Software Development for Rule-Based Spreadsheet Data Extraction and Transformation

1st Alexey Shigarov
Matrosov Institute for System Dynamics
and Control Theory of SB RAS
Irkutsk, Russia
shigarov@icc.ru

2nd Vasiliy Khristyuk
Matrosov Institute for System Dynamics
and Control Theory of SB RAS
Irkutsk, Russia
khristyuk@icc.ru

3rd Andrey Mikhailov
Matrosov Institute for System Dynamics
and Control Theory of SB RAS
Irkutsk, Russia
mikhailov@icc.ru

4th Viacheslav Paramonov
Matrosov Institute for System Dynamics
and Control Theory of SB RAS
Irkutsk, Russia
slv@icc.ru

Abstract—The paper proposes a knowledge-based software platform to generate applications for spreadsheet data extraction and transformation. The platform includes a flexible table object model and a domain-specific language for expressing user-defined rules of table analysis and interpretation. They serve to represent knowledge of table layout and content features, as well as their interpretation, depended on transformation goals. The platform enables translating such user-defined rules to Java programs. The generated source code is serialized as a project prepared for building an executable application by using the Maven tool. The execution of the generated application transforms spreadsheet data from arbitrary form defined by the rules to the canonical one. The empirical results demonstrate the applicability of the software platform to develop applications for converting data from arbitrary spreadsheet tables originated from various domains to relational flat file databases.

Index Terms—spreadsheet data transformation, table analysis, table interpretation, rule-based programming

I. Introduction

A big volume of arbitrary tables (e. g. cross-tabulations, invoices, roadmaps, and data collection forms) circulates in spreadsheet-like formats. Spreadsheets can be considered as a general form for representing tabular data with an explicitly presented layout (cellular structure) and style (graphical formatting). The arbitrary spreadsheet tables can be a valuable data source in business intelligence and data-driven research. However, difficulties that inevitably arise with extraction and integration of the tabular data often hinder the intensive use of them in the mentioned areas.

Typically, arbitrary spreadsheet tables contain unstructured and non-standardized ("messy") data. Unlike relational tables, they are not organized in a predefined manner. They lack explicit semantics required for high-level computer interpretation such as SQL queries. To be accessible for data analysis and visualization, their data need to be extracted, transformed, and loaded (ETL) into databases. Since arbitrary tables can have various complex cell layouts (structures), often the familiar industrial ETL-tools are not sufficient to populate automatically a database with their data.

Usually, researchers and developers faced with the tasks of spreadsheet data integration use general-purpose tools. They often offer their own implementations of the same tasks. In such cases, domain-specific tools can reduce the complexity of software development in the domain of the spreadsheet data integration. This is especially important when a custom software for data extraction and transformation from heterogeneous arbitrary spreadsheet tables is implemented for a short time and with a lack of resources.

A. Related Work

There are several recent studies devoted to the issues of spreadsheet data converting. The tools [1]–[6] are devoted to issues of converting data presented in spreadsheets or web tables to the linked data formats (RDF/OWL). The solutions for spreadsheet data extraction and transformation [7]–[10] are based on programming by examples. Some of them include own domain-specific languages, e.g. XLWrap [1], M^2 [2], TableProg [7], and Flare [9].

Hung et al. [11] propose TranSheet, a spreadsheet-like formula language for specifying mappings between source spreadsheet data and a target schema. DeExcelerator framework [12] aims at the development of a framework for information extraction from partially structured documents such as spreadsheets and HTML tables. The framework exploits a set of heuristics based on features of tables published as Open Data. MDSheet framework [13] also implements a technique that automatically infers relational schemes from spreadsheets. The framework [14] constructs trained spreadsheet property (e. g. aggregation rows or hierarchical header) detectors based on rule-

assisted active learning. Embley et al. [15] propose an algorithmic end-to-end solution for transforming "header-indexed" tables in CSV format to a relational form ("category tables") based on header indexing. Senbazuru, a spreadsheet database management system proposed by [16], provides extracting relational data from spreadsheets ("data frames").

