TabbyXL: Software Platform for Rule-Based Spreadsheet Data Extraction and Transformation*

Alexey Shigarov Vasiliy Khristyuk
Andrey Mikhailov Viacheslav Paramonov

Matrosov Institute for System Dynamics and Control Theory,
Siberian Branch of the Russian Academy of Sciences

shigarov@icc.ru

October 2019

* This work was supported by the Russian Science Foundation, grant number 18-71-10001

Table of Contents

Motivation

- About arbitrary spreadsheet tables
  - A large volume of valuable data for science and business applications
  - A big variety of layout, style, and content features
  - Human-centeredness (incorrect structure and messy content)
  - No explicit semantics for interpretation by computers
- Challenges
  - How to extract tables from worksheets
  - How to recognize and correct cell structure anomalies
  - How to recover semantics needed for the automatic interpretation
  - How to conceptualize extracted data by using external vocabularies

Background

Table understanding [Hurst, 2001] includes the following tasks

1. **Extraction** — detecting a table and recognizing the physical structure of its cells
2. **Role analysis** — extracting functional data items from cell content
3. **Structural analysis** — recovering internal relationships between extracted functional data items
4. **Interpretation** — linking extracted functional data items with external vocabularies (general-purpose or domain-specific ontologies)

Background

The related issues of the *table analysis and interpretation*

- **Layout properties** [Koci et al., 2017, Chen et al., 2017, Dou et al., 2018]
- **Code smells and formulas**
  [Hermans et al., 2015, Dou et al., 2017, Barowy et al., 2018, Koch et al., 2019]
- **Programming by examples**
  [Barowy et al., 2015, Singh and Gulwani, 2016, Jin et al., 2017]
- **Data model inference**
  [Amalfitano et al., 2015, Cunha et al., 2015, Cunha et al., 2016]
- **Linked Open Data** [Ritze and Bizer, 2017, Zhang, 2017]
- **Domain-specific models**
  [de Vos et al., 2017, Cao et al., 2017, Swidan and Hermans, 2017]
- **Rule-based programming**
  [Yang et al., 2017, Shigarov and Mikhailov, 2017, Yang et al., 2018]

Background

The projects with goals similar to ours

1. **TANGO**[1] (Data Extraction Group, Brigham Young University) *2005 – 2016*
   - Heuristics-based role analysis (pre-defined functional cell regions) [Embley et al., 2016]
2. **Senbazuru**[2] (Database Research Group, University of Michigan) *2013 – 2017*
   - ML-based role analysis (pre-defined functional cell regions) [Chen, 2016]
   - ML-based structural analysis (pre-defined layout properties of the header hierarchy) [Chen, 2016]
3. **DeExcelerator**[3] (Dresden Database Systems Group, TU Dresden) *2013 – 2019*
   - ML-based layout identification [Koci et al., 2016]
   - Heuristics-based role and structural analysis (pre-defined functional cell regions) [Koci et al., 2017, Koci et al., 2018]

[1] `https://tango.byu.edu`
[2] `http://dbgroup.eecs.umich.edu/project/sheets`
[3] `https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator`

Contribution

TabbyXL is a software platform aiming at the development and execution of rule-based programs for spreadsheet data extraction and transformation from arbitrary (*a*) to relational tables (*b*)

Novelty

- Table object model assigning roles to data items, not cell
- CRL, domain-specific language to express user-defined rules for table analysis and interpretation
- CRL-to-Java translator to synthesize executable programs for spreadsheet data transformation

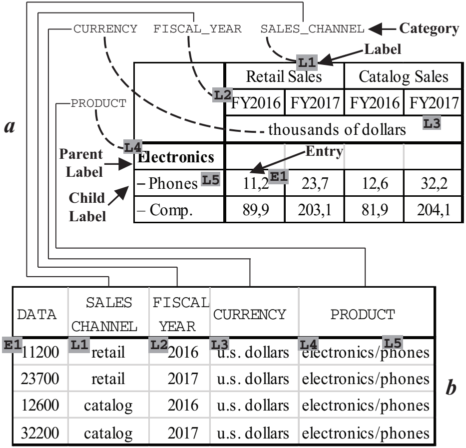


Table of Contents

User-Defined Rules

- The user-defined rules map the physical structure into the logical structure of a table
  - **WHEN-part** queries facts about the structure by using constraints
  - **THEN-part** modifies available facts and asserted new ones
- The facts are represented by items of the *table object model*
- The rules can be expressed in a rule-based language (e.g. Drools[4], Jess[5], or CRL[6])

