Compliant business process design by declarative specifications

Francesco Olivieri[1,2,3], Guido Governatori[1,2], Simone Scannapieco[1,2,3], and Matteo Cristani[3]

[1] NICTA, Queensland Research Laboratory, Australia[†]
[2] Institute for Integrated and Intelligent Systems, Griffith University, Australia
[3] Department of Computer Science, University of Verona, Italy

Abstract. We propose algorithms to synthesise the specifications modelling the capabilities of an agent, the environment she acts in, and the governing norms, into a process graph. This process graph corresponds to a collection of courses of action and represents all the licit alternatives the agent may choose to meet her outcomes. The starting point is a compliant situation, i.e., a situation where an agent is capable of reaching all her outcomes without violating the norms. In this case, the resulting process will be *compliant by design*.

Keywords: Rules, agents, and norms; Defeasible Logic; BDI paradigm

1 Introduction

Frodo, the Ring bearer, must leave Rivendell to destroy the Ring of Power in Mount Doom. Since he fears orcs, he would prefer to cross the Misty Mountains by either climbing them, or passing south of them. As a last option, he may pass through Moria. The Fellowship tries to climb the mountains, but fails due to some very unlucky weather conditions. Then, Gandalf tells Frodo not to go south since Saruman would imprison him and take the Ring for himself. Finally, Frodo may not choose to keep the Ring for himself, because he will eventually be subjugated by Sauron's will if he does.

The previous example reveals a very (un)common situation: an agent operates in an environment to reach some objectives. Such an environment is represented by the description of the world, the norms governing it, and the agent's objectives. The agent may have her own desires or intentions, and the combination of such mental attitudes with the factuality of the world defines her deliberative process, i.e., the objectives she decides to pursue. The agent may give up some of them to comply with the norms, if required. Many contexts (as the scenario of Frodo and the Ring) prevent the agent from achieving all her objectives; the agent must then understand which objectives are mutually compatible with each other and choose which objectives to attain the least of in given situations by ranking them in a preference ordering.

[†] NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

In this paper we study the following two problems: (1) determining whether an agent is able to reach her objectives without violating the norms, and (2) if this is possible, identifying what are the (alternative) courses of action to fulfil the objectives.

We can distinguish three phases an agent must pass through to bring about certain states of affairs: understanding the environment she acts in, deploying such information to deliberate which objectives to pursue, and how to act to reach them.

In the first phase, the agent gives a formal declarative description of the environment (in our case, a rule-based formalism). Rules allow the agent to represent relationships between pre-conditions and actions, actions and their effects (post-conditions), relationships among actions, which conditions trigger new obligations to come in force, and in which contexts the agent is allowed to pursue new objectives. In the second stage, the agent combines the formal description with an input describing a particular state of affairs of the environment, and determines which norms are actually in force, which objectives she decides to commit to and to which degree. The agent's decision is based on logical derivations. Since the agent's knowledge is represented by rules, during the last phase, the agent exploits information from derivation to select the activities to carry out in order to achieve the objectives. Here, it is important to notice that a derivation can be understood as a virtual simulation of the various activities involved.

In [1], authors developed a framework based on a modal variant of Defeasible Logic (DL) [2] that models agents by distinguishing three kind of knowledge: the internal constraints guiding the agent (mental attitudes), the external constraints regulating the setting (norms), and how the agent perceives the world (beliefs). Modelling rational agents in this fashion was first proposed by Bratman in [3].

Mental attitudes identified in [1] have strong similarities with those of classical BDI architecture [4–6], while the proposed framework considers desires, goals and intentions as facets of the same phenomenon: the notion of *outcome*, which is simply something an agent would like to achieve. As a result, mental attitudes were modelled through a single type of rule (*outcome rule*) by adopting the methodology of *reparative chains* to model *contrary-to-duties* (CTDs). Reparative chains were first introduced in [7] to formalise reparations of previously violated norms. Given the rule $r : \Gamma \Rightarrow o_1 \otimes o_2 \otimes \cdots \otimes o_n$, if the context described by Γ holds, then the obligation in force is o_1; in case o_1 is violated then the new obligation in force is o_2, and so on.

Using reparative chains to model outcomes reflects the natural way for an agent to give a fixed ordering in objective selection. This is natural in real-life situations when the agent gives alternatives to more preferred objectives. When settled in a given context, an agent combines preferences expressed by outcome rules with her beliefs and the norms to filter out which objectives are feasible for her, and which ones she commits to.

We make a step further in our investigation by addressing two main sequential issues. The former is to adapt the notion of *norm* and *goal compliance* presented in [8] to the (more expressive) logic proposed in [1]. This will result in determining not just whether an agent is capable of reaching a (sub)set of these outcomes without violating the norms, but also her level of commitment to them (i.e., which mental attitude the agent decides to comply with). Accordingly, the agent's mental attitudes are the result of a filtering mechanism through beliefs and norms. This allows us to pass from the (more specific) concept of goal compliance to the (more general) one of *outcome compliance*, which can be varied to suit a particular application.

The latter, and main problem, being to determine the agent's courses of action after both compliances have been established. To this end, we take inspiration from *Business Process Modelling*. A *business process* is a collection of related, structured tasks that produce a specific service, where a *task* is an activity that needs to be accomplished within a defined period of time [9]. A business process can be understood as the set of all its possible execution traces, i.e., all the possible ways in which the process can be executed, where a *trace* is just a linear sequence of tasks/actions (notice that a trace is equivalent to a classical AI plan [10]).

A derivation of a given (outcome) literal corresponds to a specific course of action which reaches that particular outcome, the reason being the information encoded in the agent's knowledge base. Thus, we can consider such a derivation a trace. One of the reasons to use business processes is because it is possible to have multiple ways in which the agent reaches her outcomes and still remains norm compliant. If this is the case, it is pointless for the agent to perform all the alternatives to bring about the same state of affairs. Instead, the agent should be equipped with a *process graph* showing her all such alternatives and allowing her to carry out the best strategy. Choosing the best strategy relies on external information that does not affect the construction of the process graph, such as risks, task cost, execution time, minimum number of task to perform, and so on.

The paper is presented as follows. Section 2 illustrates the logical formalism. In particular, in Subsection 2.3 we discuss norm and outcome compliance. Section 3 shows the algorithms used to compute a process graph starting by the declarative specifications; such processes are compliant by design with the norms and meet the objectives selected by the agent. We summarise and comment some related works in Section 4.

2 Logic

The logic exploited in this work was proposed in [1], where four different types of mental attitudes were identified: desires, goals, intentions, and social intentions.

Desires as acceptable outcomes. Consider an agent equipped with the *outcome rules*

$$r : a_1, \dots, a_n \Rightarrow_{\mathsf{U}} b_1 \odot \cdots \odot b_m \qquad\qquad s : a'_1, \dots, a'_n \Rightarrow_{\mathsf{U}} b'_1 \odot \cdots \odot b'_k$$

where the notation $\Rightarrow_{\mathsf{U}}$ denotes a rule introducing outcomes and that the situation described by $a_1, \dots, a_n$ and $a'_1, \dots, a'_n$ are mutually compatible but b_1 and b'_1 are not, namely $b_1 = \neg b'_1$. In this case $b_1, \dots, b_m, b'_1, \dots, b'_k$ are anyway all *acceptable outcomes*, including b_1 and b'_1. *Desires* are expected or acceptable outcomes, independently of whether they are compatible with other expected or acceptable outcomes.

