Artif Intell Law (2018) 26:315–344
https://doi.org/10.1007/s10506-018-9218-0

RuleRS: a rule-based architecture for decision support systems

Mohammad Badiul Islam[1] · **Guido Governatori**[1]

Published online: 23 February 2018

Abstract Decision-makers in governments, enterprises, businesses and agencies or individuals, typically, make decisions according to various regulations, guidelines and policies based on existing records stored in various databases, in particular, relational databases. To assist decision-makers, an expert system, encompasses interactive computer-based systems or subsystems to support the decision-making process. Typically, most expert systems are built on top of transaction systems, databases, and data models and restricted in decision-making to the analysis, processing and presenting data and information, and they do not provide support for the normative layer. This paper will provide a solution to one specific problem that arises from this situation, namely the lack of tool/mechanism to demonstrate how an expert system is well-suited for supporting decision-making activities drawn from existing records and relevant legal requirements aligned existing records stored in various databases.We present a Rule-based (pre and post) reporting systems (RuleRS) architecture, which is intended to integrate databases, in particular, relational databases, with a logic-based reasoner and rule engine to assist in decision-making or create reports according to legal norms. We argue that the resulting RuleRS provides an efficient and flexible solution to the problem at hand using defeasible inference. To this end, we have also conducted empirical evaluations of RuleRS performance.

Keywords Rule engine · Defeasible logic · Deontic Logic · Decision support system · SPINdle · Legal norms

✉ Mohammad Badiul Islam
Badiul.Islam@data61.csiro.au

Guido Governatori
Guido.Governatori@data61.csiro.au

[1] Data61, CSIRO, Brisbane, Australia

1 Introduction

Decision makers in governments and enterprises often have to take decisions and create reports based on regulations (and other normative and legislative documents) and organisation specific data, where the organisation data is generally stored in enterprise (relational) databases. For example, Australian financial institutions are subject to Financial Sector (Collection of Data) Act 2001 for what (financial) information to report to the relevant regulators (e.g., Australian Prudential Regulator Authority), and government departments and agencies are required to comply with the Public Governance Performance and Accountability Act 2013 and Public Governance Performance and Accountability Rule 2014 for their annual financial reporting. The requirements about what, when and in what forms to report (and related exceptions) are given in the (relevant) regulations while the (financial and related) data is stored in the databases of the institutions that have to generate reports about the data.

Accordingly, in these scenarios, one has to perform some legal reasoning (for example to understand what are the actual requirements that apply in a given case) based on the information from their enterprise databases. Legal reasoning has two distinctive features: the first is that norms, generally, set base guidelines, but then they are open to exceptions, where the exceptions are expressed themselves as norms. Also, norms prescribe behaviours using concepts such as obligations, prohibitions, and permissions. Obligations and prohibitions can be violated, and legal frameworks provide conditions to compensate for violations. Relational databases are essentially first-order logic theories, where the tables are grounded instances of predicates and queries are first-order logic expressions about the predicates stored by the tables; as such they are not suitable to reason with deontic concepts and violations (Herrestad 1991), also, Governatori (2015b) argues that most existing Deontic Logics are not able correctly handle compensatory norms, and norms where deontic concepts imply other deontic concepts. Consequently, we need a logic that can handle exceptions and complex deontic notions. Defeasible Deontic Logic (Governatori et al. 2013) is a rule-based computationally oriented logic that proved suitable to handle legal reasoning and that does not suffer from the problem affecting other Deontic Logics (Governatori 2015a).

In this paper we investigate (and discuss the implementation and empirical evaluation of) a methodology to combine Defeasible Deontic Logic to handle the reasoning about the legal requirements, and relational databases to retrieve information from enterprise databases. The key intuition is that the norms governing the scenario are encoded as rules in the logic, while the facts are encoded as database queries. Accordingly, the focus of the paper is not to study whether the chosen logic is able to properly represent norms and legal reasoning, but whether the proposed framework is scalable enough to handle massive (enterprise) databases, to provide responses in a "reasonable" time.[1] In our opinion,

[1] We fully recognise that the notion of "reasonable" time is a very open texture one. It depends on the applications: while a response in seconds could ot be suitable for a real time critical systems, an overnight computation is suitable for a task, e.g., auditing the whole set of transaction of a bank when such auditing would take months based on a limited sample data.

the work addresses a shortcoming in the field of Artificial Intelligence and Law. Most approaches to legal reasoning have been tested on a very small scale scenarios and data. To be best of our knowledge, this is the first work testing legal reasoning on large scale databases, and thus can provide benchmarks for future developments.

For the evaluation of the framework we are going to examine two case studies: the first is based on New South Wales Mandatory Reporting Guidelines (NSW Government 2016), mandating child care centres (and other professionals) to report to appropriate authorities potential child abuse or neglect cases, where the determination where a case is to a decision support system for the US Food and Drug Administration (FDA) Adverse Event Reporting System (FAERS, formerly AERS) (US Food and Drug Administration 2016). For the first case study, do the sensitive nature of the application combined to privacy and legal issues, it was not possible to use real data. Thus, instead of artificially creating a synthetic database, we decided to use an existing database (from a different domain) and test the set of rules and the database against different types of queries just to measure the performance. Therefore, the database queries were created for their structure not for their content. The second case study was evaluated for content and performance. Specifically, we use a sample database that contains information on adverse event and medication error reports submitted to the FDA generated from the publicly available FDA datasets. The datasets consist of (US Food and Drug Administration 2015) postmarket drug and biologic safety evaluations based on records and reports concerning adverse drug experiences on marketed prescription drugs for human use without approved new drug applications and the relevant regulation (US Government 2014). We illustrate the system for decision-making towards compliant or non-compliant information, in particular, on adverse event and medication error reports submitted to FDA using RuleRS system architecture and its internal components.

The paper is organised in seven sections. Section 2 briefly outlines defeasible logic. Section 3 introduces the RuleRS architecture. We include the case studies in Sect. 4 and the evaluation using case study data in Sect. 5. The following one describes some review of related works. Finally, we conclude with a discussion of implications for future research.

2 Defeasible logic

RuleRS is developed in the setting of Defeasible Logic (DL). Why DL? Indeed, DL is a formalism that has been successfully used for legal reasoning [and it has been proved that other formalisms successful in legal reasoning correspond to variants of DL (Governatori 2011)]. Defeasible Deontic Logic has been successfully used for applications in legal reasoning (Antoniou et al. 2001; Governatori 2005; Governatori and Shek 2013; Governatori 2015a) and it is has been shown that it does not suffer from problems affecting other logics used for reasoning about norms and compliance (Governatori and Hashmi 2015; Governatori 2015a). Thus Defeasible Deontic Logic is a conceptually sound approach for the representation of regulations and at the same time, it offers a computationally feasible environment to reason about them [Governatori and Rotolo (2008) proved that the logic is computationally

feasible since since we can compute the extension of a theory in linear time]. In this paper, we are going to evaluate whether the proposed combination offers a practical solution for decision systems in the legal domain combining databases and norms (represented as rules). To this end, we examine two practical systems' response time to identify a child at risk (NSW Government 2016) involving in a sample Child Care Management Systems database, and an adverse report is compliant with the relevant regulation system using two large sample databases generated from FDA datasets (US Food and Drug Administration 2015).

Defeasible Logic is "skeptical" nonmonotonic logic (meaning that it does not support contradictory conclusion), was originally proposed by Nute (1994). Since then it has been significantly used in the legal domain or closely related areas, such as modelling regulations (Antoniou et al. 1999), e-contracting (Governatori 2005; Grosof 2004), business processes compliance (Sadiq and Governatori 2015; Governatori and Rotolo 2010) and automatic negotiation system (Skylogiannis et al. 2007). The modelling of regulations in DL also offers support for "Decision support", "Explanation", "Anomaly detection", "Hypothetical reasoning" and "Debugging" tasks. Decision support is used to infer a correct answer from given rules and regulations. DL is one of the possible solutions since regulations may contradict one another using the defeasible rules do not necessarily in force; instead they may be blocked by other rules with contrary conclusions (Antoniou et al. 1999).

A *defeasible theory D* [a knowledge base in defeasible logic, or a defeasible logic program (Maher 2001)], consists of five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation. D is a triple $(F, R, \succ)$ where F and R are finite sets of facts and rules respectively, and $\succ$ is a superiority relation on R.

The language of DL consists of a finite set of literals, where a literal is either an atomic proposition or its negation. Given a literal l, $\sim l$ denotes its complement. That is, if $l = p$ then $\sim l = \neg p$, and if $l = \neg p$ then $\sim l = p$.

Facts are logical statements describing indisputable facts, represented either in the form of states of affairs [literal or modal literal (refer to Sect. 2.1)] or actions that have been performed, and are considered to be always true. For example, "John is a human" is represented by: *human(John)*.

A rule r, on the other hand, describes the relations between a set of literals [the *antecedent* $A(r)$, which can be empty] and a literal [the *consequence* $C(r)$]. We can specify the strength of the rule relation using the three kinds of rules supported by *DL*, namely: *strict*, *defeasible*, and *defeater*.

Strict rules are rules in the classical sense: whenever the premises are indisputable (e.g. a fact) then so is the conclusion. For example,

$$human(X) \rightarrow mammal(X)$$

which means "Every human is a mammal".

It is worth mentioning that strict rules with *empty* antecedents can be interpreted the same way as facts. However, in practice, facts are more likely to be used to

describe contextual information; while rules, on the other hand, are more likely to be used to represent the reasoning underlying the context.

