Verifying Business Process Compliance by Reasoning about Actions *

Davide D'Aprile[1], Laura Giordano[1], Valentina Gliozzi[2], Alberto Martelli[2], Gian Luca Pozzato[2], and Daniele Theseider Dupré[1]

[1] Dipartimento di Informatica, Università del Piemonte Orientale
{davide.daprile,laura.giordano,dtd}@mfn.unipmn.it
[2] Dipartimento di Informatica, Università di Torino
{gliozzi,mrt,pozzato}@di.unito.it

1 Introduction

Verifying the compliance of business processes with norms has become an important issue to be addressed in several domains, such as the administrative and financial domains; consider, for instance, the need for public companies and investment firms to comply with the Sarbanes-Oxley Act in the US, and the MiFID directive in the EU, regulating the financial markets.

In this paper we address the problem of verifying business process compliance with norms. To this end, we employ reasoning about actions in a temporal action theory, which is defined as a combination of Answer Set Programming and Dynamic Linear Time Temporal Logic (DLTL). The temporal action theory allows us to formalize a business process as a temporal domain description, possibly including temporal constraints. Obligations in norms are captured by the notion of commitment, which is borrowed from the social approach to agent communication. Norms are represented using (possibly) non monotonic causal laws which (possibly) enforce new obligations. In this context, verifying compliance amounts to verify that no execution of the business process leaves some commitment unfulfilled. Compliance verification can be performed by Bounded Model Checking. A feature of our approach is that the same declarative formalism is used for both the representation of processes and norms.

2 Example

As a running example we consider a fragment of the business process of an investment firm, where the firm offers financial instruments to an investor. The description of the business process in YAWL is given in Figure 1. We chose YAWL (Yet Another Workflow Language) [8] as a reference specification language for business processes, since it provides a number of advantages with respect to

* This work has been partially supported by Regione Piemonte, Project "ICT4Law - *ICT Converging on Law: Next Generation Services for Citizens, Enterprises, Public Administration and Policymakers*".

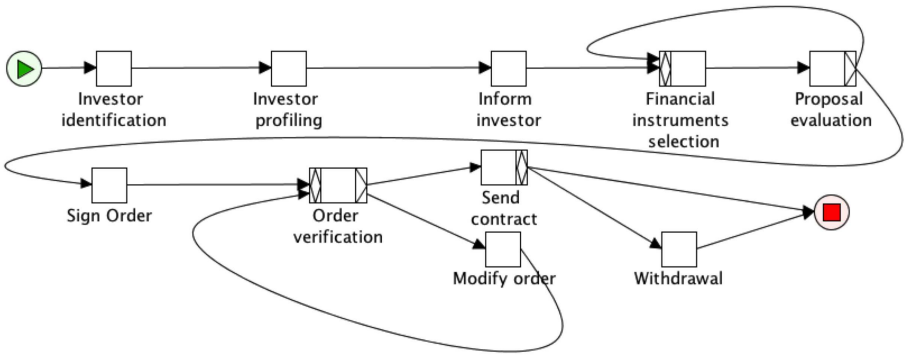

Fig. 1. Example business process in YAWL

several available alternatives: it is an open source workflow system, and it is a result of a deep analysis of business process modeling practice [9]; it comes with a formal foundation, allowing for well-founded formal analysis for achieving validation and verification goals.

Let us consider a regulation containing the following norms:

(1) the firm shall provide to the investor adequate information on its services and policies before any contract is signed;
(2) if the investor signs an order, the firm is obliged to provide him a copy of the contract.

The execution of each task in the process has some preconditions and effects. Due to the presence of norms, the execution of a task in the process above may generate obligations to be fulfilled. For instance, according to the second norm, signing an order generates for the firm the obligation to provide copy of the contract to the investor. Verifying the compliance of a business process to a regulation requires to check that, in all the executions of the business process, the obligations triggered by the norms are fulfilled.

In the following, we sketch the specification of the business process and the related norms in an action theory; and we briefly describe how the problem of verifying compliance of the business process to the norms is defined and solved as a reasoning problem in the action theory.