Goals as preferred outcomes. For rule r alone the preferred outcome is b_1, and for rule s alone it is b'_1. But if both rules are applicable, then the agent would not be rational if she considers both b_1 and $\neg b_1$ as her preferred outcomes. Hence, the agent has to decide if she prefers a state where b_1 holds to one where b'_1 (i.e., $\neg b_1$) holds, or *vice versa*. If the agent has no way to decide which is the most suitable option for her, then neither the chain of r nor that of s can produce preferred outcomes. Suppose that the agent opts for b'_1; this can be done if she establishes that the second rule overrides the first one, i.e., $s > r$. Accordingly, the preferred outcome is b'_1 for the chain of outcomes defined by s, and b_2 is the preferred outcome of r. b_2 is the second best alternative according to rule r: in fact b_1 has been discarded as an acceptable outcome given that s prevails over r.

Two degrees of commitment: intentions and social intentions. We now clarify which are the outcomes for an agent to commit to. Naturally, if the agent values some outcomes more than others, she should strive for the best, i.e., for the most preferred outcomes.

We first consider the case where only rule r applies. Here, the agent should commit to the outcome she values the most, i.e., b_1. But what if the agent *believes* that b_1 cannot be achieved in the environment, or she knows that $\neg b_1$ holds? Committing to b_1 would result in a waste of agent's resources; rationally, she should target the next best outcome, in this case b_2. Accordingly, the agent would derive the *intention* of b_2.

Suppose, now, that b_2 is *forbidden*, and the agent is social (an agent is social if the agent would not knowingly commit to anything that is forbidden [11]). Once again, in this situation the agent has to lower her expectation and settle for b_3 (the agent would obtain the *social intention* of b_3).

To complete the analysis, consider the situation where both rules r and s apply and the agent prefers s to r. As we have seen before, $\neg b_1$ (b'_1) and b_2 are the preferred outcomes since the agent stated $s > r$. Assume that, this time, the agent knows she cannot achieve $\neg b_1$ (or equivalently, b_1 holds). If the agent is rational, she cannot commit to $\neg b_1$. Thus, the best option for her is to commit to b'_2 and b_1, where she is guaranteed to be successful. In this scenario, the best course of action for the agent is where she commits herself to some outcomes that are not her preferred ones, or even that she would consider not acceptable based only on her preferences, but such that they influence her decision process given that they represent relevant external factors (either her beliefs or the norms that apply to her). The agent would have the (social) intentions of b_1 and b'_2.

2.1 Language

Let PROP be a set of propositional atoms, $\text{MOD} = \{\mathsf{B}, \mathsf{O}, \mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$ the set of modal operators, whose reading is B for *belief*, O for *obligation*, D for *desire*, G for goal, I for *intention* and SI for *social intention*. Let Lbl be a set of arbitrary labels. The set $\text{Lit} = \text{PROP} \cup \{\neg p | p \in \text{PROP}\}$ denotes the set of *literals*. The *complementary* of a literal q is denoted by $\sim q$; if q is a positive literal p, then $\sim q$ is $\neg p$, and if q is a negative literal $\neg p$ then $\sim q$ is p. The set of *modal literals* is $\text{ModLit} = \{\Box l, \neg \Box l | l \in \text{Lit}, \Box \in \{\mathsf{O}, \mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}\}$. We assume that the "$\Box$" modal operator for belief B is the empty modal operator, thus a modal literal $\mathsf{B}l$ is equivalent to literal l. Accordingly, the complementary of $\mathsf{B}{\sim}l$ and $\neg \mathsf{B}l$ is ${\sim}l$.

We define a *defeasible theory D* as a structure $(F, R, >)$, where (i) $\text{F} \subseteq \text{Lit} \cup \text{ModLit}$ is a finite set of *facts* or indisputable statements, (ii) R contains three finite sets of *rules*: for beliefs, obligations, and outcomes and (iii) $> \subseteq R \times R$ is a *superiority relation* to determine the relative strength of conflicting rules. *Belief rules* are used to relate the factual knowledge of an agent (her vision of the environment), and defines the relationships between states of the world. As such, provability for beliefs does not generate modal literals. *Obligation rules* determine when and which obligations are in force. The conclusions generated by obligation rules are modalised with obligation. *Outcome rules* establish the possible outcomes of an agent depending on the particular context. Apart from obligation rules, outcome rules derive conclusions for all modes representing possible types of outcomes: desires, goals, intentions, and social intentions.

Following ideas given in [7], rules can gain more expressiveness when a *preference operator* $\odot$ is used: an expression like $a \odot b$ means that if a is possible, then a is the first choice and b is the second one; if $\neg a$ holds, then the first choice is not attainable and b is the actual choice. This operator is used to build chains of preferences, called $\odot$*-expressions*. The formation rules for $\odot$-expressions are: (i.) every literal is an $\odot$-expression, (ii.) if A is an $\odot$-expression and b is a literal then $A \odot b$ is an $\odot$-expression.

In this paper we exploit the classical definition of *defeasible rule* in DL [2]. A defeasible rule is an expression $r : A(r) \Rightarrow_{\Box} C(r)$, where

1. $r \in \mathrm{Lbl}$ is the name of the rule;
2. $A(r) = \{a_1, \ldots, a_n\}$ with $a_i \in \mathrm{Lit} \cup \mathrm{ModLit}$ is the set of the premises (or the *antecedent*) of the rule;
3. $\Box \in \{\mathsf{B}, \mathsf{O}, \mathsf{U}\}$ represents the *mode* of the rule (from now on, we omit the subscript B in rules for beliefs, i.e., $\Rightarrow$ is used as a shortcut for $\Rightarrow_{\mathsf{B}}$);
4. $C(r)$ is the *consequent* (or *head*) of the rule, which is a single literal if $\Box = \mathsf{B}$, or an $\odot$-expression otherwise. It is worth noting that modal literals can occur only in the antecedent of rules: the reason is that the rules are used to derive modal conclusions and we do not conceptually need to iterate modalities The motivation of a single literal as a consequent for belief rules is dictated by the intended reading of the belief rules, where these rules are used to describe the environment.

We use the following abbreviations on sets of rules: $R^{\Box}$ ($R^{\Box}[q]$) denotes all rules of mode $\Box$ (with consequent q), and $R[q] = \bigcup_{\Box \in \{\mathsf{B},\mathsf{O},\mathsf{U}\}} R^{\Box}[q]$. $R[q,i]$ denotes the set of rules whose head is $\odot_{j=1}^{n} c_j$ and $c_i = q$, with $1 \leq i \leq n$.

Notice that labelling the rules of DL produces nothing more but a simple treatment of the modalities, thus two interaction strategies between modal operators are analysed.

Rule conversions and conflict-detection/resolution. It is sometimes meaningful to use rules for a modality $\Box$ as they were for another modality $\blacksquare$, i.e., to convert one type of conclusions into a different one. Formally, given rule $r : a_1, \ldots, a_n \Rightarrow_{\Box} b$ and the situation where $\blacksquare a_1, \ldots, \blacksquare a_n$ hold, the (asymmetric binary relation) $\mathrm{Convert}(\Box, \blacksquare)$ permits to derive $\blacksquare b$. In our framework, we consider $\mathrm{Convert}(\mathsf{B}, \Box)$ with $\Box \in \{\mathsf{O}, \mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$. Accordingly, we enrich the notation with $R^{\mathsf{B},\Box}$, denoting the set of belief rules that can be used for a conversion to mode $\Box$. The antecedent of all such rules is not empty, and does not contain any modal literal.

