): table, cells, rows, columns, categories
Imported Java methods for text processing, data cleansing, etc.
Recovered facts (Java objects): entries, labels, categories, and entry-label, label-label, label-category pairs

Fig. 5. Architecture of the system for rule-based canonicalization of arbitrary tables in spreadsheets

Our data structures for presenting table facts (cells, entries, labels, and categories) are Java classes developed in accordance with the naming conventions of

[2] http://poi.apache.org.
[3] http://snakeyaml.org.

JavaBeans[4] specification. This allows to use any rule engine implemented "JSR 94: Java Rule Engine API"[5] specification. Generated facts are instances of these classes. They are asserted in the working memory of the rule engine.

In the prototype we use Drools[6] rule engine. Rules for table analysis and interpretation can be expressed in either the Drools' native general-purpose DRL format or our domain-specific language, CRL. In case when loaded rules are presented in CRL they are translated to DRL format via Drools, using the DSL specification defining transformations from CRL to DRL constructs. Note that rules can use imported Java programs for text processing, data cleansing, or mathematical computations.

New facts (entries, labels, categories, and their relationships) are recovered in matching available facts against DRL-rules. Recovered facts provide generating canonical tables. The process ends with each canonical table being exported into Excel spreadsheet format, using "Apache POI" library.

The prototype of our system has been implemented as a command-line application[7] and web-service[8]. We have also developed a web-client for our web-service. It serves to transfer arbitrary tables in spreadsheets, rule sets, and category specifications to the web-service that returns corresponding canonical tables and their data provenance as a response. The client shows the canonical tables and links them with their original arbitrary forms via interactively highlighting connected source and target cells.

5 Experimental Evaluation

For experimental evaluation we use the existing dataset collected within TANGO[9] project [26]. The dataset intends to test table interpretation methods. It contains 200 arbitrary tables in Excel format imported from 10 websites with statistical data.

In the experiment, we evaluate table canonicalization where our purpose is to recover entries, labels, and entry-label, label-label relationships. Note that some cells of TANGO tables can be considered as multi-labeled, i. e. a cell includes more than one label. However, to simplify the evaluation process we admit that a cell can contain only one label. In our interpretation, we also recover hierarchical label-label relationships instead of dividing labels into groups and categorizing them. We do not process footnotes as part of a table, but a footnote reference remains a part of a recovered entry or label.

The most of TANGO tables satisfy the assumptions listed below:

– A table consists of four cell regions having different functions and separated by two invisible perpendicular lines as shown in Fig. 6, *a*: (1) each non-empty

[4] http://www.oracle.com/technetwork/java/javase/tech/spec-136004.html.
[5] http://jcp.org/en/jsr/detail?id=94.
[6] http://drools.org.
[7] Downloadable from https://github.com/shigarov/cells-ssdc.
[8] Accessible at http://cells.icc.ru:8080/ssdc.
[9] http://tango.byu.edu/data.

