Enhancing semantic consistency in anti-fraud rule-based expert systems

María del Mar Roldán-García, José García-Nieto, José F. Aldana-Montes

Dept. de Lenguajes y Ciencias de la Computación, University of Málaga, ETSI Informática, Campus de Teatinos, Málaga 29071, Spain

Keywords:
Semantic model
Ontology reasoning
Rule-based expert system
Fraud detection expert systems

A B S T R A C T

In this study, an ontology-driven approach is proposed for semantic conflict detection and classification in rule-based expert systems. It focuses on the critical case of anti-fraud rule repositories for the inspection of Card Not Present (CNP) transactions in e-commerce environments. The main motivation is to examine and curate anti-fraud rule datasets to avoid semantic conflicts that could lead the underpinning expert system to incorrectly perform, e. g., by accepting fraudulent transactions and/or by discarding harmless ones. The proposed approach is based on Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) technologies to develop an anti-fraud rule ontology and reasoning tasks, respectively. The three main contributions of this work are: first, the creation of a conceptual knowledge model for describing anti-fraud rules and their relationships; second, the development of semantic rules as conflict-resolution methods for anti-fraud expert systems; third, experimental facts are gathered to evaluate and validate the proposed model. A real-world use case in the e-commerce (e-Tourism) industry is used to explain the ontological knowledge design and its use. The experiments show that ontological approaches can effectively discover and classify conflicts in rule-based expert systems in the field of anti-fraud applications. The proposal is also applicable to other domains where knowledge rule bases are involved.

1. Introduction

Rule-based Expert Systems (RBESs) are the simplest form of artificial intelligence, which uses rules as the representation for encoding knowledge from a fairly narrow area into an automated system (Durkin, 1998). RBESs mimic the reasoning procedure of a human expert when solving a knowledge-intensive problem. A rule-based system consists of a set of `IF-THEN` rules, a set of *facts* and an interpreter controlling the application of the rules, given the facts. Rule-based systems are very simple models and can be adapted and applied to a wide set of different problems, whenever the domain of knowledge can be expressed in the form of `IF-THEN` rules.

In the case of fraud prevention and detection in e-commerce transactions, RBESs are used to identify customers' suspicious activities by automatically generating risk scoring reports of their transactions (Ketkar, Shankar, & Banwet, 2014). They analyze behaviors such as repetitive and quick access attempts, domestic/foreign transactions, and abnormal transactions compared with the customer‘s past behavior. A final decision is then delivered by the system, commonly: *Accept*, *Reject*, or *Revise*. A small subset of rules that might contribute to a negative risk assessment could be as follows (Ward, 2010): A single IP address has been used with multiple payment cards in the last few days; the shopper‘s billing address is more than “x” kilometers from the shipping address; the e-mail address has been flagged in a negative database (black list) of known fraud activity by other merchants participating in the same fraud detection strategy; the BIN (Bank Identification Number) on the payment card indicates the transaction comes from a high-risk country.

Using a combination of these and many other factors could benefit e-merchants, who are presently demanding autonomous expert systems, to quickly update their rule-bases and flag suspicious transactions (Wong, 2013). In the current market, there exist a series of tools that use rule-based knowledge engines for

☆ This work is partially funded by FP7 EU project SME-Ecompass under Grant No: 315637. It is also partially funded by Grants TIN2014-58304 (MINECO) and Regional projects P11-TIC-7529/P12-TIC-1519. Authors specially thanks to etravel.com and in particular to Orestis Papadopoulos to support this work with a set of private rules for testing and validation. José García-Nieto is recipient of a Post-Doctoral fellowship of “Captación de Talento para la Investigación” at Universidad de Málaga.

* Corresponding author.

E-mail addresses: mmar@lcc.uma.es (M.d.M. Roldán-García), jnieto@lcc.uma.es (J. García-Nieto), jfam@lcc.uma.es (J.F. Aldana-Montes).

risk scoring of e-commerce fraud: Simility,[1] Subuno,[2] Riskfield,[3] and Trustev.[4] These tools are widely used not only for tracking and scoring transactions, but also for reporting statistics of the e-commerce site. However, these tools often concentrate on high level and generic sets of rules, without the possibility of considering new *ad hoc* rules specific to each e-commerce site. When these are provided, they are available only for commercial (non-free) versions, which are rarely accessible to SMEs or individual e-merchants.

In this context, the SME-Ecompass European initiative[5] aims to provide e-commerce SMEs with accessible tools for specialized fraud prevention and detection. These software facilities are built on a rule-based expert system for the risk scoring of Card Not Present transactions (CNP). The knowledge rule base can be easily updated by the e-merchant by inserting new rules specific to his/her own e-commerce site. Nevertheless, an increasing number of anti-fraud rules (and their combinations) often provoke the emergence of conflicting rules with semantic inconsistencies. In addition, anti-fraud expert systems face a major challenge as they operate in hostile conditions, as their anticipated inference capabilities are degraded with a continuously changing environment. As a consequence, these issues can lead the underpinning expert system to perform inefficiently (Grosan & Abraham, 2011), e.g., by accepting fraudulent transactions, while discarding harmless ones. Therefore, a key task in anti-fraud applications is to inspect and curate knowledge rule bases to avoid semantic inconsistencies, before delivering a final diagnosis.

With this motivation, an ontology-driven approach is proposed for semantic inconsistent detection and classification in rule-based expert systems. The proposed approach is based on Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) technologies to develop an anti-fraud rule ontology and to perform reasoning tasks, respectively. The three main contributions of this work are:

(i) creating a conceptual knowledge model, in terms of an OWL ontology, to describe anti-fraud rules and their relationships;
(ii) developing semantic SWRL rules as conflict/inconsistency detection methods for anti-fraud expert systems;
(iii) gathering experimental facts to evaluate and validate the proposal.

A real-world use case in the e-commerce (e-Tourism) industry is used to explain the ontological knowledge design and its uses. The experiments show that the proposed semantic approach can effectively discover and classify inconsistencies and conflicts in rule-based expert systems, in the field of anti-fraud applications.

The remainder of this article is organized as follows. The next section presents background concepts and related works in the literature. In Section 3, key concepts in a real-world anti-fraud expert system are explained. Section 4 describes the proposed semantic approach, giving details of the OWL Ontology and the reasoning model. The validation procedure is reported in Section 5. The main conclusions and future work are given in Section 6.

2. Background and literature overview

This section describes the main background concepts and reviews related works in the specialized literature.

Table 1
Basic OWL-DL semantic syntax used to formally define the proposed ontology.

Descriptions	Abstract syntax	DL syntax
Operators	*intersection*$(C_1, C_2, \cdots, C_n)$	$C_1 \sqcap C_2 \sqcap \cdots C_n$
	union$(C_1, C_2, \cdots, C_n)$	$C_1 \sqcup C_2 \sqcup \cdots C_n$
Restrictions	for at least 1 value V from C	$\exists V.C$
	for all values V from C	$\forall V.C$
	R is Symmetric	$R \equiv R^-$
Class Axioms	A *partial*$(C_1, C_2, \cdots, C_n)$	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \cdots C_n$
	A *complete*$(C_1, C_2, \cdots, C_n)$	$A \equiv C_1 \sqcap C_2 \sqcap \cdots C_n$

2.1. Background concepts

- *Ontology.* Ontologies provide a formal representation of the real world by defining concepts and relationships between them (Gruber, 1993). In the context of the computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge. These primitives are typically concepts (or classes), attributes (or properties), class members (class instances) and relationships (property instances). The definitions of the primitives include information about their meaning and constraints on their logically consistent application.

- *RDF.* Graphical language used to represent information about resources on the web (Staab & Studer, 2009). It is a basic ontology language. Resources are described in terms of properties and property values using RDF statements. Statements are represented as triples, consisting of a subject, predicate and object. The RDF Schema (RDFS) (Staab & Studer, 2009) "semantically extends" RDF to enable us to talk about classes of resources, and the properties that will be used with them.

- *SPARQL.* It is a query language for ontology models and databases, capable of extracting and manipulating information stored in RDF format. Essentially, SPARQL is a graph-matching query language that can be used to extract knowledge from a model like the one proposed in this article. Given a data source D, a query is a pattern, which is matched against D. The combinations of values resulting from this matching constitute the result of the query (Pérez, Arenas, & Gutierrez, 2009).