Moreover, it is crucial to identify criteria for detecting and solving conflicts between different modalities. Formally, we define an asymmetric binary relation $\mathrm{Conflict} \subseteq \mathrm{MOD} \times \mathrm{MOD}$ such that $\mathrm{Conflict}(\Box, \blacksquare)$ means 'modes $\Box$ and $\blacksquare$ are in conflict and mode $\Box$ prevails over $\blacksquare$'. In our framework, we consider: (i.) $\mathrm{Conflict}(\mathsf{B}, \mathsf{I})$, $\mathrm{Conflict}(\mathsf{B}, \mathsf{SI})$ defining the realistic attitude of the agents (cf. [12]), and (ii.) $\mathrm{Conflict}(\mathsf{O}, \mathsf{SI})$ defining the social attitude of the agents (cf. [11]).

Notice that there are two applications of the *superiority relation*: the first considers rules of the same mode; the second when we have two rules of different mode, with complementary literals and two modes are related by the convert relation.

2.2 Inferential Mechanism

A *proof* P of *length* n is a finite sequence $P(1),\dots,P(n)$ of *tagged literals* of the type $+\partial_X q$ and $-\partial_\Box q$, where $\Box \in \text{MOD}$. As a conventional notation, $P(1..i)$ denotes the initial part of the sequence P of length i. Given a defeasible theory D, $+\partial_\Box q$ means that q is defeasibly provable in D with the mode $\Box$, and $-\partial_\Box q$ that it has been proved in D that q is not defeasibly provable in D with the mode $\Box$. From now on, the term *refuted* is a synonym of *not provable* and we use $D \vdash \pm\partial_\Box l$ iff there is a proof P in D such that $P(n) = \pm\partial_\Box l$ for an index n.

To characterise the notions of provability for all modalities, it is essential to define when a rule is *applicable* or *discarded*. To this end, the preliminary notion of when a rule is *body-applicable/discarded* is needed, stating that each literal in the body of the rule must be proved/refuted with the suitable mode.

Definition 1. *Let P be a proof and $\Box \in \{\mathsf{O},\mathsf{D},\mathsf{G},\mathsf{I},\mathsf{SI}\}$. A rule $r \in R$ is* body-applicable *(at step $n+1$) iff for all $a_i \in A(r)$:*

1. *if $a_i = \Box l$ then $+\partial_\Box l \in P(1..n)$,*
2. *if $a_i = \neg\Box l$ then $-\partial_\Box l \in P(1..n)$,*
3. *if $a_i = l \in \text{Lit}$ then $+\partial l \in P(1..n)$.*

Conditions to establish that a rule is *body–discarded* correspond to the constructive failure to prove that the same rule is body–applicable, and follow the principle of *strong negation*. The strong negation principle is related to the function that simplifies a formula by moving all negations to an inner most position in the resulting formula, and replaces the positive tags with the respective negative tags, and the other way around [13].

As already stated, belief rules allow us to derive literals with different modes. The applicability mechanism must take into account this constraint.

Definition 2. *Let P be a proof. A rule $r \in R$ is 1.* Conv-applicable*, 2.* Conv-discarded *(at step $n+1$) for $\Box$ iff*

1. *$r \in R^{\mathsf{B}}$, $A(r) \neq \emptyset$ and for all $a \in A(r)$, $+\partial_\Box a \in P(1..n)$;*
2. *$r \notin R^{\mathsf{B}}$ or $A(r) = \emptyset$ or $\exists a \in A(r)$, $-\partial_\Box a \in P(1..n)$.*

As an example, consider theory $D = (\{a, b, \mathsf{O}c\}, \{r_1 : a \Rightarrow_{\mathsf{O}} b, r_2 : b, c \Rightarrow d\}, \emptyset)$. Rule r_1 is applicable, while r_2 is not since c is not proved as a belief. Instead, r_2 is Conv-applicable for O, since $\mathsf{O}c$ is a fact and r_1 proves $\mathsf{O}b$.

The notion of applicability gives guidelines on how to consider the next element in a given chain. Since a rule for belief cannot generate reparative chains but only single literals, we can conclude that the applicability condition for belief collapses into body-applicability. The same happens to desires, where we also consider the Convert relation. For obligations, each element before the current one must be a violated obligation. A literal is a candidate to be a goal only if none of the previous elements in the chain have been proved as a goal. For intentions, the elements of the chain must pass the wishful thinking filter, while social intentions are also constrained not to violate any norm.

Definition 3. *Given a proof P, $r \in R[q,i]$ is* applicable *(at index i and step $n+1$) for*

1. *B iff $r \in R^{\mathsf{B}}$ and is body-applicable.*

2. O *iff either: (2.1.1)* $r \in R^{\mathsf{O}}$ *and is body-applicable, (2.1.2)* $\forall c_k \in C(r), k < i, +\partial_{\mathsf{O}} c_k \in P(1..n)$ *and* $-\partial c_k \in P(1..n)$*, or (2.2) r is Conv-applicable.*
3. D *iff either: (3.1)* $r \in R^{\mathsf{U}}$ *and is body-applicable, or (3.2) Conv-applicable.*
4. $\Box \in \{\mathsf{G}, \mathsf{I}, \mathsf{SI}\}$ *iff either: (4.1.1)* $r \in R^{\mathsf{U}}$ *and is body-applicable, (4.1.2)* $\forall c_k \in C(r), k < i, +\partial_{\blacksquare} {\sim} c_k \in P(1..n)$ *for some* $\blacksquare$ *such that* $\text{Conflict}(\blacksquare, \Box)$ *and* $-\partial_{\Box} c_k \in P(1..n)$ *or (4.2) r is Conv-applicable.*
   For G *there are no conflicts; for* I *we have* $\text{Conflict}(\mathsf{B}, \mathsf{I})$*, and for* SI *we have* $\text{Conflict}(\mathsf{B}, \mathsf{SI})$ *and* $\text{Conflict}(\mathsf{O}, \mathsf{SI})$.

Again, conditions to establish that a rule is discarded follow the principle of strong negation. We now describe the proof conditions for the various modal operators; we start with those for desires:

```
$+\partial_{\mathsf{D}}$: If $P(n+1) = +\partial_{\mathsf{D}} q$ then
(1) $\mathsf{D}q \in$ F or
(2) (2.1) $\neg\mathsf{D}q \notin$ F and
    (2.2) $\exists r \in R[q, i]$: $r$ is applicable for $\mathsf{D}$ and
    (2.3) $\forall s \in R[{\sim}q, j]$ either (2.3.1) $s$ is discarded for $\mathsf{D}$, or (2.3.2) $s \not> r$.
```

We say that a *desire* is each element in a chain of an outcome rule for which there is no stronger argument for the opposite desire. The proof conditions for $+\partial_{\Box}$, with $\Box \in \{\mathsf{B}, \mathsf{O}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$ are as follows:

```
$+\partial_{\Box}$: If $P(n+1) = +\partial_{\Box} q$ then
(1) $\Box q \in$ F or
(2) (2.1) $\neg\blacksquare q \notin$ F for $\blacksquare = \Box$ or Convert($\blacksquare, \Box$) and
    (2.2) $\exists r \in R[q, i]$: $r$ is applicable for $\Box$ and
    (2.3) $\forall s \in R^{\blacksquare}[{\sim}q, j]$ either
          (2.3.1) $s$ is discarded for $\blacksquare$, or
          (2.3.2) $\exists t \in R^{\Diamond}[q, k]$: $t$ is applicable for $\Diamond$ and either
                  (2.3.2.1) $t > s$ if $\blacksquare = \Diamond$, Convert($\blacksquare, \Diamond$), or Convert($\Diamond, \blacksquare$); or
                  (2.3.2.2) Conflict($\Diamond, \blacksquare$).
```

To show that a literal q is defeasibly provable with modality $\Box$ we have two choices: (1) modal literal $\Box q$ is a fact; or (2) we need to argue using the defeasible part of D. In this case, we require that a complementary literal (of the same modality, or of a conflictual modality) does not appear in the set of facts (2.1), and that there must be an applicable rule for q for mode $\Box$ (2.2). Moreover, each possible attack brought by a rule s for ${\sim}q$ has to be either discarded (3.1), or successfully counterattacked by another stronger rule t for q (2.3.2). We recall that the superiority relation combines rules of the same mode, rules with different modes that produce complementary conclusion of the same mode through conversion (both considered in clause (2.3.2.1)), and conflictual modalities (clause 2.3.2.2). Obviously, if $\Box = \mathsf{B}$, then the proof conditions reduce to those of classical defeasible logic [2].

