Application of the SDL Library to Reveal Legal Sanctions for Crime Perpetrators in Selected Economic Crimes: Fraudulent Disbursement and Money Laundering

Jaroslaw Bak, Maciej Falkowski and 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. As a part of the PPBW (Polish Platform for Homeland Security), we have developed tools that help investigators and prosecutors to conduct investigations of financial crimes and manage the data gathered. The most important task is to transform data into knowledge that allows for classification of illegal activities and assignment of sanctions based on the roles of people in companies. In this demo we present an application of the Semantic Data Library (SDL) to the problem of gathering, managing, querying and interpreting data relevant to building the evidence necessary for an indictment. The SDL uses and integrates a rule engine, a relational database, an ontology and a set of rules specific to a given crime typology. This combination allows querying and inferring a crime scheme and identifies possible charges; in particular, it discovers crime activities and roles (of particular types of owners, managers, directors and chairs) using concepts, appropriate relations and rules. We present results achieved with SDL and an ontology, called the 'minimal model,' of a fraudulent disbursement committed by management, accompanied by money laundering. Prospects on future development of the SDL tool are presented.

Keywords. SDL library, reasoning, minimal model ontology, financial crime, penal code

1 Introduction

One of the most costly and hard to trace crimes is economic crimes, such as a VAT tax fraud, or fraud. The level of complicity and amount of proxy and scam companies involved make it difficult to identify crime schemes. Another issue is to properly formulate an indictment based on evidence, that, in particular, identifies roles and activities of members of a crime group. In previous work [1, 2] we presented a model of a fraudulent disbursement crime, a subset of an asset misappropriation crime. In a 2009 survey [3], asset misappropriations constituted two-thirds of all economic crimes. These are often accompanied by money laundering schemes.

In this demo we apply our approach to fraudulent disbursement combined with money laundering [4]. The demo is based on the accompanying work [2]. The sophistication of our model lies less in ontology than in an appropriate set of rules. In the demo we present an application of the rule-based system originally called AFIZ (Analyzer of Facts and Relations) to crime scheme analysis. Our system is comprised of the SDL library [5], a relational database, an ontology and a set of rules. The relational database contains data gathered during investigation and relevant to the potential charges. The ontology provides concepts (classes) and relations to which relational data is mapped and a hierarchy of concepts and relations. Rules express dependencies and domain knowledge that allows querying about crime members' activities from a legal point of view. In the demo we do not stress completeness but simplicity. Therefore, it is based on a single fraudulent disbursement typology (the Hydra case) and core data related to this case. The rest of the data simulate variants of this crime by stochastically changing the values of the most important attributes. For some of these parameters no crime occurs, but generally various possible variants of persons' criminal activities are selected. The system goal is to detect a crime occurrence and bring proper charges based on people's activities. Despite handling a restricted class of crime, options of the crime case exhibit richness that is sufficient from a legal point of view. As seen from the demo, instantiating the ontology allows querying of various aspects of the crime.

Surprisingly, the system performs better than an average prosecutor on a charge assignment. We will elaborate on details of this statement in a future work. This demo aims at proving that the system is practical and can be of big help if extended to a larger set of crimes. Since the analytic capability of institutions, such as the Police or Prosecutor's Office in Poland, is limited, the system has to hide the complexity of data structures and reasoning engines and expose friendly interfaces.

The paper is organized as follows. Section 2 presents the SDL architecture and functionalities. Section 3 describes a Hydra case which exemplifies an important type of fraudulent disbursement scheme and the Hydra-case-like simulated input data generation. In Section 4 we execute selected queries according to the minimal model ontology. Conclusions and future work are presented in Section 5.

2 SDL Architecture and Functionalities

SDL integrates ontologies, relational data and rules that represent domain knowledge. The architecture of this system is presented in Figure 1. The central part, which gathers input from other system elements and processes rules, are the two Jess engines [10] used for forward and backward chaining.