[4] https://www.drools.org
[5] https://jessrules.com
[6] https://github.com/tabbydoc/tabbyxl/wiki/crl-language

Table Object Model

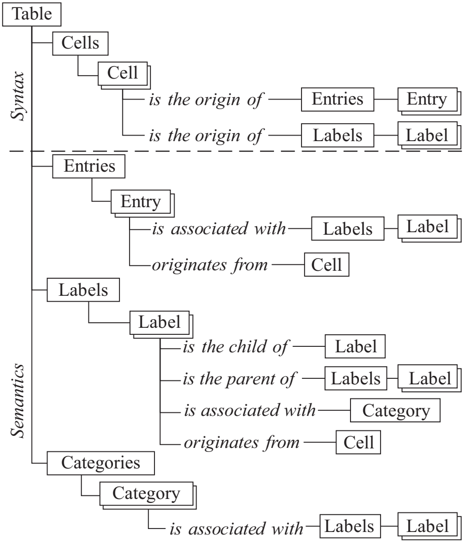
Physical Layer

Cells characterized by layout, style, and content features

Logical Layer

Functional data items and their relationships:

- entries (values)
- labels (keys)
- categories (concepts)
- entry-label pairs
- label-label pairs
- label-category pairs



CRL Grammar

```
rule        = 'rule' <a Java integer literal> 'when' condition
              'then' action 'end' <EOL> {rule} <EOF>
condition   = query identifier [':' constraint {',' constraint}
              [',' assignment {',' assignment}]] <EOL> {condition}
constraint  = <a Java boolean expr>
assignment  = identifier ':' <a valid Java expr>
query       = 'cell' | 'entry' | 'label' | 'category' | 'no cells' |
              'no entries' | 'no labels' | 'no categories'
action      = merge | split | set text | set indent | set mark |
              new entry | new label | add label | set parent |
              set category | group <EOL> {action}
merge       = 'merge' identifier 'with' identifier
split       = 'split' identifier
set text    = 'set text' <a Java string expr> 'to' identifier
set indent  = 'set indent' <a Java integer expr> 'to' identifier
set mark    = 'set mark' <a Java string expr> 'to' identifier
new entry   = 'new entry' identifier ['as' <a Java string expr>]
new label   = 'new label' identifier ['as' <a Java string expr>]
add label   = 'add label' identifier | (<a Java string expr>
              'of' identifier | <a Java string expr>)
              'to' identifier
set parent   = 'set parent' identifier 'to' identifier
set category = 'set category' identifier | <a Java string expr>
               'to' identifier
group        = 'group' identifier 'with' identifier
identifier   = <a Java identifier>
```

Cell Cleansing

The actions correct an inaccurate layout and content of a hand-coded table

- <merge> combines two adjacent cells when they share one border
- <split> divides a merged cell that spans *n*-tiles (row-column intersections) into *n*-cells
- <set text> modifies a textual content of a cell
- <set indent> modifies a text indentation of a cell

Example

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

Role Analysis

The actions recover entries and labels as functional data items presented in a table

- <set mark> annotates a cell with a user-defined tag that can be used in subsequent table analysis
- <new entry> (<new label>) creates an entry (label) from a cell content with the use of an optional string processing

Example

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

Structural Analysis

The actions recover pairs of two kinds: entry-label and label-label

- <add label> associates an entry with a label
- <set parent> binds two labels as a parent and its child

Example

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

Interpretation

The actions serve to recover label-category pairs

- <set category> associates a label with a category
- <group> places two labels to one group that can be considered as an undefined category

Example

```
when
  label l1: cell.mark == "stub"
  label l2: cell.mark == "stub", cell.rt == l1.cell.rt
then
  group l1 with l2
```