The recent papers [17], [18] develop domain-specific solutions. The work [17] proposes algorithms and accompanying software for automatic annotation of natural science spreadsheet tables. The other tool [18] extracts RDF data from French government statistical spreadsheets and populates instances of their conceptual model. The system CACheck [19] automatically detects "smelly" cell arrays by recovering their intended computational semantics. TaCLe [20] automatically identifies constraints (formulas and relations) in spreadsheets. The papers [21] consider the issues of tabular document processing.

B. Contribution

Our work shows new possibilities in spreadsheet data transformation from arbitrary to relational tables, using rule-based programming for the following stages of table understanding: (1) role analysis (extracting functional data items, entries (values) and labels (keys), from cell content); (2) structural analysis (recovering internal relationships of extracted items); (3) interpretation (associating extracted items with concepts of external vocabularies).

We develop TabbyXL, a platform for developing software for spreadsheet data extraction and transformation from an arbitrary (Fig. 1, a) to the relational (canonical) form (Fig. 1, b). Compared to the mentioned solutions it draws up this process as consecutive steps: role analysis, structural analysis, and interpretation.

The platform implements a two-layered table object model combines the physical (syntactic) and logical (semantic) table structure. Unlike others models, it is not based on using functional cell regions but determines that functional items can be placed anywhere in a table.

We also design and implement a novel domain-specific rule object model for representing rules expressed in CRL (Cells Rule Language). Such rules map the physical structure to the logical one of an arbitrary table. Our rule model enables translating CRL rules to Java programs for the spreadsheet data transformation. In contrast to the existing mapping languages, our rule language and model express the spreadsheet data conversion in terms of table understanding.

This paper extends our approach to table understanding based on executing rules for table analysis and interpretation [22], [23] by adding the rule object model for representing and translating CRL rules to the source code, as well as an implementation of CRL-to-Java translator based on this model.

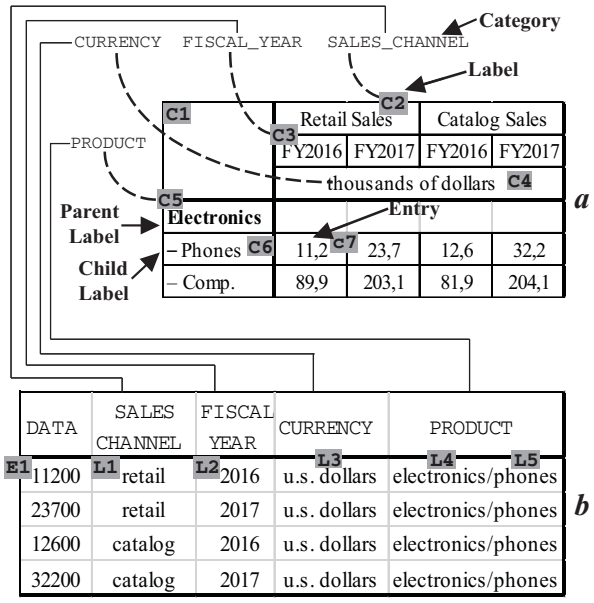

Fig. 1. A fragment of a arbitrary spreadsheet table (a) and its canonical form (b).

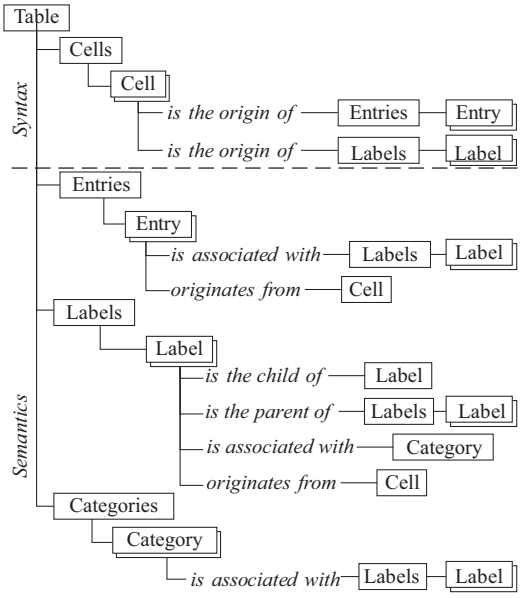

Fig. 2. Two-layered table object model [23].

