Exploring Legal Business Process Paths

Sepideh Ghanavati, Silvia Ingolfo, Alberto Siena

Exploring Legal Business Process Paths

Sepideh Ghanavati[1], Silvia Ingolfo[2], Alberto Siena[3]

[1] CRP Henri Tudor, Luxembourg City, Luxembourg
[2] University of Trento, Trento Italy
[3] FBK, Trento, Italy
sepideh.ghanavati@tudor.lu, silvia.ingolfo@disi.unitn.it, siena@fbk.eu

Abstract. Nowadays, enterprises are very complex systems, often comprised of a large number of business processes run by actors working together to achieve business objectives. Ensuring compliance with applicable laws is mandatory to avoid heavy penalties or even business failure. To this purpose, an increasingly important challenge consists of finding and resolving discrepancies between strategic goals, business processes and laws. In this paper, we envisage a formal approach that uses two modeling languages, User Requirements Notation (URN) and Nòmos, to represent enterprise goals, processes and applicable laws. Automated reasoning techniques allow us to analyze models for compliance checking and detecting conditions of unwanted concurrent executions.

1 Introduction

Modern enterprises are complex systems comprised of actors, as well as software and hardware components working together in interleaved processes to support high-level enterprise objectives. Well-designed processes are essential to ensure efficiency and effectiveness in the production of goods and services. Enterprises are also subject to laws and regulations which can be costly to comply with. Being non-compliant may introduce more cost to the enterprise, such as heavy fines, bad reputation or business disruption. Since business processes can be very large and articulated, they generate a potentially large number of alternative execution paths. Similarly, laws are generally comprised of a large set of conditional elements, such as conditions, exceptions and so on, which create an even potentially larger number of admissible paths to comply. This raises the need to ensure that (i) every possible process or process path respect complies with applicable laws; and (ii) for every law or law fragment, one (or more) process or process branch exists, which fulfills the necessary legal accomplishments. While doing (i) and (ii), the achievement of strategic objectives must also be ensured.

Several work has been done to analyze the compliance of business processes with laws and regulations. Legal-URN [1], [2], based on User Requirements Notation (URN) [3], is one of the main frameworks which aims to model regulations, organizational goals and business processes in the same notation and provides analysis for compliance. However, this approach does not explore the alternatives for compliance in detail. In [4] obligations are decomposed into operational

level rules, and Key Performance Indicators are used to measure compliance objectives and balance them with business objectives.

Although few approaches exist for business process compliance, in a recent systematic literature review [5], the authors identify the needs for having a concrete framework with guidelines on how to map legal prescriptions to business processes and how to analyze the compliance.

In [6] a modeling approach is presented, which focuses on modeling the organizational regulatory space (ORS), comprised by both, the internals of the enterprise, such as business processes and goals, and applicable regulations. In [7] a new approach is proposed for assessing compliance of business processes using the compliance checker Regorous. The used language defines a more fine-grained concept of obligation and compliance. Using FCL, a rule-based logic which combines defeasible and deontic logic, the authors formalize regulations and validate the business process model.

In this position paper we envision an approach for the exploration of business process paths to detect potential violations of the law. We define an *legal business process path* a path in the process such that the supported requirements are satisfied, while the applicable norms are not violated. We model the process in Use Case Maps (UCM), which is the scenario notation part of URN, and the requirements in Goal-oriented Requirements Language (GRL), which is the goal modeling notation of URN. Other business process modeling such as BPMN or UML activity diagrams can be used instead of UCM, however, the advantage of UCM over these approaches is that it has links to GRL for goal models and includes capabilities for analyzing scenarios, conflict detection and propagating the result of the analysis from UCM to GRL and vice-versa.

The applicable norms are represented in Nòmos [8], a modeling language for representing laws and regulations. The novelty of our approach stays in the capability to model *alternatives* in the combination of processes, law prescriptions and goals. Also in our approach, we support automated reasoning to explore *exhaustively* the models, ensuring compliance with laws and allowing at the same time, to achieve enterprise goals that the processes have to support.

The paper is structured as follows: Section 2 introduces the underlying modeling techniques; Section 3 presents the envisioned modeling approach and shows how reasoning is supported; finally, Section 4 concludes the paper and outlines some future work.