Defeasible rules are rules that can be defeated by contrary evidence. For example, typically mammal cannot fly, written formally:

$$mammal(X) \Rightarrow \neg flies(X)$$

The idea is that if we know that X is a mammal, then we may conclude that it cannot fly *unless there is other, not defeated, evidence suggesting that it may fly* (for example that the mammal is a bat). A defeasible rule with an empty antecedent can be considered as a *presumption*.

Defeaters are rules that cannot be used, on their own, to draw any conclusions. Their only use is to prevent some conclusions, i.e., to defeat some defeasible rules by producing evidence to the contrary. For example the rule:

$$heavy(X) \rightsquigarrow \neg flies(X)$$

states that an animal is heavy is not sufficient enough to conclude that it does not fly. It is only evidence against the conclusion that a heavy animal flies. In other words, we do not wish to conclude that $\neg flies$ if *heavy*, we simply want to prevent a conclusion *flies*.

A full definition of the proof theory can be found in Antoniou et al. (2001) and Billington et al. (2010). Roughly, the rules with conclusion p form a team that competes with the team consisting of the rules with conclusion $\neg p$. If the former team wins p is defeasibly provable, whereas if the opposing team wins, p is non-provable.To conclude, let us consider D as a theory in DL (as described above). A *conclusion* of D is a tagged literal and can have one of the following four forms: $+\Delta q$ meaning that q is definitely provable in D (i.e., using only facts and strict rules); $-\Delta q$ meaning that we have proved that q is not definitely provable in D; $+\partial q$ meaning that q is defeasible provable in D; and $-\partial q$ meaning that we have proved that q is not defeasible provable in D.

Strict derivations are obtained by forward chaining of strict rules while a defeasible conclusion p can be derived if there is a rule whose conclusion is p, whose prerequisites (antecedent) have either already been proved or given in the case at hand (i.e., facts), and any stronger rule whose conclusion is $\neg p$ has prerequisites that fail to be derived. In other words, a conclusion p is (defeasibly) derivable when: p is a fact, or there is an applicable strict or defeasible rules for p, and either all the rules for $\neg p$ are discarded (i.e., not suitable) or every rule for $\neg p$ is weaker than an applicable rule for p.

2.1 Defeasible Deontic Logic

It has been argued that legal reasoning requires two types of rules: constitutive and prescriptive rules. Constitutive rules are used to model definition of terms and parameters specific to legal documents, for example, the definitions of terms in an act, whereas prescriptive rules are applied for encoding the obligations, prohibitions, permissions, ..., and the conditions under which they enter into force according to a

specific legal document. To correctly model the provision corresponding to prescriptive norms, we have to supplement the language with deontic operators. In this respect we follow the classification proposed by Governatori (2013) and Hashmi et al. (2015). In addition, the logic has mechanisms to terminate and remove obligations [see Governatori and Rotolo (2010) for full details]. For obligations and permission we use the following notation:

- $[\mathrm{P}]p$: p is permitted;
- $[\mathrm{OM}]p$: there is a maintenance obligation for p[2];
- $[\mathrm{OAPP}]p$: there is an achievement preemptive and perdurant obligation for p;
- $[\mathrm{OAPNP}]p$: there is an achievement preemptive and non-perdurant obligation for p;
- $[\mathrm{OANPP}]p$: there is an achievement non preemptive and perdurant obligation for p;
- $[\mathrm{OANPNP}]p$: there is an achievement non preemptive and non-perdurant obligation for p.

Compensations are implemented based on the notion of 'reparation chain' (Governatori and Rotolo 2006). A reparation chair is an expression

$$[\mathrm{O}_1]c_1 \otimes [\mathrm{O}_2]c_2 \otimes \cdots \otimes [\mathrm{O}_n]c_n,$$

where each $[\mathrm{O}_i]$ is an obligation, and each c_i is the content of the obligation (modelled by a literal). The meaning of a reparation chain is that we have that c_1 is obligatory, but if the obligation of c_1 is violated, i.e., we have $\neg c_1$, then the violation is compensated by c_2 (which is then obligatory). But if even $[\mathrm{O}_2]c_2$ is violated, then this violation is compensated by c_3 which, after the violation of c_2, becomes obligatory, and so on.

Defeasible Deontic Logic allows deontic expressions (but not reparation chains) to appear in the body of rules. Thus we can have rules like:

$$\mathit{restaurant}, [\mathrm{P}]\mathit{sellAlcohol} \Rightarrow [\mathrm{OM}]\mathit{showLicense} \otimes [\mathrm{OAPNP}]\mathit{payFine}$$

The rule above means that if a restaurant has a license to sell alcohol (i.e, it is permitted to sell it, $[\mathrm{P}]\mathit{sellAlcohol}$), then it has a maintenance obligation to expose the license ($[\mathrm{OM}]\mathit{showLicense}$), if it does not then it has to pay a fine ($[\mathrm{OAPNP}]\mathit{payFine}$). The obligation to pay the fine is non-pre-emptive (meaning that it cannot be paid before the violation). The logic is equipped with a binary relation over rules, called superiority relation, that allows us to handle rules with conflicting conclusions: for example a rule r setting a general prohibition and a second rule s that derogates the prohibition permitting the conclusions. This type of situation is common in legal reasoning and can be modelled by saying that s is "stronger" than r, in symbols $s > r$. If both rules apply, we will say that s defeats r. For full a description of the logic and its features, see Governatori (2005), Governatori and Rotolo (2010) and Governatori et al. (2013).

[2] Prohibitions can be expressed as maintenance obligations with a negated content, i.e., $[\mathrm{OM}]\neg p$.

The reasoning to determine what obligations, prohibitions, and permissions are derivable from a set of facts and a setof rules is as follows.

An obligation $[O]p$ (where $[O]$, $[O_x]$ and $[D_y]$, in the description below, are placeholders for the obligations described above) is derivable if:

1. $[O]p$ is given as one of the facts, or
2. there is a rule

$$r : a_1, \ldots a_n \Rightarrow [O_1]p_1 \otimes [O_m]p_m \otimes [O]p \ldots$$

such that

(a) for all $1 \leq i \leq n$, a_i is provable, and
(b) for all $1 \leq j \leq m$, $[O_j]p_j$ and $\neg p_j$ are provable, and
(c) for all rules

$$s : b_1, \ldots, b_k \Rightarrow [D_1]q_1 \otimes [D_l]q_l \otimes [D]p'$$

such that p' is the negation of p, either

(i) exists $1 \leq i \leq k$ such that b_i is not provable, or
(ii) exists $1 \leq j \leq l$ such that either $[D_j]q_j$ or $\neg q_j$ is not provable, or
(iii) r defeats s.

The idea is that there must be a rule that fires: so all the elements in the antecedents are provable (a), and in case the conclusion is an obligation for a reparation, all the obligations before it have to be violated. Thus, the violated obligations were in force (thus the obligations were provable) and we have evidence that it was violated (thus the negation of the content of each violated obligation is provable) (b).Also, we have to ensure that there are no rules for the opposite that fire (c), and if they do, these rules are weaker than the rule for the obligation we want to conclude.

For permission, we have the same conditions, but where we use $[P]p$ instead of $[O]p$; also, we conclude $[P]p$ if we can conclude $[O]p$. Due to space reasons, readers interested in understanding the semantics, deontic operator conversions, conflict detections, conflict resolutions, and algorithm implementing this rule-based system are referred to Governatori and Rotolo (2004), Governatori and Rotolo (2008), Governatori and Rotolo (2010) and Governatori et al. (2013) for details.

3 RuleRS architecture

In this section, we will introduce the *RuleRS* design architecture and technical details (refer to Fig. 1). Overall, RuleRS consists of four main system components. In particular, the key system components of RuleRS are: (1) I/O Interface, (2) Predicates, (3) Formal Rules , and (4) SPINdle Reasoner. In the rest of this section, we give a short outline of the RuleRS internal components and their functions (refer to Fig. 1).

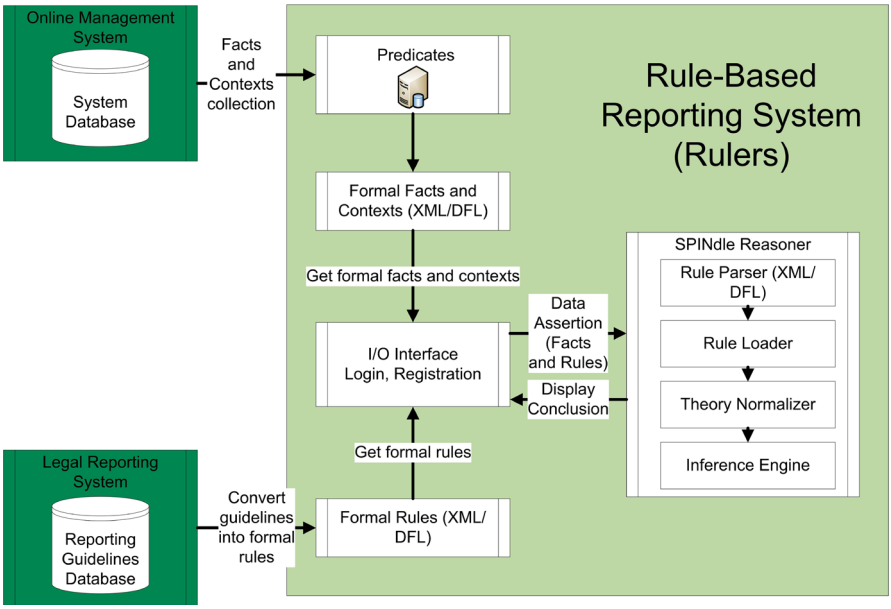

Fig. 1 Rule-based reporting system (RuleRS)

3.1 I/O interface

The I/O interface is implemented in Java to bridge RuleRS components and interacting with each other. The I/O interface is used to compile predicates (SQL or JSON files) and to generate facts and contexts in formal notation in DFL syntax, and SPINdle reasoner receives this as a parameter. The I/O interface also displays the final remarks or comments for each of the incidents and predicates. Figure 2 includes a sample RuleRS interface.

