Dyadic Obligations over Complex Actions as Deontic Constraints in the Situation Calculus

Jens Claßen , James Delgrande
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
jens_classen@sfu.ca, jim@cs.sfu.ca

Abstract

With the advent of artificial agents in everyday life, it is important that these agents are guided by social norms and moral guidelines. Notions of obligation, permission, and the like have traditionally been studied in the field of Deontic Logic, where deontic assertions generally refer to what an agent should or should not do; that is they refer to *actions*. In Artificial Intelligence, the Situation Calculus is (arguably) the best known and most studied formalism for reasoning about action and change. In this paper, we integrate these two areas by incorporating deontic notions into Situation Calculus theories. We do this by considering deontic assertions as constraints, expressed as a set of conditionals, which apply to complex actions expressed as GOLOG programs. These constraints induce a ranking of "ideality" over possible future situations. This ranking in turn is used to guide an agent in its planning deliberation, towards a course of action that adheres best to the deontic constraints. Introducing a representation for action aspects and new GOLOG constructs for joint and negated actions, we present a formalization that includes a wide class of (dyadic) deontic assertions, lets us distinguish prima facie from all-things-considered obligations, and particularly addresses contrary-to-duty scenarios. We furthermore present results on compiling the deontic constraints directly into the Situation Calculus action theory, so as to obtain an agent that respects the given norms, but works solely based on standard reasoning and planning techniques.

1 Introduction

Artificial agents are playing an ever-greater role in our everyday lives. When they interact with humans or operate in shared environments, it becomes increasingly important that they are capable of subjecting their actions to social norms and moral guidelines. Here, notions of obligation, permission, prohibition and the like come into play, which traditionally have been the subject of study in the field of Deontic Logic (von Wright 1951; Gabbay et al. 2013). The most cited and most studied system of deontic logic is *Standard Deontic Logic* (SDL), a variant of the modal logic KD (Chellas 1980), where a modal operator $\mathbf{O}\phi$ is used to express that "ϕ is obligatory" or "it ought to be that ϕ", permission is defined as the dual of obligation ($\mathbf{P}\phi = \neg\mathbf{O}\neg\phi$), and prohibition as the negation of permission ($\mathbf{F}\phi = \neg\mathbf{P}\phi$).

While SDL is simple and elegant, it is also somewhat weak and bears unintuitive consequences, traditionally called "paradoxes". In particular, it fails to handle *contrary-to-duty* obligations (Chisholm 1963), which usually take the form of conditional exhortations, and state what ought to be done if one neglects a certain other duty. Albeit somewhat brutal[1], let us consider a popular example from the literature, the "paradox of the gentle murder" (Forrester 1984):

1. Smith should not murder Jones.
2. If Smith murders Jones, he should do so gently.
3. Smith murders Jones.

This scenario is considered paradoxical because an encoding in SDL entails that Smith should murder Jones, thus leading to a contradiction with the first statement. Another well-known scenario, originally suggested by Chisholm, can be phrased as follows:

1. You ought to help your neighbour.
2. If you help your neighbour you should tell them.
3. If you don't help your neighbour, you shouldn't tell them.
4. You don't help your neighbour.

These statements intuitively appear consistent and independent from one another, yet different encodings in SDL either lead to an inconsistency or one of the statements being derivable from the others. It was soon recognized (Hansson 1969; Goble 2003) that representing such statements by means of monadic deontic modalities and material implications is insufficient, and that it rather requires *dyadic* obligations like $\mathbf{O}(tell/help)$ to express systems of *defeasible* conditionals. Thus, dyadic deontic modalities do not merely distinguish ideal from non-ideal worlds in a binary fashion, but order possible worlds according to some preference relation, allowing for differing "degrees of ideality".

When it comes to reasoning about an agent's actions and the change they bring about, the Situation Calculus (McCarthy and Hayes 1969; Reiter 2001) is (arguably) the best known and most studied formalism in Artificial Intelligence.