2 Baseline

2.1 User Requirements Notation (URN)

The User Requirements Notation (URN) [3] is an International Telecommunications Union (ITU-T) standard which helps requirements engineers and business analysts documenting requirements and/or analyzing these requirements for correctness and completeness. URN combines two modeling notations, *Goal-oriented Requirement Language* (GRL) and the *Use Case Maps* (UCM) [9]. GRL

is used to model goals and Non-Functional Requirements (NFR) with goals and softgoals, and provides means to reason about alternatives. UCM aims at modeling scenario concepts of operational and functional requirements, and help reasoning about performance and architectural decisions [10].

Goal-oriented Requirements Language (GRL) – The Goal-oriented Requirement Language (GRL) is a goal modeling notation based on i^* language and the NFR Framework's concepts and syntax. High-level business goals, NFRs and the alternatives are modeled with intentional elements in GRL. GRL also models beliefs to capture the rationales behinds stakeholders' decisions, as well as stakeholder dependencies.

GRL intentional elements are: softgoals (), goals (), tasks (), beliefs or resources (). Softgoals are abstract and have no clear measure of satisfaction while goals are quantifiable and can be fully satisfied. Softgoals usually model NFRs and quality requirements, whereas goals deal with and model functional requirements. Tasks capture solutions or alternative means to achive goals or softgoals. Resources are sometimes utilized to achieve tasks, goals and softgoals. Actors (), represent stakeholders of the system who can have certain goals to achieves and set of tasks to perform.

GRL includes a set of link to connect intentional elements to each other. These links are: decomposition links (), contribution links (), correlation links () or dependency links (). Decomposition links are used to decompose an intentional element into sub-elements and can be a type of AND, IOR, or XOR. XOR and IOR decomposition links. Contribution links which can have qualitative or quantitative values illustrate the impact of one intentional element's satisfaction on another intentional element's satisfaction value. Correlation links are similar to contribution links, but indicate side-effects of one intentional element on the other. Dependency links model relationships and the dependencies between actors.

To analyze the satisfaction value of each intentional element as well as the overall satisfaction of actors in the system, GRL provides both bottom-up and top-down evaluation mechanisms. These evaluation mechanisms can also be quantitative, qualitative, hybrid or constraint-based. The detail of these evaluations mechanism are explained in [11].

Use Case Maps (UCM) – UCM aims at modeling scenarios and use cases. It depicts the causal sequences of tasks and activities allocated to *components* (). Components can be actors, agents, roles, software modules, sub-systems, etc. and they can be decomposed into sub-components. Scenario paths connect *start points* () to *end points* (). Paths contain *responsibilities* () which are the actions and activities that need to be done. They can be performed in sequence, concurrently (), or as alternatives ().

UCM has the capability to decompose complex scenario maps into several sub-maps (i.e. plug-in maps) via *stubs* (). Stubs can be static or dynamic. Input and output of the stubs are connected to the start points and end points in the plug-in to ensure scenario continuity across various levels of details. Dynamic stubs are used to specify alternative maps in the same location.

To define *scenarios* in UCM, preconditions and postconditions are captured. Scenarios are triggered and started when a set of preconditions are satisfied. Based on the set of the preconditions, one alternative path is taken at any point in time. Resposibilities which are usually linked to tasks or other intentional elements in GRL can get the satisfaction value of the tasks or intentional elements linked to them. When the preconditions, postconditions and other values in a scenario are defined, a a *path traversal mechanism* helps simulating different scenarios and the traversed paths. *Path traversal mechanism* can be used in regression testing and it provides operational semantics to UCM models.

2.2 Nòmos 2

Nòmos 2 is a modeling language that aims at capturing the variability of compliance alternatives for norms [8]. Nòmos 2 models allow us to represent fragments of laws or regulations by representing the different conditions and rules described by a law or regulation, and the alternative ways to comply with it. The conditions on a norm's applicability and satisfiability are represented through the concept of situations denoting states-of-affairs (partial states of the world), such as "Christmas season" or "Driving on the highway". Situations are partial states of the world that we may know to hold (meaning the situation is satisfied), not hold, or neither (when we can't conclude satisfaction or denial). If some situations are satisfied, the norm will apply, and when other situations are satisfied, the norm will be satisfied. In our model, situations are linked to norms in terms of four basic relations acting as a label-propagation mechanism to identify when situations make a norm applicable/not-applicable (activate, block), or satisfied/not-satisfied (satisfy, break). When a norm (duty or right) is applicable, it is *complied with* when it is satisfied and *violated* if it is not satisfied. If the norm is not applicable or it is unknown whether the norm applies or not, it is either *tolerated* or *undefined* (see [8] for more details). Relations between norms are used to represent exceptions and special other cases where norms may make other norms applicable, not applicable, or complied with.