- *OWL.* In 2004, the W3C ontology working group (Dean & Schreiber, 2004) proposed OWL as a semantic markup language for publishing and sharing ontologies. From a formal point of view, OWL is equivalent to a very expressive description logic where an ontology corresponds to a Tbox (Gruber, 1993). This equivalence allows the language to exploit description logic researcher results. OWL extends RDF and RDFS. When compared to RDF models, OWL adds more vocabulary for describing properties and classes, among others: relationships between classes (e.g. disjointedness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes (McGuinness & Harmelen, 2004).

- *OWL-DL.* It is a syntactic variant of the SHOIN (D) description logic (Haase & Stojanovic, 2005) with a different terminology to OWL, which is based on RDF(S). Therefore, it supports data values, data types and data type properties. OWL-DL restricts OWL in two distinct ways (Horrocks & Patel-Schneider, 2003): first, some syntactic constructs, e.g., recursive descriptions in them are not permitted; second, classes, individuals and properties (respectively concepts, individuals and roles in description logics) must all be disjoint. In this approach, we use the OWL-DL syntax to formalize the proposed ontology for our semantic model. A description of the basic OWL-DL semantic syntax is shown in Table 1, where an informal logic syntax is represented (left-hand column) with regards to the corresponding OWL-DL equivalent (right).

[1] In URL https://simility.com/.
[2] In URL http://www.subuno.com/.
[3] In URL http://www.riskified.com/.
[4] In URL http://www.trustev.com/.
[5] SME-Ecompass FP7 European initiative http://www.sme-ecompass.eu/.

- *SWRL.* The Semantic Web Rule Language provides the OWL-based ontologies with procedural knowledge, which compensates for some of the limitations of ontology inference, particularly in identifying semantic relationships between individuals (Horrocks, Patel-Schneider, Bechhofer, & Tsarkov, 2005). SWRL uses the typical logic expression "Antecedent $\Rightarrow$ Consequent" to represent semantic rules. Both antecedent (rule body) and consequent (rule head) can be conjunctions of one or more atoms written as "$atom_1 \wedge atom_2 \wedge \cdots \wedge atom_n$". Each atom is attached to one or more parameters represented by a question mark and a variable (e.g., $?x$). The most common uses of SWRL include transferring characteristics and inferring the existence of new individuals (Grosof & Poon, 2004). Further information on SWRL syntax can be found in the W3C web site.[6]

2.2. Literature overview

The use of ontologies and ontology-related technologies for building knowledge databases for anti-fraud systems is considered quite beneficial for two main reasons (Alexopoulos, Kafentzis, Benetou, Tagaris, & Georgolios, 2007): (1) Ontologies provide an excellent way of capturing and representing domain knowledge, principally due to their expressive power; (2) A number of well-established methodologies, languages and tools (Gómez-Pérez, Corcho, , & Fernandez-Lopez, 2004) developed in the field of Ontological Engineering can make the building of the knowledge base easier, more accurate and more efficient, especially in the knowledge acquisition stage which is usually a bottleneck in the whole ontology development process. In this sense, Alexopoulos et al. (2007) proposed a methodology for building domain specific ontologies in the e-government domain. The main characteristic of this methodology is a generic fraud ontology that serves as the common basis on which the various domain-specific fraud ontologies can be built.

At the same time, Fang, Cai, Fu, and Dong (2007) proposed a novel method built upon ontology and ontology instance similarity for checking user behavior in CNP scenarios. Ontology is widely used to enable knowledge sharing and reuse, so some personality ontologies can be easily used to represent user behavior. By measuring the similarity of ontology instances, the authors were able to determine whether an account had been defrauded. This method decreases the data model cost and makes the system adaptable to different applications.

From a different perspective, Ramaki, Asgari, and Atani (2012) presented a technique for detecting abnormal credit card operations by exploiting an ontology. Specifically, this work used an ontology graph to model each user‘s transaction behavior and then store it in the system. During abnormality detection, only those transactions from a registered record of transactions, which are similar to entry ones, are selected for computation.

In Hu et al. (2013), a case study applying semantic technologies to social benefit fraud detection is described. The authors claim that design considerations, study outcomes, and lessons learnt could help in making decisions of how one should adopt semantic technologies in similar contexts. In a nutshell, by leveraging semantic technology, organizations are able to dynamically describe new fraud cases and facilitate the integration, analysis, and visualization of disparate and heterogeneous data from multiple sources. In addition, by generating semantic fraud detection rules, they can manage to convert labor intensive tasks into (semi-) automated processes.

Finally, in Rajput, Khan, Larik, and Haider (2014) the authors proposed an effective mechanism to detect suspicious transactions by designing an ontology based expert system. The proposed ontology consists of domain knowledge and a set of SWRL rules that together constitute an expert system. The native reasoning support in the ontology is used to deduce new knowledge from the predefined rules about suspicious transactions.

All these approaches apply a domain ontology, which tries to represent the knowledge of an expert in the domain of knowledge to pre/post rule mining processes. The semantic model proposed here seeks to go one step further as it makes use of a domain ontology, as well as an ontology representing the e-commerce rules domain itself. This is done by modeling the attributes and operators that are employed to build such rules. Furthermore, this approach is able to exploit the reasoning capabilities of the ontology to detect semantic conflicts in rules. For the sake of a better understanding, Table 2 outlines the main features of the related work with regards to the semantic approach proposed here. These features consist of specifying whether the existing approaches: are general purpose anti-fraud ontologies, focus on CNP transactions, focus on user behavior, based on knowledge rule-bases, support an expert system, and/or use SWRL reasoning rules.

3. Anti-fraud expert system for CNP transactions

Data representing the credit card usage profiles of the customers consist of variables, each of which discloses a behavioral characteristic. A profile allows the merchant to differentiate among various business segments. The variables may show the spending habits of the customer with respect to geographical locations, days of the month, hours of the day or Merchant Category Codes (Hand & Blunt, 2009). Credit card data comprises close to 70 variables per transaction: Transaction ID, transaction type, date and time of transaction (to nearest second), amount, currency, local currency amount, merchant category, card issuer, and chip card verification results, are among the most used. Using combinations of these and many other variables, it is possible to formulate a set of behavior rules (in the IF-THEN format) that compose rule-based knowledge engines for risk scoring in CNP transactions.

An innovative example of a knowledge rule base system has been recently released in the SME-Ecompass European initiative,[7] whose principal aim is to provide e-commerce SMEs with accessible tools for specialized fraud prevention and detection. This knowledge rule base can be easily updated by the e-merchants by inserting new rules specific to their own e-commerce sites. In addition, based on previous experience, a series of shared black and white lists of suspicious attributes are integrated in the application that allow the user to consider highly specific fraudulent cases.

In SME-Ecompass‘s anti-fraud services, each newly arrived purchase order (in a given e-shop) flows through the inference engine and receives a risk score (RS) depending on its characteristics. This score reflects the confidence with which the order can be regarded as fraudulent. Once scoring is completed, the transaction is routed according to the three-event fraud-detection protocol:

- If the risk score is above an upper cut-off point (threshold), the order is accepted and executed automatically.
- If the risk score is below a lower cut-off point, the order is rejected without further notice.
- If the risk score lies between lower and upper cut-off points, the order is sent to fraud analysts for further investigation (under quarantine).

A fraud assessment rule has a series of main attributes that are described in Table 3. The rule logic is on the antecedent side. It is composed of one or more "Rule Conditions" (RC), which are

[6] https://www.w3.org/Submission/SWRL/.

[7] SME-Ecompass FP7 European Initiative http://www.sme-ecompass.eu/.

Table 2
Summary ontologies' main features with regards to the proposed approach.

Feature/Ontology	Alexopoulos et al. (2007)	Fang et al. (2007)	Ramaki et al. (2012)	Hu et al. (2013)	Rajput et al. (2014)	Proposal
Generic anfi-fraud	✓			✓		✓
CNP transactions		✓	✓		✓	✓
User behavior		✓	✓	✓		✓
Expert systems					✓	✓
SWRL reasoning	✓				✓	✓
knowledge rule-base						✓

Table 3
Main attributes of the fraud assessment rules in the SME-Ecompass knowledge rule base.

