Automated Reasoning for Regulatory Compliance

Alberto Siena[1], Silvia Ingolfo[1], Anna Perini[2], Angelo Susi[2], and John Mylopoulos[1]

[1] University of Trento, via Sommarive 14, Trento, Italy
{a.siena,silvia.ingolfo,jm}@unitn.it
[2] FBK-Irst, via Sommarive 18, Trento, Italy
{perini,susi}@fbk.eu

Abstract. Regulatory compliance is gaining attention from information systems engineers who must design systems that at the same time satisfy stakeholder requirements and comply with applicable laws. In our previous work, we have introduced a conceptual modelling language called Nòmos 2 that aids requirements engineers analyze law to identify alternative ways for compliance. This paper presents an implemented reasoning tool that supports analysis of law models. The technical contributions of the paper include the formalization of reasoning mechanisms, their implementation in the NRTool, as well as an elaborated evaluation framework intended to determine whether the tool is scalable with respect to problem size, complexity as well as search space. The results of our experiments with the tool suggest that this conceptual modelling approach scales to real life regulatory compliance problems.

Keywords: Conceptual Modeling, Automated Reasoning, Experimental Evaluation, Regulatory Compliance.

1 Introduction

The risk of information system non-compliance with relevant laws is gaining increasing attention from government and business alike, partly because of potentially staggering losses and partly because of growing public concern that, somehow, information systems need to be reined in. This trend has made regulatory compliance of software systems an important topic for Software and Information Systems Engineering: systems must comply with applicable laws (legal norms), in addition to fulfilling stakeholder requirements.

To deal with the problem of regulatory compliance, we need formal models of law that can be formally analyzed through various forms of reasoning to help requirements engineers find compliant solutions. Modeling approaches intended for law have been studied for decades in AI (more precisely, AI and Law), generally grounded on expressive, often modal, logics. Other approaches, grounded in Natural Language Processing and Information Retrieval, support different forms of analysis such as determining case similarity and relevance. We contend that neither heavy-handed logical representations, nor natural language ones properly support the analysis requirements engineers need when they tackle the problem of regulatory compliance. Instead, we propose to use conceptual models of law that sit somewhere between logical and natural language models with respect to complexity, to help requirements engineers answer questions such as "In situation S, what are my alternative ways for complying with law fragment L?" , and if stakeholders have given preferences in addition to requirements, "What is a preferred compliance solution for law L, given situation S?"

The modeling framework for building conceptual models of law and capturing preferences (named Nòmos 2) has been presented in two companion papers [9,20]. This paper focuses on tool support for Nòmos 2. Given the size of law models, tool support is essential for any type of analysis. Accordingly, we have implemented such a tool that is founded on an inference engine (DLV [2,11]) to answer questions concerning compliance solutions in different situations, taking into account stakeholder preferences.

The specific questions addressed in this paper are: (1) What kind of automated reasoning is useful in tackling the compliance problem? (2) Can we have a reasoning tool that scales with the size and/or complexity of law models? In order to answer these questions, we first formalized reasoning mechanisms in terms of axioms, then conducted a series of experiments with the implemented tool. Artificial models of increasing sizes and with different properties were generated automatically and analyzed by the tool. The performance data from a series of runs were collected and analyzed. Our conclusions suggest that indeed the NRTool scales to problems of moderate law size.

The rest of the paper is organized as follows. Section 2 recalls the Nòmos 2 modeling language. Section 3 details the formal framework for reasoning on law models. Section 4 presents the NRTool that answers compliance queries through automated reasoning. Section 5 evaluates the efficiency of the tool through a scalability analysis involving a series of experiments. Section 6 surveys the state of the art and related work, while Section 7 concludes.

2 Baseline

Nòmos 2 [20] is a modeling framework that aims at capturing the variability of law. Indeed, legal texts often contain both a set of norms and elements such as conditions, exceptions and derogations, which make different norms hold under different conditions. These elements define a variability space, intended as alternative ways to comply with the set of norms within the legal text. This trait is captured in Nòmos 2 by differentiating *applicability* and *satisfiability* values for norms, and by defining *compliance* on the bases of the two.

Specifically, a norm is defined as a 5-tuple (*type*, *hol*, *ctrpart*, *ant*, *cons*): *type* is the type of the norm (e.g., duty or right); *hol* is the *holder* of the norm, the role having to satisfy the norm; *ctrpart* is the *counterpart*, the role whose interests are helped if the norm is satisfied; *ant* is the *antecedent*, the conditions to satisfy to make the norm applicable; *cons* is the *consequent*, the conditions to satisfy

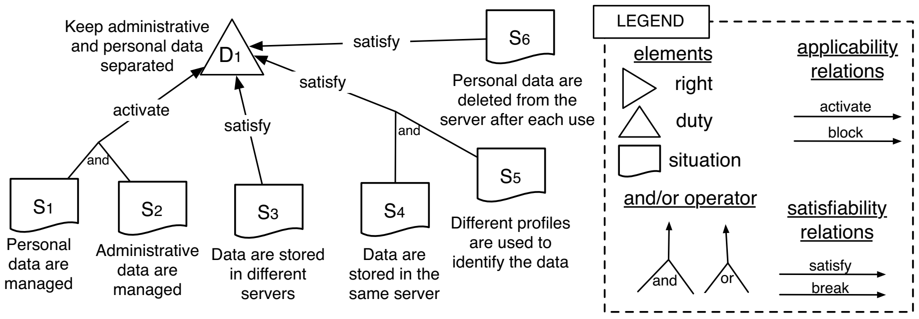

Keep administrative and personal data separated
D1
satisfy
S6
Personal data are deleted from the server after each use
satisfy
activate
satisfy
and
and
S5
S1
S2
S3
S4
Personal data are managed
Administrative data are managed
Data are stored in different servers
Data are stored in the same server
Different profiles are used to identify the data
LEGEND
elements
right
duty
situation
and/or operator
and
or
applicability relations
activate
block
satisfiability relations
satisfy
break