Again, the negative counterparts ($-\partial_{\mathsf{D}}$ and $-\partial_{\Box}$) are derived by strong negation applied to conditions for $+\partial_{\mathsf{D}}$ and $+\partial_{\Box}$, respectively.

Let us consider the theory $D = (\{\neg b_1, \mathsf{O}\neg b_2, \mathsf{SI} b_4\}, \{r : \Rightarrow_{\mathsf{U}} b_1 \odot b_2 \odot b_3 \odot b_4\}, \emptyset)$. Then r is trivially applicable for D and $+\partial_{\mathsf{D}} b_i$ holds, for $1 \leq i \leq 4$. Moreover, we have $+\partial_{\mathsf{G}} b_1$ and r is discarded for G after b_1. Since $+\partial \neg b_1$ and $\text{Conflict}(\mathsf{B}, \mathsf{I})$, $-\partial_{\mathsf{I}} b_1$ holds (as well as $-\partial_{\mathsf{SI}} b_1$); the rule is applicable for I and b_2, and we are able to prove $+\partial_{\mathsf{I}} b_2$, thus the rule becomes discarded for I after b_2. Given that $\mathsf{O}\neg b_2$ is a fact, r is discarded for SI and b_2 and $-\partial_{\mathsf{SI}} b_2$ is proved, which in turn makes the rule applicable for SI at b_3, proving $+\partial_{\mathsf{SI}} b_3$. As we have argued before, this would make the rule discarded for b_4. Nevertheless, b_4 is still provable with modality SI (in this case because it is a fact, but in other theories there could be more rules with b_4 in their head).

In [1], authors showed coherency and consistency of the logical apparatus.

2.3 Norm and Outcome Compliance

Our logic is able to model in a natural way the concept of being compliant with respect to norms and outcomes. We introduce the literal $\perp$ whose interpretation is an un-compliant situation, and we exploit the modal derivations of $\perp$ to formally characterise norm compliant ($-\partial_{\mathsf{O}}\perp$) and goal compliant ($-\partial_{\Box}\perp, \Box \in \{\mathsf{G}, \mathsf{I}, \mathsf{SI}\}$) situations.

$-\partial_{\mathsf{O}}\perp$: If $P(n+1) = -\partial_{\mathsf{O}}\perp$ then
(1) $\forall r \in R^{\mathsf{O}} \cup R^{\mathsf{B},\mathsf{O}}$ either r is discarded, or either
  (2.1) $\forall c_i \in C(r)$. $-\partial_{\mathsf{O}} c_i$ or
  (2.2) $\exists c_i \in C(r)$ such that $+\partial_{\mathsf{O}} c_i \in P(1..n)$ and $+\partial c_i \in P(1..n)$.

To be norm compliant, all applicable rules producing an obligation are such that either all elements in the consequent are not actually active obligations (condition (2.1)), or one element c_i is an obligation in force which is fulfilled, i.e., we prove it as a belief (condition (2.2)). The situation is slightly different when addressing outcome compliance.

$-\partial_{\Box}\perp$: If $P(n+1) = -\partial_{\Box}\perp$ ($\Box \in \{\mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$) then
(1) $\forall r \in R^{\mathsf{U}} \cup R^{\mathsf{B},\Box}$, Conflict($\blacksquare, \Box$), either r is discarded or
(2) $\exists c_i \in C(r)$ such that $+\partial_{\Box} c_i, \forall c_j \in C(r), j < i, -\partial_{\Box} c_j, \exists c_k, k \geq i\colon +\partial c_k$ and $-\partial_{\Box} {\sim} c_k$.

First, the agent chooses her level of commitment, that is the mode $\Box$ among D, G, I, or SI to comply with. Then, we select the first element proved with mode $\Box$ in the consequent of any applicable rule (e.g., element c_i in rule r). We are outcome compliant with respect to r if a following element c_k has been proved as a belief, and its opposite has not been chosen as an outcome to achieve (in this case r is an *outcome-compliant* rule).

3 Algorithmic Results

We now show how to model a process graph starting from a compliant situation. Accordingly, the process graph resulting by the execution of Algorithm 1 COMPLIANCEBYDESIGN will be compliant by design. Here, we choose intention as the mental attitude the agent has to comply with, thus theory describing the agent capabilities derives $-\partial_{\mathsf{O}}\perp$ as well as $-\partial_{\mathsf{I}}\perp$. The algorithm can be easily modified to treat outcome compliance with respect to the other mental attitudes.

The intuitive idea of the approach is the following. First, for each outcome-compliant rule we select the first element in the corresponding chain proved as an intention (u_i in the algorithm). Elements in such chains are ranked in a preference order and each represents an acceptable outcome for the agent. Thus, we create an X–OR SPLIT/JOIN pattern among u_i and all the elements in the chain following it that are desires with a factual derivation. We propose the exclusive choice pattern among elements since achieving any one of these elements implies the outcome compliance of the entire chain.

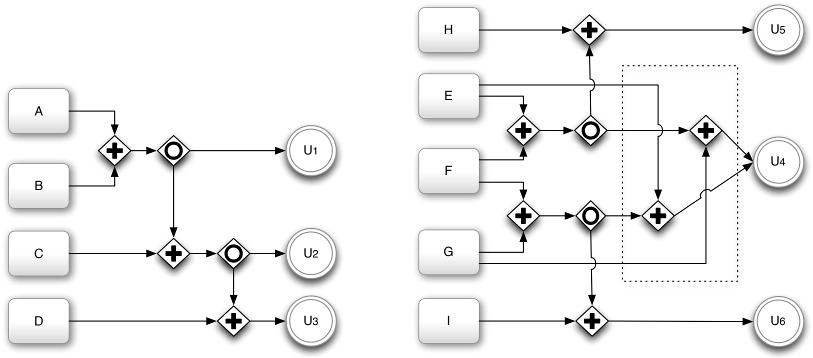
As a second step, we navigate backwards the derivation tree, rule by rule, until the facts of the theory are met. We create a node for every l proved within the theory, and we collect all the rules proving it. All rules deriving such a literal correspond to the different ways in which the agent can bring about the state of affairs described by l. These multiple choices can be naturally represented by an OR JOIN pattern. Moreover, we recall that a rule consists of a conclusion and a set of premises: each premise must be proved individually for the rule to be applicable. Again, this property has a natural counterpart in the AND JOIN pattern among the premises. This reasoning is iterated for each antecedent of each rule for l and is processed by Algorithm 2 BACKTRACK.