II. Table Object Model

The table object model is designed for representing both a physical structure and logical data items of an arbitrary table in the process of its analysis and interpretation (Fig. 2). Our model adopts the terminology of Wang's table model [24]. It includes two interrelated layers: physical represented by the collection of cells and logical that consists of three collections of entries (values), labels (keys), and categories (concepts). This model is presented more detail in our previous paper [23].

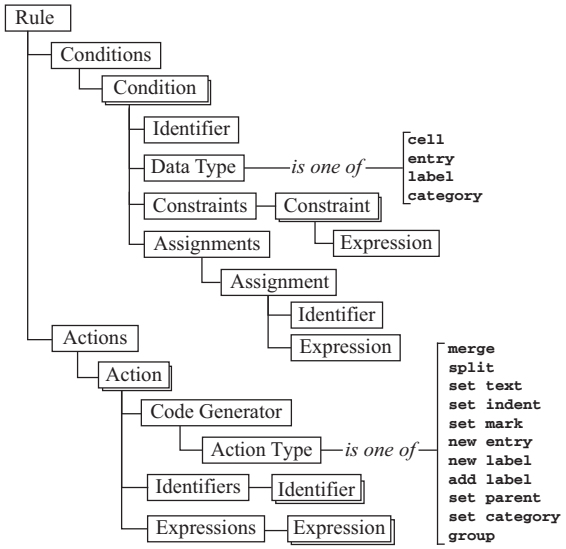

Fig. 3. CRL rule object model.

The physical layer consists of a set of cells. "Cell" object represents common features of a cell that can be presented in a spreadsheet format, including: positions in rows and columns; style (font, alignment, etc.); text content, data types; annotation (user-defined tags to annotate the cell). These features are extracted from an arbitrary table (Fig. 1, a).

The logical layer includes sets of the following data items. "Entry" object serves to represent a data value of a table. An entry can be associated with only one label in each category. "Label" object represents a label (key) that addresses one or more entries (data values). A label can be associated with other label as child-parent, "Category" object represents a category of labels. Each label is associated with only one category. Labels combined into a category can be organized as one or more trees. The recovered data items of the logical layer can be presented as a relational table in the canonical form (Fig. 1, b).

III. Table Analysis and Interpretation Rules

Table analysis and interpretation rules are intended to map explicit features (layout, style, and text of cells) of an arbitrary table into its implicit semantics (entries, labels, and categories) Such rules can be expressed in a general-purpose rule-based language and executed with a rule engine [22]. To determine a functionality needed for developing them, we design CRL[1], a domain-specific rule language, which specializes in expressing such rules. Our language hides details which are inessential for us and allows to focus on the logic of table analysis and interpretation. The syntax of CRL rules is presented more detail in our previous paper [23].

[1]https://github.com/tabbydoc/tabbyxl/wiki/crl-language

In this paper, we propose the rule object model for representing CRL rules (Fig. 3). This model enables to translate CRL rules to the source code of application software for converting spreadsheet data from an arbitrary to the relational form.

The rule object model includes a set of conditions and a set of actions as shown in Fig. 3. The set of conditions is built according to the left hand side ("when") of a rule. They enable to query available facts of a table such as cells, entries, labels, and categories.

The set of actions describes the right hand side ("then") of the rule. They are executed when all conditions of the set being true. The actions modify the existed facts or generate new ones.

A. Conditions

A condition consists of the following items: (1) Data Type denoting the fact type ("cell", "entry", "label", or "category"); (2) Identifier names "variable" of the specified fact type; (3) Constraints define (Java Boolean expression) for restricting the requested facts; (4) Assignments are used for binding additional variables with some values.

A rule can use two kinds of conditions. The first requires that there exists at least one fact of a specified data type, which satisfies a set of constraints:

```
cell | entry | label | category
variable: constraints, assignments
```

The second kind of conditions determines that there exist no facts satisfied to specified constraints:

```
no cells | entries | labels | categories:
constraints
```

Such conditions have no identifiers and assignments.

B. Actions for Cell Cleansing

In practice, hand-coded tables often have messy layout (e. g. improperly split or merged cells) and content (e. g. typos, homoglyphs, or errors in indents). We address several actions to the issues of cell cleansing that can be used as the preprocessing stage.