Fig. 1. An example of a Nòmos 2 model representing the duty for an administration office to keep the personal and administrative data of their clients/employee separated

in order to comply with the norm. Consequent and antecedents are modeled in terms of *situations.* A situation is a partial state of the world – or state-of-affairs – expressed through a proposition. For example, "Christmas season 2012" is a situation. A situation can be true, false, or have an unknown truth value. We use abbreviations ST, SF, SU to refer to truth values Satisfied True/False/Unknown, while AT, AF, AU refer to truth values Applicable True/False/Unknown. If the situations make the antecedent true, the norm applies; if the situations make the consequent true, the norm is satisfied. Situations are related to norms and to other situations by four basic relations. The *activate* relation, from a situation to a norm, means that if the situation is satisfied the norms is applicable; viceversa, the *block* relation makes the norm not applicable. The *satisfy* relation, from a situation to a norm or another situation, means that if the situation is satisfied the norm or the target situation is satisfied; viceversa, the *break* relation makes it not satisfied. Additionally, three shortcut relations have been defined between norms, in order to model the cases where one norm *derogates*, *endorses* or *implies* another one (see [20] for more details).

Depending on its applicability and satisfiability value, a norm may have value: *complied with*, *violated*, *tolerated* or *inconclusive.* For example, Figure 1 shows an example of a Nòmos 2 model describing the duties of an administration office that should keep personal data of their employees/clients separated from the administrative data used for running their business. When the two situations s_1 and s_2 hold (both have label ST), the conjunction of their labels is processed by the *activate* relation that propagates an applicability value to the norm (the label AT). Should s_3 hold, the duty will receive a label indicating that it is satisfied (ST). Since there is evidence that the duty is both applicable and satisfied, we say it is *complied.* Should there be no evidence of satisfiability for any of the situations that are linked with a satisfy relation (s_3–s_6 have label SU), the duty would be applicable and not satisfied — i.e., *violated.*

In a Nòmos 2 model when several relations target the same situation or norm (e.g., $s_3 \xrightarrow{sat} D_1$, ($s_4$ and s_5) $\xrightarrow{sat} D_1$, $s_6 \xrightarrow{sat} D_1$), the values propagated to that target are treated as being in disjunction. So, as we can see in the example, different sets of situations can satisfy the duty D_1 and make it *complied with.*

Table 1. Axiom schema for invariants and propagation rules in norm models

Nr.	**Description**	**Axiom**	**Definition**
A1	prioritization for satisfiability	$ST(\phi) > SU(\phi) > SF(\phi)$	$ST(\phi) \wedge SU(\phi) \rightarrow ST(\phi)$,
			$ST(\phi) \wedge SF(\phi) \rightarrow ST(\phi)$,
			$SF(\phi) \wedge SU(\phi) \rightarrow SU(\phi)$,
			$ST(\phi) \wedge SF(\phi) \wedge SU(\phi) \rightarrow ST(\phi)$
A2	prioritization for applicability	$AT(\phi) > AU(\phi) > AF(\phi)$	$AT(\phi) \wedge AU(\phi) \rightarrow AT(\phi)$,
			$AT(\phi) \wedge AF(\phi) \rightarrow AT(\phi)$,
			$AF(\phi) \wedge AU(\phi) \rightarrow AU(\phi)$,
			$AT(\phi) \wedge AF(\phi) \wedge AU(\phi) \rightarrow AT(\phi)$
A3	default satisfiability value	ϕ	$\neg ST(\phi) \wedge \neg SF(\phi) \rightarrow SU(\phi)$
	default applicability value	ϕ	$\neg AT(\phi) \wedge \neg AF(\phi) \rightarrow AU(\phi)$
A4	compliance rule	ϕ	$AT(\phi) \wedge ST(\phi) \rightarrow Com(\phi)$
A5	not applicability rule	ϕ	$\neg AT(\phi) \rightarrow Tol(\phi)$
A6	compliance subsumption	ϕ	$Com(\phi) \rightarrow Tol(\phi)$
A7	duty violation	ϕ	$AT(\phi) \wedge \neg ST(\phi) \wedge Duty(\phi) \rightarrow Vio(\phi)$
A8	right non-exercisation	ϕ	$AT(\phi) \wedge \neg ST(\phi) \wedge Right(\phi) \rightarrow Tol(\phi)$
A9	inconclusiveness	ϕ	$\neg Tol(\phi) \wedge \neg Vio(\phi) \rightarrow Inc(\phi)$
	Description	**Relation**	**Axiom**
A10	satisfy	$\phi \xrightarrow{\text{satisfy}} \psi$	$ST(\phi) \rightarrow ST(\psi)$
A11	break	$\phi \xrightarrow{\text{break}} \psi$	$ST(\phi) \rightarrow SF(\psi)$
A12	activate	$\phi \xrightarrow{\text{activate}} \psi$	$ST(\phi) \rightarrow AT(\psi)$
A13	block	$\phi \xrightarrow{\text{block}} \psi$	$ST(\phi) \rightarrow AF(\psi)$
A14	and-satisfy	$(\phi_1 \wedge \phi_2) \xrightarrow{\text{satisfy}} \psi$	$ST(\phi_2) \wedge ST(\phi_1) \rightarrow ST(\psi)$
A15	and-break	$(\phi_1 \wedge \phi_2) \xrightarrow{\text{break}} \psi$	$ST(\phi_2) \wedge ST(\phi_1) \rightarrow SF(\psi)$
A16	and-activate	$(\phi_1 \wedge \phi_2) \xrightarrow{\text{activate}} \psi$	$ST(\phi_2) \wedge ST(\phi_1) \rightarrow AT(\psi)$
A17	and-block	$(\phi_1 \wedge \phi_2) \xrightarrow{\text{block}} \psi$	$ST(\phi_2) \wedge ST(\phi_1) \rightarrow AF(\psi)$