Illustrative Example

The transformation of arbitrary tables with the same layout features (*a* and *c*) to their canonicalized versions (*b* and *d*)

a1		a2	
b1	1	b4	4
b2	2	b5	NA
b3		b6	6

a

DATA	A	B
1	a1	b1
2	a1	b2
4	a2	b4
6	a2	b6

b

a1		a2		a3	
b1		b3	3	b5	5
b2	2	b4	NA	b6	6

c

DATA	A	B
2	a1	b2
3	a2	b3
5	a3	b5
6	a3	b6

d

The ruleset for the cell cleansing (*a*), role analysis (*b*, *c*), structural analysis (*d*, *e*), and interpretation (*f*, *g*)

a

```
when cell c: c.text.matches("NA")
then set text "" to c
```

b

```
when cell c: (cl % 2) == 0, !blank
then new entry c
```

c

```
when cell c: (cl % 2) == 1
then new label c
```

d

```
when
   entry e
   label l: cell.cr == e.cell.cr
then add label l to e
```

e

```
when
   entry e
   label l: cell.rt == e.cell.rt, cell.cl == e.cell.cl - 1
then add label l to e
```

f

```
when label l: cell.rt == 1
then set category "A" to l
```

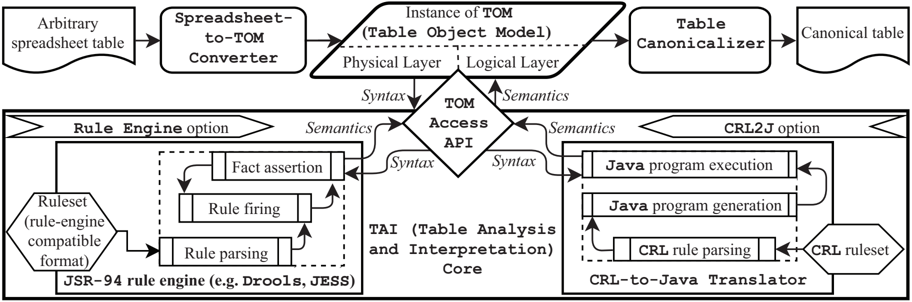
g

```
when label l: cell.rt > 1
then set category "B" to l
```

This example is reproducible at `https://codeocean.com/capsule/5326436`

Table of Contents

Architecture



Two options are provided

Rule Engine option

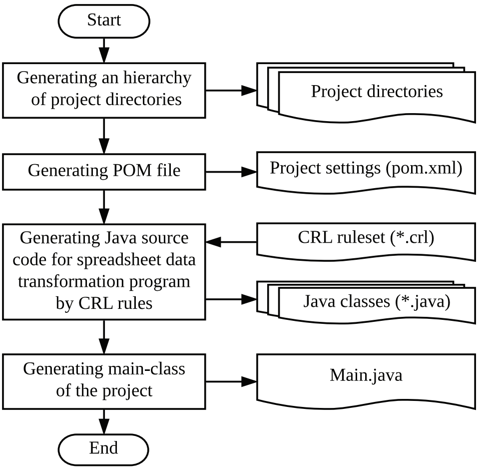
Executing a ruleset in an appropriate format with a JSR-94 compatible rule engine (e.g. Drools, Jess)

CRL2J option

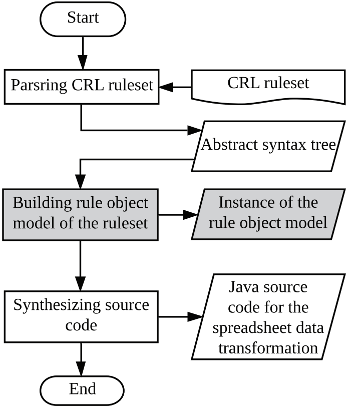
Translating a ruleset expressed in CRL to an executable Java program

CRL2J Translation

Workflow for generating a Maven-project of a spreadsheet data transformation program

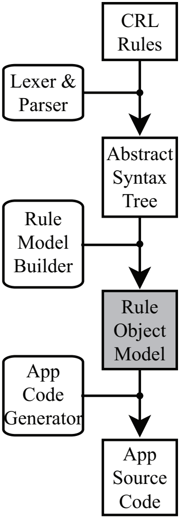


Workflow for translating a CRL ruleset to Java source code

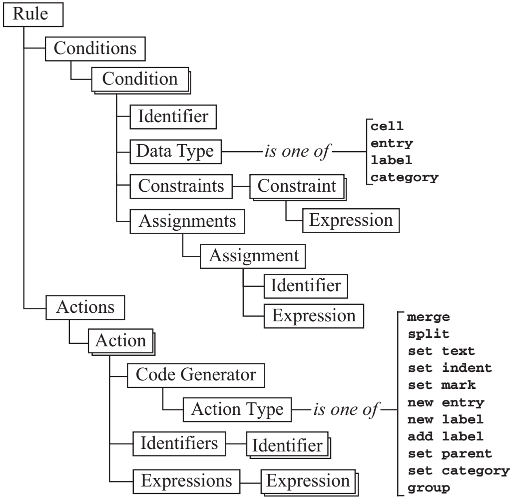


CRL2J Translation

In the Workflow



Rule Object Model



CRL2J Translation

Example (Source Rule)

```
when
  cell corner: cl == 1, rt == 1, blank
  cell c: cl > corner.cr, rt > corner.rb, ! marked
then
  set mark "@entry" to c
  new entry c
```