Attribute	Description
id	A unique key, that uniquely identifies the rule
name	A name that visually distinguishes the rule
profile_id	The profile that owns the particular rule. A profile allows the merchant to differentiate among various business segments
type	Identifies the type of the rule
order	The position of the rule relative to the other rules in the same profile
active	Whether the rule is active in the system or not. Only active rules are taken into account when evaluating a case
score	An integer number that is assigned to the case being evaluated when the rule matches (becomes true)
result	"Accept", "Reject" or "Review". When present, it will turn the rules into decision rules. Decision rules, when matched, assign their result to case fraud assessment result irrespective of the score

Table 4
Main attributes used in rule conditions in the SME-Ecompass knowledge rule base.

Attribute	Description
id	Uniquely identifies the rule condition
condition_group_id	Groups rule conditions
left_argument_attribute_id	The reference to the left operand. The left operand will always be a session attribute
operator	The operator that will be used to operate the left operand over the right one. This will be a string representation of the operator as described in Table 5
argument_type	Literal, List or Attribute. This defines the type of the right operand, which might be: a custom value (Literal), a reference to a List or a reference to a session attribute
literal_argument_value	If the right operand is a custom value and its argument_type is Literal, then this attribute will hold the actual custom value. This will be a string representation of the value
literal_argument_data_type	If the right operand is a custom value, it will be specified by the datatype. The datatype is specified with one of the string values, as cataloged in Table 5: "String", "Country", "Number", "Date", "Flag", "Email"
right_argument_attribute_id	A foreign key to the session attribute that will be used as the right operand of the condition. This will take a value only if the argument_type column has the value "Attribute"
right_argument_list_id	A foreign key to the "List" that will be used as the right operand of the condition. This will take a value only if the argument_type column has the value "List"

Table 5
List of possible operators to be used in rule conditions, with regards to the datatype in the SME-Ecompass knowledge rule base.

Datatype	Operators
String	Equals, NotEquals, Contains, DoesNotContain, Matches, DoesNotMatch, EndsWithAnyFromList, IncludedInList, NotIncludedInList, DoesNotWithAnyFromList, ContainsAnyFromList, DoesNotContainAnyFromList
Number	Equals, NotEquals, GreaterThan, LessThan, GreaterThanOrEquals, LessThanOrEquals
Flag	Equals, NotEquals
Email	Equals, NotEquals, Contains, DoesNotContain, Matches, DoesNotMatch, IncludedInList NotIncludedInList, EndsWithAnyFromList, DoesNotEndWithAnyFromList ContainsAnyFromListOperator, DoesNotContainAnyFromListOperator
Date	Equals, NotEquals, Before, After, SameDate, DifferentDate
Country	Equals, NotEquals, IncludedInList, NotIncludedInList

joined in "Condition Groups" (CG). The condition group will return a true or false value depending on the true or false value that each one of its rule conditions returns, i.e., by internally computing the boolean operator AND. In order for a rule to return true, at least one condition group should return true, that is, by computing the boolean operator OR. A rule condition is specified by a boolean condition between the left and right operands. If the value of the left operand matches the value of the right one, then the rule condition will return true. Otherwise, it will return false. The attributes involved in the rule conditions are described in Table 4 and the set of possible operators to be used are shown in Table 5, in accordance with their corresponding data types. To illustrate this, Fig. 1 shows the structure of a typical rule where some condition groups are formulated in the antecedent side. In this example, the first condition group is composed of several condition rules, in which the attribute "BIN Country" is operated with regards to "Device Country" and "Similar Countries to GB". In the consequent side, the preliminary result and the score of the rule are specified.

The complete rule-based expert system comprises a well-grounded set of fraud assessment rules (like the one explained in Fig. 1), and the final score is then computed as an aggregation of all the partial scores of the rules. Finally, the three-event fraud-

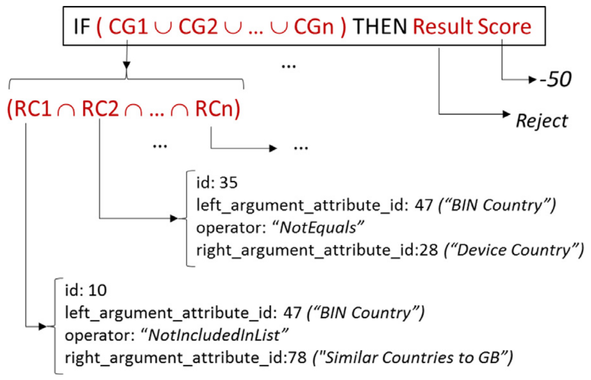

IF (CG1 ∪ CG2 ∪ ... ∪ CGn) THEN Result Score
...
-50
Reject
(RC1 ∩ RC2 ∩ ... ∩ RCn)
...
...
id: 35
left_argument_attribute_id: 47 ("BIN Country")
operator: "NotEquals"
right_argument_attribute_id:28 ("Device Country")
id: 10
left_argument_attribute_id: 47 ("BIN Country")
operator: "NotIncludedInList"
right_argument_attribute_id:78 ("Similar Countries to GB")

Fig. 1. Example of anti-fraud rule with condition groups (CG) and rule conditions (RC) in which the attribute "BIN Country" is operated with regards to "Device Country" and "Similar Countries to GB" (GB = Great Britain).

detection protocol is applied to recommend a final decision: "Accept", "Reject" or "Review". In the example of Fig. 1, assuming a lower cut-off point of 0, the score is -50 and the result is "Reject".

4. Semantic approach

In general, in anti-fraud detection services, the analytic algorithms must examine the transactional data in real time in order to quickly detect whether a transaction is suspicious, or not, by applying a set of rules. The user can define his/her own rules and insert them in the expert knowledge rule base. It is logical to think that the number of rules will be large and therefore some kind of filtering and post-processing task must be carried out after inserting a new rule. For example, pairs of rules for which items in the antecedent are semantically correlated can be simplified as one single association rule, either more comprehensive or more specific, depending on the domain.

In this context, the semantic web offers a series of useful methods for processing a set of rules by bringing information from the specific domain of knowledge, to filter and analyze the rules. Specifically, the semantic knowledge can be provided by a domain ontology, so semantic methods based on this ontology can be used for cleaning redundant and inconsistent rules.

To this end, the semantic model proposed is driven by an OWL ontology that covers all the concepts and relationships concerning the anti-fraud assessment rules, although focusing on the real-world case of SME-Ecompass anti-fraud service.

The proposed ontology has been designed by following the standard Ontology 101 development process (Noy & McGuinness, 2001) comprising seven steps:

1. **Determine the domain and scope of the ontology**: As a starting point, the ontology definition is based on the relational database that constitutes the data model of the SME-Ecompass anti-fraud service. In this model, the anti-fraud rules are related to a user profile as explained in Section 3.
2. **Consider reusing existing ontologies**: To the best of our knowledge, there are no existing similar ontologies for modeling anti-fraud rules. Moreover, we could not find an ontology or vocabularies for modeling associate rules, so the proposed ontology here has been designed from scratch.
3. **Enumerate important terms in the ontology**: Important terms in the ontology were extracted from the expert rule-based relational schema. Examples of such terms are: profile, rule, condition group, rule condition, operator, email, country, date, score, result, argument type, etc. These terms are described in Tables 3–5.
4. **Define classes and the class hierarchy**: The initial list of ontology classes is obtained from the list of important terms. A general overview of the ontology class diagram is illustrated in Fig. 2, where the main classes are: *Rules, ConditionGroups, RuleConditions* and *Operators*. A set of subclasses is also defined to create a class hierarchy, to classify rules and to compare them. For example, *Inconsistent_condition_groups* is a subclass of *ConditionGroups* whose members are inconsistent condition groups.
5. **Define the properties of classes and slots**: In order to link related classes and to define their attributes, objects and data properties are identified. Examples of object and data properties are reported inTables 6–9 with their definitions in description logic (as specified in Table 1): a condition group has conditions, a rule has condition groups, a rule condition has operators, etc. Some additional object properties are included to relate rules, for example: two rules can be contradictory and two rules can be the same.
6. **Define the facets of the slots**: This step includes the definition of cardinality constraints and value restrictions. In order to classify and to compare condition groups and rules, several value restrictions are needed. For example, a *ConditionGroups1* is a condition group whose individuals are those with only 1 condition. Similarly, *Rules2* are those rules whose individuals have two condition groups.
7. **Create instances**: Instances (individuals in OWL) correspond to the specific rule obtained from the expert rule base. Individuals are obtained by mapping the relational database to RDF in accordance with the ontology. Furthermore, operators of rule conditions are also included as ontology individuals. For example: *Contains, DoesNotMatch, IncludedInList, NotEqual*, etc., are individuals of the class *Operator*. In addition, to detect contradictory rule conditions, the opposite operator of each operator has to be specified. Instances of the object property *oppositedOperators* are: *(Matches, DoesNotMatch), (Contains, DoesNotContain)*, etc.