In order to select one (or a few) way of complying, out of many possible ones, we have extended our framework with a preference relation between situations [9]. The problem of compliance becomes then the *Preferred Compliance Problem* — i.e., the problem of finding a compliant solution to a norm model, given a set of applicable norms, such that the chosen solution best fits stakeholder preferences.

For example in Figure 1, deleting the data every time after each use can be considered a task more time consuming than using different server: we say that s_6 is less desirable than s_3 according to the time criterion ($s_6 <_{time} s_3$). The use of different server profiles in one server can be evaluated at least as desirable as using different servers: ($s_5 \leq_{time} s_3$). However, from an economical perspective, using two servers is more expensive than using one ($s_3 <_{cost} s_4$). Given all the preferences above, possible solutions to the norm model are evaluated and ranked.

3 Formal Analysis of Norm Models

In order to be analyzed, Nòmos 2 models need to be translated into sets of FOL formulas. Formally, a *norm model* is a pair $\{\mathcal{P}; \mathcal{R}\}$ where $\mathcal{P}$ is a set of propositions and $\mathcal{R}$ is a set of relations over $\mathcal{P}$. If $(\phi_1; ...; \phi_n) \rightarrow \phi$ is a relation in $\mathcal{R}$, we call $\phi_1 ... \phi_n$ source propositions and ϕ the target proposition of the relation.

Axioms. In Table 1 we introduce the axioms used to formalize the propagation of satisfiability and applicability values. We use six primitive predicates over propositions: $ST(\phi)$, $SF(\phi)$, $SU(\phi)$, $AT(\phi)$, $AF(\phi)$, $AU(\phi)$, meaning respectively that there is evidence that a given proposition ϕ is satisfied ($ST(\phi)$), not satisfied ($SF(\phi)$) or its satisfaction is undefined ($SU(\phi)$), that its applicability is true ($AT(\phi)$), false ($AF(\phi)$) or undefined ($AU(\phi)$). We establish a total order over satisfiability predicates $ST(\phi) > SU(\phi) > SF(\phi)$, meaning that $x > y \rightarrow \{(x \wedge y) \rightarrow x\}$); e.g., if there is conflicting evidence over ϕ, say $ST(\phi)$ and $SF(\phi)$ then $(ST(\phi) \wedge SF(\phi)) \rightarrow ST(\phi)$. Similarly, we have a total order over applicability predicates: $AT(\phi) > AU(\phi) > AF(\phi)$. Axioms A4–A9 state the four derived predicates for compliance. $Com(\phi)$ indicates that ϕ is *complied with*, being applicable ($AT(\phi)$) and satisfied ($ST(\phi)$). A tolerated norm ($Tol(\phi)$) is also complied with (Tol subsumes Com). A first case of tolerated norm is when the norm is not applicable (A5). Another tolerated case is when a right is applicable but not satisfied (A8) (e.g., a subject having a right but not exercising is a tolerated case). A violation ($Vio(\phi)$) is a case of an applicable duty that is not satisfied (A7). When none of the 3 cases above applies (compliance, tolerance or violation), we say that a norm is inconclusive (A9). Relation axioms define how the relations in a norm model propagate labels in order to deduce primitive and derived predicates. Satisfy/break relations define how positive/negative satisfiability values are propagated (A10, A11). Activate/block define how positive/negative applicability values are propagated (A12, A13). If neither a positive or negative value is propagated, an undefined value is propagated by default (A3). Axioms A14–A17 are used to characterize satisfiability/applicability values in case of a conjunction of values. The axioms for disjunction are defined similarly. For more details see [8].

