Graph-based rule editor

Maciej Nowak, Jaroslaw Bak, Czeslaw Jedrzejek

Institute of Control and Information Engineering,
Poznan University of Technology,
M. Sklodowskiej-Curie Sqr. 5, 60-965 Poznan, Poland
{firstname.lastname}@put.poznan.pl

Abstract. In this paper we present a prototypical implementation of a graphical tool for creating rules. This tool uses a graph-based Palantir tool environment as a user interface to model rule conditions and conclusions. It is also used to visualize data and results of reasoning. We present a process of converting graph models stored in an XML format file into the Jess knowledge base and rules. Results obtained in the reasoning process are presented to the user in the same form as source data.

Keywords. graphical rule representation, Jess, Palantir, reasoning

1 Introduction

Rule engines are becoming one of the most commonly used technologies in bc business and engineering projects. The usage of rule engines enables the reasoni process which enriches gathered data and searching methods, utilizing pattern matc ing applied in rules. Rules and rule engines are successfully used in: expert systen business processes, data integration and transformation, and in other applicatic requiring intelligent data processing. Despite the clear advantages of rule-based tec nologies, there are many software application areas where they occur in a relative simple form, e.g. by using filters. In the area of criminal analysis, the addition of ru to investigation systems would enable analysts to discover very complex cri schemes, totally beyond the capacity of traditional systems.

A wide range of rule environments have been proposed, each with its own synt and semantics for a rule engine and interface. The process of acquiring tl knowledge can be simplified with the use of a graphical representation of rules anc user-friendly interface.

The main aim of this demo paper is to present a graph-based tool, in which an u trained analyst is able to construct a set of simple rules and use them in order to obta new (inferred) information. The rules constitute the expert's knowledge of a giv domain, while facts represent data. Both rules and facts are expressed graphically the form of directed graphs. Rules can be applied to facts using a reasoning engi After the inference process, a user gets the result which can be a graph with:

- the addition of new objects and/or relations,
- the modification of existing objects and/or relations,

- the lack of objects and/or relations that were deleted.

The paper is organized as follows. Section 2 presents the main overview of the proposed approach and related work. Section 3 describes a prototypical implementation and applied tools. Section 4 demonstrates an example which reflects a fragment of analysis of the real-world crime case. Section 5 contains concluding remarks.

2 Graph-based Rule Representation

2.1 Existing methods

Graphical rule representation and creation has been the subject of research conducted by many investigators and companies. Some efforts have sought to standardize graphical notations, for example: Unified Modeling Language/Object Constraints Language (UML/OCL) [5], UML-based Rule Modeling Language (URML) [6] or Object Role Modeling (ORM) [7]. Among them, the ORM language is the most intuitive and easy to use for people who are not familiar with the complex syntax of rules and UML/OCL. The ORM concepts were adopted [8] also in the SBVR (Semantics of Business Vocabulary and Business Rules) standard [8]. Our ideas are based on ORM and graph-based representation. Other popular rule representations include: decision tables, decision trees and eXtended Tabular Trees [10].

Tools that implement graphical rules representation are:

- Visual Rules [11] – supports building flow rules and decision tables using rich and intuitive graphical editors.
- Drools Guvnor [12] – provides many ways of representing rules: guided editor (easy to use, but not graphical), guided decision tables creation, rule flows (which represent the flow of logic).
- VisiRule [13] – is an extension to Win-Prolog which supports building decision models using a graphical paradigm. It offers graphical representation of forward chaining rules, with access to Prolog.
- OntoStudio Graphical Rule Editor [14] – is based on ObjectLogic [15] internally. It supports drawing rule diagrams, which consist of concepts, attributes of these concepts and relations between them. It does not allow the comparison of variables (only comparisons between values and variables are allowed).

In this work we give only a short overview of the main differences. Detailed comparisons among the mentioned standards will be presented elsewhere.

In most of the current approaches, rules are created to control data workflows and making decisions, while we apply rules to discover new information and to process data. Accordingly, one rule (LHS and RHS respectively) is represented by two graphs. Tools like Visual Rules, Drools Guvnor etc. are rule authoring frameworks while our approach is only an attempt to integrate rules and data in one graph-based form and perform reasoning. Such work, to the best of our knowledge, has not yet been done for the Jess engine.