3.2 Predicates

The values encoding a regulation and the databases (schemas) used in conjunction with the rules are in general developed independently are likely have a different vocabulary in general. There is no direct correspondence between the literals used by the rules and the table/attributes of the database schema. Accordingly, we have to establish a mapping among them to enable the integration of rules and instances in the database. The fundamental idea behind predicates is that data stored in the database corresponds to facts in a defeasible theory and these facts can be retrieved from the database using queries. Thus each fact corresponds to a (SQL) query and a predicate is a statement that can be true or false depending on the value of its arguments/variables. A predicate with n arguments is just an n-ary relation.

A predicate in RuleRS corresponds to a database view, i.e., a named query, where the name is literal to be used by the defeasible rules. The details are the query to be run to determine if the predicate is true or false for a given set of parameters. In case

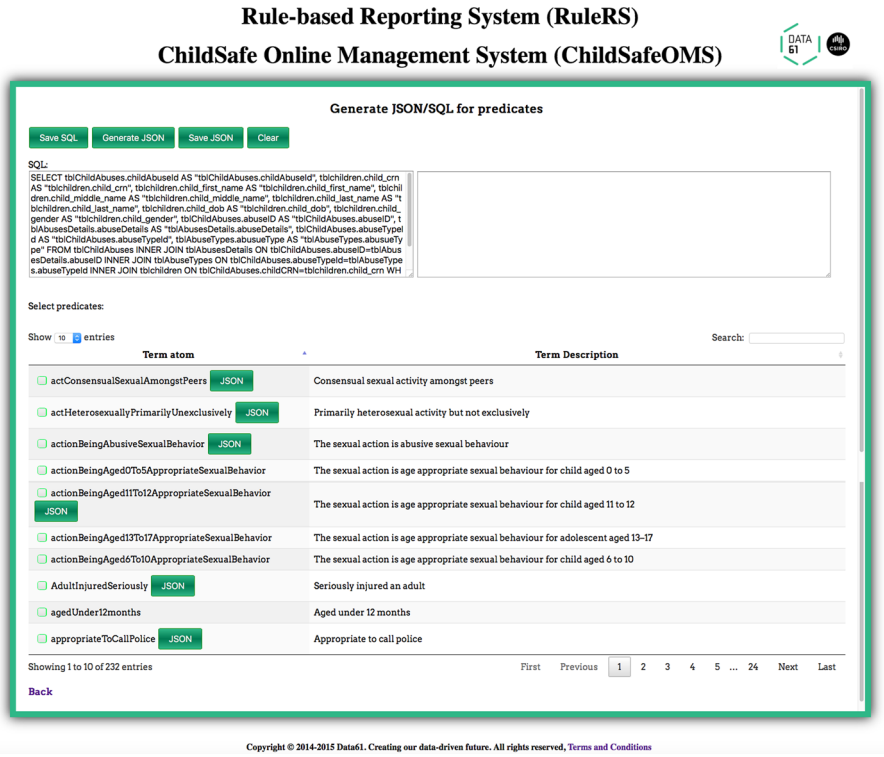

Fig. 2 Generate SQL/JSON for predicate

the output of the query is not empty, the predicate is true and is passed to the defeasible theory as fact.

In RuleRS, predicate consists of two components: (1) predicate name and (2) predicate details. *Predicate name* represents the action(s), condition(s) or indisputable statement(s), and passed on to the rule engine, SPINdle as a defeasible fact (literal and modal literal) (Governatori and Rotolo 2004, 2008, 2010) or actions that have been performed. For example, the fact "There is a risk for an incident" is represented by "*riskForIncident*" and passed as "$>>$ *riskForIncident*" to SPINdle if it is returned as true from the relational database. "Predicate details" includes the "incident details" and is stored either as an SQL Statement or (after conversion) in lightweight data-interchange JSON (JavaScript Object Notation) syntax[3] (easy for humans to read and write, easy to generate, and easy to parse for machines) to create a bridge between the data stored in the database and the terms passed as predicates (input case) to the rule engine. The SQL or JSON statements can be created in the initialisation of RuleRS with all of the incidents along with all of the predicates for each of the incidents or dynamically add it later.

Incident ID and relevant details of the incidents are also included for each of the predicates and named the predicates with relevant incident information such as

[3] http://json.org accessed on 19 January 2017.

"riskForIncident.sql" (for SQL statement) or "riskForIncident.json" (for JSON Statement). The following snippet illustrates the syntax adopted by RuleRS: "riskForIncident" predicate using SQL Statement:

```
SELECT incidentID, IncidentDetails, IncidentDetails1,
   IncidentDetails2
FROM tblIncident
WHERE incidentID='XXXXXX'
```

In this example, *IncidentDetails*, *IncidentDetails1*, *IncidentDetails2* are substituted for the place-holders in the "riskForIncident" predicate from relational databases for the *incidentID* *'XXXXXX*.

"riskForIncident" predicate using JSON Statement:

```
{"riskForIncident":
   { "incidentID":"XXXXXX",
     "IncidentDetails":"ABC",
     "IncidentDetails1":null,
     "IncidentDetails2":"XYZ"}}
```

(Yu et al. 1992; International Business Machines Corporation 1993, 2001), We could further classify predicates according to the query types[4] and SQL statement[5] composition to each kind of constructs such as WHERE, GROUP BY, HAVING and ORDER BY.

In the next step, the records and incidents for which there is a match in the relational database are transformed into predicates to be used by the SPINdle rule engine (Lam and Governatori 2009), and forwarded to SPINdle for further processing using the I/O interface to make the process dynamic.

3.3 Formal rules

One of the features of RuleRS is its ability to perform reasoning based on (legal) requirements. As we alluded to in the introduction such regulatory requirements are represented as formal rules in Defeasible Deontic Logic (Antoniou et al. 2001; Nute 1994); to enable their use with the rule engine used by RuleRS (SPINdle, see the next section) the rules are stored in the DFL format (Lam and Governatori 2009). At this stage the rules are created manually by legal knowledge engineers and stored in a knowledge-base.

[4] The are various query types as basic (compares two values), quantified (compares a value or values with a collection of values), BETWEEN (compares a value with a range of values), NULL (tests for null values), LIKE (searches for strings that have a certain pattern), EXISTS (tests for the existence of specific information), IN [another approach to compare a value with a collection of values (Yu et al. 1992; International Business Machines Corporation 1993, 2001)] and different aggregate functions [AVG, MAX, MIN, SUM, COUNT(*), or COUNT(DISTINCT)].

[5] When "SELECT" statement is used in a predicate, is called a "subquery" (International Business Machines Corporation 1993, 2001).

3.4 SPINdle reasoner

SPINdle Reasoner[6] (Lam and Governatori 2009) is a Java-based implementation of DL (Antoniou et al. 2001; Nute 1994) that computes the extension of a defeasible theory. SPINdle supports Modal DL, all types of DL rule, such as facts, strict rules, defeasible rules, defeaters, and superiority. A full description of SPINdle is out of scope for the current paper, we refer the interested readers to Lam and Governatori (2009) for details. In summary, SPINdle is a tool which accepts rules, facts, monotonic and non-monotonic (modal) rules for reasoning with inconsistent and incomplete information. In RuleRS, SPINdle Reasoner receives the formal facts, contexts as predicates from predicate file generated for data stored in the associated relational databases and computes definite or defeasible inferences which are then displayed by the I/O interface.

4 Case studies

The fitness of Defeasible Deontic Logic for modelling norms has been demonstrated elsewhere (Antoniou et al. 2001; Governatori 2005; Governatori and Shek 2013; Governatori 2015a; Islam and Governatori 2015) (refer to Sect. 2). Accordingly, the aims of this section is to evaluate whether RuleRS offers a practical solution to the issue of providing a viable combination of norms and databases. Specifically, the focus is to determine whether the response time is appropriate for practical applications. In general, the response time consists of two components: the time required to execute a SQL query and the time needed to compute the extension of a theory using SPINdle.

In this section, we illustrate RuleRS with the help of two case studies to demonstrate possible applications of the RuleRS architecture.

4.1 Case study: sample online system

In this section, we introduce Case Study 1 a *ChildSafe Care Online Management Systems (ChildSafeOMS)* and its relational database. ChildSafeOMS maintains a relational database to record child enrolment, attendance, abuse and/or neglect data and relevant system information.

The integration of RuleRS and the underlying relational database extend the ChilSafeOMS's functionality. In particular, ChildSafeOMS introduces new functionalities including the automation of early identification of children at risk of abuse and neglect.