Once this backwards procedure ends, we are ready to synthesise the process graph in three subsequent steps.

First, we identify the co-occurrence of literals. We say that a set $S \subseteq (\mathrm{Lit} \cup \mathrm{ModLit})$ *co-occurs* iff whenever one of its element is in the antecedent $A(r)$ of a rule, then all the other elements of S are in $A(r)$ as well. The idea is to represent literals in S as a separate building block able to interact with the other parts of the graph.

Second, based on the idea that a trace is a sequence of only tasks, we remove nodes for conditions and modal literals which become labels of annotated edges.

Third, we recognise SPLIT patterns (Algorithm 3 SPLITPATTERN). This step is needed because literals may occur together in more than one antecedent but not co-occur. One can be tempted to solve the problem by a simple incremental approach, that is by identifying those literals which appear together with more frequency, grouping them and iterating the step. For instance, given the rules $r_1 : a,b \Rightarrow u_1$, $r_2 : a,b,c \Rightarrow u_2$, $r_3 : a,b,c,d \Rightarrow u_3$ the process described above is depicted in Figure 1(a). However, it can be shown that this method fails for certain configurations. For example, if we consider the sets of antecedents $r_4 : e,f,g \Rightarrow u_4$, $r_5 : e,f,h \Rightarrow u_5$, $r_6 : f,g,i \Rightarrow u_6$, the approach would produce the graph reported in Figure 1(b) which is incorrect since it represents two ways to achieve u_4 and not one, namely r_4. Therefore, given a set of antecedents and a pivot literal (resp., sL and l in the algorithms), the proposed solution is to identify a set S such that: (i) the intersection between sets in sL is either l or S, and (ii) S is minimal.



(a) Correct synthesis (b) Wrong synthesis

Fig. 1: Two instances of SPLIT pattern recognition.

At this level of analysis we make no distinction between OR SPLIT and AND SPLIT, and we consider them in the graph simply as OR SPLIT (OR-S).

Algorithm Notation. As already pointed out, we model the collection of possible alternative courses of action as a process graph $G = (V, E)$, namely *ComplianceGraph* in the algorithms, where V is a set of *nodes* and E a set of *edges*. Nodes represent relevant tasks, while (possibly labelled) edges represent logical dependency among literals, based on rules in the initial theory. Each literal $l \in \mathrm{Lit}$ has a unique counterpart in the graph, namely a node labelled with L. This will ease the definition of the formal properties and operations in the algorithms since we can talk about a given node and then refer to the corresponding literal in the theory (and the other way around) without formally bind them every time. For any $\mathrm{X} \in V$, we define $in_V(\mathrm{X}) = \{\mathrm{Y} \in V | (\mathrm{Y}, \mathrm{X}) \in E\}$ and $out_V(\mathrm{X}) = \{\mathrm{Y} \in V | (\mathrm{X}, \mathrm{Y}) \in E\}$.

We are using a semi-formalised language, a pseudocode, that makes use of reserved words. In particular we use the word *procedure* to refer to any kind of subprogram in the code. Moreover, we use the two reserved words **Exit** and **Break** with the following semantics. The first command ends the actual run of the algorithm, while the latter ends the computation of the loop construct.

Here follows a detailed description of the above mentioned algorithms. We begin with Algorithm 1 COMPLIANCEBYDESIGN, the startup of the entire computation.

Algorithm 1 COMPLIANCEBYDESIGN starts by selecting the set of *active* rules (lines 1–3). An *active rule* is an applicable rule which (i) contributes to derive a literal in the positive extension, and (ii) is not defeated by an applicable rule for the opposite. We recall that a modal literal $\Box l$ can be derived in two ways: either directly through a rule $r \in R^{\Box}$, or by exploiting the Convert relation through a belief rule $r \in R^{\mathsf{B},\Box}$.

The **for** cycle at lines 5–23 models the backtrack procedure briefly described above. Line 6 selects the compliant choices for each active outcome rule. If the step picks more than one element (**if** condition at line 7), then we represent them with an X-OR pattern (**then** branch at lines 8–14), otherwise no additional construct is needed (lines 19–20).

Depending on the number of outcomes selected by line 7, Algorithm 2 BACKTRACK is invoked with a different node as second parameter (lines 16 and 21). In both cases, the computation only considers the factual part of the rules leading to the current outcome u_j. Figure 2(a) shows the graph resulting from the execution of Algorithm 2 BACKTRACK with input theory D of Example 1. In the figures we consider only the subprocesses connecting the facts to the outcomes ignoring START and END nodes.

The **for** cycle at lines 24–32 recognises the co-occurrence of nodes. This modification is necessary only when they appear in more than one antecedent (**if** condition at line 27). For each set of co-occurrent literals, we create an AND-J node and OR-S node (line 28), and then we adjust the graph accordingly (line 29).

The algorithm now synthesises the SPLIT patterns (lines 34–38). Line 33 selects only the nodes representing literals; *sL* is the set of antecedents where the current literal l appears in (line 35).

Finally, we remove nodes representing conditions and modal literals by substituting the corresponding node and edges attached to it with an unique labelled edge. This process combined with the synthesis of literal co-occurrence and SPLIT pattern may