2.2 Overview of the approach

The main goal of this paper is to present the graph-based tool, in which a user can: import data, construct rules, perform reasoning and obtain results. Rules and data, represented graphically, can be more easily understood by an untrained analysts and by engineers without intensive training. Our aim is to provide an easy-to-use and easy-to-understand analytical tool which can be used in many domains where rules and graphs can be employed to support a user's work.

The process of rule creation consists of creating two graphs which will later serve as sides of the constructed rule: the LHS (left hand side, called the body) and the RHS (right hand side, called the head). In our approach rules should be understood as *if LHS then RHS* statements. These rules (expressed in the Jess language) can be used to infer new information in a given rule-based knowledge base.

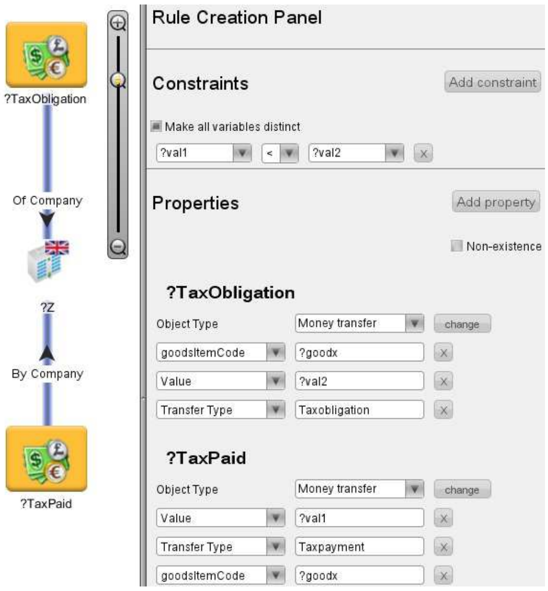
The LHS is built from condition elements (patterns) that need to be fulfilled in order to execute instructions written in the RHS. There are two types of conditions: the (non-) existence of a fact in the knowledge base with specified attributes, and the relationship between two attributes of existing facts. The execution of the rule may cause one of the following results: modification or removal of an existing fact, or addition of a new one. These operations are defined in the RHS of a rule.

It is possible to represent conditions from the body of a rule in a graphical form, more precisely in a graph. The graph consists of nodes and edges. The nodes are graphical representation of objects from the Palantir ontology (see Section 3.1), and the edges are the relations between them. Objects can have many properties; the type of an object is the most important one. Relations do not possess properties other than a type. The presence of an object in the graph means that a representing fact should exist in the knowledge base with attributes equal to the properties of the object. The presence of an object or a relation on a red background means that these artefacts should not exist in the knowledge base. The red colour on the graph expresses the negation of existence of objects or relations.

The construction of a rule is made with the following steps (within the Palantir environment):

1. A user creates a graph which constitutes the body of the rule, the conditions. The user creates objects and relations between them. Values of objects' properties, variables and constraints are specified in the Rule Creation Panel (RCP).
2. The user creates another graph, a modification of the first one which constitutes the head of a rule. The user adds/removes/modifies objects or relations of this graph. Conclusions - changes in the knowledge base after the application of the rule - can be modelled as the difference between two graphs.

Such an approach allows modelling of rules depending on object types, relations between them and values of objects' properties. It allows comparing attributes' values with each other, which is a significant advantage over some other tools (e.g. OntoStudio). There is no graphical way of presenting the comparison on the graph, so the only solution is to present it in the corresponding panel. For this purpose, we use a simple tab called Rule Creation Panel, which is presented in Figure 1.



Rule Creation Panel

Constraints

☒ Make all variables distinct

?val1 < ?val2

Properties

□ Non-existence

?TaxObligation

Object Type: Money transfer
goodsItemCode: ?goodx
Value: ?val2
Transfer Type: Taxobligation

?TaxPaid

Object Type: Money transfer
Value: ?val1
Transfer Type: Taxpayment
goodsItemCode: ?goodx

Figure 1. Attributes and relationships of selected objects (highlighted in yellow).

Created rules need to be applied to the working memory built from facts. In this paper we present a converter (see Section 3.3), which transforms rules and data to the Jess engine according to the structure expressed in the Palantir ontology. After the reasoning process, a user obtains results presented on a new graph (in comparison to the source data graph).