More specifically, the system contains three sources of information: (1) a child care database ChildSafeOMS with the data described above; (2) a set of defeasible rules encoding the decision trees, definitions and (normative) guidelines of New South Wales Mandatory Reporter Guidelines (NSW Government 2016). We have

[6] SPINDle Reasoner is available to download freely from http://spindle.data61.csiro.au/spindle/tools.html accessed on 31 January 2018 under LGPL license agreement. https://opensource.org/licenses/lgpl-license accessed on 31 January 2018.

translated the rules manually from the source document; and (3) a set of bridging statements relating the terms of the regulations and the fields (and data) in the ChildSafeDB. These statements are obtained from cross-analysis of the terms and fields in the two related components.

One of the aims of the system is the early identification of children at risk, in particular, a risk of abuse and neglect. In the rest of the section, we describe the component outlined above.

4.1.1 Sample database

We start by describing the relational database we used in this experiment. Sample RuleRS is a large dataset with more than 11 million records in total, and takes up 2017 MB. To make sure that the readers are not overwhelmed by the size of the data, we have provided brief introductions to RuleRS dataset given the high sensitivity and the nature of the data and application (and legal implications of analysis such data), it is not possible to use real-life databases. However, the aim of this paper is not to build such an application but to validate the methodology of integrating defeasible (legal) reasoning with database technology. As we have already discussed, a vital component of the method is the definition of the predicates/literals to be passed to the reasoners as the facts of a Defeasible theory obtained as an SQL query. In other words, SQL queries define the meaning of literals used in a defeasible theory regarding data and the schema of a database. As we have explained our focus is the response time; thus we can ignore the actual content of the undergoing database. The time taken to answer an SQL query depends on the structure and the size of the database and the structure of the SQL query itself (Osman and Knottenbelt 2012).

For this case study, we have created sample RuleRS database based on dataset available from stat-computing.[7] Most of the data comes initially from RITA.[8] These data have derivable variables removed, are packaged in yearly chunks and have been more heavily compressed than the originals. We have downloaded data from 1998 to April 2008. Along with other columns in the tables, each of the tables contain various fields such as "Year", "Quarter" (1-4), "Month", "DayofMonth" (Day of Month), "DayOfWeek" (Day of Week Analysis). There are more than 1 million records in each of the tables, and we stored the data in a separate table for each of the years. The data consists of flight arrival and departure details for all commercial flights within the USA. We have inserted to a new column "child_crn" (Child's Child Reference Number) in each of the year's data to map data with "children" table. The idea of column insertion using a "sample scenario" is that the child visited an airport during that year. For this experiment, we skipped other details of the year's columns such as if the children stuck in an airport because their flight was delayed or canceled.

Sample RuleRS database also contains many tables with information about children in a child care centre, types of abuses and their descriptions, and records for possible instances of abuses. The typical work-flow to populate the information is

[7] http://stat-computing.org/dataexpo/2009/the-data.html accessed on 19 January 2017.

[8] http://www.transtats.bts.gov/Fields.asp?Table_ID=236 accessed on 19 January 2017.

that educators of child care insert data about a child in the database to report day-to-day activities. In case they have concerns for some forms of abuse or neglect, the system offers an interactive questionnaire where the educator answers a set of 'Health and Welfare' emotional and physical questions of the type "In your knowledge, do any of the following environments apply to the child?". Here the questions pertain to any of the abuse types and ask if they were aware of any child was abused or if they have any concern about a particular child. The systems populate the database according to answers provided by the educators/carers. Additionally, the educator/child carer also fills up five additional information along with the abuses types: (i) Child Abuse ID (auto-incremental); (ii) Child crn (Child Reference Number); (iii) Abuse ID; (iv) Abuse type id (Type of child abuse); and Abuse date.

4.1.2 From database records to predicates

As we discussed above to aim of RuleRS is to draw inferences from a database supplemented by a rule systems encoding legal knowledge. In this setting, the information stored in the database is queried to provide the facts for the SPINdle reasoner in the form of predicates. After the facts are queried from the ChildSafeDB, the SPINdle reasoner draws a set of conclusions using the facts and the defeasible rules encoding the decision trees of the NSW mandatory reporter guidelines and a set of additional rules, eventually provided by domain experts, giving further indicators for risks of neglect or abuse.

For the extraction of facts, RuleRS is equipped with SQL/JSON files containing the definitions of the predicates used by the SPINdle component regarding either simple attribute names and values or by SQL queries. For example, according to the guideline, a person is considered a child if her/his age is less than fifteen years. Several steps in the decision trees depend on whether a person at risk of abuse is a child or not. RuleRS In RuleRS, this is represented by the predicate *beingChild* and contains the following (JSON) definition:

```
{"predicate":"beingChild",
"SQL":"SELECT *
       FROM tblchildren
       WHERE tblchildren.child_crn = $id
       AND tblchildren.child_dob >= ${today - 15y}"}
```

where `$id` is a parameter given as input by the end-user, and `${today - 15y}` is a computation using the system variable `$today`.

The I/O interface takes the user input, examines the definitions, transforms the descriptions into appropriate SQL queries, and executes the queries to return facts and context as output. For example, for the predicate *beingChild* and `$id=123456789A` ChildsafeOMS produces the following JSON snippet, encoding the predicate and the context that it holds.

```
{"beingChild":{
    "tblchildren.child_crn":"123456789A",
    "tblchildren.child_first_name":"Simone",
    "tblchildren.child_middle_name":"Ruby",
    "tblchildren.child_last_name":"Shirazi",
    "tblchildren.child_dob":"2013-04-02",
    "tblchildren.child_gender":"Female"}}
```

The predicate is then used in the successive reasoning phase, while the context is used in case of reasoning phase determines that a report has to be submitted to a relevant authority; the RuleRS is used to populate the required reports.

4.1.3 Sample rules

This section includes sample some rules in SPINdle format. Some of the formal rules are converted from the NSW mandatory reporter guidelines (NSW Government 2016). The guidelines are formulated as decision trees, and they specify (i) when an educator of child carer has to report an abuse or potential abuse (ii) under what conditions one has to inform. Also it contains definitions of types of violation or possible abuses and indicators of these.

Consider for example the provision (as part of a decision tree) prescribing to report immediately to *CS* if there is a situation where you are aware that a child has been sexually abused even though he/she hasn't told you or the child made a clear, unambiguous statement of sexual abuse (NSW Government 2016). We can represent the rule as *If "ChildAbusedSexually" has occurred, then there is an obligation (non-persistent, pre-emptive, achievement) to do "reportToCSImmediately"* [refer to Governatori and Rotolo (2010), Governatori (2013) and Hashmi et al. (2015)) for other types of obligations]. Its formal representation in SPINdle is as follows:

$$ChildAbusedSexually \Rightarrow [\text{OANP}]\, reportToCSImmediately$$

Furthermore, the rule set contains more rules obtained from the definitions of what counts as a child being sexually abused and possible indicators for different types of abuse, where such definitions are given as decision trees themselves.

The next example concerns the situation where a child or young person exhibits a (persistent) problematic sexual behaviour versus other. In this case, an educator has different options to whom report. In case the parent of carers are aware of the problem and they have responded appropriately, the educator has to report to the Child Well fare Unit (*consultWithCWU*). However, if the initiating child/young person has continuing or imminent contact with the victim and there where some coercion or the victim is in a situation of inferiority, then the situation has to be reported immediately to Community Services (CS). Otherwise, the normal procedure is to file a formal report to CS (NSW Government 2016, p. 19). In case of a problematic behaviour without further conditions, the educator has to

continue to monitor the situation.The following rules can represent this part of the decision tree:

$$r_1 : PersistentSexualBehaviourVsOther,$$
$$ParentRespondedAppropriately, \neg [\text{OANP}]\, reportCS,$$
$$\neg [\text{OANP}]\, reportCSImmediately \Rightarrow [\text{OANP}]\, consultWithCWU.$$

The rule above describes the decision when the parents or carers are aware of the situation and responded appropriately. However, to ensure a single outcome (behaviour) we have to derive the failure of other obligations, namely report to CS or report to CS immediately.

$$r_2 : SexualBehaviourVsOther, CoercionOrInferior,$$
$$ContactWithVictim \Rightarrow [\text{OANP}] reportToCSImmediately$$

Above is the most specific and its antecedent subsumes all other antecedents thus there is no need to include the failure of the other obligations.

$$r_3 : SexualBehaviourVsOther, CoercionOrInferior,$$
$$\neg [\text{OANP}]\, reportToCSImmediately \Rightarrow [\text{OANP}]\, reportToCS$$

Above is the second most specific rule; thus its antecedent has to include the failure of the rule that is more specific that it, that is the obligation of the conclusion of r_2.

Finally, the following rule, whose conclusion is a maintenance obligation, is triggered when the other obligations do not apply.

$$r_4 : SexualBehaviourVsOther, \neg [\text{OANP}]\, reportCSImmediately,$$
$$\neg [\text{OANP}]\, consultCWU, \neg [\text{OANP}]\, reportCS \Rightarrow [\text{OM}]\, monitor$$

The SPINdle rules are derived directly from the decision trees provided in the NSW Mandatory Reporter Guidelines (NSW Government 2013). A decision tree can be represented as a set of classical nested if-then-else rules, where for every node (binary decision, question) there is a predicate/proposition corresponding to it. For the representation of a decision tree in Defeasible Deontic Logic we needed a rule for each "positive" outcome (in our case a decision to report a case to a relevant authority or to the determination that a particular type of situation occurred). The conclusion of the rule is the proposition labelling the leaf node, and in the antecedent or body of the rule we have the proposition labelling the nodes from the root of the tree to the particular leaf node, negated or not depending on the conditions on the path. Arguably, the advantage of using Defeasible Deontic Logic as formal representation and implementation of a decision tree is the reduced complexity (no need to create nested if-then-else structures), and reduced number of rules [since, conditions leading to no outcome, i.e., no report in our case, are not needed, and are such situations are handled directly by the defeasibility of the rules, see Governatori et al. (2009)]. Furthermore, the adoption of Defeasible Deontic Logic allows us to use negated deontic literals in the body of rules. This corresponds to use a form of negation of failure, which requires either the duplication of the conditions leading to

the failure of other reporting conditions, or the explicit representation of all else conditions in the if-then-else.