4.1. Ontology knowledge model

The proposed ontology, called "`afro.owl`" (Anti Fraud Rules Ontology), resulting from the development process described above has a total number of 19 classes (groups of individuals sharing the same attributes), 12 object properties (binary relationships between individuals), and 16 data properties (individual attributes), 99 restriction axioms and 16 individuals. The complete ontology is available in the GitHub repository.[8]

For simplicity, we describe here a representative subset of the four main classes including some of their most interesting object and data properties. These classes are: *ConditionGroups, RuleConditions, Operators*, and *Rules*. Each class requires a set of properties or conditions in order to be conceptualized. An individual that satisfies those properties is considered a member of that class.

- *ConditionGroups*. This class represents the condition groups in the antecedent part of a rule. Each condition group has a number of conditions expressed as data property in Table 6, as well as object properties oriented to consider overlapping, same, and tautological condition groups. This entity can be classified depending on its number of conditions. Therefore, *ConditionGroups1, ConditionGroups2, ConditionGroups3, ConditionGroups4, ConditionGroups5* and *ConditionGroups6* are subclasses of *ConditionGroups*. Finally, *inconsistent_condition_groups* is also modeled as a subclass of *ConditionGroups*.

- *RuleConditions*. It specifies a boolean condition with a left and right operand. If the value of the left operand matches the

[8] URL link https://github.com/KhaosResearch/AFRO.

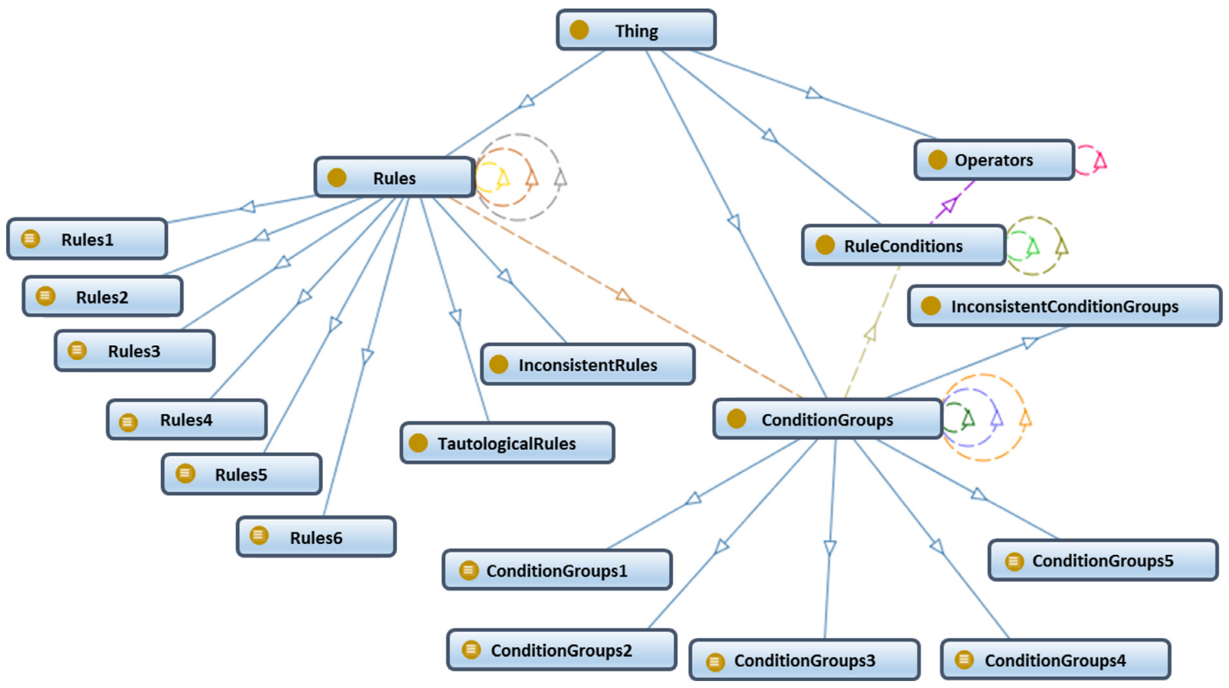


Fig. 2. General overview of the "afro" ontology. Solid arrows mark sub class of. Dotted arrows mark specific properties.

Table 6
ConditionGroups: object and data properties.

Object properties	Description logic
hasCondition	∃ hasCondition.Thing ⊑ ConditionGroups ⊤ ⊑ ∀ hasCondition.RuleConditions
overlappedConditionGroups	∃ overlappedConditionGroups.Thing ⊑ ConditionGroups ⊤ ⊑ ∀ overlappedConditionGroups ConditionGroups
sameConditionGroups	TransitiveProperty sameConditionGroups ∃ sameConditionGroups.Thing ⊑ ConditionGroups ⊤ ⊑ ∀ sameConditionGroups ConditionGroups
tautologicalConditionGroups	∃ tautologicalConditionGroups.Thing ⊑ ConditionGroups ⊤ ⊑ ∀ tautologicalConditionGroups.ConditionGroups
Data Properties	Description Logic
hasNumberOfConditions	∃ hasNumberOfConditions.Datatype Literal⊑ ConditionGroups

Table 7
RuleConditions: object and data properties.

Object properties	Description logic
hasOperator	∃ hasOperator.Thing ⊑ RuleConditions ⊤ ⊑ ∀ hasOperator Operators
inconsistentRuleConditions	∃ inconsistentRuleConditions.Thing ⊑ RuleConditions ⊤ ⊑ ∀ inconsistentRuleConditions.RuleConditions
sameRuleConditions	TransitiveProperty sameRuleConditions ∃ sameRuleConditions.Thing ⊑ RuleConditions ⊤ ⊑ ∀ sameRuleConditions.RuleConditions
Data Properties	Description Logic
hasArgumentDataType	∃ hasArgumentDataType.Datatype Literal ⊑ RuleConditions
hasArgumentType	∃ hasArgumentType.Datatype Literal ⊑ RuleConditions
hasLeftArgumentAttributeID	∃ hasLeftArgumentAttributeID.Datatype Literal ⊑ RuleConditions
hasLiteralArgumentDataType	∃ hasLiteralArgumentDataType.Datatype Literal ⊑ RuleConditions
hasLiteralArgumentValue	∃ hasLiteralArgumentValue.Datatype Literal ⊑ RuleConditions
hasRightArgumentAttributeID	∃ hasRightArgumentAttributeID.Datatype Literal ⊑ RuleConditions
hasRightArgumentListID	∃ hasRightArgumentListID.Datatype Literal ⊑ RuleConditions

value of the right one, then the rule condition will return true (otherwise, false). Object and data properties of rule conditions are formally described in Table 7.

- *Operators*. This class is modeled with just one object property: *oppositedOperators* as described in Table 8, but comprising a set of members related to operators with data types for boolean operations in rule conditions, as specified in Table 5.

- *Rules*. This class represents the anti-fraud rules. Each rule belongs to a profile, contains one or more condition groups and has a name, a result, a score, an URL and an attribute to indicate whether the rule is active, or not (see Table 9). Furthermore, the *overlappedRules* and *contradictoryRules* properties relate two rules which are overlapping or contradictory, respectively (see Section 4.2). Rules are classified in accordance with their number of condition groups.

Table 8
Operators: object properties.