- "merge", the action combines two adjacent cells when they share one border.
- "split", the action divides a merged cell that spans n-tiles into n-cells. Each of the n-cells completely copies content and style from the merged cell and coordinates from the corresponding tile.
- "set text", the action provides modifying textual content of a cell. Some string processing (e. g. regular expressions and string matching algorithms) implemented as Java-methods can be involved in the action.
- "set indent", the action modifies an indentation of a cell.

C. Actions for Role Analysis

This stage aims to recover entries and labels as functional data items presented in tables. We also enable associating cells with user-defined tags (marks) that can assist in both role and structural analysis.

- "set mark", the action annotates a cell with a word or phrase. The assigned tag can substitute the corresponding constraints in subsequent rules. The typical practice is to set a tag to all cells, which play the same role or are located in the same table functional region. Thereafter, we can use these tags in subsequent rules instead of repeating constraints on cell location in the regions.
- "new entry", the action generates an entry, using a specified cell as its origin. Usually, a value of the created entry is an expression obtained as a result of string processing for its textual content of the cell.
- "new label", the action generates a label in a similar way.

D. Actions for Structural Analysis

The next stage recovers pairs of two kinds: entry-label and label-label.

- "add label", the action binds an entry with an added label. A label can be specified as a value of a category indicated by its name.
- "set parent", the action connects two labels as a parent and its child.

E. Actions for Interpretation

The stage includes actions for recovering label-category pairs.

- "set category", the action associates a label with a category.
- "group", the action places two labels in one group. Arbitrary tables often place all labels of one category in the same row or column. Consequently, we can suppose that the labels belong to a category without defining its name. In the cases, grouping two or more labels means that they all belong to an undefined category. All labels of a group can be associated with one category only.

IV. Development of Spreadsheet Data Transformation Programs

TabbyXL[2] implements both the presented table object model and the approach to rule-based spreadsheet data extraction and transformation. It enables automatically generating Java source code from CRL rules. A generated program can be compiled to executable Java bytecode. We use ANTLR[3], the parser generator, to implement the CRL-to-Java translator. This allows to parse CRL rules and to build their object model which is then translated to Java source code. The generated source code can be serialized as a project prepared for building an executable application by using the Maven tool. Fig. 4 shows the workflow for generating Maven-projects for such programs (a), including Java source code generating from CRL rules (b).

As an alternative, the rules can be executed by Drools rule engine. In this case, they should be expressed in a dialect of CRL that is implemented as DSL in corresponding of Drools requirements. Unlike the pure CRL, this dialect supports DRL attributes in rule declarations. CRL rules are automatically translated into DRL, a native format of Drools, through the DSL-specification that defines CRL-to-DRL mappings. The rule engine matches asserted facts (cells from an instance of the table object model) against the rules. The instance is augmented by recovered facts (entries, labels, and categories). At the end of the transformation process, the instance is exported as a flat file database.

V. Application Example

The application we considered aimed at developing an interactive statistical atlas of the Irkutsk region (StatAtlas). It required extracting data from government statistical reports distributed by IrkutskStat (Irkutsk Regional Committee of the Russian Federal State Statistics Service). Source data were presented as arbitrary tables in Word documents. All tables were extracted and converted to Excel format, using VBA macro. We also accompanied each table by two tags $START and $END for specifying its boundaries on the worksheet. We developed a ruleset in CRL format for recovering functional roles and relationships inside each table. All tables were transformed into relational ones by running the ruleset. Note that our software platform was used only in this phase of the workflow. As a result, all transformed tables were aggregated and converted to CSV format with geocoded names of territories and loaded into the database of StatAtlas.

Fig. 5 depicts an arbitrary table (a) as an example of a layout form being used in IrkutskStat reports, as well as a corresponding relational table in the canonical form that is generated as a result of the table transformation (b). The ruleset we used contains only 7 rules expressed in CRL format as follows: (1) generating "column" labels from cells located in the head part of a table; (2) associating child column labels with parent ones; (3) generating "row" labels from cells located in the stub part; (4) associating child "row" labels with parent ones; (5) generating entries from cells located in the body part; (6) associating entries with column labels (7) and with row labels. The data and steps to reproduce this example are available at GitHub[4].