Example (Fragment of the Generated Java Code)

```
...
Iterator<CCell> iterator1 = getTable().getCells();
while (iterator1.hasNext()) {
  corner = iterator1.next();
  if ((corner.getCl() == 1) && (corner.getRt() == 1) && ...
    Iterator<CCell> iterator2 = getTable().getCells();
    while (iterator2.hasNext()) {
...
```

Table of Contents

Performance Evaluation

The results of the transformation of 200 tables of Troy200 dataset [Nagy, 2016]

Metrics	Role analysis		Structural analysis	
	Type of instances			
	entries	labels	entry-label pairs	label-label pairs
Recall	0.9813 $\frac{16602}{16918}$	0.9965 $\frac{4842}{4859}$	0.9773 $\frac{34270}{35066}$	0.9389 $\frac{1951}{2078}$
Precision	0.9996 $\frac{16602}{16609}$	0.9364 $\frac{4842}{5171}$	0.9965 $\frac{34270}{34389}$	0.9784 $\frac{1951}{1994}$
F-score	0.9904	0.9655	0.9868	0.9582

Metrics

$$\text{recall} = {}^{|R \cap S|}/_{|S|} \quad \text{precision} = {}^{|R \cap S|}/_{|R|}$$

S is a set of instances in a source table, *R* is a set of instances in its canonical form

All data and steps to reproduce the results are available at `http://dx.doi.org/10.17632/ydcr7mcrtp.5`

Performance Evaluation

The comparison of the running time by using TabbyXL with three different options for transforming 200 tables of Troy200 dataset [Nagy, 2016]

Running time of	CRL2J	Drools	Jess
Ruleset preparation (t_1)	2108* ms	$1711^{\dagger}$ ms	$432^{\dagger}$ ms
Ruleset execution (t_2)	367** ms	$1974^{\ddagger}$ ms	$4149^{\ddagger}$ ms

* t_1 — a time of parsing and compiling the original ruleset into a Java program
** t_2 — a time of executing the generated Java program

† t_1 — a time of parsing the original ruleset and adding the result into a rule engine session
‡ t_2 — a time of asserting facts into the working memory and matching rules against the facts

For testing, we used 3.2 GHz 4-core CPU

Comparison with Others

Role Analysis

- *Contest task*: The segmentation of a table into typical functional cell regions
- *Testing dataset*: Troy200 [Nagy, 2016]
- *Contestant*: MIPS (TANGO) [Embley et al., 2016]
- *Accuracy*: MIPS (TANGO) — **0.9899** vs. TabbyXL — **0.9950**

Structural Analysis

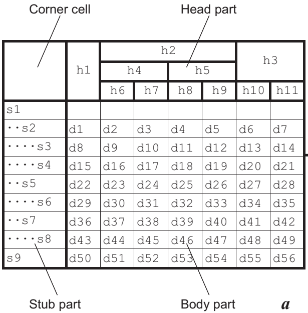
- *Contest task*: The extraction of header hierarchies from tables
- *Testing dataset*: A random subset of SAUS[a]
- *Contestant*: Senbazuru [Chen and Cafarella, 2014]
- *F-score*: Senbazuru — **0.8860** vs. TabbyXL — **0.8657**