Object properties	Description logic
oppositedOperators	∃ oppositedOperators.Thing ⊑ Operators ⊤ ⊑ ∀ oppositedOperators.Operators

Therefore, *Rules1, Rules2, Rules3, Rules4, Rules5* and *Rules6* are subclasses of *Rules*. Finally, *InconsistentRules* and *TautologicalRules* are also subclasses of Rules.

4.2. Semantic rules

The semantic rules are built on top of the OWL ontology to deduce new information from the existing knowledge. It is worth clarifying that semantic rules are formulated in SWRL and used to perform semantic reasoning tasks (to detect errors in anti-fraud rules), whereas anti-fraud rules, as explained in Section 3, are used to compose the knowledge rule base of the anti-fraud expert system (to identify suspicious CNP transactions).

In the proposed inference model, after a deep inspection of the existing anti-fraud rules, a series of OWL subclasses and axioms have been defined to classify condition groups with 1,2,3,4 and 5 conditions, as well as rules with 1,2,3,4,5, and 6 condition groups, based on the data properties *hasNumberOfConditions* and *hasNumberOfConditionGroups* (see Table 10). These subclasses have been explicitly defined in the ontology for practical and efficiency issues, since using data properties to consider the number of condition groups/conditions led the reasoner to explore an extensive number of combinations, hence to perform inefficiently.

Then, a set of SWRL rules have been defined to infer possible mistakes in the anti-fraud rule dataset. They are intended to discover anti-fraud rules that are: duplicated (same rules), overlapping, inconsistent, tautological and contradictory. In this way, the SWRL rules are evaluated by the reasoner after classifying rules and condition groups in accordance with axioms in Table 10.

• **Same rules**. Those anti-fraud rules that belong to the same profile and have the same antecedent and consequent parts, but with different IDs (i.e. duplicated). To detect them, it is necessary to previously check the existence of **same condition groups** (also **same conditions**) in the antecedent part. Rules with the same conditions have the same parameters, but with different IDs. In all likelihood, the human-expert (unintentionally) duplicated the conditions in the rule database. An example of SWRL code to detect "same rules" with two condition groups is:

Table 10
Additional subclasses to enhance the reasoning tasks.

Subclass	Description logic
ConditionGroups1	≡ *hasValue*.hasNumberOfConditions "1" ⊑ ConditionGroups
ConditionGroups2	≡ *hasValue*.hasNumberOfConditions "2" ⊑ ConditionGroups
ConditionGroups3	≡ *hasValue*.hasNumberOfConditions "3" ⊑ ConditionGroups
ConditionGroups4	≡ *hasValue*.hasNumberOfConditions "4" ⊑ ConditionGroups
ConditionGroups5	≡ *hasValue*.hasNumberOfConditions "5" ⊑ ConditionGroups
Rules1	≡ *hasValue*.hasNumberOfConditionGroups "1" ⊑ Rules
Rules2	≡ *hasValue*.hasNumberOfConditionGroups "2" ⊑ Rules
Rules3	≡ *hasValue*.hasNumberOfConditionGroups "3" ⊑ Rules
Rules4	≡ *hasValue*.hasNumberOfConditionGroups "4" ⊑ Rules
Rules5	≡ *hasValue*.hasNumberOfConditionGroups "5" ⊑ Rules
Rules6	≡ *hasValue*.hasNumberOfConditionGroups "6" ⊑ Rules

```
Rules2(?x) ^ Rules2(?y)
^ hasConditionGroup(?x, ?a)
^ hasConditionGroup(?x, ?b)
^ hasConditionGroup(?y, ?c)
^ hasConditionGroup(?y, ?d)
^ sameConditionGroups(?a, ?c)
^ sameConditionGroups(?b, ?d)
^ hasId(?a, ?ida) ^ hasId(?b, ?idb)
^ hasId(?c, ?idc) ^ hasId(?d, ?idd)
^ hasId(?x, ?idx) ^ hasId(?y, ?idy)
^hasProfileId(?x,?p) ^hasProfileId(?y,?p)
^ hasResult(?x, ?r) ^ hasResult(?y, ?r)
^ hasScore(?x, ?s) ^ hasScore(?y, ?s)
^lessThan(?idx, ?idy) ^notEqual(?ida,?idb)
^ notEqual(?idc, ?idd)
-> sameRules(?x, ?y)
```

This example uses `sameConditionGroups(?x, ?y)`, which is the consequent of others SWRL rules formulated in accordance with the number of condition groups in the antecedent part. The reason is that it should consider all the combinations of properties

Table 9
Rules: object and data properties.

Object properties	Description logic
hasConditionGroup	∃ hasConditionGroup.Thing ⊑ Rules ⊤ ⊑ ∀ hasConditionGroup.ConditionGroups
contradictoryRules	∃ hasOperator.Thing ⊑ RuleConditions ∃ contradictoryRules.Thing ⊑ Rules
overlappedRules	∃ overlappedRules.Thing ⊑ Rules ⊤ ⊑ ∀ overlapped_rules.Rules
sameRules	TransitiveProperty sameRules ∃ sameRules.Thing ⊑ Rules ⊤ ⊑ ∀ sameRules.Rules
Data Properties	Description Logic
hasName	∃ hasName.Datatype Literal ⊑ Rules
hasNumberOfConditionGroups	∃ hasNumberOfConditionGroups.Datatype Literal ⊑ Rules
hasProfileId	∃ hasProfileId.Datatype Literal ⊑ Rules
hasResult	∃ hasResult.Datatype Literal ⊑ Rules
hasScore	∃ hasScore.Datatype Literal ⊑ Rules
hasURL	∃ hasURL.Datatype Literal ⊑ Rules
isActive	∃ isActive.Datatype Literal ⊑ Rules

to satisfy the required inference task. An example of a SWRL rule with 4 condition groups is as follows:

```
ConditionGroups4(?x)
^ ConditionGroups4(?y)
^ hasCondition(?x, ?a)
^ hasCondition(?x, ?b)
^ hasCondition(?x, ?c)
^ hasCondition(?x, ?d)
^ hasCondition(?y, ?e)
^ hasCondition(?y, ?f)
^ hasCondition(?y, ?g)
^ hasCondition(?y, ?h)
^ sameRuleConditions(?a, ?e)
^ sameRuleConditions(?b, ?f)
^ sameRuleConditions(?c, ?g)
^ sameRuleConditions(?d, ?h)
^ hasId(?a, ?ida) ^ hasId(?b, ?idb)
^ hasId(?c, ?idc) ^ hasId(?d, ?idd)
^ hasId(?e, ?ide) ^ hasId(?f, ?idf)
^ hasId(?g, ?idg) ^ hasId(?h, ?idh)
^ hasId(?x, ?idx) ^ hasId(?y, ?idy)
^ lessThan(?idx, ?idy)
^ notEqual(?ida, ?idb)
^ notEqual(?ida, ?idc)
^ notEqual(?ida, ?idd)
^ notEqual(?idb, ?idc)
^ notEqual(?idb, ?idd)
^ notEqual(?idc, ?idd)
^ notEqual(?ide, ?idf)
^ notEqual(?ide, ?idg)
^ notEqual(?ide, ?idh)
^ notEqual(?idf, ?idg)
^ notEqual(?idf, ?idh)
^ notEqual(?idg, ?idh)
-> sameConditionGroups(?x, ?y)
```

Similarly, this rule uses the function `same_rule_conditions(?x, ?y)`, which checks all the left arguments and operators to detect those anti-fraud rules in the knowledge base that share the same condition.