[2]https://github.com/tabbydoc/tabbyxl/releases/tag/v1.0.4
[3]https://www.antlr.org
[4]https://github.com/tabbydoc/tabbyxl/wiki/example-3

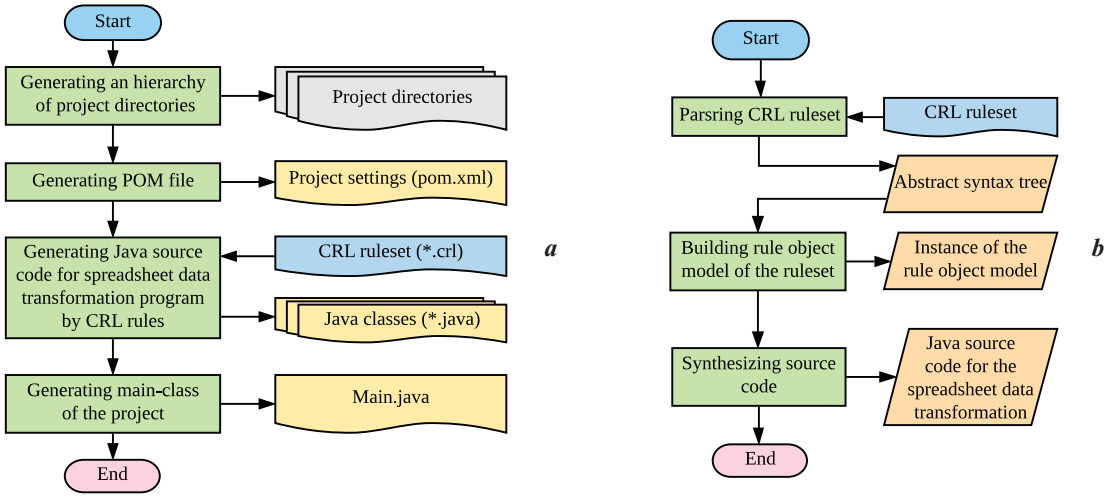

Fig. 4. Workflow for generating Maven-projects for spreadsheet data transformation programs (a), including translating CRL rules to Java source code (b).

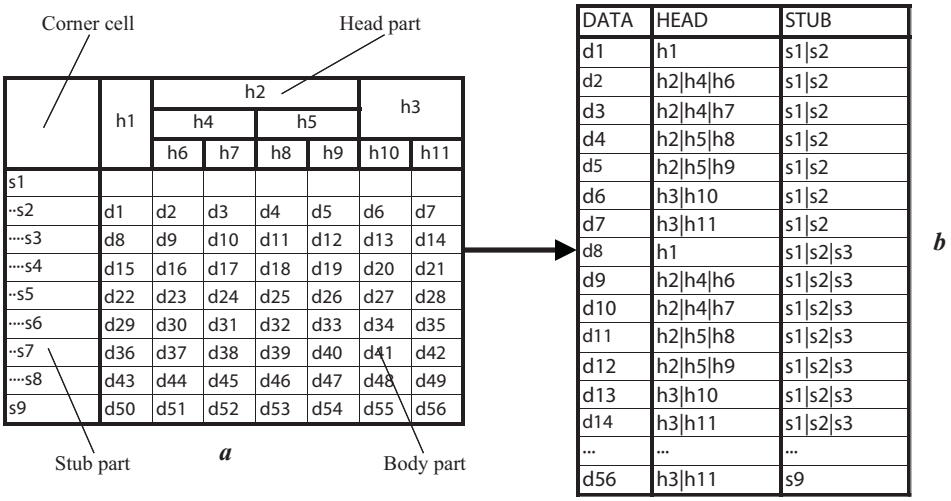

Fig. 5. Source table (a) and its target representation in the canonical form (b).

VI. Experimental Results

The purpose of the experiment is to show a possibility of using our platform for converting tables, which originate from various sources produced by different authors but pertain to the same document domain. The experiment includes two parts: (i) designing and implementing an experimental rule-set for tables of the same genre, and (ii) evaluating the performance of the rule-set on a set of these tables. All data and rules for reproducing the experiment results are available as the published dataset [25].