[a] `http://dbgroup.eecs.umich.edu/project/sheets/datasets.html`

Table of Contents

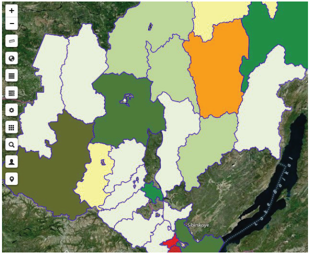
Application Experience

Populating a web-based statistical atlas of the Irkutsk region — (*b*) via extracting data from government statistical reports — (*a*)



a

DATA	HEAD	STUB
d1	h1	s1 | s2
d2	h2 | h4 | h6	s1 | s2
d3	h2 | h4 | h7	s1 | s2
d4	h2 | h5 | h8	s1 | s2
d5	h2 | h5 | h9	s1 | s2
d6	h3 | h10	s1 | s2
d7	h3 | h11	s1 | s2
d8	h1	s1 | s2 | s3
d9	h2 | h4 | h6	s1 | s2 | s3
d10	h2 | h4 | h7	s1 | s2 | s3
d11	h2 | h5 | h8	s1 | s2 | s3
d12	h2 | h5 | h9	s1 | s2 | s3
d13	h3 | h10	s1 | s2 | s3
d14	h3 | h11	s1 | s2 | s3
...	...	...
d56	h3 | h11	s9

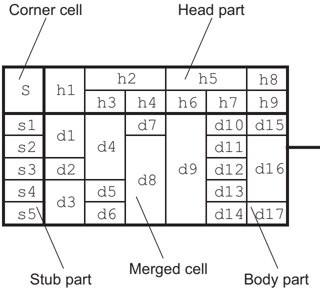


b

The more detail can be found at `https://github.com/tabbydoc/tabbyxl/wiki/statistical-atlas`

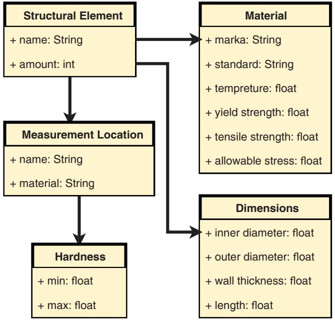
Application Experience

Generating conceptual models — (*b*) from arbitrary tables presented in industrial safety inspection reports — (*a*)



a

DATA	HEAD	S
d1	h1	s1
d1	h1	s2
d2	h1	s3
d3	h1	s4
d3	h1	s5
d4	h2 | h3	s1
d4	h2 | h3	s2
d4	h2 | h3	s3
d5	h2 | h3	s4
d6	h2 | h3	s5
...	...	...
d17	h8 | h9	s5



b

The more detail can be found at `https://github.com/tabbydoc/tabbyxl/wiki/industrial-safety-inspection`

Table of Contents

Conclusions & Further Work

- Impact on software development for spreadsheet data management
  - Table object model associating functional roles with data items
  - Table analysis and interpretation driven by user-defined rules
  - Formulated actions to recover missing semantics of arbitrary tables
  - Translation of rules to executable spreadsheet transformation programs
- Limitations
  - The inaccurate cell structure prevents the table analysis
  - The very limited interpretation (without external vocabularies)
- Further work
  - Rearrangement of cell structure by using visual (human-readable) cells
  - Detecting derived data by spreadsheet formulas
  - Enriching the table analysis by named entity recognition
  - Linking extracted data items with LOD cloud

Table of Contents

References I

Amalfitano, D., Fasolino, A. R., Tramontana, P., De Simone, V., Di Mare, G., and Scala, S. (2015).
A reverse engineering process for inferring data models from spreadsheet-based information systems: An automotive industrial experience.
In *Data Management Technologies and Applications*, pages 136–153.

Barowy, D. W., Berger, E. D., and Zorn, B. (2018).
ExceLint: Automatically finding spreadsheet formula errors.
Proc. ACM Program. Lang., 2(OOPSLA):148:1–148:26.

Barowy, D. W., Gulwani, S., Hart, T., and Zorn, B. (2015).
FlashRelate: Extracting relational data from semi-structured spreadsheets using examples.
SIGPLAN Not., 50(6):218–228.

Cao, T. D., Manolescu, I., and Tannier, X. (2017).
Extracting linked data from statistic spreadsheets.
In *Proc. Int. Workshop Semantic Big Data*, pages 5:1–5:5.

Chen, Z. (2016).
Information Extraction on Para-Relational Data.
PhD thesis, University of Michigan, US.

Chen, Z. and Cafarella, M. (2014).
Integrating spreadsheet data via accurate and low-effort extraction.
In *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 1126–1135.

Chen, Z., Dadiomov, S., Wesley, R., Xiao, G., Cory, D., Cafarella, M., and Mackinlay, J. (2017).
Spreadsheet property detection with rule-assisted active learning.
In *Proc. ACM on Conf. on Information and Knowledge Management*, pages 999–1008.