Algorithm 1 COMPLIANCEBYDESIGN

```
Require: A modal defeasible theory $D$ and its extension $E(D)$
Ensure: A process graph $G$
 1: $R_{ACT} \leftarrow \{r \in R^{\Box} \cup R^{\mathsf{B},\Box} : C(r) \in +\partial_{\Box}\}$ with $\Box \in \mathrm{MOD} \setminus \{\mathsf{B}\}$
 2: $R_{ACT} \leftarrow R_{ACT} \setminus \{r \in R_{ACT} : \exists \blacksquare a \in A(r)$ such that $a \in -\partial_{\blacksquare}\}$
        ▷ if $r \in R^{\mathsf{B},\Box}$ then $\blacksquare = \Box$ and $\exists a \in A(r)$ such that $a \in -\partial_{\blacksquare}$
 3: $R_{ACT} \leftarrow R_{ACT} \setminus \{r \in R_{ACT} : s > r, s \in R_{ACT}\}$
 4: $ComplianceGraph = (V = \{\textsc{start}, \textsc{end}\}, E = \emptyset)$
 5: for $r \in R^{\mathsf{U}} \cap R_{ACT}$ such that $\exists u_i \in C(r)$ such that $u_i \in +\partial_{\mathsf{I}}$ and $\forall u_j, j < i, u_j \notin +\partial_{\mathsf{I}}$ do
 6:     $goals \leftarrow \{u_j \in C(r) | j \geq i, u_j \in +\partial_{\mathsf{D}}$, and $u_j \in +\partial_{\mathsf{B}}\}$
 7:     if $|goals| \geq 2$ then
 8:         $V \leftarrow V \cup \{\text{XOR–S}_r\} \cup \{\text{XOR–J}_r\}$
 9:         $E \leftarrow E \cup \{(\text{XOR–J}_r, \textsc{end})\}$
10:         for $u_j \in goals$ do
11:             $V \leftarrow V \cup \{\mathrm{U}_j\}$
12:             $E \leftarrow E \cup \{(\text{XOR–S}_r, \mathrm{U}_j)\}$
13:             $E \leftarrow E \cup \{(\mathrm{U}_j, \text{XOR–J}_r)\}$
14:         end for
15:         for $u_j \in goals$ do
16:             BACKTRACK($u_j$, XOR–S$_r$, $\{r \in R_{ACT} \cap R^{\mathsf{B}} | r$ proves $u_i\}$)
17:         end for
18:     else
19:         $V \leftarrow V \cup \{\mathrm{U}_i\}$
20:         $E \leftarrow E \cup \{(\mathrm{U}_i, \textsc{end})\}$
21:         BACKTRACK($u_i$, U$_i$, $\{r \in R_{ACT} \cap R^{\mathsf{B}} | r$ proves $u_i\}$)
22:     end if
23: end for
24: $V' \leftarrow V$
25: for L $\in V'$ do                                                     ▷ Co-occurrence
26:     $cO \leftarrow \{\mathrm{M} \in V' | out_V(\mathrm{L}) = out_V(\mathrm{M})$ and $m \neq l\}$
27:     if $|out_V(\mathrm{L})| \geq 2$ and $|cO| \geq 2$ then
28:         $V \leftarrow V \cup \{\text{AND-J}_{cO}\} \cup \{\text{OR-S}_{cO}\}$
29:         $E \leftarrow E \cup \{(\mathrm{M}, \text{AND-J}_{cO}) | \mathrm{M} \in cO)\} \cup \{(\text{AND-J}_{cO}, \text{OR-S}_{cO})\} \cup \{(\text{OR-S}_{cO}, \mathrm{N}) | \mathrm{N} \in out_V(\mathrm{L})\}$
                $\setminus \{(\mathrm{M}, \mathrm{N}) | \mathrm{M} \in cO$ and $\mathrm{N} \in out_V(\mathrm{L})\}$
30:     end if
31:     $V' \leftarrow V' \setminus cO$
32: end for
33: Let $Vlist$ be the list of nodes representing literals
34: for L $\in Vlist$ do                                                  ▷ SPLIT pattern
35:     $sL \leftarrow \{A(r) | r \in R_{ACT}$ and $l \in A(r)\}$
36:     SPLITPATTERN($l, sL, null$)
37:     Remove L from $Vlist$
38: end for
39: while $\exists \mathrm{L} \in V$ such that: i. L is an AND-J node and $|in_V(\mathrm{L})| = 1$, ii. $l$ is a condition, or iii. $l = \Box m$, with
    $\Box \in \mathrm{MOD} \setminus \{\mathsf{B}\}$ do                                  ▷ Contraction
40:     $E \leftarrow E \cup \{e = (\mathrm{X}, \mathrm{Y}) | (\mathrm{X}, \mathrm{L}) \in E$ and $(\mathrm{L}, \mathrm{Y}) \in E\} \setminus \{(\mathrm{X}, \mathrm{Y}) |$ either $\mathrm{X} = \mathrm{L}$, or $\mathrm{Y} = \mathrm{L}\}$
41:     $label(e) \leftarrow \{l\} \cup label((X,Y))$ with $\mathrm{X} = \mathrm{L}$ or $\mathrm{Y} = \mathrm{L}$
42:     $V \leftarrow V \setminus \{\mathrm{L}\}$
43: end while
44: return $G$
```

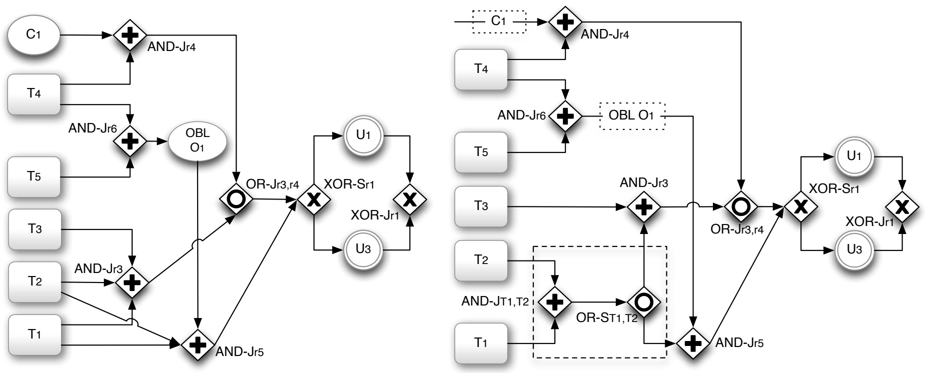
have created AND–J nodes with only one incoming edge, that have to be erased. The **while** cycle at lines 39–43 implements these two operations.

Figure 2(b) shows the final process graph for theory D presented in Example 1 (notice that here Algorithm 3 SPLITPATTERN keeps the graph unchanged). Figure 3 shows the execution of Algorithm 1 COMPLIANCEBYDESIGN on theory D' of Example 2, where Algorithm 3 SPLITPATTERN effectively modifies the graph.

Example 1. Let $D = (\{\Gamma, \Delta, t_1, \ldots, t_5, c_1\}, R, \emptyset)$ be a theory such that

$$\begin{aligned} R = \{ r_1 &: \Gamma \Rightarrow_{\mathsf{U}} u_1 \odot u_2 \odot u_3 & r_2 &: \Delta \Rightarrow_{\mathsf{U}} \neg u_2 \\ r_3 &: t_1, t_2, t_3 \Rightarrow u_1 & r_4 &: t_4, c_1 \Rightarrow u_1 \\ r_5 &: t_1, t_2, \mathsf{O} o_1 \Rightarrow u_3 & r_6 &: t_4, t_5 \Rightarrow_{\mathsf{O}} o_1 \}. \end{aligned}$$

Labels of the edges for c_1 and $\mathsf{O}o_1$ are highlighted by *dotted* squares; the co-occurrence of tasks T_1 and T_2 is highlighted by a *dashed* square.



(a) Run of the algorithm up to line 23 (b) Labelled edges and co-occurrence

Fig. 2: Process graph of theory D resulting from Algorithm 1 COMPLIANCEBYDESIGN

Algorithm 2 BACKTRACK performs the backwards construction of the process graph. It receives as input the last processed literal, the set of active rules to be considered (R^{bT}), and the node (N) that will be attached to the edges introduced by the current run of the algorithm. Such a node is either the literal l itself, or the XOR–S whenever l is in the chain of an outcome rule.

The algorithm performs different operations depending on: (i) how many rules prove l, and (ii) the number of elements in the antecedent. If l is a fact (lines 3–6), then we link N to START. Otherwise, we create a node for each element in the antecedent not already in V. If the antecedent contains more than one element, we add an AND–J node between the nodes representing the antecedent and L (**if** condition at line 13, **else** at line 31). Furthermore, if l is derived by more than one rule, the algorithm adds an OR–S node and links it to the suitable nodes (**if** condition at line 23). The algorithm invokes itself on every antecedent of every rule found (lines 10, 19, 30, and 37).

Algorithm 3 SPLITPATTERN takes three input parameters. The third one distinguishes if the algorithm was invoked either (1) by the **while** loop at lines 2–25, or (2.1) by the **while** loop at lines 26–32 or (2.2) by Algorithm 1 COMPLIANCEBYDESIGN at line 36; moreover, the AND-J node created at line 5 will be linked to it.