Figure 1 represents an example of a Nòmos 2 of a hypothetical tax law. When a product is bought from a seller the duty to pay taxes applies ($S_1 \xrightarrow{\text{activate}} D_1$). In order to satisfy the norm — and therefore comply with it — it is possible to either pay the taxes ($S_2 \xrightarrow{\text{satisfy}} D_1$), or by filling in the VAT-claim tax form ($S_3 \xrightarrow{\text{satisfy}} D_1$). For example, since VAT-free product are untaxed by definition, if the product is VAT-free (and S_4 holds), the duty is no longer applicable ($S_4 \xrightarrow{\text{block}} D_1$). For the purpose of this example, we consider that the stores allow a return policy when a valid receipt of the purchase is shown. The two situations — of 1) having bought a product and 2) having a valid receipt — activate the right to return the product to the store, which in turn is complied with when the product is indeed returned ($S_6 \xrightarrow{\text{satisfy}} R_1$). However, if the product is damaged, then the right is no longer applicable ($S_7 \xrightarrow{\text{break}} R_1$). The main idea behind Nòmos 2 is that leveraging on the applicability and satisfiability of different norms and

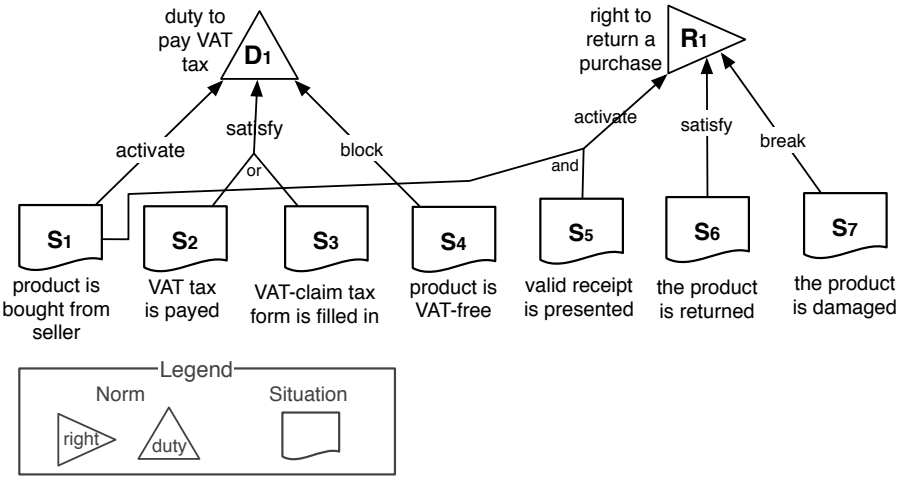

Fig. 1. An example of a Nòmos 2 model.

situations holding, it is possible to identify how to comply with a law in different ways (e.g., by paying the VAT tax, by buying a VAT-free product, ...).

3 Legal Business Process Paths

A business process consists of a set of activities to be executed in sequence. Gateways, such as decision points and parallelisms, split the activity sequence into different possible flows, potentially executed in parallel by multiple actors (lanes). The presence of parallelism, in particular, generates a sort of indeterminism in process execution, which in turn causes multiple execution instances to be possible, depending on the concurrency conditions. For example, given two activities, a and b, a parallelism between them implies that three executions exist: a is executed before b, b is executed before a, and a and b simultaneously. If there is a legal constraint on the output of a, b, or both, it becomes necessary to identify these alternative executions and explore them to detect potential violation conditions of a business process, in order to change the process structure.

3.1 Modeling

To design a process that satisfies strategic goals while complying with applicable laws, we need a modeling language capable to model business processes, goals and laws. There are mature, standardized languages to model business processes, such as BPMN; also, there are modeling languages to model goals and some to model legal prescriptions. However, to model the three aspect of the problem and be able to support automated reasoning, we need to combine two or more existing languages. In this paper, we evaluate the combination of URN and

Nòmos since the integration of goals and business process models has already been made in URN standard.