[1] It is straightforward to conceive of similarly structured, yet harmless examples that are more appropriate in the context of artificial agents. For example, we may want a household robot be subject to the constraint that it ought not touch a valuable antique vase, but if it has to move the vase, it ought to do so gently. Nevertheless, we will use Forrester's example in this paper due to its popularity in the literature.

As a dialect of standard first-order logic, it offers a great deal of expressivity for formulating action theories that define the preconditions and effects of an agent's primitive actions. These in turn can be composed into more complex behaviours by means of the agent programming language GOLOG (Levesque et al. 1997), which has been found especially useful for the control of mobile robots (Burgard et al. 1999; Ferrein and Lakemeyer 2008). Consequently, the integration of deontic concepts into the Situation Calculus has been investigated previously (Demolombe and del Pilar Pozos Parra 2005; Demolombe and del Pilar Pozos Parra 2009), but this line of research followed formalizations based on SDL very closely. In particular, deontic modalities were applied to state properties ("ought-to-be"), but additionally could *change* due to actions. For example, a vehicle is permitted to be in an intersection if the traffic light is green; turning the light red then removes this permission.

In this paper, we take a different approach and assume that the actions themselves are subject to deontic constraints ("ought-to-do"). Inspired by the aforementioned classical approaches to "contrary-to-duty" scenarios, we propose to employ *dyadic* obligations in the form of defeasible conditionals, with the difference that instead of propositions, they will apply to complex actions from a fragment of the GOLOG language. A rule like $\delta \Rightarrow \gamma$ then is to be understood as a *deontic constraint* saying that in case the agent is committed to a course of action according to δ (e.g., murdering Jones), it ought to follow a course of action according to γ (e.g., murdering Jones gently). Instead of possible worlds, such conditionals rank the *reachable situations* by their deontic "ideality", and so when deliberating about what to do next, such ranks will guide the agent towards a course of action adhering best to the given deontic constraints. Our formalization includes a representation for *action aspects* such as "gently" in the Situation Calculus, and extends GOLOG by constructs for joint and negated actions. We also present a method for compiling deontic constraints into standard Situation Calculus action theories, thus making them available to existing Situation Calculus/GOLOG systems, without the need for additional reasoning machinery.

The rest of this paper is organized as follows. Section 2 presents the formal preliminaries for the Situation Calculus and GOLOG, which we extend in Section 3 by the concepts of action aspects, joint actions, and action negation. In Section 4 we present our main framework, where we first distinguish prima facie from all-things-considered obligations, and afterwards discuss two systems of dyadic obligations over complex actions: Non-temporal constraints talk about alternatives for single actions (e.g., the Forrester scenario), whereas temporal constraints refer to subsequent or previous actions (e.g., the Chisholm scenario). Section 5 shows an approach for compiling constraints into action theories, after which we review related work and conclude.

2 Preliminaries

2.1 The Situation Calculus

The Situation Calculus (McCarthy and Hayes 1969; Reiter 2001) is a dialect of first-order logic, with some second-order features, for reasoning about action and change. The universe of discourse is assumed to consist of the three sorts *actions*, *situations*, and *objects* (everything else). Situations describe possible sequences of actions, where a term $do(\alpha, \sigma)$ denotes the situation resulting from applying action α in situation σ, and S_0 is a constant that stands for the initial situation. For example, the term $do(eat(pizza), do(put(napkin), S_0))$ means the situation reached by first putting a napkin on one's lap, and then eating a pizza. As notional convention, we will use σ (possibly with decorations) to denote *terms* of sort situation, and s (possibly with decorations) as situation *variables* in formulas. Similarly, α and β will refer to action *terms*, whereas a is used as an action variable. For notational convenience we further extend do to also work on *sequences* $\vec{\alpha}$ of actions:

$$do(\langle\rangle, \sigma) \doteq \sigma, \quad do(\vec{\alpha} \cdot \beta, \sigma) \doteq do(\beta, do(\vec{\alpha}, \sigma))$$

We use $\langle \ldots \rangle$ for sequence literals, so $\langle\rangle$ stands for the empty sequence, and the above example situation could be written as $do(\langle put(napkin), eat(pizza)\rangle, S_0)$.