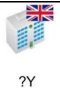
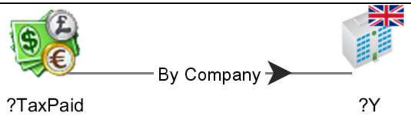
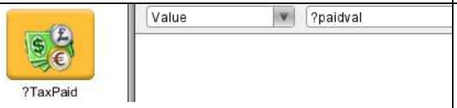
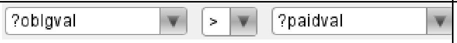
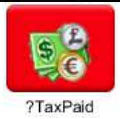
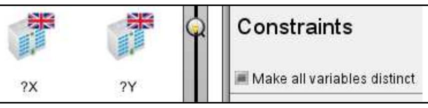
In the Jess language, we represent objects and relations from the graph as *triples* (as in RDF) [16] in order to express dependencies between objects and their attributes. The triple consists of subject, predicate and object. Each relation (edge) from the graph is mapped as the predicate, with its starting point as the subject and the ending point as the object of a triple. Each subject (node) has a set of properties, where: the *id* of the node constitutes the subject; property name corresponds to the predicate; and the value constitutes the object (in triple-based representation). Such an approach can be compared to the OWL Web Ontology Language, where ObjectProperties represent relations between objects and DatatypeProperties represent links from individuals (objects) to data values. Employing the triple-based representation we are able to apply OWL ontology in the future.

We have defined mapping for a bidirectional transformation between Palantir and Jess, executed by the XMLtoJess converter. Table 1 presents available expressions, with examples in the Jess language and graph elements.

The authors of this paper have successfully used rule engines in the past [16, 18, 19]. They were used during investigations of a number of cases. For some economic crime, the complete model of a crime investigation was constructed. That allowed achieving a result in a fully automatic way, but each time the rule set was made by a programmer experienced in the Jess language after consultation with a business specialist. We want to shorten this process with the help of the proposed system, and to increase analytical flexibility by including new elements of crime schemes.

The introduction of the rule engine offers not only the possibility of reasoning about complex dependencies, but also to performing queries. Any graph containing nodes and edges can be entered as a search phrase. Rule engine will search the whole knowledge base for a given set of conditions, and return all objects that meet the specified requirements.

Table 1. Representations of main elements, using a Graph and RCP panel, in the Jess code.

Element	Graph and RCP panel representation	Jess code
Object		(triple (subject ?Y) (predicate "Object Type") (object "GBOrganization"))
Relation		(triple (subject ?TaxPaid) (predicate "relation-By Company") (object ?Y))
Attribute Value	 Value ?paidval	(triple (subject ?TaxPaid) (predicate "property-Value") (object ?paidval))
Comparison of attribute values	 ?oblgval > ?paidval	(test (> ?oblgval ?paidval))
Declaration of non-existence (red background)		(not (triple (subject ?TaxPaid) (predicate "Object Type") (object "MoneyTransfer")))
Distinction of Variables	 Constraints ■ Make all variables distinct	(test (neq ?X ?Y))
Addition of an object/relation/attribute	New object/relation/attribute on the RHS (We add a new object/relation/attribute to a graph.)	(assert (triple (subject ?TaxObligation) (predicate "Object Type") (object "MoneyTransfer")))
Modification of existing object/relation/attribute	Modified object/relation/attribute on the RHS (We modify an object/relation/attribute in a graph.)	(modify ?f (object "DefaultingTrader"))
Removal of object/relation/attribute	Lack of object/relation/attribute on the RHS (We delete object/relation/attribute from a graph.)	(retract ?f)

3 Technologies Used

3.1 Palantir Government Graph Application

Palantir Government [1] is a Java-based platform for analysing and visualizing data. It is widely used by financial (Palantir Finance) and government agencies. It is capable of importing data structured in many various formats (such as Excel), and, due to the Palantir Dynamic Ontology (PDO) [2], objects inside the platform possess some semantic background meaning, which can be easily transformed into rules. The PDO is very simple; it only indicates that two objects are connected with a certain relation (represented then on a graph by an icon or relation).

Graph is the most sophisticated part of the Palantir Platform. It provides visualization of input data, with the structure defined in the given ontology. Properties of each object are not visible directly on the graph; they are reachable under the "Browser" tab. It is possible to export information from the graph into an external XML file, which is an essential element of the integration with a rule engine.