Figure 2 depicts an example of our approach. The picture represents an electronic commerce scenario, in which tax laws have to be taken into account. Two actors are represented, the Buyer and the Seller, and two business processes are shown: one process involves both the Buyer and the Seller, while the other concerns only the Buyer. Different activities have been introduced in both processes to comply with the legal provisions already depicted in Figure 1. The processes run in parallel, so there is no way to ensure that a certain flow of activities is executed. While we can ensure that when each activity is performed, its corresponding legal requirement is satisfied, it is possible that performing a combination of activities results in a violation.

To represent a satisfiability conditions between the business process and the law, we use traceability links from the path branch of the business process to the corresponding situation in Nòmos model. For example, in the business process related to the tax law, two conditional verifications must happen. First, it is necessary to check if the product is tax free. If it is, situation S4 is satisfied. Next, if the product is not tax-free, the second check has to be done to verify whether the buyer filled up the tax exemption form or not. The two paths created from this satisfy S2 and S3 in Nòmos.

In the process of 'returning the product', the seller only accepts the product as returned if three conditions (i.e. having bought the product, having valid receipt, and returning the purchased product with no damage) are satisfied. These three conditions satisfy the situations, S5, S6 and S7 in Nòmos respectively. If the buyer does not satisfy all of the three conditions, then the main goal, `return product` will not be satisfied, as shown with red X in Figure 2.

In previous works [8], we have used Nòmos to model pieces of real laws and studied the scalability of this modeling approach for legal document. Therefore — despite using a preliminary small example for this paper — we are confident in the ability of this approach in capturing all of the possible combinations that may emerge from very complex processes.The combinations emerge in particular from the design of the business processes and law, and their execution. This is achieved by using formal reasoning on the models, as described in the following.

3.2 Formal reasoning

Once a process starts, at any given time t_k, a set of activities, from 0 to n, are executed. The trace of the process at time t_k consists of all the activities that have been executed from time t_0 to time t_k. Figure 3 illustrates the exhaustive generation of multiple paths within a given business process. In the left part, the figure depicts a simple business process with 5 activities and two branches executed in parallel. Due to the parallelism, many different execution traces can be defined: in the right part, 9 different partial traces defined by the process are shown. In the first trace, the process has been started (activity 's' executed); in the second, 'x1' has been executed; and so on. At a certain point, when the activities can be executed in parallel, the number of possible traces grows rapidly