The SWRL code to obtain “same rule conditions” is as follows:

```
RuleConditions(?x) ^ RuleConditions(?y)
^ hasId(?x, ?idx)
^ hasId(?y, ?idy)
^ hasLeftArgumentAttributeID(?x, ?lai)
^ hasLeftArgumentAttributeID(?y, ?lai)
^ hasLiteralArgumentValue(?x, ?lav)
^ hasLiteralArgumentValue(?y, ?lav)
^ hasOperator(?x, ?o)
^ hasOperator(?y, ?o) ^ lessThan(?idx, ?idy)
-> sameRuleConditions(?x, ?y)
```

• **Overlapping rules**. Those rules that belong to the same profile and the antecedent part of one of them is contained in the antecedent part of the other. Therefore, the overlapping rule can be deleted. In the example below, the rule with ID 5648 can be removed from the rule database because it is overlapped by the rule with ID 5656.

```
ProfileID:164;
RuleID:5648;
IF-Part: "Credit Card Hash"
      IsIncludedInList
      "Black list - credit card hash";
Score: 0;
Result: Reject

ProfileID:164;
RuleID:5656;
IF-Part: "Credit Card Hash"
  IsIncludedInList
    "Black list - credit card hash"
       OR
       "Bin Country" NotEqual "Andorra";
Score: 0;
Result: Reject
```

The SWRL code to infer the existence of overlapping rules of this example is as follows:

```
Rules1(?x) ^ Rules3(?y)
^ hasConditionGroup(?x, ?a)
^ hasConditionGroup(?y, ?b)
^ same_condition_groups(?a, ?b)
^ hasId(?x, ?idx) ^ hasId(?y, ?idy)
^ hasProfileId(?x, ?p)
^ hasProfileId(?y, ?p)
^ hasResult(?x, ?r) ^ hasResult(?y, ?r)
^ hasScore(?x, ?s) ^ hasScore(?y, ?s)
^ lessThan(?idx, ?idy)
-> overlappedRules(?x, ?y)
```

• **Inconsistent rules**. Rules with inconsistent conditions and/or condition groups. Therefore, they cannot be applied at the same time. Inconsistent conditions are those with inconsistent boolean operators. For example, `(''Number of Access from Same IP'' GreaterThan 2) AND (''Number of Access from Same IP'' LessThan 2)`. The function `inconsistent_rule_conditions(?x, ?y)` is formulated to cover condition groups.

The SWRL code to infer inconsistencies are:

```
RuleConditions(?x)
^ RuleConditions(?y)
^ hasOperator(?x, ?op1)
^ hasOperator(?y, ?op2)
^ oppositedOperators(?op1, ?op2)
^ hasId(?x, ?idx) ^ hasId(?y, ?idy)
^ hasLeftArgumentAttributeID(?x, ?lai)
^ hasLeftArgumentAttributeID(?y, ?lai)
^ hasRightArgumentAttributeID(?x, ?lav)
^ hasRightArgumentAttributeID(?y, ?lav)
^ lessThan(?idx, ?idy)
->
inconsistentRuleConditions(?x, ?y)

ConditionGroups(?cg)
^ hasCondition(?cg, ?c1)
^ hasCondition(?cg, ?c2)
^ inconsistentRuleConditions(?c1, ?c2)
-> InconsistentConditionGroups(?cg)

Rules(?x)
^ InconsistentConditionGroups(?cg)
^ hasConditionGroup(?x, ?cg)
-> InconsistentRules(?x)
```

• **Tautological rules**. Those rules that always satisfy, that is, their IF-Part that is always true. For example:

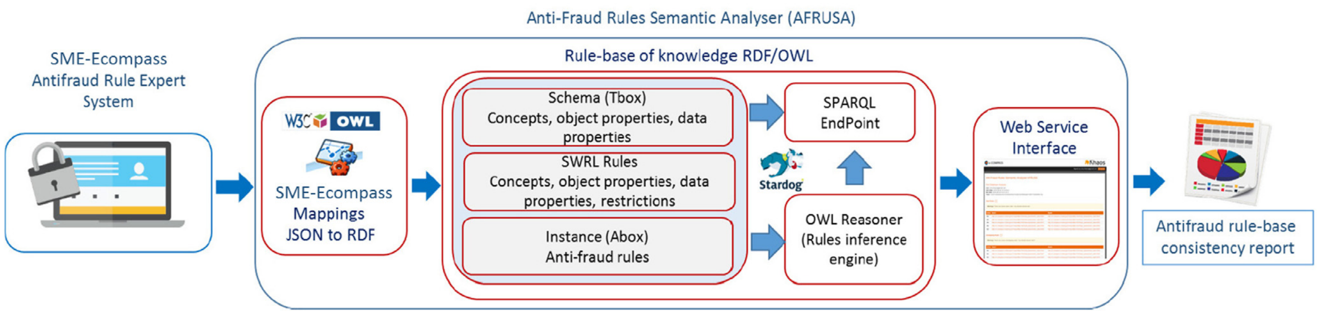

Anti-Fraud Rules Semantic Analyser (AFRUSA)
SME-Ecompass Antifraud Rule Expert System
Rule-base of knowledge RDF/OWL
OWL
SME-Ecompass Mappings JSON to RDF
Schema (Tbox) Concepts, object properties, data properties
SWRL Rules Concepts, object properties, data properties, restrictions
Instance (Abox) Anti-fraud rules
Stardog
SPARQL EndPoint
OWL Reasoner (Rules inference engine)
Web Service Interface
Antifraud rule-base consistency report

Fig. 3. Overview of the anti-fraud rules semantic analyzer.

```
ProfileID:164;
RuleID:3963;
IF-Part: "Device city" Equals "Malaga"
     OR
     "Device city" NotEquals "Malaga";
Score: 0;
Result: Accept
```

Therefore, these rule can be deleted from the knowledge rule base. The corresponding SWRL code is as follows:

```
ConditionGroups1(?cg1)
^ ConditionGroups1(?cg2)
^ hasCondition(?cg1, ?c1)
^ hasCondition(?cg2, ?c2)
^ inconsistentConditionRule(?c1, ?c2)
^ hasId(?cg1, ?idx)
^ hasId(?cg2, ?idy)
^ lessThan(?idx, ?idy)
->
tautologicalConditionGroups(?cg1, ?cg2)

Rules(?x)
^ hasConditionGroup(?x, ?cg1)
^ hasConditionGroup(?x, ?cg2)
^tautologicalConditionGroups(?cg1,?cg2)
-> TautologicalRule(?x)
```

• **Contradictory rules**. Rules that are formulated with the same antecedent, but with a different consequent. That is, they are accepted and rejected at the same time. The SWRL code to infer the existence of contradictory rules is:

```
Rules1(?x) ^ Rules1(?y)
^ hasConditionGroup(?x, ?a)
^ hasConditionGroup(?y, ?b)
^ same_condition_group(?a, ?b)
^ hasId(?a, ?ida) ^ hasId(?b, ?idb)
^ hasId(?x, ?idx) ^ hasId(?y, ?idy)
^ hasProfileId(?x, ?p)
^ hasProfileId(?y, ?p)
^ hasResult(?x, ?r1) ^ hasResult(?y, ?r2)
^ hasScore(?x, ?s) ^ hasScore(?y, ?s)
^lessThan(?idx,?idy)^notEqual(?r1,?r2)
->
contradictory_rules(?x, ?y)
```

4.3. Overall approach

The proposed approach, called AFRUSA (Anti-Fraud Rules Semantic Analyzer), consists of a Java Web service that connects to the SME-Ecompass Anti-fraud application to assist in rule curation and conflict resolution. An overview of the AFRUSA approach is illustrated in Fig. 3. According to the figure, the anti-fraud rules are obtained in form of JSON files, which are served by a REST-API hosted in the SME-Ecompass application. A set of mapping functions are then used to translate the anti-fraud rules from JSON to triples in RDF format, which follow the ontology knowledge model, as explained in Section 4.1.

The anti-fraud rules in RDF format (Abox) are then stored in a Stardog[9] repository, which is a commercial version of the Pellet OWL 2 reasoner (Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007), but enhanced with persistence capabilities. Once the ontology (Tbox) has been loaded together with the SWRL rules, a series of reasoning tasks are launched by using the Stardog OWL 2 reasoner to derive new information that is not explicitly expressed in the knowledge rule base. The new information will indicate, when applicable, which of the analyzed anti-fraud rules are: duplicated, overlapping, inconsistent, contradictory or tautological.

Finally, the SWRL rules and a series of SPARQL queries are evaluated by means of the Stardog reasoner and EndPoint, so that a final report with the results of the analysis is issued to the human-expert through the web interface.[10]

5. Validation

For the validation of the proposed semantic model, a real-world database is used, which comprises 2155 anti-fraud rules generated in the context of the e-Tourism site eTravel.com. This company offers services for reserving and booking flights, hotels, cars and cruises to thousands of users around the world. Most of these operations are successfully booked through on-line CNP transactions, thus the use of efficient fraud prevention applications is highly recommendable to detect and examine suspicious clients.