First, the set S is computed based on the constraints previously discussed (line 2). Notice that condition $a)$ prevents S to contain only l when the algorithm is invoked at line 28 or by Algorithm 1 COMPLIANCEBYDESIGN. The algorithm links each element in S with a new AND–J node, unique for each distinct S (lines 10–12), which is in turn

Algorithm 2 BACKTRACK

```
1:  procedure BACKTRACK(literal $l$, node N, set of rules $R^{bT}$)
2:      $R_{ACT} \leftarrow R_{ACT} \setminus R^{bT}$
3:      if $l \in$ F then
4:          $E \leftarrow E \cup \{(\text{START}, \text{N})\}$
5:          Exit
6:      end if
7:      if $R^{bT} = \{r\}$ and $A(r) = \{a\}$ then
8:          $V \leftarrow V \cup \{\text{A}\}$
9:          $E \leftarrow E \cup \{(\text{A}, \text{N})\}$
10:         BACKTRACK($a$, A, $\{r \in R_{ACT} \cap (R^{\Box} \cup R^{\mathsf{B},\Box}) | a \in C(r)\}$)
                    ▷ $\Box = \mathsf{O}$ if $a = \mathsf{O}b$, $\Box = \mathsf{U}$ if $a = \blacksquare b$ with $\blacksquare = \{\mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$
11:         Exit
12:     end if
13:     if $R^{bT} = \{r\}$ and $|A(r)| > 1$ then
14:         $V \leftarrow V \cup \{\text{AND–J}_r\}$
15:         $E \leftarrow E \cup \{(\text{AND–J}_r, \text{N})\}$
16:         for $a \in A(r)$ do
17:             $V \leftarrow V \cup \{\text{A}\}$
18:             $E \leftarrow E \cup \{(\text{A}, \text{AND–J}_r)\}$
19:             BACKTRACK($a$, A, $\{r \in R_{ACT} \cap (R^{\Box} \cup R^{\mathsf{B},\Box}) | r$ proves $a\}$)
20:         end for
21:         Exit
22:     end if
23:     if $|R^{bT}| > 1$ then
24:         $V \leftarrow V \cup \{\text{OR–J}_N\}$
25:         $E \leftarrow E \cup \{(\text{OR–J}_N, \text{N})\}$
26:         for $r \in R^{bT}$ do
27:             if $A(r) = \{a\}$ then
28:                 $V \leftarrow V \cup \{\text{A}\}$
29:                 $E \leftarrow E \cup \{(\text{A}, \text{OR–J}_N)\}$
30:                 BACKTRACK($a$, A, $\{r \in R_{ACT} \cap (R^{\Box} \cup R^{\mathsf{B},\Box}) | r$ proves $a\}$)
31:             else
32:                 $V \leftarrow V \cup \{\text{AND–J}_r\}$
33:                 $E \leftarrow E \cup \{(\text{AND–J}_r, \text{OR–J}_N)\}$
34:                 for $a \in A(r)$ do
35:                     $V \leftarrow V \cup \{\text{A}\}$
36:                     $E \leftarrow E \cup \{(\text{A}, \text{AND–J}_r)\}$
37:                     BACKTRACK($a$, A, $\{r \in R_{ACT} \cap (R^{\Box} \cup R^{\mathsf{B},\Box}) | r$ proves $a\}$)
38:                 end for
39:             end if
40:         end for
41:         Exit
42:     end if
43: end procedure
```

attached to an OR–S node (line 6). This operation removes only the edges (N, X), with $n \in S$, such that L is linked to X (i.e., $(\text{L}, \text{X}) \in E$).

Sets in S can be grouped based on whether they contain another subset $S' \neq S$ joining properties in line 2, or not (resp., *sLrest* in line 18 and *onlyS* in line 14). Thus, for the sets of antecedents in *onlyS*, we just link the OR–S$_S$ to the AND–J$_r$ which was added by Algorithm 2 BACKTRACK (**for** loop at lines 15–17). If *sLrest* is not empty, we choose a new pivot element (m in line 22) and we invoke Algorithm 3 SPLITPATTERN (line 23).

Once the **while** loop at lines 2–25 ends, the remaining sets in sL suffer the issue shown in Figure 1(b). This means that no set S satisfying condition c) of line 2 exists. The **while** loop at lines 26–32 solves this problem. By selecting a set of antecedents

Algorithm 3 SPLITPATTERN

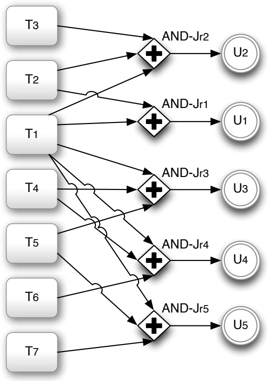
```
1:  procedure SPLITPATTERN(literal $l$, set $sL$, node OR-NODE)
2:      while $\exists S \subseteq \bigcup_{A(r)\in sL} A(r)$ such that
            $a)$ if OR-NODE $= null$, then $S \neq \{l\}$
            $b)$ $\exists T, U \in sL$ such that $T \neq U$ and $S \subseteq T, S \subseteq U$
            $c)$ $\forall X, Y \in sL$. either $X \cap Y = \{l\}$, or $S \subseteq X \cap Y$
            $d)$ $\forall X \in \bigcup_{M \in sL}$ if $X \subseteq S$, then $X = S$ do
3:          $supp_{sL} \leftarrow \{A(r) \in sL | S \subseteq A(r)\}$
4:          $sL \leftarrow sL \setminus supp_{sL}$
5:          $V \leftarrow V \cup \{\text{AND-J}_S\} \cup \{\text{OR-S}_S\}$
6:          $E \leftarrow E \cup \{(\text{AND-J}_S, \text{OR-S}_S)\}$
7:          if OR-NODE $\neq null$ then
8:              $E \leftarrow E \cup \{(\text{OR-NODE}, \text{AND-J}_S)\}$
9:          end if
10:         for $n \in S$ do
11:             $E \leftarrow E \setminus \{(\mathrm{N}, \mathrm{X}) | \mathrm{X} \in out_V(\mathrm{L}) \cap out_V(\mathrm{N})\} \cup \{(\mathrm{N}, \text{AND-J}_S)\}$
12:         end for
13:         $supp_{sL} \leftarrow \{A(r) \setminus S | A(r) \in supp_{sL}\}$
14:         $onlyS \leftarrow \{A(r) \in supp_{sL} | \left(\bigcup_{A(t)\in supp_{sL}, r\neq t} A(t)\right) \cap A(r) = \emptyset\}$
15:         for $A(r) \in onlyS$ do
16:             $E \leftarrow E \cup \{(\text{OR-S}_S, \text{AND-J}_r)\}$
17:         end for
18:         $sLrest \leftarrow supp_{sL} \setminus onlyS$
19:         if $sLrest = \emptyset$ then
20:             Break
21:         else
22:             choose $m \in \bigcap_{A(r)\in sLrest} A(r)$
23:             SPLITPATTERN($m, sLrest, \text{OR-S}_S$)
24:         end if
25:     end while
26:     while $\exists W \subseteq \bigcup_{A(r)\in sL} A(r)$ such that satisfies $b)$ and $d)$ do
27:         $supp_W \leftarrow \{A(r) \in sL | W \subseteq A(r)\}$
28:         SPLITPATTERN($l, supp_W, null$)
29:         $sL \leftarrow sL \setminus supp_W$
30:         $intersec_W \leftarrow \left(\bigcup_{A(r)\in supp_W} A(r) \cap \bigcup_{A(t)\in sL} A(t)\right) \setminus \{l\}$
31:         $sL \leftarrow \{A(r) \setminus intersec_W | A(r) \in sL\}$
32:     end while
33: end procedure
```

satisfying conditions $b)$ and $d)$ ($supp_W$ in line 28), we exclude those sets for which condition $c)$ would fail ($A(r_3)$, $A(r_4)$, and $A(r_5)$ in Example 2). Accordingly, if we now run Algorithm 3 SPLITPATTERN on $supp_W$, a new S will be found.