3 Action theories in Temporal ASP

The action theory comprises a set of laws describing the effects of actions as well as their executability preconditions. Actions may have direct effects on *fluents* (propositions whose truth value describes the state of the world), that are described by *action laws*, and indirect effects, that capture the causal dependencies among fluents and are described by *causal laws*.

Consider the nondeterministic action $order\_verification(T, C)$ in the example: it checks whether the order of the financial product T by customer C is

correct or not. In the first case, the order is accepted, otherwise it is not. The action is modeled in the following action law:

$$\Box([order\_verification(T,C)]confirmed(T,C)\ or\\ [order\_verification(T,C)]\neg confirmed(T,C)$$

where $\Box$ is the modal temporal operator "always" and $[action]effect$ represents an effect of the action, i.e. that *effect* holds in the state right after the action.

In case of deterministic actions, there is a single disjunct in the head of the action law. For instance, the action of informing the investor has the effect that the investor has acquired information: $\Box([inform(C)]informed(C)$

Causal laws are used to represent indirect effects of actions, e.g.,

$$\Box(\neg order\_confirmed(T,C) \leftarrow order\_deleted(T,C))$$

where "confirmed" means "confirmed by the firm" and is a possible effect of order verification, while "deleted" means "withdrawn by the customer", models the fact that the direct effect "deleted" of withdrawal has the indirect effect of making the order no longer effective for the firm as well.

Precondition laws state preconditions for actions, e.g.:

$$\Box([proposal\_evaluation(T,C)]\bot \leftarrow \neg selected(T,C) \vee \neg informed(C))$$

states that an investor can be requested to evaluate a proposed investment only if the proposal has been selected and the investor has been already informed of the firm policy. Similar preconditions can either be asserted in the model, or verified to be true. The second option is suitable for the case where the process explicitly includes, as in figure 1, activities that make the precondition true; the first one is suitable for the case where such activities are abstracted away.

Temporal constraints are arbitrary temporal formulas of DLTL. They are used to restrict the space of the possible extensions. DLTL [6] extends LTL by allowing the until operator $\mathcal{U}^\pi$ to be indexed by a program π, as in Propositional Dynamic Logic (PDL). For instance, $\neg sent\_contract\ \mathcal{U}\ signed$ states that the contract is not sent to the customer until it has been signed. The program π can be any regular expression built from atomic actions by making use of sequence (;), non-deterministic choice (+) and finite iteration ($*$).

For instance, the program

$$\pi = inform(C); select\_financial\_instrument(T,C);\\ ((sign\_order(T,C); send\_contract) + withdraw(T,C))$$

is the program for a simplified version of the process in figure 1: it describes a process in which: the investor C is informed, a financial instrument T is selected for C, then C either signs the contract and a copy of the contract is set to him, or C withdraws.

4 Normative Specification and Compliance Verification

According to the normative specification, the execution of each task in the business process can, in addition to effects in terms of changes in the state of the

world, trigger some normative position (obligation, permission, prohibition). For instance, the *identification* task in the business process in Figure 1, which introduces a new investor C, also generates the obligation to inform the investor. This obligation must be fulfilled during the course of execution of the business process, for the process to be compliant with the norm stating that the firm has the obligation to inform customers.

In order to model obligations, we rely on the notion of commitment introduced in multi-agent systems [7]. As in [3], we introduce two kinds of commitments: *Base-level commitments* having the form $C(i, j, A)$ and meaning that agent i is committed to agent j to bring about A (where A is an arbitrary propositional formula not containing commitment fluents); *Conditional commitments* having the form $CC(i, j, B, A)$ and meaning that agent i is committed to agent j to bring about A, if condition B is brought about.

A commitment $C(i, j, A)$, created at a given state of a run of the process, is regarded to be fulfilled in the run if there is a later state of the run in which A holds. As soon as committment is fullfilled in a run, it is considered to be satisfied and no longer active: it can be discharged.

The norms in Section 2 can be represented by the following precondition and causal law:

$\Box([sign\_order(T, C)]\bot \leftarrow informed(C))$
$\Box(C(firm, C, sent\_contract(T, C)) \leftarrow order\_signed(T, C))$

The first one is a precondition for $sign\_order(T, C)$, and it is quite obviously true in the example process model, because $informed(C)$ is the effect of the action $inform(C)$ which is always executed before $sign\_order(T, C)$ is reached (and there is no action making $informed(C)$ false). Verifying preconditions may be more interesting in more complex processes where the action may be reached via several paths.