Notice that is it possible to formally show that the rules we created are equivalent[9] to the decision trees provided in the NSW Mandatory Reporter Guidelines (NSW Government 2013). However, since this is to investigate the combination of rules and database, we refrain from proving the equivalence.

4.1.4 Sample predicates

As we have already alluded to the database used in this case study has been chosen for it being available and it size. The subject matter of the database is irrelevant to the content of the rules and the intended application of the rules. The queries are not relevant for content, but just to test the performance of RuleRS with different types of SQL queries (Yu et al. 1992; The PostgreSQL Global Development Group 1996–2015; International Business Machines Corporation 1993, 2001). Alternatively, we could use JSON format too. Each of the queries represents a single predicate, which passes as an input parameter to Spindle (see Sect. 3.4). The following snippet illustrates sample test predicates in SQL format used for this case[10]:

```
SELECT tbl.child_crn, tbl.child_first_name,
       tbl.child_middle_name, tbl.child_last_name,
       tbl.child_dob, tbl.child_gender
FROM tblchildren as tbl
WHERE AGE(tbl.child_dob)<'15 years 0 mons 0 days'
```

Predicate 1 beingChild

```
SELECT tbl.child_crn, tbl.child_first_name,
       tbl.child_middle_name, tbl.child_last_name,
       tbl.child_dob, tbl.child_gender
FROM tblchildren as tbl
WHERE child_crn
IN (SELECT tbl2008.child_crn
    FROM tbl2008
    WHERE tbl2008.child_crn=tbl.child_crn)
```

Predicate 2 riskForIncident6

[9] By equivalent, we mean that given an input the decision trees and our rules produce same outcome.

[10] The snippets reported here have been selected for the type of SQL query they employ to evaluate the corresponding response time, not for the meaning of the data associated with the query.

```
SELECT tbl.child_crn, tbl.child_first_name,
       tbl.child_middle_name, tbl.child_last_name,
       tbl.child_dob, tbl.child_gender
FROM tblchildren as tbl
WHERE AGE(tbl.child_dob)<
     (SELECT AVG(AGE(tblchildren.child_dob))
      FROM tblChildren)
```

Predicate 3 riskForIncident8

```
SELECT tbl.child_crn, tbl.child_first_name,
       tbl.child_middle_name, tbl.child_last_name,
       tbl.child_dob, tbl.child_gender
FROM tblchildren as tbl
INNER JOIN tblchildabuses as tbl1 ON
      tbl1.childcrn=tbl.child_crn
LEFT JOIN tbl2008 ON tbl2008.child_crn=tbl.child_crn
GROUP BY tbl.child_crn, tbl.child_first_name,
         tbl.child_middle_name, tbl.child_last_name,
         tbl.child_dob, tbl.child_gender
HAVING COUNT(tbl1.childabuseid)>0
ORDER BY substring(tbl.child_crn,'^[0-9]+')::int,
         substring(tbl.child_crn,'[^0-9]*$')
```

Predicate 4 riskForIncident13

```
SELECT tbl.child_crn, tbl.child_first_name,
       tbl.child_middle_name, tbl.child_last_name,
       tbl.child_dob, tbl.child_gender
FROM tblchildren as tbl
LEFT JOIN tbl2008 ON tbl2008.child_crn=tbl.child_crn
   INTERSECT
SELECT tbl.child_crn, tbl.child_first_name,
       tbl.child_middle_name, tbl.child_last_name,
       tbl.child_dob,  tbl.child_gender
FROM tblchildren as tbl
LEFT JOIN tbl2007 ON tbl2007.child_crn=tbl.child_crn
```

Predicate 5 riskForIncident16

4.2 Case study: FAERS

In this section, we illustrate RuleRS as decision support system with the help of a case study to demonstrate possible applications of the RuleRS architecture.

Accordingly, we introduce the sample database based on FAERS (US Food and Drug Administration 2016) records.

4.2.1 Sample database

The sample database based on the FAERS records that we have downloaded and converted from into PostgreSQL database format. For the experiment reported in this paper, we use the records for the first quarter 2014. The selected sample FAERS is a large dataset with more than three million forty thousand records in total and makes up 342 MB. To make sure that the readers are not overwhelmed by the size of the data, we have provided brief introductions to the dataset based on FAERS records.

The structure and format of the seven tables of FAERS from the first quarter of 2014 are explained in Table 1. This table also includes the sample examples, FAERS tables organisation, and content details.

4.2.2 From database records to predicates

As we discussed above the aim of RuleRS is to draw inferences from a database supplemented by a rule systems encoding legal knowledge. In this setting, the information stored in the database is queried to provide the facts for the SPINdle reasoner. After the facts are queried from the FAERS database, the SPINdle reasoner draws a set of conclusions using the facts and the defeasible rules encoding the regulations specify the records and reports concerning adverse drug experiences on marketed prescription drugs for human use without approved new drug applications (US Government 2014).

For the extraction of facts, RuleRS is equipped with SQL/JSON files containing the definitions of the predicates used by the SPINdle component regarding either simple attribute names and values or by SQL queries.

4.2.3 Sample predicates

We have used several predicates and queries for testing scenarios which are written in SQL. Alternatively, we could use JSON format too. Each of the queries represents a single predicate, which passes as an input parameter to SPINdle (see Sect. 3.4). The following snippet illustrates sample test predicates: *ICSRs_contain_Patient_age* in SQL format:

```
1 SELECT primaryid,
2 CASE WHEN age IS NOT NULL THEN '
     report_Patient_age_to_FDA'
3 ELSE '-report_Patient_age_to_FDA'
4 END FROM DEMO14Q1
```

Predicate 6 report_Patient_age_to_FDA

Table 1 FAERS tables

No.	Table name	Table description	Examples, organisation and content detail
1.	DEMO14q1	Patient demographic and administrative information, a single record for each event report	The case (*PRIMARYID*-100108481, *CASEID* 10010848 *CASEVERSION*-1), received by the FDA on 2014/03/10. Like any Case, it appears only once in the Demographic table, "DEMO14q1"
2.	DRUG14q1	Drug/biologic information for as many medications as was reported for the event, 1 or more per event	Two drugs were reported for the Case (*PRIMARYID*-100108481): ALBENDAZOLE as Primary Suspect and MECTIZAN (IVERMECTIN) as Secondary Suspect. *PRIMARYID* appears two times in the Drug file, DRUG14q1, with a different *DRUG_SEQ* for each drug
3.	THER14q1	Drug therapy start dates and end dates for the reported drugs, 0 or more per drug per event	Dates of therapy for ALBENDAZOLE and MECTIZAN (IVERMECTIN) were reported as "2013/07/10 (ongoing)" with same *PRIMARYID* and the same *DRUG_SEQ* as in the drug table using *DSG_DRUG_SEQ*
4.	INDI14q1	Contains all "Medical Dictionary for Regulatory Activities" (MedDRA)[a] terms coded for the indications for use (diagnoses) for the reported drugs, 0 or more per drug per event	*DRUG_SEQ* from drug table, the fields labeled *INDI_DRUG_SEQ* in indication table, REAC14q1. Three cases were reported FILARIASIS, FILARIASIS LYMPHATIC and ONCHOCERCIASIS with different *INDI_DRUG_SEQ* using the same *PRIMARYID*-100108481
5.	REAC14q1	Contains all MedDRA terms coded for the adverse event, 1 or more per event	Four reactions were reported using the *PRIMARYID*-100108481 as Abdominal pain, Gait disturbance, Headache and Vomiting in reaction table, REAC14q1
6.	OUTC14q1	Patient outcomes for the event, 0 or more	HO (Hospitalization—Initial or Prolonged) was reported for the *PRIMARYID* 100108481
7.	RPSR14q1	Report sources for the event, 0 or more	HP (Health Professional) and FGN (Foreign) was reported in report source table, RPSR14q1 using the *PRIMARYID*-100108481

[a] http://www.meddra.org

```
SELECT primaryid,
CASE WHEN drugname IS NOT NULL THEN '
    report_Suspect_medical_product_name_to_FDA'
ELSE '-report_Suspect_medical_product_name_to_FDA'
END
FROM drug14q1
```

Predicate 7 report_Suspect_medical_product_name_to_FDA

The predicates are then used in the successive reasoning phase (we may process several predicates by joining them in a single run), while the context is used in case of reasoning phase determines that a report has to be submitted to a relevant authority; the context is used to populate the required decision and reports.

4.2.4 Sample rules

This section includes sample rules which are manually converted from U.S. ELECTRONIC CODE OF FEDERAL REGULATIONS–Title 21: Food and Drugs–PART 310–NEW DRUGS–Subpart D–Records and Reports US Government (2014) in SPINdle format. The regulations specify the records and reports concerning adverse drug experiences on marketed prescription drugs for human use without approved new drug applications. In particular, the regulations list the reporting requirements for Manufacturers, Packers, and Distributors (MPD) and information reported on various life-threatening serious and unexpected adverse drug experience for Individual Case Safety report (ICSR). Additionally, the regulations contain definitions of various adverse drug experiences.