We used Troy200 [26], the existing dataset of tables, for the performance evaluation. It contains 200 arbitrary tables as CSV files collected from 10 different sources of the same genre, government statistical websites predominantly presented in English language.

We designed and implemented a rule-set that transforms the tables of the dataset into the relational form according to the presented target requirements. We also prepared ground-truth data, including reference entries, labels, entry-label, and label-label pairs extracted from Troy200 tables. Their form is designed for human readability. Moreover, they are independent of the presence of critical cells in tables. The performance evaluation is based on comparing the ground-truth data with the tables generated by executing the presented rule-set for the experiment

TABLE I
The performance evaluation results of the tested ruleset for recovering data items and their relationships.

	data items		relationships	
metrics	entries	labels	entry-label	label-label
recall	0.9813	0.9965	0.9773	0.9389
precision	0.9996	0.9364	0.9965	0.9784
F-score	0.9904	0.9655	0.9868	0.9582

dataset.

We used the standard metrics: recall, precision, and F-score, to evaluate our rule-set for both role and structural analysis. The experiment results are shown in Table I. Among 200 tables of the dataset, only 25 are processed with errors (1249 false negatives in 25 tables, and 488 false positives in 14 ones). total 1256 false negatives in 25 tables, and 498 173 false positives in 14 ones Only one table is not processed. This results in 948 false negative and 316 false positive errors, which amount to about 73% of all errors. In this case, entries are not recovered because they are not numeric as it is assumed in the used rule-set.

This experiment exemplifies the use of our language for developing a task-specific rule-set. The performance evaluation confirms the applicability of the implemented rule-set in accomplishing the stated objectives of this application.

VII. Conclusions and Further Work

The presented approach can be applied to develop software for extraction and transformation data of arbitrary spreadsheet tables. We expect TabbyXL to be useful in cases when data from a large number of tables appertaining to a few table types are required for populating a database. Our application experience and the experiment results demonstrate that the platform can be used for developing programs for the transformation of spreadsheet data into the relational form. One rule-set can process a wide range of tables of the same genre, e. g. government statistical websites. Our platform can be used for populating databases from arbitrary tables, which share common features.

The work focuses rather on table analysis than on issues of interpretation. We only recover categories as sets of labels, without binding them with an external taxonomy of concepts (e.g. linked open data). The further work on table content conceptualisation can overcome this limitation. Moreover, we observe that arbitrary tables can contain messy and useless data. It seems to be interesting to incorporate additional techniques of data cleansing in the platform.

Acknowledgment

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

References

[1] A. Langegger and W. Wöß, XLWrap - Querying and Integrating Arbitrary Spreadsheets with SPARQL, 2009, pp. 359–374.

[2] M. J. O'Connor, C. Halaschek-Wiener, and M. A. Musen, Mapping Master: A Flexible Approach for Mapping Spreadsheets to OWL, 2010, pp. 194–208.

[3] V. Mulwad, T. Finin, and A. Joshi, A Domain Independent Framework for Extracting Linked Semantic Data from Tables, 2012, pp. 16–33.

[4] M. Fiorelli, T. Lorenzetti, M. T. Pazienza, A. Stellato, and A. Turbati, "Sheet2RDF: a flexible and dynamic spreadsheet import&lifting framework for RDF," in Proc. 28th Int. Conf. Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2015, pp. 131–140.

[5] M. Galkin, D. Mouromtsev, and S. Auer, "Identifying web tables: Supporting a neglected type of content on the web," in Proc. 6th Int. Conf. Knowledge Engineering and Semantic Web, 2015, pp. 48–62.

[6] I. Ermilov and A.-C. N. Ngomo, TAIPAN: Automatic Property Mapping for Tabular Data, 2016, pp. 163–179.

[7] W. R. Harris and S. Gulwani, "Spreadsheet table transformations from examples," SIGPLAN Not., vol. 46, no. 6, pp. 317–328, 2011.

[8] S. Gulwani, W. R. Harris, and R. Singh, "Spreadsheet data manipulation using examples," Commun. ACM, vol. 55, no. 8, pp. 97–105, 2012.