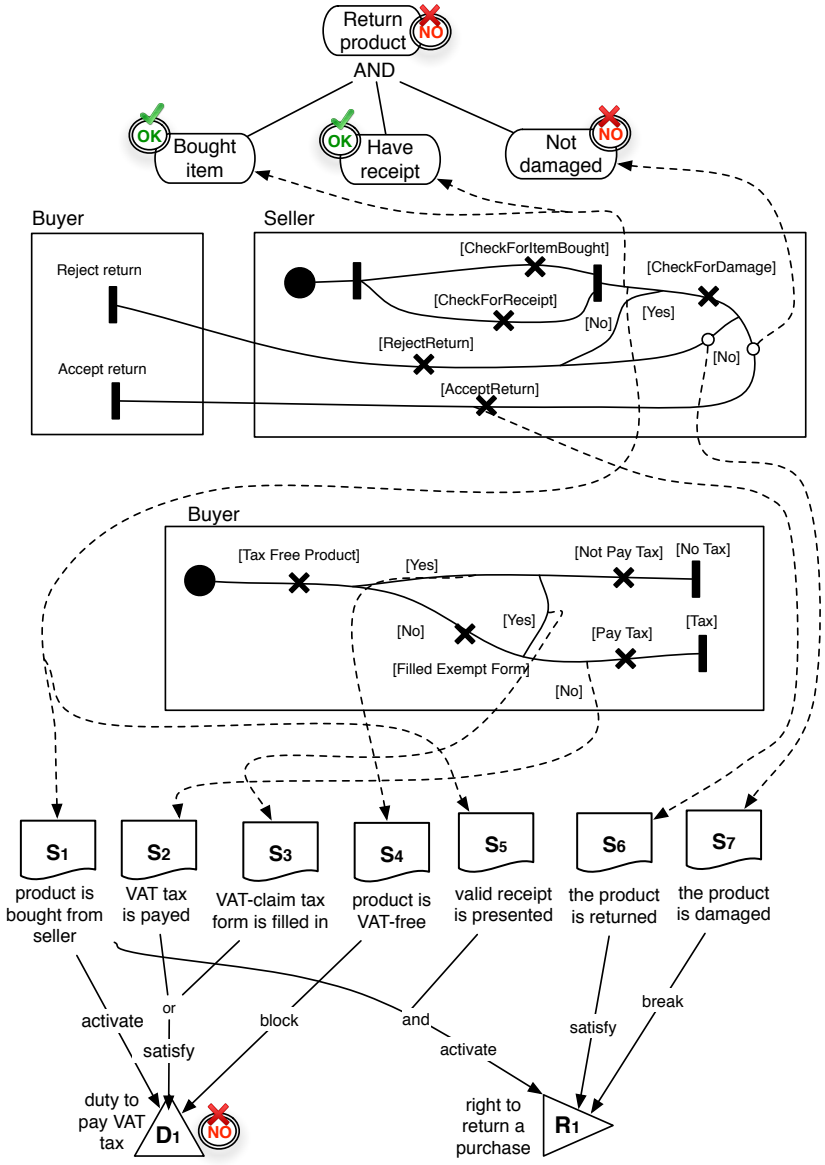

Fig. 2. Example of URN-Nòmos Reasoning

because of the concurrency conditions described above. We hypothesize that, given a process and a law, each activity of the process can bring about an arbitrary number of conditions defined in the law to be compliant with, that are the legal prescriptions. The set of activities in a trace form an execution assignment to that prescriptions: the activities that belong to the trace are considered to

actually fulfill the legal prescriptions; while the activities that do not belong to the trace leave the corresponding prescriptions not fulfilled. Depending on how the law is structured, failing in fulfilling one or more prescriptions can generate a violation to that law. To generalize, given a process P and a set of norms L, we define a *legal business process path* as a path in the process, such that for each possible trace in the path L is not violated. The intuition behind the present work is that we can use URN and Nómos to enforce modeling business processes that satisfy stakeholder requirements and at the same time comply with applicable laws. In particular, URN offers basics to model requirements with GRL, and business processes with UCM, and to establish *traceability links* between them. Nòmos allows modeling of the legal prescriptions and provides support for automated reasoning. Specifically, Nòmos enables for *exhaustive* exploration of legal alternatives within a given model of law. Adopting similar traceability links between business processes and laws, we can perform exhaustive search in the space of traces generated by a given business process.

Table 1 illustrates how our approach can be formalized using Disjunctive Logic Programming (DLP) [12]. DLP is a declarative, first-order logic language and a deductive system, where facts and deduction rules are expressed as predicates of the logic language. Disjunctions may appear in the rule heads to allow multiple alternative consequences to be drawn from a rule. Solvers, such as

Table 1. An excerpt of the formalisation in DLP

```
done(s,p) :- start(p).

done(x1,p) v pending(x1,p) :- done(s,p).

%parallelism
done(x21,p) v pending(x21,p) :- done(x1).
done(x22,p) v pending(x22,p) :- done(x1).

done(x23,p) v pending(x23,p) :- done(x22).

done(x3,p) v pending(x3) :- done(x21,p), done(x23,p).

done(e,p) v pending(e,p) :- done(x3,p).

end(p) :- done(e,p).

% link to situations
st(s1) :- done(x1,p).
sf(s1) :- not done(x1,p).

start(p) v not_start(p).
```

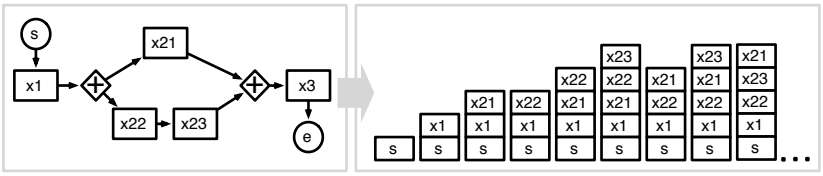

Fig. 3. Set of possible traces defined on a given process.

DLV [13] allows *exhaustively* exploring the alternative models defined on a set of predicates, searching for desired properties.

We use DLP to introduce alternative traces, as in [14]. Each activity in a given process is represented as a variable. If an activity `x` is executed in a process `p`, the predicate `done(x,p)` is triggered. If the activity that precedes `x` has been executed, but `x` still has not been executed, the predicate `pending(x,p)` is triggered. After `x` has been executed, it is possible that either the next one, say `y` is executed, or that the transition has not accomplished, so we have that `done(y,p) v pending(y,p) :- done(x,p)`. If an activity is far from being executed, nothing is deducted. In both cases, whether the activity is pending or discarded, `not done(y,p)` holds. This mechanism allows us to provide a Nòmos model with the true values coming from the process trace. In the table, a certain situation `s1` is considered satisfied (i.e., id *holds*) when the activity `x1` is executed in the process `p`.