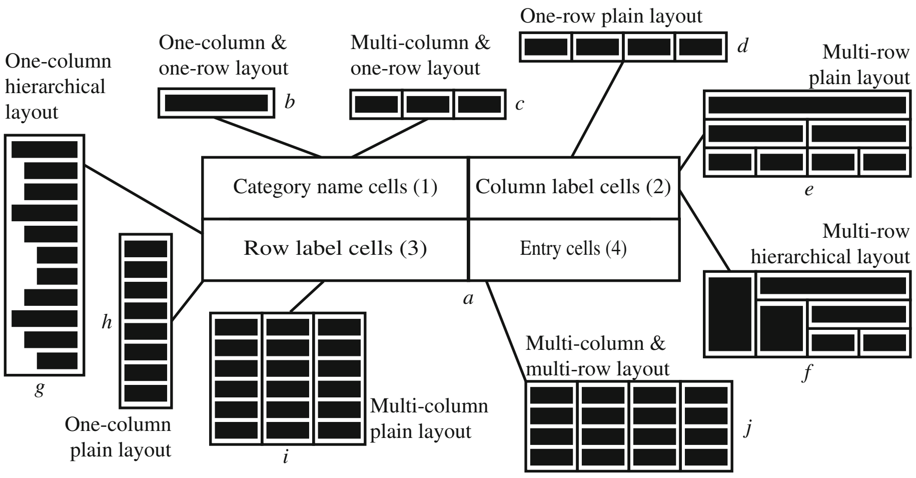

One-column hierarchical layout
One-column & one-row layout
b
Multi-column & one-row layout
c
One-row plain layout
d
Multi-row plain layout
e
Category name cells (1)
Column label cells (2)
Row label cells (3)
Entry cells (4)
a
Multi-row hierarchical layout
f
g
h
One-column plain layout
i
Multi-column plain layout
Multi-column & multi-row layout
j

Fig. 6. Cell regions (a) and their layouts (b–j) in tables of TANGO dataset

cell of the top-left region contains a category name; (2) each non-empty cell of the top-right region contains a column label; (3) each non-empty cell of the bottom-left region contains a row label; (4) each non-empty cell of the bottom-right region contains an entry.

– Each cell region has one of the appropriate layouts as shown in Fig. 6 and enumerated in Table 1.
– If a cell located in a column contains a category name then all row labels produced from the rest of cells in this column belong to this category.
– Column labels can form a hierarchy (Fig. 6, e and f). If in the top-right region a not empty cell c located on i-row spans several columns and not empty cells $c_1,\dots,c_n$ are located in these columns on $i+1$-row, then the cell c contains a parent label for labels produced from the cells $c_1,\dots,c_n$.
– Row labels located in the leftmost column can form a hierarchy (Fig. 6, g). Three typographical ways can denote the presence of a label hierarchy: (1) each level of label nesting appends one additional indent equaled two spaces; (2) hyphen char ('-') at the beginning of a label indicates that the label is nested; (3) text highlighted by the bold font can signalize spanning label.
– An entry is either a number or special word (e.g. '#', 'F', '...', 'x', 'NA').

These assumptions are formally presented as a set of 16 CRL-rules combined into the CRL-program called TANGO-200.

To evaluate table canonicalization we use the following measures:

$$Completeness = \frac{\text{number of completely recovered tables}}{\text{number of all tables}}$$

where a table is completely recovered, when all entries, labels, entry-label pairs, and label-label pairs which are contained in its source form are included in its

Table 1. Using cell region layouts in TANGO tables

Region	Layout	Fig. 6	Cases
(1) Category name cells	One-column one-row	*b*	94.5 %
	Multi-column one-row	*c*	5.5 %
(2) Column label cells	One-row plain	*d*	65.5 %
	Multi-row plain	*e*	26 %
	Multi-row hierarchical	*f*	8.5 %
(3) Row label cells	One-column hierarchical	*g*	47.5 %
	One-column plain	*h*	47 %
	Multi-column plain	*i*	5.5 %
(4) Entry cells	Multi-column multi-row	*j*	100 %

canonical form.

$$Preciseness = \frac{\text{number of precisely recovered tables}}{\text{number of all tables}}$$

where a table is precisely recovered, when all entries, labels, entry-label pairs, and label-label pairs which are included in its canonical form are contained in its source form.

The evaluation is carried out as follows. Two experts independently compare the source arbitrary tables with their canonical forms generated automatically. They referee that each table is processed successfully or not in terms of the completeness and preciseness. In case when two experts make opposite decisions on a table then the third expert makes a final decision. The experimental results are presented in Table 2.

The completeness is 87 % and preciseness is 88.5 % for TANGO-200 program. Among 200 tables of the dataset, total 33 are processed with errors, which reduce the completeness and preciseness. There are various causes of these errors, including the following: (1) ambiguity of text highlighting, e.g. the use of the bold font to emphasize a hierarchy of row labels or data aggregation, (2) absence of spatial or style features, when relationships are expressed by natural language only, and (3) messy layout and data of cells, e.g. two parts of a label can improperly be placed in two different cells. For the most part, errors appear in recovering label-label relationships between row labels located in the leftmost column with one-column hierarchical layout (Fig. 6, *g*), 28 of 33 cases, 85 %.