The complete AFRUSA service, comprising: the semantic model, Stardog repository, reasoner, SPARQL EndPoint, and web interface, is deployed on a Linux CentOS machine with Intel(R) Xeon(R) CPU (2 core) 2.70GHz and RAM 4GBs. In this scenario, an analysis of the rule database of eTravel.com takes 5 min on average, which entails a set of SPARQL queries to infer problematic rules. It is worth saying that the Stardog reasoning[11] covers all the OWL 2 profiles as well as user-defined rules via SWRL. This ensures that the reasoning task of AFRUSA detects all the errors (duplicates, overlapping, etc.) in anti-fraud rules taking into account the defined SWRL rules. This was indeed manually tested for the knowledge rule-base used in this experimentation (eTravel.com).

The SPARQL queries are specified in Table 11, together with a brief description of the information they obtain.

After the analysis, the resulting report is shown in Fig. 4, which consists in: 5 duplicated rules (Same Rules category), 5 overlapping, 2 inconsistencies, 1 tautology, and 1 case of contradictory rules. Among rules that are the same (duplicated with different

[9] http://www.stardog.com/.
[10] The AFRUSA service http://ecompass.khaos.uma.es/afra/login.jsp.
[11] http://www.stardog.com/docs/4.1.3/#_reasoning_types.

Table 11
SPARQL queries which are defined to analyze anti-fraud rules.

Description	SPARQL query
Get pairs of conditions that are the same	`select ?x ?y where { ?x rul:same_rule_conditions ?y }`
Get conditions that are the same as a specific condition (e.g. conditionX)	`select ?y where { rul:conditionX rul:same_rule_conditions ?y }`
Get pairs of condition groups that are the same	`select ?x ?y where { ?x rul:same_condition_groups ?y }`
Get condition groups which are the same as a specific condition group (e.g. conditiongroupY)	`select ?y where { rul:conditiongroupY rul:same_condition_groups ?y }`
Get pairs of rules which are the same	`select ?x ?y where { ?x rul:same_rules ?y }`
Get rules which are the same as a specific rule (e.g. ruleX)	`select ?y where { rul:ruleX rul:same_rules ?y }`
Get all the rules	`select ?x where { ?x rdf:type rul:Rules }`
Get all the rules with 2 condition groups	`select ?x where { ?x rdf:type rul:Rules2 }`
Get condition groups with 3 conditions	`select ?x where { ?x rdf:type rul:ConditionGroups3 }`
Get pairs of rules which overlap	`select ?x ?y where { ?x rul:overlapped_rules ?y }`
Get inconsistent rules	`select ?x where { ?x rdf:type rul:inconsistent_rules }`
Get tautological rules	`select ?x where { ?x rdf:type rul:tautological_rules }`

Table 12
Example of rules detected in category "Same rules".

ProfileID	RuleID	IF-Part	Score	Result
164	3951	"Number of Checkouts within the last 2 days for Credit Card" Equals "4"	0	Reject
164	3967	"Number of Checkouts within the last 2 days for Credit Card" Equals "4"	0	Reject

Table 13
Example of rules detected in category "Overlapped Rules".

ProfileID	RuleID	IF-Part	Score	Result
164	3920	"Credit Card Hash" IsIncludedInList "Blacklist-CreditCardHas"	0	Reject
164	3959	"Credit Card Hash" IsIncludedInList "Blacklist-CreditCardHas" OR "BIN Country" NotEquals "Spain" OR "Card Holder's Name" IsIncludedInList "Email Providers (domain)"	0	Reject

Table 14
Example of rules detected in category "Inconsistent Rules".

ProfileID	RuleID	IF-Part	Score	Result
164	3964	"Billing Address City" NotEquals "Device City" OR ("Card Holder's Name" IsIncludedInList "Email Provider's Domain" AND "Card Holder's Name" IsNotIncludedInList "Email Provider's Domain")	0	Reject

IDs), a recurring mistake is detected when a numerical custom value is involved in one condition of the antecedent.

Table 12 shows an example of a duplicated rule that is detected for IDs 3951 and 3967, which is for examining the number of checkouts with a credit card in 2 days. This is a typical error generated in setting phase, when the human-expert is fine-tunning thresholds in rules in accordance with his/her own past experience.

In the case of overlapping, Table 13 shows an example of the rules detected with IDs 3920 and 3959, which use the same condition (``Credit Card Hash'' IsIncludedInList ``Blacklist-CreditCardHas'') that are satisfied in both cases. Therefore, rule 3920 can be removed from the rule database, since it is actually covered by rule 3959.

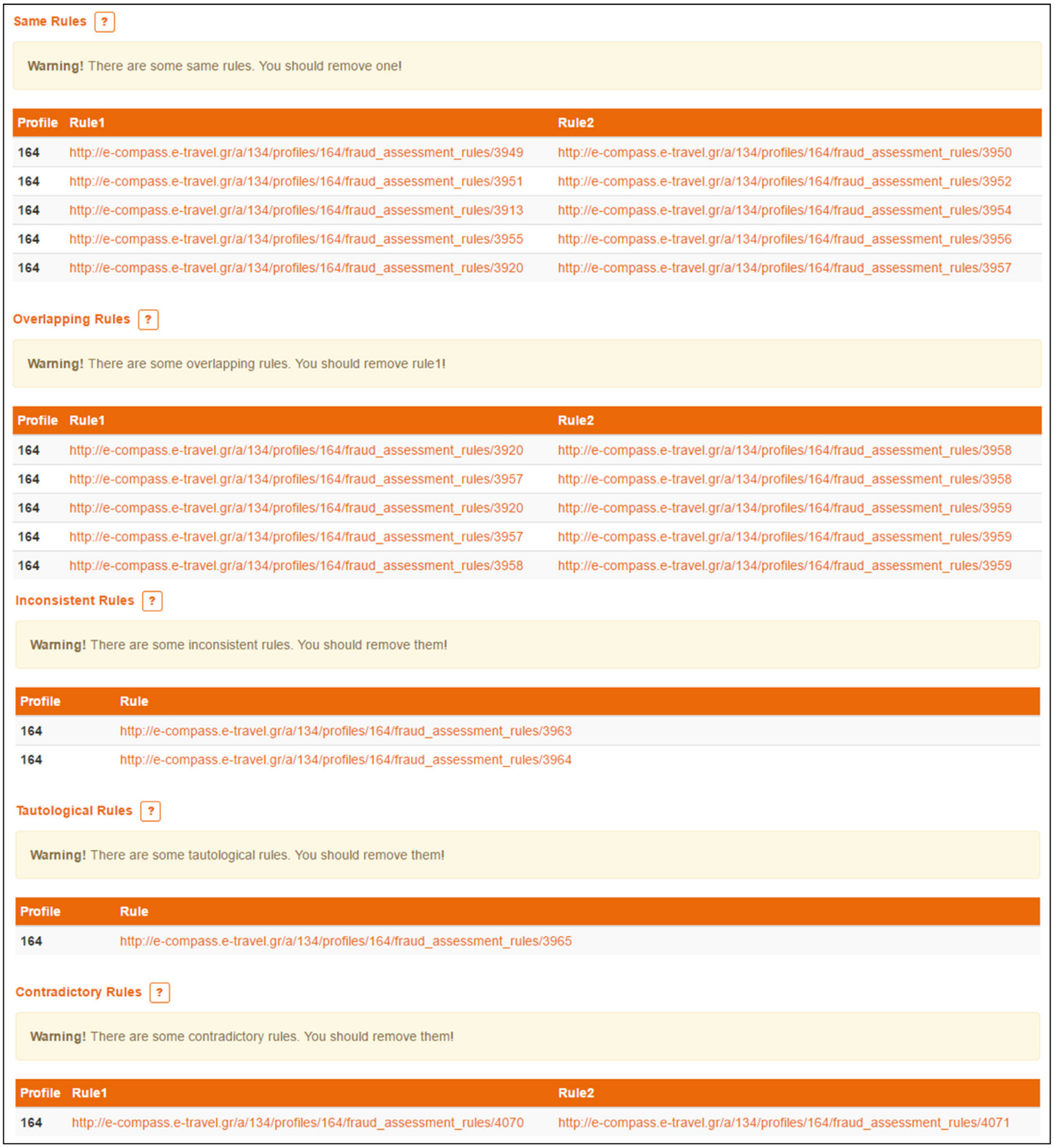
An interesting result can be observed in Table 14, where an inconsistent rule is detected as it contains a condition that can never be satisfied, e.g., ``Card Holder's Name'' IsIncludedInList ``Email Provider's Domain'' AND ``Card Holder's Name'' IsNotIncludedInList ``Email Provider's Domain''. In this regard, it is worth differentiating inconsistent rules from tautological ones, for which the antecedent part is always true. An example of a tautological rule detected by the semantic analyzer is shown in Table 15, for which the condition involving ``Device Country'' and ``Andorra'' always satisfies.