4 Discussion and Conclusion

In this position paper, we have proposed an approach to support modeling enterprises in their strategic dimension (goals) as well as operational dimension (business processes), while ensuring compliance to applicable laws. Our approach relies on two existing modeling languages, URN and Nòmos, and extends them to allow linking activities of the business processes to the situations belonging to the legal model. Models are translated into disjunctive logic programs and exhaustively checked by means of an automated reasoning tool. Our approach allows exhaustively generating all possible execution traces on a given process, and contrast them to the linked situations to check for compliance.

While aligning business processes to the goals is a consolidated method in both research and practice, to the best of our knowledge there is a lack in analysis capabilities to what concerns the alignment of the internal structure of the enterprise to the legal environment, where the enterprise operates in. The main drawback of this approach is the need to explicitly model the links between business process and law models. However, supporting this kind of analysis through automated reasoning, is of particular importance when considering large enterprises, with many processes distributed over several legislations.

In our current work in progress, we are planning to apply our approach to a larger example for a more thorough evaluation of our proposal. This will allow us

to evaluate its feasibility in a real-size case study, and to evaluate the scalability of the approach in this larger settings by means of artificial models (see for example [14]). An important limitation that we plan to investigate in our future work is for sure the creations of our models (currently done manually), where we envision a systematic process to help the analyst in the compliance evaluation of the model, as well as in the amendment of such model to reach compliance.

Acknowledgments. This work has been partially funded by AFR - PDR grant #5810263 and by the ERC advanced grant 267856 "Lucretius: Foundations for Software Evolution".

References

1. S. Ghanavati, "Legal-URN framework for legal compliance of business processes," http://hdl.handle.net/10393/24028, 2013, ph.D. thesis, University of Ottawa, Canada.
2. D. A. Sepideh Ghanavati and A. Rifaut, "Legal goal-oriented requirement language (Legal-GRL) for modeling regulations," in *MiSE @ICSE (To be appeared)*, India, 2014.
3. ITU-T, "Recommendation Z.151 (10/12), User Requirements Notation (URN) - Language definition," http://www.itu.int/rec/T-REC-Z.151/en, 2012.
4. A. Shamsaei, "Indicator-based policy compliance of business processes," PhD Thesis, University of Ottawa, Canada, 2012.
5. S. Ghanavati, D. Amyot, and L. Peyton, "A systematic review of goal-oriented requirements management frameworks for business process compliance," in *RELAW*, 2011, pp. 25–34.
6. J. Grabis, M. Kirikova, J. Zdravkovic, and J. Stirna, Eds., *The Practice of Enterprise Modeling - 6th IFIP WG 8.1 Working Conference, PoEM 2013, Riga, Latvia, November 6-7, 2013, Proceedings*, ser. Lecture Notes in Business Information Processing, vol. 165. Springer, 2013.
7. G. Governatori, "Ict support for regulatory compliance of business processes," *CoRR*, vol. abs/1403.6865, 2014.
8. A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini, and J. Mylopoulos, "Capturing variability of law with Nòmos 2," *ER'12*, 2012.
9. M. Weiss and D. Amyot, "Designing and evolving business models with URN," in *MCETECH'05*, 2005, pp. 149–162.
10. D. Amyot and G. Mussbacher, "User Requirements Notation: The first ten years, the next ten years (invited paper)," *Journal of Software (JSW)*, vol. 6, no. 5, pp. 747–768, 2011.
11. D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. S. K. Yu, "Evaluating goal models within the goal-oriented requirement language," *Int. J. Intell. Syst.*, vol. 25, pp. 841–877, August 2010.
12. J. Minker, "Overview of disjunctive logic programming," *Ann. Math. Artif. Intell.*, vol. 12, no. 1-2, pp. 1–24, 1994.
13. M. Alviano et al., "The disjunctive datalog system dlv," in *Datalog*, 2010, pp. 282–301.
14. A. Siena, S. Ingolfo, A. Perini, A. Susi, and J. Mylopoulos, "Automated reasoning for regulatory compliance," in *ER*, ser. Lecture Notes in Computer Science, vol. 8217. Springer, 2013, pp. 47–60.