Additionally, we evaluate our system for a subset of the dataset where we exclude all tables having one-column hierarchical layout of the bottom-left region (Fig. 6, *g*). Only 105 tables of the dataset belong to the subset. To process them, we use a subset of TANGO-200 rules where we remove 3 rules intended for recovering label-label relationships between row labels located in the leftmost column. The subset of 13 CRL-rules is combined into TANGO-SUB program. The experimental results for this case is also presented in Table 2.

Table 2. Experimental results

CRL-program	TANGO-200	TANGO-SUB
Total tables	200	105
Completely recovered tables	174	100
Precisely recovered tables	177	100
Completeness	87 %	95.2 %
Preciseness	88.5 %	95.2 %

The evaluation shows that particular CRL-programs can be developed for different sets of tables with similar features to automate table canonicalization with high accuracy. One set of rules can be suitable for processing a wide range of arbitrary tables. Furthermore, our experiment demonstrates that narrowing a table type, like from TANGO-200 to TANGO-SUB, can cause simplifying rules and increasing the completeness and preciseness in table canonicalization. The experimental results, including the generated canonical forms and the rule sets, are presented in more detail at address http://cells.icc.ru/pub/crl.

6 Conclusions and Further Work

The presented tools, system and rule language, are mainly designed for data integration. They can be applied to develop specialized ETL software where tagged arbitrary tables in spreadsheets, word documents, or web pages can be used as data sources. We expect that they are useful in business intelligence applications when data from a large number of arbitrary tables with similar features of their layout, style and text are required for populating a database.

We observe that arbitrary tables presented in spreadsheets can contain messy (not standardized) data. It seems to be interesting for the further work to integrate our tools with data cleansing techniques. The use of domain ontologies and global taxonomies (e.g. FreeBase, DBpedia, YAGO) is of special interest for recovering categories. In addition, the further work can be focused on developing table analysis and interpretation methods for widely used features, e.g. for automatically recovering a row label hierarchy in the leftmost column. We believe this can improve expressiveness and usefulness of our tools.

Acknowledgements. We thank Prof. George Nagy and all members of TANGO research group(http://tango.byu.edu) for providing and discussing the TANGO dataset for our experiments.

This work was financially supported by the Russian Foundation for Basic Research (Grant No. 15-37-20042 and 14-07-00166) and Council for Grants of the President of Russian Federation (Grant No. NSh-8081.2016.9). The presented web-service for table canonicalization is performed on resources of the Shared Equipment Center of Integrated information and computing network of Irkutsk Research and Educational Complex(http://net.icc.ru).