The set of functionalities allows the SDL library to answer queries to the relational database using ontology and rules. The tool uses hybrid reasoning (forward and backward) for query execution. The backward method is responsible for gathering data from the relational database and the forward chaining is used to answer a given query. The Minimal Model ontology that conceptualizes financial crimes (presented in [2] and on the demo page) is expressed in OWL-DL [6]. Before it can be

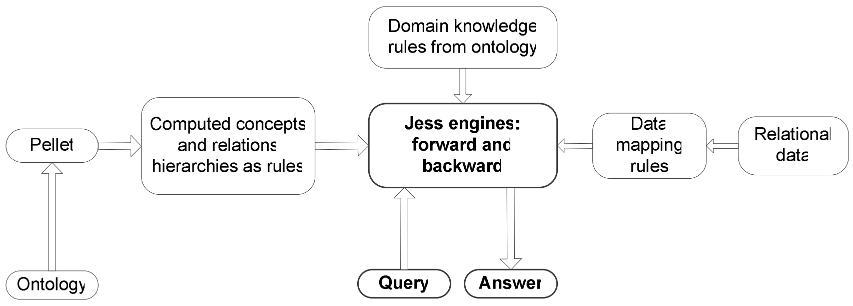
Domain knowledge rules from ontology
Pellet
Computed concepts and relations hierarchies as rules
Jess engines: forward and backward
Data mapping rules
Relational data
Ontology
Query
Answer

Fig. 1. The architecture of the Semantic Data Library.

used by the Jess engine, it has to be transformed into a set of rules. We adopt an approach that first calculates hierarchies of concepts and relations and then transforms these hierarchies into a set of rules. A part of the ontological knowledge is lost during this process, but for our purposes this is sufficient, and after adding rules the model remains decidable. A few chosen OWL properties are transformed into Jess rules as well, e.g., owl:SymmetricProperty. We use Pellet [7] to calculate all the ontological dependencies and taxonomy. In the next step, the Pellet output, the computed ontology has to be transformed into a set of Jess-format rules, which is done by the SDL-API function.

Another set of rules is the domain knowledge set, which is expressed in SWRL [8] language. SWRL extends the expressivity of OWL, supporting the use of ontology axioms in rules. We also use SWRLB language [9] to extend SWRL with additional functions. We use these rules to infer new facts in the knowledge base; in our case, about connections between persons, documents, money transfers and legal sanctions. These two sets constitute deduction rules.

There is one more set of rules – production rules; these are used to map between ontology axioms (properties and classes) and data stored in a relational database. These are defined in Jess-specific language [10] and their creation is supported by the SDL-GUI. They map some of the axioms (“essential” axioms) to appropriate SQL queries. “Essential” means that the instance of this axiom cannot be obtained from the taxonomy or rules, only directly from a database. Usually, “essential” axioms are lowest level taxonomy axioms.

3 The Hydra Case

3.1 The Hydra case as an example of a fraudulent disbursement scheme

Our so-called minimal model ontology comes from experience based on detailed analysis of descriptions, indictments and sentences of around 10 criminal cases. The most “clean” case of fraudulent disbursement is the so-called *Hydra Case*.

In the Hydra case, the Chief Executive Officer (CEO) of company A (*Hydra*) subcontracted construction work. The work was then consecutively subcontracted through a chain of phony companies B, C, and D (Hermes, Dex, Mobex). Each company was getting a commission for money laundering and falsified documents stating that the contracted work had been done. Actually, company A itself did what was identified as “subcontracted construction work”.

At the end of the chain, the owner of a single-person company D attempted to withdraw cash, and there was a suspicion that this cash would reach the management of company A “under the table”. The crime scheme of the Hydra case is presented on the demo description site (http://150.254.41.181/).

A definition of the minimal model in application to financial crimes, expressed in OWL language using the editor Protégé 4.0, is presented in [11]. This ontology has a modular structure and contains the modules listed in the accompanying paper [2]. The ontology is added to the demo material.