3.2 The Jess Rule Engine

Jess [3] is a rule engine and rule-based environment for building expert systems. It uses an enhanced version of the Rete [4] algorithm, which processes rules and facts in a very efficient way. Jess supports forward and backward chaining, working memory queries and many other useful features. Jess is provided as a library written in the Java language. It can easily be embedded into other Java applications. We applied Jess and its forward reasoning as extensions to the Palantir Government tool.

3.3 XMLtoJess Converter

XML is used as the interchange format between Jess and Palantir modules. Rule engines require input knowledge in form of facts, and that is why XMLtoJess converter is an essential part of the presented method.

The XMLtoJess converter is used to extract objects and relations stored in a Palantir XML (pXML) document generated from Palantir and create the Jess knowledge base. pXML format is the default output structure of the Palantir Platform. It holds information about objects in the Graph and all properties related to selected objects.

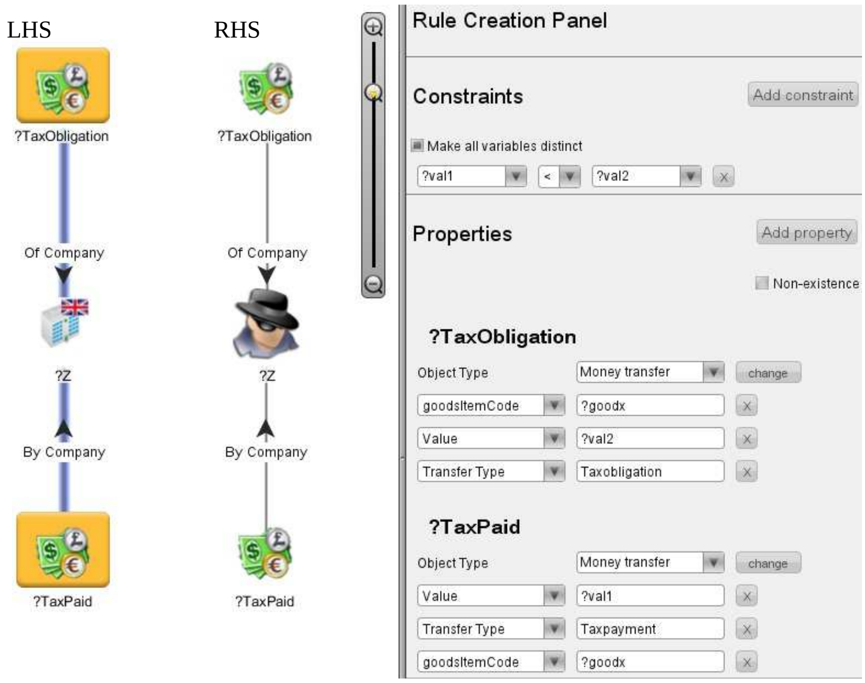
4 Example

In this section, we provide an example which reflects part of the analysis of a real-world crime case, the VAT carousel crime, also called the Missing Trader Intra-Community crime (MTIC). It is a sophisticated international fraud exploiting Value Added Tax (VAT) evasion, in order to create large amounts of unpaid VAT liabilities and VAT repayment claims connected with them. More information can be found on the demo site [17] and in [18].

Figure 2 depicts a graphical representation of the rule presented on the next page (where letters are used as shortcuts: *s* – subject, *p* – predicate, *o* – object).

```
(defrule VATFraudsterRule
   ?f <- (triple (s ?Z) (p "Object Type") (o "GBOrganization"))
   (triple (s ?TaxObligation) (p "Object Type") (o "MoneyTransfer"))
   (triple (s ?TaxObligation) (p "property-Transfer Type") (o "Taxobligation"))
   (triple (s ?TaxObligation) (p "property-goodsItemCode") (o ?good1))
   (triple (s ?TaxObligation) (p "relation-Of Company") (o ?Z))
   (triple (s ?TaxObligation) (p "property-Value") (o ?oblgval))
   (triple (s ?TaxPaid) (p "Object Type") (o "MoneyTransfer"))
   (triple (s ?TaxPaid) (p "property-Transfer Type") (o "Taxpayment"))
   (triple (s ?TaxPaid) (p "property-goodsItemCode") (o ?good1))
   (triple (s ?TaxPaid) (p "relation-by Company") (o ?Z))
   (triple (s ?TaxPaid) (p "property-Value") (o ?paidval))
   (test (> ?oblgval ?paidval))
   (test (neq ?TaxObligation ?TaxPaid))
   =>
   (modify ?f (object "VATFraudster")))
```

This rule modifies the icon of the company which pays obligatory tax, less than it should be. As a result, this company is called a VAT fraudster. Unfortunately, the Palantir Government tool limits objects to one type (some additional types can be deduced only by an engineer according to the given Palantir Dynamic Ontology).