[9] D. W. Barowy, S. Gulwani, T. Hart, and B. Zorn, "FlashRelate: Extracting relational data from semi-structured spreadsheets using examples," SIGPLAN Not., vol. 50, no. 6, pp. 218–228, 2015.

[10] Z. Jin, M. R. Anderson, M. Cafarella, and H. V. Jagadish, "Foofah: Transforming data by example," in Proc. ACM Int. Conf. Management of Data, 2017, pp. 683–698.

[11] V. Hung, B. Benatallah, and R. Saint-Paul, "Spreadsheet-based complex data transformation," in Proc. 20th ACM Int. Conf. Information and Knowledge Management, 2011, pp. 1749–1754.

[12] J. Eberius, C. Werner, M. Thiele, K. Braunschweig, L. Dannecker, and W. Lehner, "DeExcelerator: a framework for extracting relational data from partially structured documents," in Proc. 22nd ACM Int. Conf. on Information & Knowledge Management, 2013, pp. 2477–2480. [Online]. Available: http://doi.acm.org/10.1145/2505515.2508210

[13] J. Cunha, M. Erwig, J. Mendes, and J. Saraiva, "Model inference for spreadsheets," Autom Softw Eng, vol. 23, no. 3, pp. 361–392, 2016.

[14] Z. Chen, X. Rong, S. Dadiomov, R. Wesley, G. Xiao, D. Cory, M. Cafarella, and J. Mackinlay, "Spreadsheet property detection with rule-assisted active learning," Tech. Rep. CSE-TR-601-16, 2016. [Online]. Available: https://www.cse.umich.edu/techreports/cse/2016/CSE-TR-601-16.pdf

[15] D. W. Embley, M. S. Krishnamoorthy, G. Nagy, and S. Seth, "Converting heterogeneous statistical tables on the web to searchable databases," Int. J. Document Analysis and Recognition, vol. 19, no. 2, pp. 119–138, 2016.

[16] Z. Chen, "Information extraction on para-relational data," Ph.D. dissertation, University of Michigan, US, 2016.

[17] M. de Vos, J. Wielemaker, H. Rijgersberg, G. Schreiber, B. Wielinga, and J. Top, "Combining information on structure and content to automatically annotate natural science spreadsheets," International Journal of Human-Computer Studies, vol. 103, pp. 63–76, 2017.

[18] T. D. Cao, I. Manolescu, and X. Tannier, "Extracting linked data from statistic spreadsheets," in Proc. Int. Workshop Semantic Big Data, 2017, pp. 5:1–5:5.

[19] W. Dou, C. Xu, S. C. Cheung, and J. Wei, "Cacheck: Detecting and repairing cell arrays in spreadsheets," IEEE Transactions on Software Engineering, vol. 43, no. 3, pp. 226–251, 2017.

[20] S. Kolb, S. Paramonov, T. Guns, and L. De Raedt, "Learning constraints in spreadsheets and tabular data," Machine Learning, vol. 106, no. 9, pp. 1441–1468, 2017.

[21] S. Yang, J. Guo, and R. Wei, "Semantic interoperability with heterogeneous information systems on the internet through automatic tabular document exchange," Information Systems, vol. 69, pp. 195–217, 2017.

[22] A. Shigarov, "Table understanding using a rule engine," Expert Systems with Applications, vol. 42, no. 2, pp. 929–937, 2015.

[23] A. O. Shigarov and A. A. Mikhailov, "Rule-based spreadsheet data transformation from arbitrary to relational tables," Information Systems, vol. 71, pp. 123–136, 2017.

[24] X. Wang, "Tabular abstraction, editing, and formatting," Ph.D. dissertation, University of Waterloo, Waterloo, Ontario, Canada, 1996.

[25] V. K. A. Shigarov, "TabbyXL: Dataset for the performance evaluation of a software platform for rule-based spreadsheet data extraction and transformation," Mendeley Data, v5, 2018.

[26] G. Nagy, "TANGO-DocLab web tables from international statistical sites (Troy_200), 1, ID: Troy_200_1," 2016. [Online]. Available: http://tc11.cvc.uab.es/datasets/Troy_200_1