The second one, a causal law, states that when an order is signed by C, the firm is committed to C to send her the information required. The commitment remains active until some action is executed, which makes $sent\_contract(T, C)$ true. In the business process, the commitment is fulfilled by the execution of the action $send\_contract(T, C)$.

Causal laws are needed for modeling the interplay of commitments. In particular, for each commitment $C(i, j, \alpha)$, we introduce the following causal laws in the domain description:

(i) $\Box(C(i, j, \alpha)) \wedge \bigcirc\alpha \rightarrow \bigcirc\neg C(i, j, \alpha))$
(ii)$\Box((CC(i, j, \beta, \alpha) \wedge \bigcirc\beta) \rightarrow \bigcirc C(i, j, \alpha))$
(iii)$\Box((CC(i, j, \beta, \alpha) \wedge \bigcirc\beta) \rightarrow \bigcirc\neg CC(i, j, \beta, \alpha))$

A commitment to bring about α is considered fulfilled and is discharged (i) as soon as α holds. A conditional commitment $CC(i, j, \beta, \alpha)$ becomes a base-level commitment $C(i, j, \alpha)$ when β has been brought about (ii) and, in that case, the conditional commitment is discharged (iii).

Even though we do not deal with the issue in detail here, the nonmonotonic nature of ASP allows for a natural representation of exceptions and priorities in norms; it also allows to model reparation chains in [4].

Once the specification of the business process has been given as a domain description in an action theory, the problem of verifying its compliance with some regulation can be modeled as the problem of verifying that all the executions of the business process fulfil the obligations that are generated during the execution of the process.

In our approach, this is done as follows. The notions of *temporal answer set* and *extension* are defined, extending the notion of *answer set* [2].

Extensions represent the possible events and states that occur in an execution of the modeled process. Even for a single action sequence σ, a domain description may have more than one extension, due to different possible initial states (when the initial state is incompletely specified) and different possible effects of nondeterministic actions.

To check if there is an execution which violates some obligation we model the need to fulfil a commitment $C(i, j, \alpha)$ as the temporal formula:

$$\Box(C(i, j, \alpha) \rightarrow \Diamond\alpha)$$

Such formulae, together with the precondition formulae corresponding to norms, is the set of formulae to be verified in order to check compliance. The verification of compliance can be performed by using bounded model checking techniques, which generalize LTL bounded model checking [5].

A feature of our approach is that it provides business process specification, normative specification, and compliance verification in an integrated representation and reasoning framework. We refer to [1] for details as well as for comparison with related work.

References

1. D. D'Aprile, L. Giordano, V. Gliozzi, A. Martelli, G. Pozzato, and D. Theseider Dupré. Reasoning about Actions with Temporal Answer Sets. *Technical Report, Dipartimento di Informatica, Università del Piemonte Orientale*, 2010.
2. M. Gelfond. Answer Sets. *Handbook of Knowledge Representation, chapter 7, Elsevier*, 2007.
3. L. Giordano, A. Martelli, and C. Schwind. Specifying and Verifying Interaction Protocols in a Temporal Action Logic. *Journal of Applied Logic (Special issue on Logic Based Agent Verification)*, 5:214–234, 2007.
4. G. Governatori and S. Sadiq. The journey to business process compliance. *Handbook of Research on BPM, IGI Global*, pages 426–454, 2009.
5. K. Heljanko and I. Niemelä. Bounded LTL model checking with stable models. *Theory and Practice of Logic Programming*, 3(4-5):519–550, 2003.
6. J.G. Henriksen and P.S. Thiagarajan. Dynamic Linear Time Temporal Logic. *Annals of Pure and Applied logic*, 96(1-3):187–207, 1999.
7. N.R. Jennings. Commitments and Conventions: the foundation of coordination in multi-agent systems. *The knowledge engineering review*, 8(3):233–250, 1993.
8. W. van der Aalst and A. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
9. W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14:5–51, 2003.