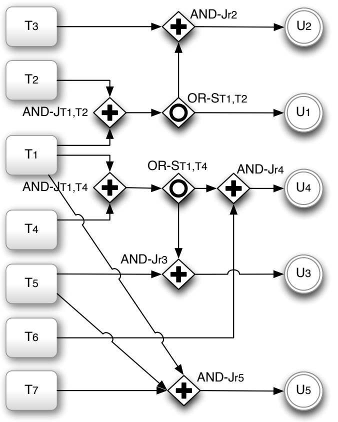
Example 2. Let $D' = (\{t_1, \ldots, t_7, \lnot u_1, \ldots, \lnot u_5\}, R, \emptyset)$ be a theory such that

$$\begin{aligned} R = \{ r_1 &: t_1, t_2 \Rightarrow u_1 & r_2 &: t_1, t_2, t_3 \Rightarrow u_2 \\ r_3 &: t_1, t_4, t_5 \Rightarrow u_3 & r_4 &: t_1, t_4, t_6 \Rightarrow u_4 \\ r_5 &: t_1, t_5, t_7 \Rightarrow u_5 \}. \end{aligned}$$

S is initially $\{t_1, t_2\}$, thus the algorithm creates AND–$\mathrm{J}_{\mathrm{T}_1,\mathrm{T}_2}$ and OR–$\mathrm{S}_{\mathrm{T}_1,\mathrm{T}_2}$ at lines 26–32. Then, the algorithm enters the **while** guard at line 26 with $sL = \{A(r_3), A(r_4), A(r_5)\}$. There are two Ws which satisfy conditions $b)$ and $d)$, that is $\{t_1, t_4\}$ and $\{t_1, t_5\}$. In this example, the algorithm chooses the former set and invokes itself by passing $l = t_1$ and $supp_W = \{A(r_3), A(r_4)\}$.



(a) Run of the algorithm up to line 33



(b) Final process graph

Fig. 3: Process graph of theory D' resulting from Algorithm 1 COMPLIANCEBYDESIGN

Computational results. For space reason, we only present a sketch of proof for termination and soundness of the algorithms. Termination of Algorithm 1 COMPLIANCEBYDESIGN depends on termination of its sub-routines, being the set of literals and rules of theory finite. Algorithm 2 BACKTRACK terminates since: (1) literals and rules of the theory are finite, and (2) each rule is considered exactly once. Algorithm 3 SPLITPATTERN terminates since: (1) set *sL* is finite and this implies that the set possible candidates S is finite as well; (2) rules of the theory in *sL* are considered once, (3) each operation from line 3 to 18 operates on finite sets, (4) recursive invocation at line 23 takes as input set *sLrest* which is a subset of *sL* and contains only (antecedents of) rules not yet handled. The same reasoning applies for operations at lines 26–32.

The soundness of the algorithms is ensured if (1) all the traces generated by the algorithms are compliant and (2) the operations preserve the structure of the theory, i.e. if rule r is $a \Rightarrow b$, in the graph node B does not occur before node A. (1) is guaranteed since R_{ACT} contains only (not defeated) rules with both antecedents and conclusion(s) in the positive extensions. (2) is ensured by Algorithm 2 BACKTRACK which, given a rule $r : a_1, \dots, a_n \Rightarrow c$, establishes an equivalence between (i) the causality among $a_1, \dots, a_n$ and c, and (ii) the sequentiality of nodes representing $a_1, \dots, a_n$ with respect to c. Algorithm 3 preserves these properties. Specifically, causality is guaranteed by conditions $b)$ and $c)$ being T, U, X and Y subsets of *sL*, and since the AND-J$_S$ and OR-S$_S$ gates generated at line 5 are always placed between the elements in S and the nodes representing conclusions of rules where such elements appear as antecedents.

4 Conclusions and Related Work

The contribution of this paper is twofold: (1) we extended the definition of compliance provided in [8] to accommodate the notion of outcome, and, as the main contribution of the paper, (2) we presented algorithms to generate outcome and norm compliant business processes starting from the description of the environment, norms and the agent's capabilities.

The approach we have presented departs from the standard BDI architecture and agent programming languages implementing it (e.g., 2APL [14], Jason [15]), and extensions with norms (e.g., BOID [12], and see [16] for an overview) in several respects. First, the use of reparation chains allows us to use the same deductive process to compute what are the norms in force, the outcomes the agent subscribe to, and whether agents are outcome and norm compliant. Second, while in the above mentioned approaches the agent has to select (partially) predefined plans from a plan library, we propose that the agent generates on the fly a business process (corresponding to a set of plans) to meet the objectives without violating the norms she is subject to.

There are agent frameworks where agents generate plans (e.g., KPG [17] and Golog [18]), but these are typically based on classical AI planning and they do not consider norms and their interactions with other mental attitudes.

As future research, we will prove that our algorithms work in polynomial time.

References

1. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S., Cristani, M.: Picking up the best goal: An analytical study in defeasible logic. In Paschke, A., Morgenstern, L., Stefaneas, P., eds.: RuleML. Volume 8035 of LNCS., Springer (2013)
2. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic **2**(2) (2001) 255–287
3. Bratman, M.E.: Intentions, Plans and Practical Reason. Harvard University Press (1987)
4. Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence **42**(2-3) (1990) 213–261
5. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In Allen, J.F., Fikes, R., Sandewall, E., eds.: KR, Kaufmann, M. (1991) 473–484
6. Rao, A.S., Georgeff, M.P.: Decision procedures for bdi logics. JLC **8**(3) (1998) 293–342
7. Governatori, G., Rotolo, A.: Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. Australasian Journal of Logic **4** (2006) 193–215
8. Governatori, G., Olivieri, F., Scannapieco, S., Cristani, M.: Designing for compliance: Norms and goals. In Olken, F., Palmirani, M., Sottara, D., eds.: RuleML America. Volume 7018 of LNCS., Springer (2011) 282–297
9. Davenport, T.H.: Process innovation : reengineering work through information technology. Harvard Business School Press (1993)
10. Ghallab, M., Nau, D., Traverso, P.: Automated planning - theory and practice. Elsevier (2004)
11. Governatori, G., Rotolo, A.: BIO logical agents: Norms, beliefs, intentions in defeasible logic. Journal of Autonomous Agents and Multi-Agent Systems **17**(1) (2008) 36–69
12. Broersen, J., Dastani, M., Hulstijn, J., van der Torre, L.: Goal generation in the BOID architecture. Cognitive Science Quarterly **2**(3-4) (2002) 428–447
13. Antoniou, G., Billington, D., Governatori, G., Maher, M.J., Rock, A.: A family of defeasible reasoning logics and its implementation. In: ECAI 2000. (2000) 459–463
14. Dastani, M.: 2APL: A practical agent programming language. AAMAS **16**(3) (2008) 214–248
15. Bordini, R.H., Wooldridge, M., Hubner, J.: Programming multi-agent systems in agentspeak using Jason. John Wiley & Sons (2007)
16. Alechina, N., Bassiliades, N., Dastani, M., De Vos, M. Logan, B., Mera, S., Morris-Martin, A., Schapachnik, F.: Computational models for normative multi-agent systems. In: Normative Multi-Agent Systems. (2013) 71–92
17. Kakas, A., Mancarella, P., Sadri, F., Stathis, K., Toni, F.: The kgp model of agency. In: ECAI. (2004) 33–37
18. Gabaldon, A.: Making golog norm compliant. In: CLIMA XII, Springer (2011) 275–292