Different propositions may be preferable over others. To capture this information, we add a set of binary reflexive, antisymmetric and transitive relations $\leq_C \in \mathcal{P} \times \mathcal{P}$, each $\leq_C$ defining a partial order on propositions. Informally, we call these relations *preference relations*, and we read "'$\phi \leq_C \psi$" as "ψ is at least as preferred as ϕ according to criterion C". We let "$\phi =_C \psi$" abbreviates "$\phi \leq_C \psi$ and $\psi \leq_C \phi$", so that "$\phi <_C \psi$" abbreviates "$\phi \leq_C \psi$ and not $\phi =_C \psi$". Informally reads "ψ is strictly more desirable than ϕ according to criterion C". Each criterion C defines a partial order over propositions. Preference relations allow us to record relative preference of stakeholders between propositions, according to different criteria for comparison. Let $\mathcal{C}$ denote the set of all criteria. We can further add relations between criteria, to help comparisons. We can define a hierarchy of domain-specific criteria for comparison, such as, for example: criterion *cost* is an aggregate of criteria *production cost*, *infrastructure cost*, *transportation cost*, etc. Such a structuring can help define aggregation functions and/or procedures to automatically rank alternative sets of propositions.

We do not discuss how preferences are negotiated between stakeholders, since different stakeholders can have opposing preferences over the same criteria. Both the definition of aggregation functions of preferences over criteria, and the negotiation of conflicting preferences are outside the scope of this paper.

Table 2. Propagation rules for satisfiability propagation

Rules for Satisfiability Propagation
R1 `sat_evidence(X1,true)` $\leftarrow$ `sat_evidence(X2,true), situation(X2), satisfy(X2,X1).`
R2 `sat_evidence(X1,undefined)` $\leftarrow \neg$ `sat_evidence(X2,true), situation(X2), satisfy(X2,X1).`
R3 `sat_evidence(X1,false)` $\leftarrow$ `sat_evidence(X2,true), situation(X2), break(X2,X1).`
R4 `sat_evidence(X1,undefined)` $\leftarrow \neg$ `sat_evidence(X2,true), situation(X2), break(X2,X1).`

Propagation Rules. The formal semantics defined in this section allows the support of formal analysis on norm models. To do this we represent a Nòmos 2 model as a database of facts, and using a declarative logic programming language we have define propagation rules that implement the Nòmos 2 axioms of previous section.

For example the propagation rules for the satisfiability of a proposition specify that, if there is a satisfy (or break) relation between two propositions and the source proposition is satisfied, then the target is also satisfied (or not satisfied, respectively). As we can see in table 2, the propagation rules associated with these axioms (A10, A11) ensure that the correct label is propagated depending on the evidence of satisfiability for the source proposition (Rule R1, R3). When there is not evidence of such satisfiability, indeed both relations propagate undefined evidence of satisfiability (Rule R2, R4).

These propagation rules have been defined for all our axioms in accordance with the rules in [20].[1] Once these rules are encoded, the user can therefore query this database of facts and infer the truth values of axioms. In the following section we describe the architecture of a tool (called NRTool), which implements the rules and exploits the DLV framework [11] to compute and verify the norm models.

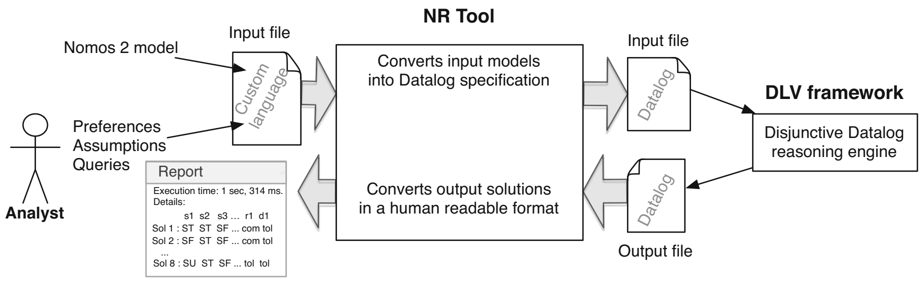
4 Automated Reasoning with Norm Models

The specification and analysis of Nòmos 2 models — formalized in the previous section — is supported by a tool called Norm-Reasoning Tool, or NRTool. With this tool we can perform automated bottom-up and top-down analysis of a Nòmos 2 model, as described in [20], in order to support the analyst answer questions about these models; e.g., what are the applicable norms? Do we comply with a set of norms? What are the alternative ways to comply with a set of norms? etc.

A preliminary evaluation of this reasoning tool and its assessment on a small part of a case study is presented in [9].

Figure 2 describes the overall behaviour of the tool. The tool takes a structured representation of a norm model as input, and converts it into a Datalog logic program. Datalog [1] is a first-order logic program for querying deductive databases. A Datalog program is a set of rules of the form $r\text{:-}\ l_1 \wedge ... \wedge l_n$, where

[1] See the technical report [8] for the full details of all propagation rules.


NR Tool
Input file
Nomos 2 model
Preferences
Assumptions
Queries
Analyst
Custom language
Converts input models into Datalog specification
Converts output solutions in a human readable format
Input file
Datalog
DLV framework
Disjunctive Datalog reasoning engine
Datalog
Output file
Report

```
Execution time: 1 sec, 314 ms.
Details:
          s1  s2  s3 ...  r1  d1
Sol 1 : ST  ST  SF ... com tol
Sol 2 : SF  ST  SF ... com tol
  ...
Sol 8 : SU  ST  SF ... tol  tol
```

Fig. 2. Overall architecture of the NRTool tool. It takes in input the Nòmos 2 model, the preferences, the assumptions and the queries from the analyst and transforms them into a disjunctive Datalog program as input for the DLV engine. The NRTool reports the output of DLV to the analyst in a human readable format.

r (called the head of the rule) is a positive literal, and $l_1, ..., l_n$ are literals (called the body of the rule). Intuitively, the rule states that if $l_1, ..., l_n$ are true then r must be true.