It is important to correctly model the sequence of activities in the company structure that lead to decisions and transactions. We illustrate this in the example of the three-level structure of authorization (this is easy to generalize to more levels, but the intent is to make it compatible with the Hydra case). The chain of activities is the following: in the Hydra case, acceptance of construction work done by B at a given site is first signed by a manager in A responsible for a work supervision at this site (MiddleLevelManager); this is followed by a signature of the higher level manager – a Director of the company responsible for supervision of all sites. A Director may be authorized to accept invoices and order a payment – technically this is fraud and in the case of Hydra, was done by a written authorization on the back of the invoice. The role of the Principal (the top level of authorization, which, however, could have not been exercised) was analyzed in detail in [1], where we modeled all possible options of the Principal’s behavior.

The Principal might not have known that the work has not been done. However, he was the one who signed the contract for subcontracting and thus could be implicated. Had the Principal of company A been a person who on the basis of the work acceptance document had ordered the payment of A to B, upon issuance of an invoice by B, he would be directly implicated.

3.2 Generation of the Hydra-case-like simulated input data

For a practical demonstration of the minimal model ontology, we need data stored in a relational database. We implemented a generation tool in Java which enables us to generate a relational database with the size of a case as parameters: the number of companies and number of documents (invoices, work approval documents and money turnovers). The Hydra case generator generates data concerning:

- Information about employees and their position in a company
- Invoices with all obligatory elements (payer, seller, product, etc.)
- Work approval documents (or the lack of them)
- Signatures on documents
- Goods and services
- Companies and their legal form
- Money turnovers: money transfers, payments and withdrawals
- Legal articles (name, ID and content)
- Information about illicit personal gains and damages to companies (with values)

- Other facts, like who knows about what (*Person knowsAbout document*) – these data result from testimonies. These facts are in the form of RDF triples (the table contains three columns: subject, predicate, object)

The seed of the generator is constant, so if the number of the documents is growing (the number of companies is fixed), the query results (i.e., the number of cases found criminal) from the bigger database contains all the results from the smaller database (and some extra results, possibly). The tool generates the Hydra case in four variants [2] and also generates some obscuring data (other documents, invoices, etc.). Every generated element has its own identification number. In this manner data are connected and internally coherent.

4 Queries and Query Execution

To realize what could be possible questions to the system, we present the relevant part of one count of charges in the Polish Penal Code:

"Article 296.

§ 1. Whoever, while under an obligation resulting from provisions of law, a decision of a competent authority or a contract to manage the property or business of a natural or legal person, or an organizational unit which is not a legal person, by exceeding powers granted to him or by failing to perform his duties, causes it to suffer considerable material damage, shall be subject to the penalty…

- § 2. If the perpetrator of the offence specified in § 1 acts in order to gain a material benefit…
- § 3. If the perpetrator of the offence specified in § 1 or 2 causes significant material damage of great extent…
- § 4. If the perpetrator of the offence specified in § 1 or 3 acts unintentionally…"

The specificity and the simple nature of the Hydra case is that all persons, possibly including the CEO, knew that they were committing a crime; therefore, the model does not have concepts and data explaining their intentions. It is extremely difficult to model and answer predicates like: "exceed powers granted to him" (which could depend, for example, on taking an excessive risk) or "fail to perform his duties". A judge has to answer these questions that pertain to a given crime's attributes. At this stage our model does not contain such soft crime attributes. It contains facts and hard concepts, such as "cause a company to suffer considerable material damage". The system is not prepared to answer directly all questions a judge might ask. However, we could ask questions: all counts of criminal activities of a person X, or all persons subject to counts of a charge C.

At present, the system does not reason on a partial set of facts. Rather, it assumes that the set of facts is complete and allows querying for elements of crime attributes. We do not have an option that allows a user to pose his/her own query. We restrict the use to a preselected query.