Changing properties are represented by means of *fluents*, which are functions and predicates that take a situation as their last argument. For example, $Hungry(x, s)$ may express that person x is hungry in situation s, and so $\neg Hungry(jones, do(eat(pizza), S_0))$ says that Jones is not hungry anymore after eating pizza. A formula is called *uniform in* σ if the only situation term it mentions is σ, and σ only occurs in the situation argument of fluents (so quantification or equalities over situations are ruled out). For example, $Hungry(jones, s)$ is uniform in s.

Following (Reiter 2001), we use a basic action theory (BAT) $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_{post} \cup \mathcal{D}_{fnd}$ to encode a dynamic domain. The *initial theory* $\mathcal{D}_0$ is a description of the initial situation consisting of formulas uniform in S_0. $\mathcal{D}_{post}$ is a set of *successor state axioms* (SSA) of the form $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$, one for each relational fluent F, where $\Phi_F(\vec{x}, a, s)$ is a formula uniform in situation s, and similar for functional fluents. (As convention, free variables are understood as being universally quantified from the outside.) For the sake of simplicity, we ignore preconditions completely. Finally, a BAT contains *foundational axioms* $\mathcal{D}_{fnd}$, which include unique names axioms for actions, a unique names axiom

$$do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2 \qquad (1)$$

for situations, as well as

$$\neg s \sqsubset S_0 \quad \text{and} \quad s \sqsubset do(a, s') \equiv s \sqsubseteq s', \qquad (2)$$

that define "$\sqsubset$" as the subhistory relation among situations, with $s \sqsubseteq s'$ abbreviating $s \sqsubset s' \vee s = s'$.

2.2 GOLOG

The agent programming language GOLOG (Levesque et al. 1997) is defined on top of the Situation Calculus, allowing for composing more complex actions (also called programs) out of the primitive actions described in a BAT. Here, we consider the fragment consisting of the constructs of *primitive actions* α, *test conditions* $\phi?$ (where ϕ is a formula

with all situation arguments suppressed), *sequential execution* $\delta_1;\delta_2$, *nondeterministic branching* $\delta_1 + \delta_2$, and *nondeterministic choice* ("pick") $\pi v.\delta$. We leave out nondeterministic iteration for simplicity. Programs are interpreted through predicate $Do(\delta, s, s')$, intuitively expressing that a successful execution of δ exists that starts in situation s and leads to situation s' (note that due to nondeterminism, a program δ in general admits several different executions):

Definition 1. $Do(\delta, s, s')$ is defined as a macro:

$$Do(\alpha, s, s') \doteq s' = do(\alpha, s)$$
$$Do(\phi?, s, s') \doteq \phi[s] \wedge s = s'$$
$$Do(\delta_1;\delta_2, s, s') \doteq \exists s''.\, Do(\delta_1, s, s'') \wedge Do(\delta_2, s'', s')$$
$$Do(\delta_1 + \delta_2, s, s') \doteq Do(\delta_1, s, s') \vee Do(\delta_2, s, s')$$
$$Do(\pi v.\delta, s, s') \doteq \exists v Do(\delta, s, s')$$

where $\phi[s]$ denotes the result of restoring situation argument s to any fluent occurring in ϕ (e.g. if ϕ is $Hungry(jones)$, then $\phi[s]$ stands for $Hungry(jones, s)$).