Various examples can be made as the rule set (US Government 2014) contains more rules obtained from information reported on ICSRs. Consider for an example the provision (as part of informed on ICSRs) prescribing to report electronically to *FDA* as ICSRs include "Patient age" while report to FDA. We can represent the rule as *While an obligation (persistent, non-pre-emptive, achievement) report on ICSRs to FDA, there is an obligation (persistent, non-pre-emptive, achievement) to include "Patient age"* . Its formal representation in SPINdle is given in rule r_1:

$$r_1 : [\text{OAPN}]\, \textit{report on ICSRs to FDA} \Rightarrow [\text{OAPN}]\, \textit{report Patient age to FDA}$$

Similarly, ICSRs include "Suspect medical product name" while report to FDA. We can represent the rule as *While an obligation (persistent, non-pre-emptive, achievement) to report on ICSRs to FDA, there is an obligation (persistent, non-pre-emptive, achievement) to include "Suspect medical product name"*. Its formal representation in SPINdle is as r_2:

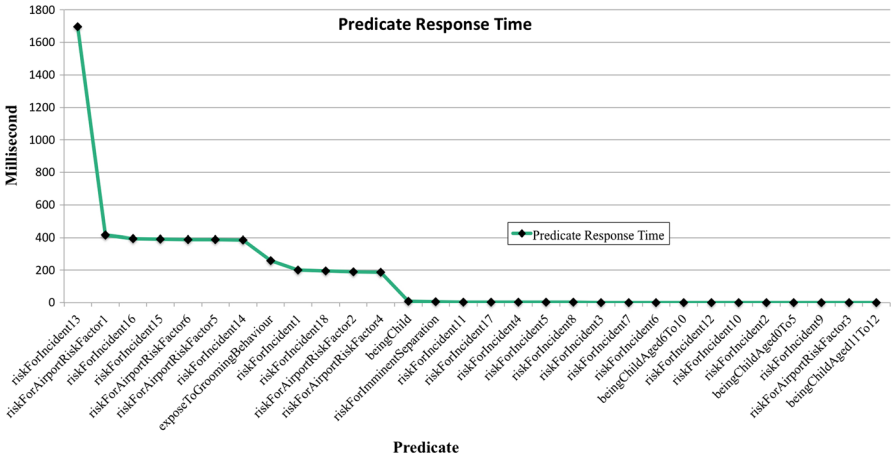

Fig. 3 Case Study1: predicate response time

$$r_2 : \text{[OAPN]}\ \textit{report on ICSRs to FDA}$$
$$\Rightarrow \text{[OAPN]}\ \textit{report Suspect medical product name to FDA}$$

The rules r_1 and r_2 describe the decision when voluntary (aka direct) and mandatory report to FDA by companies are compliant or non-compliant. However, to ensure a single outcome (behaviour) we have to derive the failure of other obligations too. The rules r_1 and r_2, thus their antecedent (i.e., Patient age and Suspect medical product name) has to include the failure of the rules that is more specific, that is the obligation of the conclusion of r_1 and r_2 respectively for ICSRs. Finally, whose conclusion is an achievement preemptive obligation non-persistent (OAPN) obligations, is triggered when the other obligations do not apply. Figure 4 includes sample RuleRS output, where the predicates with "-" sign were non-compliant and have not been included in the ICSRs while reported to FDA.

```
Run# 1
Event primaryid 100033001
-report Patient information to FDA
report outcome attributed to adverse drug experience to FDA
report Suspect medical product name to FDA
report Suspect medical product route of administration used t
o FDA
-report Patient weight to FDA
report Adverse drug experience term to FDA
report Suspect medical Lot number to FDA
report Patient gender to FDA
-report MPD Report source to FDA
-report Whether initial report to FDA
-report Whether 15 day Alert report followup to FDA
-report Suspect medical product Expiration date to FDA
-report Whether followup report to FDA
report Suspect medical product NDC number to FDA
report Date of ICSR submission to FDA
report Whether the initial reporter is a health care profession
al to FDA
-report Date of adverse drug experience to FDA
report MPD name to FDA
report Patient age to FDA
-report Whether 15 day Alert report to FDA

+d [OAPNP]report Suspect concomitant medical products to FDA(X)
+d report Suspect medical Lot number to FDA(X)
+d [OAPNP]report Suspect medical Lot number to FDA(X)
+d [OAPNP]report Suspect medical product Expiration date to FDA(X)
+d -report Suspect medical product Expiration date to FDA(X)
+d report Suspect medical product NDC number to FDA(X)
+d [OAPNP]report Suspect medical product NDC number to FDA(X)
+d [OAPNP]report Suspect medical product concomitant therapy dates to FDA(X)
+d [OAPNP]report Suspect medical product diagnosis for use indication to FDA(X)
+d [OAPNP]report Suspect medical product information to FDA(X)
+d report Suspect medical product name to FDA(X)
+d [OAPNP]report Suspect medical product name to FDA(X)
+d report Suspect medical product route of administration used to FDA(X)
+d [OAPNP]report Suspect medical product route of administration used to FDA(X)
+d [OAPNP]report Suspect medical product to FDA(X)
+d [OAPNP]report Unique case identification number to FDA(X)
+d -report Whether 15 day Alert report followup to FDA(X)
+d [OAPNP]report Whether 15 day Alert report to FDA(X)
+d -report Whether 15 day Alert report to FDA(X)
+d [OAPNP]report Whether followup report to FDA(X)
+d -report Whether followup report to FDA(X)
+d [OAPNP]report Whether initial report to FDA(X)
+d -report Whether initial report to FDA(X)
+d report Whether the initial reporter is a health care professional to FDA(X)
+d [OAPNP]report Whether the initial reporter is a health care professional to FDA(X)
+d [OAPNP]report to FDA Suspect medical product whether prescription product(X)
```

Fig. 4 Sample RuleRS output

5 The evaluation of RuleRS

As we have already pointed out, we are interested in evaluating whether the methodology underlying provides a sound solution but we want to identify if the solution offered by RuleRS is a viable solution for practical applications. To this end, we provide an empirical evaluation of the response time. As we remarked in Sect. 4.1.1 the database for ChildSafeOMS is used only to evaluate the response time, and it does not provide any insight about the domain application.

The database used for the second case study presents data specific to the application. Accordingly, we evaluated the application for correctness, this means to evaluate whether the rules we have encoded properly translate the existing FDA regulation. To this end, we randomly selected twenty events from the database, and we manually check them for compliance with the regulation. In the second phase, the same twenty events were automatically analysed using RuleRS. We have a perfect match between the manual analysis and the automated analysis. The principal evaluation, however, concerned the response time.

5.1 Evaluation methodology

See Algorithm 1 that is used for the evaluation process. In this algorithm, the evaluation process measures the predicate response time which is measured based on the process used in a database. We use a list of predicates and inputs and receive Defeasible Conclusion (a set of literals) as output. Since there is a caching time involved, each query is executed for 10 times consecutively and the average time is computed. We limit the query for top 20 events from FAERS database for each test case. In this process, predicates are executed under six groupings (randomly selected): 1 predicate, 2 predicates, 5 predicates, 10 predicates, 15 predicates and 20 predicates. The operating systems provide various system calls that return the current time (Currim et al. 2013). We use Java method **currentTimeMillis()**,[11] which returns the operating system current time in milliseconds (ms). We measure the time required to execute each of the predicates (i.e., SQL query). We record query response time into several steps using **currentTimeMillis()** method, examining (a) Predicate response time (b) Total response time in millisecond for 10 executions and generate inferences using RuleRS. that is used for the evaluation process.

[11] http://bit.ly/1Lyg6Z4 accessed on 19 January 2017.

Algorithm 1: The evaluation process algorithm

```
Data: List of predicates
Result: Defeasible Conclusion (a set of literals)
initialization;
for each predicate do
    open the database recordset;
    issue the predicate;
    Execute the predicate on the database;
    obtain the result set;
    traverse that result set sequentially;
    if match condition for predicate then
        generates facts for each of the predicates;
        record predicate name as Defeasible fact in Defeasible theory/rules
          set;
        collect the elapsed time for each predicate run time;
        compute and record the average response time per predicate in
          millisecond;
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;
    end
    open the SPINdle Reasoner;
    pass the Defeasible theory/rules (including the formal
      facts/contexts/predicates);
    for for each of Defeasible theory/rules do
        obtain the inference set;
        collect elapsed time for the Spindle run time;
    end
    close the SPINdle Reasoner;
    collect elapsed time for total run time;
    calculate mean for each of the recorded time;
    traverse that inference set sequentially;
    for for each inference do
        print Defeasible Conclusion (a set of literals);
    end
end
print predicate mean runtime in millisecond;
print predicate runtime in millisecond;
print Spindle run time in millisecond;
print Total run time in millisecond;
```

5.2 Experimental set-up

After deciding to experiment, RuleRS, our next task was to set-up the experiment using sample database. We have performed the test on a Mac computer. The environment is as follows:

Model Name: MacBook Pro
Model Identifier: MacBookPro11,2
Processor Name: Intel Core i7
Processor Speed: 2.2 GHz
Number of Processors: 1
Total Number of Cores: 4
L2 Cache (per Core): 256 KB
L3 Cache: 6 MB
Memory: 16 GB

5.3 The evaluation results

We now present some of the information obtained after applying RuleRS to study the system evaluation mentioned above. In the particular environment we studied, the following observation can be made for the execution behaviour of predicates.