The NRTool relies on the DLV reasoning engine [2, 11] to execute the logic program and perform queries on the norm model. DLV is an Answer Set system that extends Datalog in different ways. It adds *disjunctions* in the rule heads, thus generating multiple alternatives; it adds support for true negations; it also supports weak constraints – i.e., constraints that can be violated at a cost, allowing solutions to be ranked according to the number of violations occurring. The search techniques and heuristics used by DLV are: backward search (similar to SAT algorithms), and advanced pruning operators, for model generation and innovative techniques for answer-set checking. DLV generates as output a complete set of models produced by the set of predicates and assignments to the variables or a pruned set of models that depends on input preferences.

Soft constraints allow us to identify the best compliance solution(s) in terms of their minimality. Since a large number of solutions could be returned – possibly too many – we are interested in having only the *best* solutions. To do this, we adopt the criterion of maximizing the number of not assigned values. The idea is that the fewer are the situations whose satisfiability is set to true or false in a compliance solution, the less analysts are constrained; viceversa, the more "undefined" situations we have, the more analysts are free to make their own decisions. By adding a soft constraint that situations' satisfiability should be neither true nor false, we force the selection of the solutions with the highest possible number of "undefined" values.

The NRTool maps situations and norms into Datalog facts, while relations are mapped into deduction rules. Moreover, the Nòmos 2 model is encoded via ground formulae (without variables and logical quantifiers). The output of DLV is parsed by NRTool that presents it to the user in the form of a report specifying the truth value of the situations in the model and their respective compliance values.

Example: which set of norms is applicable? In this example we will see how the tool works in order to evaluate the applicable norms to a scenario. Answering to this question involves performing a forward reasoning analysis on the Nòmos 2 model. Given an initial values assignment for situations (expressed by the assumption), forward reasoning focuses on the propagation of these values to the norms accordingly to the propagation rules of Nòmos 2. The norms will receive applicability and satisfiability value depending on the relations in the law model. After translating this model into Datalog, the NRTool enables the reasoner to apply the propagation rule and calculate the applicability values. The NRTool then parses the output and returns to the analyst the set of norms of the model that are applicable.

5 Evaluation

Laws usually consist of tens or even hundreds of pages of natural language text, resulting in large models that may involve tens of thousands of concepts and links. In this section we investigate the scalability of our proposed reasoning tool with respect to the following three criteria (research questions):

RQ1. How does the tool scale with respect to the **size** of the problem, defined as the number of elements in the model?
RQ2. How does the tool scale with respect to the **complexity** of the problem, defined as the number of relationships constraining the different elements of the model? Also, how does the tool scale w.r.t. the number of solutions?
RQ3. How does the tool scale with respect to the **space of alternatives**, defined as the number of alternative refinements?

To answer these questions we have set up a testing framework, capable of producing artificial norm models with desired properties, run compliance analysis and record execution times. All experiments have been performed on an Intel i7 eight core 2.80 GHz computer equipped with 6 GB of memory running Linux version 2.6.18. The tool, the setting data, and the results generated by the experiments are available at `http://selab.fbk.eu/lawvariability/`.

5.1 Results

RQ1. To answer the first research question we have set up an experiment that tests the behaviour of the tool when the size of the norm model grows. A first model (the *input model*) was initially manually created. It consisted of 4 norms and 10 situations. Starting from this input model, 50 *test models* were then automatically generated. The generation algorithm consisted in: (i) creating a number (from 1 to 50) of replicas of the input model, resulting in models of size from 15 to 13875 nodes; (ii) creating a root norm that represents the full law; the root norm is and-refined into the root norm of each replica, through the *imply* relation; and (iii) adding a fixed number (10) of random relations from each replica to others. The number 10 was selected to ensure that our model has a sufficient connectivity — indeed in our second experiment we study the impact of having different number of relations between replicas. The randomness of these relations was controlled by a parameter in our configuration called 'seed'. It is worth noting that by changing this seed, the random relations also change, thus creating similar but not identical test models. The experiment was run 5 times with the same input model but different seeds.

The results of this experiment are reported in Figure 3. The figure reports on the x-axis the size of the model, expressed as number of nodes (including all types of Nòmos 2 concepts) of the test model. The y-axis reports the time taken at each execution to identify the set of solutions. There is a difference in one run, which results to be steeper, indicating a dependence of the slope from the seed; but the overall trend is quite linear in the considered problem size interval. In [20] we show an extract of a Nòmos 2 model for one column of HIPAA's section 164.502, where we identified 15 situations. Given that the entire law consists of approximately 250 columns, an estimate of 4000 situations for the whole law is well within the boundaries of the models we tested.