Finally, contradictory rules are probably the most critical for the e-merchant, as they could lead the expert system to issue incorrect recommendations. An example of a contradictory rule was detected (see Table 16) that reflects a typical scenario where a custom substring is contained in one attribute of the card transaction, in this case the ``Billing Address''. The e-shopper is now able to decide on which rule is incorrect and remove it, thereby preventing the anti-fraud expert system from operating wrongly.

6. Conclusions

In this paper, a semantic approach driven by ontology has been proposed as a mediated schema for the representation and consolidation of actual rule-based expert systems. The specific case of anti-fraud rule repositories for Card Not Present (CNP) transactions in e-commerce environments has been analyzed. The proposed approach is based on Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) technologies to develop reasoning tasks aimed at finding mistakes in rules, which lead the expert system to perform inaccurately.

The proposed approach is materialized in the form of a web service called AFRUSA. It has been evaluated in the context of a real-world scenario in the e-commerce (e-Tourism) industry. The experiments have shown that AFRUSA can effectively discover and classify mistakes in rule databases of anti-fraud expert knowledge systems. The results obtained comprise: duplicates, overlaps, inconsistencies and contradictions in rules that can now be curated by the e-merchant.


Same Rules ?

Warning! There are some same rules. You should remove one!

Profile	Rule1	Rule2
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3949	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3950
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3951	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3952
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3913	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3954
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3955	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3956
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3920	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3957

Overlapping Rules ?

Warning! There are some overlapping rules. You should remove rule1!

Profile	Rule1	Rule2
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3920	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3958
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3957	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3958
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3920	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3959
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3957	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3959
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3958	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3959

Inconsistent Rules ?

Warning! There are some inconsistent rules. You should remove them!

Profile	Rule
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3963
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3964

Tautological Rules ?

Warning! There are some tautological rules. You should remove them!

Profile	Rule
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/3965

Contradictory Rules ?

Warning! There are some contradictory rules. You should remove them!

Profile	Rule1	Rule2
164	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/4070	http://e-compass.e-travel.gr/a/134/profiles/164/fraud_assessment_rules/4071

Fig. 4. Screenshot of the results panel of Anti-Fraud Rules Semantic Analyser (AFRUSA) web interface.

Table 15
Example of rules detected in category “Tautological Rules”.

ProfileID	RuleID	IF-Part	Score	Result
164	3965	“Device Country” NotEquals “Andorra” OR “Device Country” Equals “Andorra”	100	Accept

Table 16
Example of rules detected in category “Contradictory Rules”.

ProfileID	RuleID	IF-Part	Score	Result
164	4070	“Billing Address City” Contains “La vella”	0	Accept
164	5072	“Billing Address City” Contains “La vella”	0	Reject

The semantic model elaborated here is also applicable to other domains where rule-base expert systems are involved. This motivates our main future line of research. In addition, ongoing work is focusing on the incorporation of Open Linked Data to enrich the semantic model with new perspectives of information, such as habits of fraudsters, commonly demanded products in fraudulent transactions and open blacklists of IPs address, card numbers, etc.

References

Alexopoulos, P., Kafentzis, K., Benetou, X., Tagaris, T., & Georgolios, P. (2007). Towards a generic fraud ontology in e-government. In *Proceedings of the second international conference on e-business - volume 1: Ice-b, (icete 2007)* (pp. 269–276).

Dean, M., & Schreiber, G. (2004). OWL web ontology language reference. *Technical Report*. W3C Recommendation, 10 February 2004.

Durkin, J. (1998). *Expert systems: design and development* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Fang, L., Cai, M., Fu, H., & Dong, J. (2007). Ontology-based fraud detection. In Y. Shi, G. D. van Albada, J. Dongarra, & P. M. A. Sloot (Eds.), *Lecture notes in computer science, Part III 7th international conference on computational science: Vol.3* (pp. 1048–1055)). Springer Berlin Heidelberg.

Gómez-Pérez, A., Corcho, O., & Fernandez-Lopez, M. (2004). *Ontological engineering*. Springer-Verlag London Limited.

Grosan, C., & Abraham, A. (2011). Rule-based expert systems. In *Intelligent systems: A modern approach* (pp. 149–185)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Grosof, B. N., & Poon, T. C. (2004). Sweetdeal: Representing agent contracts with exceptions using semantic web rules, ontologies, and process descriptions. *International Journal of Electronic Commerce, 8*(4), 61–97.

Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition,, 5*(2), 199–220.

Haase, P., & Stojanovic, L. (2005). Consistent evolution of owl ontologies. In A. Gómez-Pérez, & J. Euzenat (Eds.), *The semantic web: Research and applications*. In *Lecture notes in computer science: Vol. 3532* (pp. 182–197). Springer Berlin Heidelberg.

Hand, D. J., & Blunt, G. (2009). Estimating the iceberg : How much fraud is there in the u. k.? *Journal of Financial Transformation, 25*(1), 19–29.

Horrocks, I., & Patel-Schneider, P. (2003). Reducing owl entailment to description logic satisfiability. In D. Fensel, K. Sycara, & J. Mylopoulos (Eds.), *The semantic web - iswc 2003*. In *Lecture notes in computer science: Vol. 2870* (pp. 17–29). Springer Berlin Heidelberg.

Horrocks, I., Patel-Schneider, P. F., Bechhofer, S., & Tsarkov, D. (2005). OWL rules: A proposal and prototype implementation. *Web Semantics: Science, Services and Agents on the World Wide Web, 3*(1), 23–40.

Hu, B., Carvalho, N., Laera, L., Lee, V., Matsutsuka, T., Menday, R., & Naseer, A. (2013). Applying semantic technologies to public sector: A case study in fraud detection. In H. Takeda, Y. Qu, R. Mizoguchi, & Y. Kitamura (Eds.), *Semantic technology: Second joint international conference, JIST 2012, Nara, Japan, December 2–4, 2012. Proceedings* (pp. 319–325)). Springer Berlin Heidelberg.

Ketkar, P. S., Shankar, R., & Banwet, K. D. (2014). Telecom kyc and mobile banking regulation: An exploratory study. *Journal of Banking Regulation, 15*(2), 117–128.

McGuinness, D., & Harmelen, F. (2004). OWL web ontology language overview. *Technical Report*. W3C Recommendation.

Noy, N. F., & McGuinness, D. L. (2001). DOntology development 101: A guide to creating your first ontology. *Technical report*. Stanford University Knowledge Systems Laboratory Technical Report KSL-01-05. http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

Pérez, J., Arenas, M., & Gutierrez, C. (2009). Semantics and complexity of sparql. *ACM Transactions on Database Systems, 34*(3), 16:1–16:45.

Rajput, Q., Khan, N. S., Larik, A., & Haider, S. (2014). Ontology based expert-system for suspicious transactions detection. *Computer and Information Science, 7*(1), 103–114.

Ramaki, A. A., Asgari, R., & Atani, R. E. (2012). Credit card fraud detection based on ontology graph. *International Journal of Security, Privacy and Trust Management (IJSPTM), 1*(5), 1–12.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web, 5*(2), 51–53.

Staab, S., & Studer, R. (2009). Handbook on ontologies. *International handbooks on information systems*. Springer.

Ward, T. (2010). Strategies for reducing the risk of ecommerce fraud. *Technical report a first data white paper*. https://www.firstdata.com/downloads/thought-leadership/ecommfraudwp.pdf.

Wong, L. (2013). Money-laundering in southeast asia: Liberalism and governmentality at work. *Contemporary Politics, 19*(2), 221–233.