In the evaluation process using Case Study 1 (Sect. 4.1), 30 predicates were executed and Fig. 3 sketches response times for each of the predicates. The average response time per predicate is 171.19 ms, while that "riskForIncident13"[12] has highest average response time 1696.1 ms and "beingChildAged11To12" (return 0 response with BETWEEN predicate) has lowest average response time 0.4 ms. On average (per run), total response time for 20 predicates is 5142 ms, while that Spindle response time is 42.2 ms and total response Time 5184.2 ms. Figure 5 sketches the response time for each of the predicates used for the evaluation. Furthermore, 60% of the predicates response within a small amount of time (between 0.4 and 7.6 ms). There are 12 predicates; each takes a significant amount of time to the response which is listed in Table 2. These predicates used large transactions, and there is a presence of cross joins with other large tables.

We now present some of the information obtained for the second case study.

In this evaluation process, SQL response time (mean) was calculated under six groupings (randomly selected): 1 predicate, 2 predicates, 5 predicates, 10 predicates, 15 predicates and 20 predicates which can be found in Fig. 5. For example, using "10 predicates", the average total SQL response time was 302.695 ms and SQL response time per predicate is 30.2695 ms and for "15 predicates", the total response time is 311.89 ms and per predicate is 20.791 ms. Figure 5 also reveals that there has been a gradual increase in the total SQL response time while steady decrease in per predicate response time. From the experiment data we have

[12] with 2-way joins (1-INNER JOIN and 1-LEFT JOIN), "GROUP BY", "HAVING", "COUNT" and "ORDER BY" predicate.

extrapolated that the query for each predicate would take approximately 5ms. Furthermore, the FDA regulation for FAERS requires approximately 200 rules, we estimate that the time need to compute a report for each single incident would take about 1.5 s. Thus we believe that the proposed approach offers a viable solution for an online system to check whether a new record complies with the data collection requirements mandated by the underlying FDA regulation. Furthermore, it would take approximately 8 h on a laptop to audit the whole data that makes up 342 MB with more than three million forty thousand records in total for a quarter.

6 Related work

We began the RuleRS architecture development by describing our analysis of the fact that various businesses and agencies may have to generate decisions and reports (along with other obligation) based on regulations and policies align with their existing database records. Whereas traditional database management systems (DBMSs) are passive in the sense that their general focus has been to execute commands (e.g., query, update, delete) as and when requested by the user or application program (Paton and Díaz 1999), rule engines cooperate with the rule repository and the underlying database.

Why we need rule engine? Prior studies have established various advantages of improving systems using rule engines.[13] The key aspects of using rule-based system can be listed as follows: natural knowledge representation, uniform structure, separation of knowledge from its processing and dealing with incomplete and uncertain knowledge.[14]

Prior approaches for rule engines can be classified into one of five types as Prolog-based, deductive databases, production and reactive rules, triple engines, and knowledge bases (Liang et al. 2009). The prolog-based system is defined as a top-down inference engine. While various applications have been used prolog-based system (e.g., XSB[15]; Yap[16]), their common focus has been to provide a complete environment for building applications (Liang et al. 2009). Deductive databases, which are the extension of the database management system, typically structure query language and design storage around a logical model of data (Kozák 2011). These approaches can be seen in various systems: DLV[17]; IRIS[18]; and Ontobroker.[19] Production and reactive rules-based systems are the bottom-up engines where rules have actions in the consequent (Liang et al. 2009). All of these systems (e.g.,

[13] http://java.sys-con.com/node/45082 accessed on 19 January 2017.

[14] http://intelligence.worldofcomputing.net/expert-systems-articles/rule-based-expert-systems.html accessed on 20 January 2017.

[15] http://xsb.sourceforge.net accessed on 20 January 2017.

[16] http://www.dcc.fc.up.pt/~vsc/Yap/ accessed on 20 January 2017.

[17] http://www.dlvsystem.com/ accessed on 20 January 2017.

[18] http://sourceforge.net/projects/iris-reasoner/ accessed on 20 January 2017.

[19] http://www.semafora-systems.com/en/products/ontobroker/ accessed on 20 January 2017.

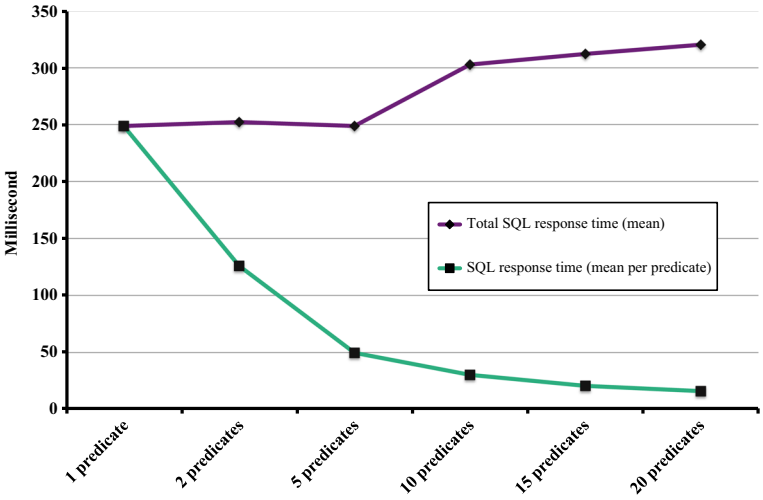

Fig. 5 Predicate response time

Table 2 Sample Predicate Response Time and analysis

Predicate	Response time (ms)	Analysis on predicate
RiskForIncident13	1696.1	With 1-INNER JOIN and 1-LEFT JOIN, "GROUP BY", "HAVING", "COUNT" and "ORDER BY" predicate
RiskForAirportRiskFactor1	415.6	With 2-LEFT JOIN
RiskForIncident16	395.11	With 2-LEFT JOIN and INTERSECT
RiskForIncident15	391.5	With 3-INNER JOIN and 1-LEFT JOIN UNION with 3-INNER JOIN and 1-LEFT JOIN
RiskForAirportRiskFactor6	388.6	With IN predicate apply to a large table
RiskForAirportRiskFactor5	387	Using EXISTS predicate on other large table
RiskForIncident14	386.9	With 2-LEFT JOIN with 2 large table
ExposeToGroomingBehaviour	259.5	With 3-INNER JOIN and 1-LEFT JOIN large table using WHERE condition
RiskForIncident1	202	Query a large table based on other table's column data
RiskForIncident18	195.2	With 3-INNER JOIN and 1-LEFT JOIN large table using WHERE condition
RiskForAirportRiskFactor2	190.8	Using NULL predicate and WHERE condition
RiskForAirportRiskFactor4	186.8	Using EXISTS predicate on other tables

Drools[20]; Jess[21] Prova[22]) are based on (a version of) the Rete algorithm (Liang et al. 2009). Triple engines have been used two rule engines: a bottom-up engine and a top-down one for Semantic Web applications (Liang et al. 2009). A few prior systems that used rule engines based on triples are Jena,[23] SwiftOWLIM[24] and BigOWLIM.[25] Knowledgebase engine such as CYC[26] is an integration of domain knowledge for solving classical and common sense reasoning based complex problems. According to Liang et al. (Liang et al. 2009), although Ontobroker was not the best system in all of these rule-based system, Ontobreaker (and partly DLV) was assigned to be the best performing systems.

The architecture of RuleRS is inspired by the PLIS+ application which is a rule-based Personalised Location Information System (Viktoratos et al. 2012). Similar to PLIS+, RuleRS collects and evaluates rules on-the-fly and delivers specialised and contextualised information/inferences according to rule-based policies.

Indeed, another critical issue of the RuleRS architecture is to search and extract relevant information from a relational database and represent it as defeasible facts. RuleRS employs SQL/JSON syntax for describing facts and contextualised information as an intermediate step to bridge relational database information with a rule engine/reasoner.

To summarise, there is a need for a more fine-grained architecture to integrate relational databases, with a logic-based reasoner and rule engine for assisting to take decisions or create reports according to guidelines, policies or regulations that, in addition to being developed using appropriate research methods, should be readily evaluated. A researcher should be able to ensure designed artifacts are reliable and valid by checking for evaluation with other evidence. The evidence is not limited but we can include as experimental models for validating technology could be comprehensively classified as project monitoring, assertion, field study, literature search, legacy, lessons learned, static analysis, replicated, synthetic analysis, dynamic analysis, and simulation (Zelkowitz and Wallace 1998). Experiments could also be classified as "in vivo" (when it is run at a development location) and "in vitro" (when it is run in an isolated, controlled setting) (Basili 1996). Also, experiments could be classified as nine types which could further divide into three general categories: a quantitative experiment (identify measurable benefits of using a method or tool), a qualitative experiment (assess the features provided by a method or tool, and a benchmarking experiment to determine performance (Kitchenham 1996).

[20] http://www.drools.org accessed on 20 January 2017.

[21] http://www.jessrules.com/jess/index.shtml accessed on 20 January 2017.

[22] https://prova.ws accessed on 20 January 2017.

[23] http://jena.apache.org accessed on 20 January 2017.

[24] https://confluence.ontotext.com/display/OWLIMv35/SwiftOWLIM+Introduction last accessed on 20 January 2016.

[25] https://confluence.ontotext.com/display/OWLIMv35/BigOWLIM+Introduction accessed on 20 January 2017.

[26] http://www.cyc.com accessed on 20 January 2017.