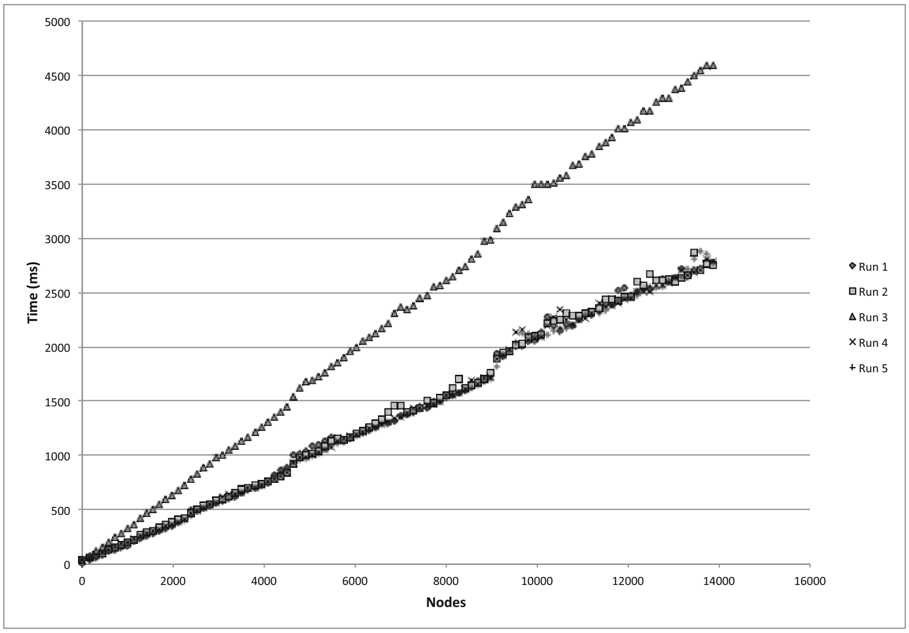


Fig. 3. Results from the experiments testing scalability wrt the size of the problem

RQ2. Answering the second research question requires understanding how the tool behaves when the connectivity of its input model changes. As in the previous case, we started from an input model and produced 700 test models. Differently from the previous case, now we kept fixed the size of the model, expressed as number of nodes, to a value of 225. Then, a random number of relations, varying from 0 to 750, were added to produce the test models.

The results of the experiments are reported in Figure 4. The x-axis reports the "connectivity" parameter — i.e., how many random relations have been added to the model. The y-axis reports the time taken at each execution to find the solutions. The left figure reports the execution time for all the connectivity values. A timeout of 60 seconds had been set, and in the first 7 runs the timeout was reached. For the remaining connectivity values execution time decreases significantly. The right figure magnifies the runs from a value of connectivity 8 to 500 and basically highlight the trend that is not possible to see in the left figure. In these cases the execution time decreases slightly and then increases again, smoothly. The reason of this behavior is that for unconstrained Nòmos 2 models (i.e., models with few relations among nodes) the number solutions depends exponentially on the number of nodes N (3^N, to be exact). As relations are added, the number of solution decreases, as does the time to find all of them. As more relations are added, the complexity of the problem to be solved — defined by the number of relations over a fixed graph — overtakes the cost of finding all solutions. This peculiarity results in the increasing trend shown on the graph of Figure 4. Besides this, we see a trend that decreases until a connectivity of 50, and then it increases smoothly.

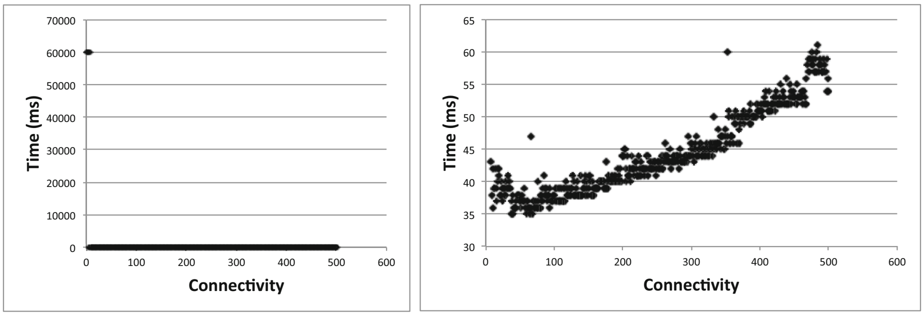


Fig. 4. Results from the experiments testing scalability wrt the complexity of the problem

RQ3. To answer the last research question we set up an experiment to analyze how the behaviour of the tool changes when specific constructs are introduced into the models. The constructs are those that theoretically change the number of available solutions, and so the space of solutions change without changing neither the size of the model nor its connectivity. The experiment was run with a model of 14000 nodes. 5 variations of the input model have been created. At each variation, the proportion between AND-relations and OR-relations has changed, from a value of OR of 0% (i.e., all the relations are in AND) to 100% (i.e. all the relations are in OR).

The results are shown in Figure 5. Here, the results are roughly the same with connectivity values of 0, 25% and 50%. With values of 75% and 100% (where the OR-decomposition becomes prevalent) times increase by approximately 20% passing from an average of 230 ms to 275 ms.