We have prepared five queries to test different aspects of the query answering mechanism. Queries were executed with the use of the hybrid reasoning process (forward and backward chaining). Graphical representation of the queries is presented on the demo description site.

The first query contains only variables (without any values) and exploits hierarchy rules. The second query contains variables and values; it exploits ontological rules (for inComplicityWith symmetric property). The third query contains only variables and exploits hierarchy rules. The fourth and fifth queries contain variables and values, and exploit various characteristics of the knowledge base as coded by rules. The last two queries are computationally demanding - the property *fallsUnder* needs almost all rules to be fired, because it requires evidence why a person falls under a given article. It is obvious that a rule can be fired more than once (if appropriate facts exist in the Jess working memory).

The demo queries were executed on three databases, generated according to methodology presented in Section 3.2. These databases differ in the size of the generated documents, values of money, turnovers, etc. The numbers of companies and employees are the same in every database (20 companies and 240 people). Generated databases contain the following numbers of documents (and money turnovers): 20, 100, 200. A mapping between a relational database and ontology consists of 57 mapping rules, and the set of domain knowledge rules contains 291 rules (42 SWRL rules from the ontology, instead 41 as in [2]).

All queries exploit mapping rules (because they are responsible for gathering data from the relational databases). For anonymity, the numbers in responses to queries are IDs of objects (persons, companies); if needed they can be transformed into real names. Queries without any values sometimes need more time to execute because SDL has to check all possible variable bindings from the database. Results of the executed queries are presented in Table 1. The SDL Demo is available on this site: http://150.254.41.181/. *Rules fired* means how many rules fired in the backward chaining engine during the reasoning process. Queries where executed on a computer with the following parameters: Core2Duo 2GHz, 2GB Ram; Java Heap Space was set at 1024MB.

Table 1. Results of the queries execution

Query and info		**Database 20**	**Database 100**	**Database 200**
Query 1	[ms]	781	1328	1922
Results	[number]	54	474	1036
Rules Fired	[number]	74	441	796
Query 2	[ms]	2734	37141	163968
Results	[number]	1	1	1
Rules Fired	[number]	1076	36260	225381
Query 3	[ms]	2875	36344	183047
Results	[number]	18	322	1004
Rules Fired	[number]	1367	38457	232583
Query 4	[ms]	5437	128719	Time exceeded 10 minutes
Results	[number]	1	1	
Rules Fired	[number]	2040	57091	
Query 5	[ms]	9312	Time exceeded 10 minutes	Time exceeded 10 minutes
Results	[number]	1		
Rules Fired	[number]	2540		

As we can see, simple queries (1, 2, 3) are executed in an efficient way (if we look at how many rules were fired). For more complex reasoning (queries 4 and 5), further optimization is needed. It is worth noticing that it takes 4 minutes for Pellet to classify our Minimal Model ontology with instantiation of the Hydra case. Our tool is more efficient for simplified reasoning without the full advantages of the OWL DL language (we mostly use hierarchies of concepts and relations computed by Pellet and SWRL rules). More sophisticated reasoning with the Jess engine is available using the OWL Meta-model from OWL2Jess [13].

We also compared our results with the Jess engine using forward and backward chaining separately. Appropriate scripts were loaded into two Jess engines. Facts (the same as in databases) were loaded from files. The times of executing queries for the same database were approximated, because the reasoning process is the main part of the query execution. Differences between queries execution were not higher than 500ms.

The time of executing queries in the forward chaining engine (data from the first database) was around 250ms for each query (data loading took 375ms). From the second database: 16s (query), 15s (facts). While loading data from the third database, the size of the Java heap space was reached (both engines), so the queries could not be executed. The times in a backward chaining engine are the following: 325ms (each query), 485ms (facts) for data from the first database, and 20s (each query), 183s (facts) in the second. It is obvious that for small databases, it is better to store data (facts) in the engines' working memory. But for the bigger databases, the problem with scalability occurs. Future investigation and modification of our hybrid reasoning method are necessary to obtain performance comparable to the forward chaining in the Jess engine. At the demo time we demonstrate the result of a method that avoids backward chaining and improves performance to a level comparable to the forward chaining in the Jess engine.