References II

Cunha, J., Erwig, M., Mendes, J., and Saraiva, J. (2016).
Model inference for spreadsheets.
Autom Softw Eng, 23(3):361–392.

Cunha, J., Fernandes, J. P., Mendes, J., and Saraiva, J. (2015).
Embedding, evolution, and validation of model-driven spreadsheets.
IEEE Transactions on Software Engineering, 41(3):241–263.

de Vos, M., Wielemaker, J., Rijgersberg, H., Schreiber, G., Wielinga, B., and Top, J. (2017).
Combining information on structure and content to automatically annotate natural science spreadsheets.
Int. J. Human-Computer Studies, 103:63–76.

Dou, W., Han, S., Xu, L., Zhang, D., and Wei, J. (2018).
Expandable group identification in spreadsheets.
In *Proc. 33rd ACM/IEEE Int. Conf. on Automated Software Engineering*, pages 498–508.

Dou, W., Xu, C., Cheung, S. C., and Wei, J. (2017).
CACheck: Detecting and repairing cell arrays in spreadsheets.
IEEE Transactions on Software Engineering, 43(3):226–251.

Embley, D. W., Krishnamoorthy, M. S., Nagy, G., and Seth, S. (2016).
Converting heterogeneous statistical tables on the web to searchable databases.
Int. J. Document Analysis and Recognition, 19(2):119–138.

Hermans, F., Pinzger, M., and Deursen, A. (2015).
Detecting and refactoring code smells in spreadsheet formulas.
Empirical Softw. Engg., 20(2):549–575.

References III

Hurst, M. (2001).
Layout and language: Challenges for table understanding on the web.
In *Proc. 1st Int. Workshop Web Document Analysis*, pages 27–30.

Jin, Z., Anderson, M. R., Cafarella, M., and Jagadish, H. V. (2017).
Foofah: Transforming data by example.
In *Proc. ACM Int. Conf. Management of Data*, pages 683–698.

Koch, P., Hofer, B., and Wotawa, F. (2019).
On the refinement of spreadsheet smells by means of structure information.
Journal of Systems and Software, 147:64–85.

Koci, E., Thiele, M., Lehner, W., and Romero, O. (2018).
Table recognition in spreadsheets via a graph representation.
In *Proc. 13th IAPR Int. Workshop on Document Analysis Systems*, pages 139–144.

Koci, E., Thiele, M., Romero, O., and Lehner, W. (2016).
A machine learning approach for layout inference in spreadsheets.
In *Proc. 8th Int. Joint Conf. Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 77–88.

Koci, E., Thiele, M., Romero, O., and Lehner, W. (2017).
Table identification and reconstruction in spreadsheets.
In *Proc. 29th Int. Conf. Advanced Information Systems Engineering*, pages 527–541.

Nagy, G. (2016).
TANGO-DocLab web tables from international statistical sites (Troy_200), 1, ID: Troy_200_1.

References IV

Ritze, D. and Bizer, C. (2017).
Matching web tables to dbpedia - A feature utility study.
In *Proc. 20th Int. Conf. on Extending Database Technology*, pages 210–221.

Shigarov, A. O. and Mikhailov, A. A. (2017).
Rule-based spreadsheet data transformation from arbitrary to relational tables.
Information Systems, 71:123–136.

Singh, R. and Gulwani, S. (2016).
Transforming spreadsheet data types using examples.
SIGPLAN Not., 51(1):343–356.

Swidan, A. and Hermans, F. (2017).
Semi-automatic extraction of cross-table data from a set of spreadsheets.
In Barbosa, S., Markopoulos, P., Paternò, F., Stumpf, S., and Valtolina, S., editors, *End-User Development*, pages 84–99.

Yang, S., Guo, J., and Wei, R. (2017).
Semantic interoperability with heterogeneous information systems on the internet through automatic tabular document exchange.
Information Systems, 69:195–217.

Yang, S., Wei, R., and Shigarov, A. (2018).
Semantic interoperability for electronic business through a novel cross-context semantic document exchange approach.
In *Proc. ACM Symposium on Doc. Eng.*, pages 28:1–28:10.

Zhang, Z. (2017).
Effective and efficient semantic table interpretation using TableMiner$^+$.
Semantic Web, 8(6):921–957.

Thanks

Read more about the project at
http://td.icc.ru

The project source code is available at
https://github.com/tabbydoc/tabbyxl