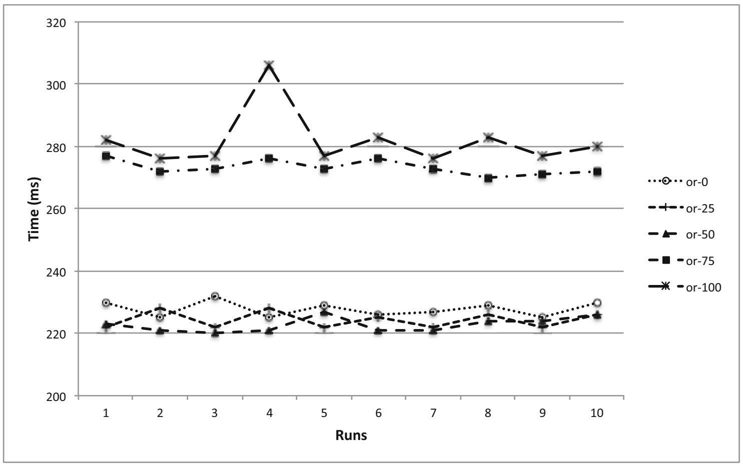


Fig. 5. Results from the experiments testing scalability wrt the space of alternatives

5.2 Discussion

The results of these experiments are twofold. On the one hand, we see a very encouraging linear trend for execution times, which generally corresponds to at most a few seconds. On the other hand, in the second experiment, we see in some cases times running out of bounds. This is due to the difference between *searching for a solution set* and *exploring the solution set*. The exploration time may overcome search time and diverge if the model is highly sparse and the space of alternatives is extremely large. Also with the third experiments we confirmed how the space of alternatives directly influences execution time. Moreover, as we can see from the second experiment, the initial constraints added to the model resulted first in a reduction of the time (as the number of solutions was decreasing) but then complexity kicks-in increasing overall execution times. The result of this experiment is comparable with similar investigations performed, e.g., in [17]. The lesson learnt from these experiments is that conceptual models can be a viable solution in analyzing laws for compliance, but only if the modelled laws are not too under-constrained. Given that laws are generally comprised of a high number of conditions, exceptions, derogations, cross-references and so on, we expect that real law models are not under-constrained.

6 Related Work

The main focus of this paper is the investigation of automated reasoning techniques to enable compliance analysis on real-size conceptual models of laws, including experimental evaluation of scalability. Similar approaches can be found in conceptual modelling for complex socio-technical systems [14], and for goal modelling in requirements engineering [7]. Relevant works on experimental evaluation of scalability of reasoning techniques and related tools, include the following.

In [11] the evaluation of DLV has been performed considering several problems, such as Traveling Salesperson or Quantified boolean formulas, having an increasing theoretical complexity (from NP to Σ_2^P). For each one of these problems, models of growing dimension have been considered; the performance of the solver has been measured in terms of the time necessary to solve the problems. In [17] is presented a method for randomly generating clausal formulae in modal logics. The paper describes several expected properties of good test sets, such as representativeness, reproducibility, parametrization, and presents the generating algorithms that produce 3CNF formula of growing complexity. Finally [5] characterizes hard SAT problems and identifies a "phase transition" in the problem attribute space.

In our work, we generate artificial norm models by cloning a manually defined input model. A similar procedure is used in [22], where a goal based framework for monitoring and diagnosing software requirements is presented. Thus, in our experimental evaluation we can generate models with increasing number of nodes (i.e. norms and situations), and increasing percentage of "AND/OR", "activate/block", and "satisfy/break" relationships, in a controlled way.

Worth pointing out that empirical evaluation in conceptual modelling has a wider scope with respect to what addressed in our paper, which, as reminded above, focuses on one specific but essential property to enable conceptual reasoning for regulatory compliance, namely scalability of automated reasoning. Addressing a different purpose with respect to ours in this paper, the empirical evaluation of conceptual models has been investigated from a domain understanding perspective. In this direction, [4] proposes a framework for the empirical evaluation of conceptual modeling grammars. [15] instead propose four criteria to evaluate conceptual modeling techniques. Differently from our work, these guidelines also focus on the effectiveness of the grammar modeled and the criteria to chose for its evaluation (independent variables, participants, ...). Recker [18] pursues a more philosophical-paradigmatic directions and discusses how existing evaluation methods can be assessed through the Bunge-Wand-Weber ontology. For a general overview on quality frameworks for conceptual modeling, Daniel L. Moody [12] presents a review of research in this field, identifies some theoretical and practical issues, and advocates the need of a common standard for the evaluation of quality of conceptual models.

Concerning the underlying approach in our work, namely law modeling for supporting compliance analysis, relevant related work are modeling approaches for RE and law, which extend existing RE modeling languages. For example, in [16] time line visualizations and decision trees are used to model legal terms or regulations in contracts. In [13] a Semantic Process Language (SPL) was created by combining Petri nets and a formal language, to describe legal regulations. In [21] business process models are checked for legal compliance through a modeling method called Event-driven Process Chain (EPC). However, all these approaches assess a different and specific aspect of legal compliance and appear therefore relatively isolated. [10] proposes a framework that supports analyzing the compliance of legacy Information Systems, which rests on the alignment of a model of the transactions in the legacy system with an ontology of the laws that regulated the IS domain. This law ontology explicates the organizational roles, which correspond to the legal subjects of the laws governing the IS domain, with the domain artifacts and processes under their responsibility. Goal oriented techniques have also been used to represent legal prescriptions. For example, Darimont and Lemoine have used KAOS to represent objectives extracted from regulation texts [3]. Ghanavati et al. [6] use URN (User Requirements Notation) to model goals and actions prescribed by laws. Likewise, Rifaut and Dubois use $i*$ to produce a goal model of the Basel II regulation [19].