5 Conclusions and future work

We demonstrated that the SDL library can be used to solve practical problems with formally defined semantics (in our case the minimal model ontology). We realize that our tools need extensions and optimization.

In the next version of the SDL tool, we intend to apply a new conflict resolution strategy which will be a combination of the "depth" and "breadth" strategies. In the implemented strategy, SQL queries (data retrieval) will be executed in a more efficient way, which will make our approach more scalable and useful. Another useful feature we are going to implement is the reasoning path, which can be presented to the user and can explain what evidence proves that a person falls under a concrete legal article. Analysis of justifications [14] may give important information on ontology incompleteness, which often happens when a model is extended. We also want to add generation of more ontological rules than owl:SymmetricProperty.

The demo serves several purposes. It is evidence of the growing power of rule-based systems. It can already be employed by prosecutors and judges for training purposes. Moreover, upon a crime typology extension, it can be used as an expert system. Our plan is to commercialize such a tool under the name “Anti-Fraud Future”.

Extensions in some directions are relatively easy. To account for fraudulent disbursement committed by non-management workers, one has to think of possible schemes [15]. It can be either stealing money, e.g., from the register, falling under Art. 284 PC, or falsifying documents (joint Art. 271PC and Art. 284 PC).

Acknowledgement. This work was supported by the Polish Ministry of Science and Higher Education, Polish Technological Security Platform grant 0014/R/2/T00/06/02 in the first stage of the work and by PUT DS 45-083/10 grant in the later stage.

References

1. Bak J., Jedrzejek C., Application of an Ontology-based Model to a Selected Fraudulent Disbursement Economic Crime. In Casanovas, P., Pagallo, U., Ajani, G., and Sartor, G., editors, AI approaches to the complexity of legal systems, LNAI, Vol 6237, 2010
2. Bak J., Jedrzejek , C. and Falkowski M., Application of an Ontology-based and Rule-based Model to Selected Economic Crimes – Fraudulent Disbursement and Money Laundering, RuleML 2010 Conference, LNCS 6403
3. PricewaterhouseCoopers Global economic crime survey 2009 (2009), http://www.pwc.com/gx/en/economic-crime-survey/download-economic-crime-people-culture-controls.jhtml
4. Financial Action Task Force (FATF), http://www.fatf-gafi.org
5. Bak, J., Jedrzejek, C., Falkowski, M.: Usage of the Jess engine, rules and ontology to query a relational database. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 216–230. Springer, Heidelberg (2009)
6. McGuinness D., van Harmelen, F.:. Owl web ontology language overview. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-features/
7. Pellet Reasoner, http://clarkparsia.com/pellet/
8. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member Submission (May 21 2004), http://www.w3.org/Submission/SWRL/
9. SWRL Built-ins, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/
10. Jess (Java Expert System Shell), http://jessrules.com/
11. Jedrzejek C., Cybulka J., Minimal Model of financial crimes (In Polish) Definitions In OWL, Technical report PPBW 07/2009 extended 09/2009, http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl; the present version is in Open Source http://150.254.41.181/SDL_demo_ontology.zip
12. OWL API 3.0, http://owlapi.sourceforge.net/
13. OWL Meta-model, http://www.ag-nbi.de/research/owltrans/owlmt.clp
14. Horridge, M., Parsia, B, and Sattler. U., Laconic and Precise Justifications in OWL. In Proceedings of the 7th International Semantic Web Conference (ISWC). Springer, Oct. 2008.
15. Jedrzejek C., Bak J., Application of an Ontology-Based Model to Wide-Class Fraudulent Disbursement Economic Crimes, Jurix 2010, submitted