■ Make all variables distinct

?val1 < ?val2

□ Non-existence

?TaxObligation

Object Type: Money transfer

goodsItemCode: ?goodx

Value: ?val2

Transfer Type: Taxobligation

?TaxPaid

Object Type: Money transfer

Value: ?val1

Transfer Type: Taxpayment

goodsItemCode: ?goodx

Figure 2. An exemplary VAT fraudster rule.

5 Conclusions

In this paper we have demonstrated a tool which supports graph-based creation of rules for the Jess engine. The tool integrates data and rules in the Palantir Government tool. Graph-based representation is convenient and intuitive for an untrained analysts. Such tool can be used in many domains where rules and graphs can be employed to support a user in her/his work.

Because of copyright issues connected with the Palantir Government application, we provide only the presentation which contains screenshots of executing the example concerning an analysis of a real-world criminal case. The presentation with a more detailed description is available on the demo site [17].

Acknowledgement. This work was supported by DS-MK 45-102/12 and 45-085/11 DS-PB grants.

References

1. Palantir Government Platform, http://palantir.com/government
2. Palantir Dynamic Ontology, https://wiki.palantir.com/pgdz/palantir-dynamic-ontology-properties.html
3. Jess (Java Expert System Shell), http://jessrules.com/
4. Forgy C., Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19, pp. 17-37, 1982.
5. Object Constraint Language (OCL), v2.0. http://www.omg.org/spec/OCL/2.0/
6. UML-based Rule Modelling Language, http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=URML
7. Halpin T.: Object-Role Modeling: an overview, 2001, http://www.orm.net/pdf/ORMwhitePaper.pdf
8. Lukichev S., Jarrar M.: Graphical Notations for Rule Modeling. In: A. Giurca, D. Gasevic, and K. Taveter (Eds), Handbook of Research on Emerging Rule-based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, 2009
9. Semantics of Business Vocabulary and Business Rules, http://www.omg.org/spec/SBVR/1.0/
10. Grzegorz J. Nalepa, Antoni Ligęza, and Krzysztof Kaczor. 2011. Overview of knowledge formalization with XTT2 rules. In Proceedings of the 5th international conference on Rule-based reasoning, programming, and applications (RuleML'2011), Nick Bassiliades, Guido Governatori, and Adrian Paschke (Eds.). Springer-Verlag, Berlin, Heidelberg, 329-336.
11. Visual Rules, http://www.visual-rules.com/business-rules-management-software-rules-engine.html
12. Drools Guvnor Rules Authoring, http://docs.jboss.org/drools/release/5.4.0.Final/drools-guvnor-docs/html/ch04.html
13. VisiRule, http://www.lpa.co.uk/vsr.htm
14. OntoStudio Graphical Rule Editor, http://ontorule-project.eu/showcase/OntoStudio_Graphical_Rule_Editor
15. Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object oriented and frame-based languages. J. ACM, 42(4):741–843, 1995.
16. Bak J., Jedrzejek C., Falkowski M.: Usage of the Jess Engine, Rules and Ontology to Query a Relational Database. In Proceedings of the 2009 International Symposium on Rule Interchange and Applications (RuleML '09), Guido Governatori, John Hall, and Adrian Paschke (Eds.). Springer-Verlag, Berlin, Heidelberg, 216-230.
17. Demo site: http://draco.kari.put.poznan.pl/ruleml2012/
18. Jedrzejek C., Bak J., Falkowski M., Cybulka J., Nowak M., On the Detection and Analysis of VAT Carousel Crime, in: Frontiers in Artificial Intelligence Applications, vol. 235, Proceedings of JURIX 2011: The Twenty-Fourth Annual Conference Legal Knowledge and Information Systems, pp. 130 – 134, IOS Press, 2011
19. Nowak M., Jedrzejek C., Bak J., Szwabe A., A rule-based expert system for building evidence in VAT-carousel, Proceedings MISSI’12, Multimedia and Internet Systems: New Solutions, in print.