7 Conclusions

In this paper we have presented an implemented reasoning tool that supports situational analysis of law models. The technical contributions of this work include an axiomatic formalization of the reasoning mechanism realized by the tool, as well as its implementation based on an off-the-shelf inference engine (DLV). In addition, we report on a series of experiments that evaluated the tool for scalability with respect to problem size (the size of the model being analyzed), problem complexity (measured by the inter-connectivity of nodes in a model), and the space of alternatives (measured by the number of alternative refinements in a model). The results of these experiments suggest that the tool scales to real regulatory compliance problem involving a full law such as HIPAA.

The main limitation of our work is that our evaluations used artificial models. Accordingly, an important future research task will be the evaluation of the tool using Nòmos 2 models of real law. To this end, we need tools that support the generation of law models that are a good-enough approximation of a real law. Our future plans include exploring how to exploit existing tools for legal text analysis to support the extraction of Nòmos 2 models from text. Also, we plan to extend our language and reasoning tool to provide support for compliance analysis based on legal and social roles, delegations and related concepts.

Acknowledgments. This work has been supported by the ERC advanced grant 267856 “Lucretius: Foundations for Software Evolution” (April 2011 – March 2016) http://www.lucretius.eu.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Alviano, M., Faber, W., Leone, N., Perri, S., Pfeifer, G., Terracina, G.: The disjunctive datalog system DLV. In: de Moor, O., Gottlob, G., Furche, T., Sellers, A. (eds.) Datalog 2010. LNCS, vol. 6702, pp. 282–301. Springer, Heidelberg (2011)
3. Darimont, R., Lemoine, M.: Goal-oriented analysis of regulations. In: ReMo2V, held at CAiSE 2006 (2006)
4. Gemino, A., Wand, Y.: A framework for empirical evaluation of conceptual modeling techniques. Requirements Engineering 9, 248–260 (2004)

5. Gent, I.P., Walsh, T.: Beyond np: the qsat phase transition. In: Hendler, J., Subramanian, D. (eds.) AAAI/IAAI, pp. 648–653. AAAI Press / The MIT Press (1999)
6. Ghanavati, S., Amyot, D., Peyton, L.: Towards a framework for tracking legal compliance in healthcare. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 218–232. Springer, Heidelberg (2007)
7. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the tropos methodology. Eng. Appl. of AI 18(2), 159–171 (2005)
8. Ingolfo, S., Siena, A., Jureta, I., Susi, A., Perini, A., Mylopoulos, J.: Reasoning with stakeholder preferences and law. research report. Technical report, University of Trento, Italy, TR-DISI-12-042 (2012), `http://selab.fbk.eu/lawvariability/`
9. Ingolfo, S., Siena, A., Jureta, I., Susi, A., Perini, A., Mylopoulos, J.: Choosing compliance solutions through stakeholder preferences. In: Doerr, J., Opdahl, A.L. (eds.) REFSQ 2013. LNCS, vol. 7830, pp. 206–220. Springer, Heidelberg (2013)
10. Khadraoui, A., Leonard, M., Thi, T.T.P., Helfert, M.: A Framework for Compliance of Legacy Information Systems with Legal Aspect. In: AIS Trans. Enterprise Sys. GITO Publishing GmbH (2009)
11. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Trans. Comput. Log. 7(3), 499–562 (2006)
12. Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data & Knowledge Engineering 55(3), 243–276 (2005)
13. Olbrich, S., Simon, C.: Process modelling towards e-government - visualisation and semantic modelling of legal regulations as executable process sets. In: European Conference on E-Government (ECEG), pp. 405–414 (June 2007)
14. Paja, E., Dalpiaz, F., Poggianella, M., Roberti, P., Giorgini, P.: Sts-tool: Socio-technical security requirements through social commitments. In: Heimdahl, M.P.E., Sawyer, P. (eds.) RE, pp. 331–332. IEEE (2012)
15. Parsons, J., Cole, L.: What do the pictures mean? guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. Data & Knowledge Engineering 55(3), 327–342 (2005)
16. Passera, S., Haapio, H.: Facilitating collaboration through contract visualization and modularization. In: ECCE 2011, pp. 57–60. ACM (2011)
17. Patel-Schneider, P.F., Sebastiani, R.: A new general method to generate random modal formulae for testing decision procedures. J. Artif. Intell. Res (JAIR) 18, 351–389 (2003)
18. Recker, J.C.: Conceptual model evaluation towards more paradigmatic rigor. In: CAiSE 2005 Workshops, pp. 569–580 (2005)
19. Rifaut, A., Dubois, E.: Using goal-oriented requirements engineering for improving the quality of iso/iec 15504 based compliance assessment frameworks. In: RE 2008, pp. 33–42 (2008)
20. Siena, A., Jureta, I., Ingolfo, S., Susi, A., Perini, A., Mylopoulos, J.: Capturing variability of law with Nòmos 2. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 383–396. Springer, Heidelberg (2012)
21. Speck, A., Feja, S., Witt, S., Pulvermüller, E., Schulz, M.: Formalizing business process specifications. Comput. Sci. Inf. Syst. 8(2), 427–446 (2011)
22. Wang, Y., McIlraith, S.A., Yu, Y., Mylopoulos, J.: Monitoring and diagnosing software requirements. Autom. Softw. Eng. 16(1), 3–35 (2009)