In this paper, we also argued that DL is suitable for making decisions based on regulations, policies and relational database information. Recent research has shown that DL is suitable for modelling and inferring from evidence and reasoning in real-world applications where the conflicting situation may appear simultaneously. In the introduction section, we included some of these claims.

None of the prior studies found met all of these requirements, as their focus had largely been on establishing the rule engine by restricting only human behaviour, without direct connection with any software system (Hu et al. 2009) or database systems. In the remainder of the paper, we examine the premise that combining databases with logic reasoner and rule engine into fine-grained report according to regulations and policies.

7 Conclusion

Our preliminary step was to describe an ongoing work on the integration of existing *Online Management system* with compliance and regulations, which also combines Defeasible Logic and SPINdle rule reasoner. In particular, we extend the SPINdle rule reasoner to generate reports for a possible consequence of the risk involving business processes. Classically, compliance and regulations perform a significant role in society, and business processes have to comply with such policies to avoid future risks. It turned out that this type of motivations is defeasibly captured by a rule-based approach to static decision trees for Online Mandatory Reporter Guide (NSW Government 2016).

As a next step, a rule-based architecture, RuleRS has been introduced. It collects information from the relational database of an online management system and represents the data into an intermediate file format using SQL/JSON syntax which is further processed by SPINdle rule reasoner.

Thirdly, we demonstrated that external systems such as as reporting to meet legal requirement could integrate rule reasoner, SPINdle. Typically, these reporting systems contain compliance and regulatory requirements which were converted into a rule knowledge-base. We used a pre-defined DL format which can also be represented using predefined plain text or XML file format. We argued that using this technique external stakeholders are can also be connected with the RuleRS.

On the next, we demonstrated a case study based on an Online Child Care Management, ChildSafeOMS to automate the early identification of children at risk, in particular of abuse and neglect. Preliminary results appear to be very promising, showing a close correspondence between compliance, regulations and online management system. RuleRS.

Finally, on account of motivations of the RuleRS as expert system, we demonstrated a case study based on an Adverse Event Reporting System, FAERS to automate the decision-making for compliant and non-complaint information, in particular on adverse event and medication error reports submitted to FDA. Preliminary results appear to be very promising, showing a close correspondence between compliance, regulations and online management system. Last but not least, RuleRS allows users or agents to produce conclusions and consequences

automatically, based on a given compliance and regulatory knowledge base or generated on the fly by other applications and existing relational database information of an online management system.

Acknowledgements Preliminary version of the material included in this paper appeared at ICAIL 2015 (Islam and Governatori 2015).

References

Antoniou G, Billington D, Governatori G, Maher MJ (1999) On the modeling and analysis of regulations. In: Australian conference on information systems

Antoniou G, Billington D, Governatori G, Maher MJ (2001) Representation results for defeasible logic. ACM Trans Comput Logic 2(2):255–287

Basili VR (1996) The role of experimentation in software engineering: past, current, and future. In: Proceedings of the 18th ICSE, IEEE Computer Society, pp 442–449

Billington D, Antoniou G, Governatori G, Maher MJ (2010) An inclusion theorem for Defeasible Logics. ACM Trans Comput Logic 12(1):1–27

Currim S, Snodgrass RT, Suh YK, Zhang R, Johnson MW, Yi C (2013) DBMS metrology: measuring query time. In: Proceedings of the 2013 ACM SIGMOD, ACM, pp 421–432

Governatori G (2005) Representing business contracts in RuleML. Int J Coop Inf Syst 14(2–3):181–216

Governatori G (2011) On the relationship between Carneades and Defeasible Logic. In: Proceedings of the ICAIL 2011, ACM, pp 31–40

Governatori G (2013) Business process compliance: an abstract normative framework. Inf Technol 55(6):231–238

Governatori G (2015a) The Regorous approach to process compliance. In: 2015 IEEE 19th international enterprise distributed object computing workshop, IEEE Press, pp 33–40

Governatori G (2015) Thou shalt is not you will. In: Atkinson K (ed) Proceedings of the fifteenth international conference on artificial intelligence and law. ACM, New York, pp 63–68

Governatori G, Hashmi M (2015) No time for compliance. In: Enterprise distributed object computing conference (EDOC), 2015 IEEE 19th international, IEEE, pp 9–18

Governatori G, Rotolo A (2004) Defeasible logic: agency, intention and obligation. In: Proceedings of the DEON 2004, Springer, vol 3065 in LNCS, pp 114–128

Governatori G, Rotolo A (2006) Logic of violations: a Gentzen system for reasoning with contrary-to-duty obligations. Australas J Logic 4:193–215

Governatori G, Rotolo A (2008) BIO logical agents: norms, beliefs, intentions in defeasible logic. Auton Agent Multi-Agent Syst 17(1):36–69

Governatori G, Rotolo A (2010) A conceptually rich model of business process compliance. In: Proceedings of the APCCM 2010, ACS, vol 110 in CRPIT, pp 3–12

Governatori G, Shek S (2013) Regorous: a business process compliance checker. In: Proceedings of the fourteenth international conference on artificial intelligence and law, pp 245–246

Governatori G, Padmanabhan V, Rotolo A, Sattar A (2009) A defeasible logic for modelling policy-based intentions and motivational attitudes. Logic J IGPL 17(3):227–265

Governatori G, Olivieri F, Rotolo A, Scannapieco S (2013) Computing strong and weak permission in Defeasible Logic. J Philos Logic 42(6):799–829

Grosof BN (2004) Representing e-commerce rules via situated courteous logic programs in RuleML. Electron Commer Res Appl 3(1):2–20

Hashmi M, Governatori G, Wynn MT (2015) Normative requirements for regulatory compliance: an abstract formal framework. Inf Syst Front 18:429

Herrestad H (1991) Norms and formalization. In: Proceedings of the 3rd international conference on artificial intellilgence and law, ACM, pp 175–184

Hu YJ, Yeh CL, Laun W (2009) Challenges for rule systems on the web. In: Governatori G, Hall J, Paschke A (eds) Rule interchange and applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 4–16

International Business Machines Corporation (1993, 2001) Ibm db2 universal database sql reference version 8. http://bit.ly/IBMdb2s1e80, http://bit.ly/IBMdb2s2e80. Accessed 4 Apr 2016

Islam MB, Governatori G (2015) Ruleoms: a rule-based online management system. In: Proceedings of the ICAIL 2015, ACM, New York, NY, USA, pp 187–191

Kitchenham BA (1996) Evaluating software engineering methods and tool part 1: the evaluation context and evaluation methods. ACM SIGSOFT Notes 21(1):11–14

Kozák J (2011) Rules in database systems. In: Proceedings of contributed papers WDS'11, pp 131–136

Lam HP, Governatori G (2009) The making of spindle. In: Governatori G, Hall J, Paschke A (eds) Rule interchange and applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 315–322

Liang S, Fodor P, Wan H, Kifer M (2009) Openrulebench: an analysis of the performance of rule engines. In: Proceedings of the 18th international conference on World wide web, ACM, pp 601–610

Maher MJ (2001) Propositional defeasible logic has linear complexity. Theory Pract Logic Program 1(06):691–711

NSW Government (2013) Online mandatory reporter guide. http://sdm.community.nsw.gov.au/mrg/screen/DoCS/en-GB/summary?user=guest. Accessed 30 Sept 2014

NSW Government (2016) The nsw mandatory reporter guide. http://www.keepthemsafe.nsw.gov.au/reporting_concerns/mandatory_reporter_guide. Accessed 15 June 2016

Nute D (1994) Defeasible logic. In: Gabbay DM, Hogger CH, Robinson J (eds) Handbook of logic in artificial intelligence and logic programming, vol 3. Oxford University Press, Oxford, pp 353–395

Osman R, Knottenbelt W (2012) Database system performance evaluation models: a survey. Perform Eval 69:471–493

Paton NW, Díaz O (1999) Active database systems. ACM Comput Surv CSUR 31(1):63–103

Sadiq S, Governatori G (2015) Managing regulatory compliance in business processes. In: vom Brocke J, Rosemann M (eds) Handbook of business process management, vol 2, 2nd edn. Springer, Berlin, pp 265–288

Skylogiannis T, Antoniou G, Bassiliades N, Governatori G, Bikakis A (2007) Dr-negotiate—a system for automated agent negotiation with defeasible logic-based strategies. Data Knowl Eng 63:362–380

The PostgreSQL Global Development Group (1996–2015) Postgresql 9.4.4 documentation. http://www.postgresql.org/files/documentation/pdf/9.4/postgresql-9.4-A4.pdf. Accessed 15 Nov 2015

US Food and Drug Administration (2015) FDA adverse event reporting system (FAERS): latest quarterly data files. http://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/ucm082193.htm. Accessed 15 Mar 2016

US Food and Drug Administration (2016) FDA adverse event reporting system (FAERS). https://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/. Accessed 16 Feb 2018

US Government (2014) Records and reports concerning adverse drug experiences on marketed prescription drugs for human use without approved new drug applications. http://bit.ly/eCFR310_305. Accessed 7 Mar 2016

Viktoratos I, Tsadiras A, Bassiliades N (2012) Plis+: a rule-based personalized location information system. In: Proceedings of the RuleML2012@ECAI challenge, vol 874 in CEUR workshop proceedings

Yu P, Chen M, Heiss H (1992) On workload characterization of relational database environments. IEEE Trans Softw Eng 18:347–355

Zelkowitz MV, Wallace DR (1998) Experimental models for validating technology. Computer 31(5):23–31