References

1. Unstructured information management architecture (UIMA) version 1.0 (2009). http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html
2. Abraham, R., Erwig, M.: UCheck: A spreadsheet type checker for end users. J. Vis. Lang. Comput. **18**(1), 71–95 (2007)
3. Astrakhantsev, N., Turdakov, D., Vassilieva, N.: Semi-automatic data extraction from tables. In: Selected Papers of the 15th All-Russian Scientific Conference on Digital Libraries: Advanced Methods and Technologies, Digital Collections, pp. 14–20 (2013)
4. Cafarella, M.J., Halevy, A., Wang, D.Z., Wu, E., Zhang, Y.: WebTables: Exploring the power of tables on the web. Proc. VLDB Endow. **1**(1), 538–549 (2008)
5. Chambers, C., Erwig, M.: Automatic detection of dimension errors in spreadsheets. J. Vis. Lang. Comput. **20**(4), 269–283 (2009)
6. Chen, Z., Cafarella, M.: Automatic web spreadsheet data extraction. In: Proceedings 3rd International Workshop on Semantic Search Over the Web, pp. 1: 1–1: 8. ACM, New York, NY, USA (2013)
7. Chen, Z., Cafarella, M.: Lntegrating spreadsheet data via accurate and low-effort extraction. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1126–1135. ACM, New York, NY, USA (2014)
8. Cunha, J., Saraiva, J.A., Visser, J.: From spreadsheets to relational databases and back. In: Proceedings ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp. 179–188. ACM, New York, PEPM 2009, NY, USA (2009)
9. Embley, D.W., Krishnamoorthy, M.S., Nagy, G., Seth, S.: Converting heterogeneous statistical tables on the web to searchable databases. Int. J. Doc. Anal. Recogn. **19**, 1–20 (2016)
10. Embley, D.W., Seth, S., Nagy, G.: Transforming web tables to a relational database. In: Proceedings 22nd International Conference on Pattern Recognition, pp. 2781–2786. ICPR 2014, IEEE Comp. Soc., Washington, DC, USA (2014)
11. Embley, D., Tao, C., Liddle, S.: Automating the extraction of data from HTML tables with unknown structure. Data Knowl. Eng. **54**(1), 3–28 (2005)
12. Galkin, M., Mouromtsev, D., Auer, S.: Identifying web tables: Supporting a neglected type of content on the web. In: Proceedings of the 6th International Conference Knowledge Engineering and Semantic Web, Moscow, Russia. Communications in Computer and Information Science, vol. 518, pp. 48–62 (2015)
13. Gatterbauer, W., Bohunsky, P., Herzog, M., Krpl, B., Pollak, B.: Towards domain-independent information extraction from web tables. In: Proceedings 16th International Conference on World Wide Web, pp. 71–80. New York, US (2007)
14. Govindaraju, V., Zhang, C., Ré, C.: Understanding tables in context using standard NLP toolkits. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL. vol. 2: Short Papers, pp. 658–664 (2013)
15. Hung, V.: Spreadsheet-Based Complex Data Transformation. Ph.D. thesis, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia (2011)
16. Hung, V., Benatallah, B., Saint-Paul, R.: Spreadsheet-based complex data transformation. In: Proceedings 20th ACM International Conference on Information and Knowledge Management, pp. 1749–1754. ACM, New York, CIKM 2011, NY, USA (2011)

17. Kim, Y.S., Lee, K.H.: Extracting logical structures from html tables. Comput. Stand. Interfaces **30**(5), 296–308 (2008)
18. Kudinov, P.Y.: Extracting statistics indicators from tables of basic structure. Pattern Recogn. Image Anal. **21**(4), 630–636 (2011)
19. Nagy, G., Embley, D., Seth, S.: End-to-end conversion of html tables for populating a relational database. In: Proceedings 11th IAPR International Workshop on Document Analysis Systems, pp. 222–226. IEEE Computer Society, Tours Loire Valley, France, April 2014
20. Pivk, A., Cimiano, P., Sure, Y.: From tables to frames. Web Semant. **3**(2–3), 132–146 (2005)
21. Pivk, A.: Thesis: Automatic ontology generation from web tabular structures. AI Commun. **19**(1), 83–85 (2006)
22. Pivk, A., Cimiano, P., Sure, Y., Gams, M., Rajkovič, V., Studer, R.: Transforming arbitrary tables into logical form with TARTAR. Data Knowl. Eng. **60**(3), 567–595 (2007)
23. Seth, S., Nagy, G.: Segmenting tables via indexing of value cells by table headers. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 887–891, August 2013
24. Shigarov, A.: Rule-based table analysis and interpretation. In: Proceedings of the 21st International Conference on Information and Software Technologies. Communications in Computer and Information Science, vol. 538, pp. 175–186 (2015)
25. Shigarov, A.: Table understanding using a rule engine. Expert Syst. Appl. **42**(2), 929–937 (2015)
26. Tijerino, Y., Embley, D., Lonsdale, D., Ding, Y., Nagy, G.: Towards ontology generation from tables. World Wide Web: Int. Web Inf. Syst. **8**(3), 261–285 (2005)
27. Wang, J., Wang, H., Wang, Z., Zhu, K.Q.: Understanding tables on the web. In: Johannesson, P., Lee, M.L., Liddle, S.W., Opdahl, A.L., López, Ó.P. (eds.) ER 2015. LNCS, vol. 9381, pp. 141–155. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34002-4_11
28. Wang, X.: Tabular Abstraction, Editing, and Formatting. Ph.D. thesis, University of Waterloo, Waterloo, Ontario, Canada (1996)