3 Extensions to the Situation Calculus

3.1 Action Aspects

When encoding a dynamic domain with a BAT, we would typically represent a fact such as "the agent eats a pizza" by means of an action term $eat(pizza)$. That is to say, an action name usually corresponds to the verb describing the act in question, and its parameter(s) to the object(s) being affected. (In a multi-agent scenario it makes sense to also name the subject, i.e. the acting agent, as parameter, but here we only consider single-agent domains and hence leave the agent implicit.) For the purpose of planning, this is usually sufficient, as we are only interested in the outcome of actions. Eating something will have the effect that the agent is satiated, and closing a door will result in the door being closed, no matter in what exact manner the agent performs these actions. This changes once we take social aspects into consideration, where some ways of doing something may be more or less appropriate, and where this may vary depending on the circumstances. For example, when eating pizza, eating *with one's hands* may be acceptable, but not when eating a steak. Similarly, closing a door by letting it slam shut will not be frowned upon in a loud factory environment, but in a library, we should rather close it *gently*.

Aspects like these, typically expressed through adverbials, go unmentioned most of the time when they are not relevant in the given context. We could encode them by means of additional parameters (e.g., $eat(pizza, hands)$), but this would mean that we have to foresee every one such aspect when defining the symbols of our action theory, and may result in actions taking a large number of parameters most of which are irrelevant in most contexts. Instead, we propose the following encoding, which among other things allows for a convenient notation that is closer to how we would express things in natural language.

We reserve the last argument of any action function for its *aspects*, for better notational distinction separated by a semicolon from "normal" parameters. Aspects of a concrete instance of an action are given in terms of a finite set of constant symbols. For example, instead of a unary $eat(x)$ action, we would use $eat(x; y)$, where x is the normal argument representing the food being eaten, while y stands for the finitely many aspects being explicitly involved. Instances of the eat action then include $eat(pizza; \{hands\})$, $eat(pizza; \{\})$, and $eat(x, \{hands, noisy\})$, and closing a door gently is encoded as $close(door; \{gently\})$. The intended reading is that an action term mentioning more aspects is more specific than one that mentions fewer (in terms of set containment). Hence, $eat(pizza; \{\})$ (eating a pizza) subsumes $eat(pizza; \{hands\})$ (eating a pizza with one's hands) in the sense that the latter requires that one eats with one's hands, while the former may or may not involve using one's hands. Aspects can also be negated, so that e.g. $eat(pizza; \{\overline{hands}\})$ means eating pizza *not* with ones hands, which is again a more specific case of $eat(pizza; \{\})$.

To define aspects formally, we make use of reification and introduce a special rigid, binary predicate $HasAspect(y, x)$. For a finite set of (both positive and negative) aspects $\Gamma = \{c_1, \ldots, c_k, \overline{c}_{k+1}, \ldots, \overline{c}_{k+m}\}$, let

$$HasAspects(y, \Gamma) \doteq \bigwedge_{i=1}^{k} HasAspect(y, c_i) \wedge \bigwedge_{i=k+1}^{k+m} \neg HasAspect(y, c_i) \quad (3)$$

where y serves as a placeholder for the set of aspects associated with the action, and the $(\neg)HasAspect(y, c_i)$ conjuncts express the presence (or absence) of the corresponding c_i in that set. For every such Γ, we include the statement

$$\exists y.\, HasAspects(y, \Gamma) \quad (4)$$

into the foundational axioms $\mathcal{D}_{fnd}$ of our BAT $\mathcal{D}$.

We may then make use of actions with aspects within complex actions and formulas. For formulas, without loss of generality, we assume that action terms $A(\vec{x}; \Gamma)$ only occur in equational subformulas of the form $(a = A(\vec{x}; \Gamma))$, where the left-hand side is an action variable a.[2] Then let

$$a = A(\vec{x}; \Gamma) \doteq \exists y.\, HasAspects(y, \Gamma) \wedge a = A(\vec{x}, y). \quad (5)$$

Aspects in complex actions can be understood as a macro:

$$A(\vec{x}; \Gamma) \doteq \pi y.\, HasAspects(y, \Gamma)?; A(\vec{x}, y). \quad (6)$$

The corresponding $A(\vec{x}, y)$ is a standard action term whose effects are defined as usual in the BAT, only with the extra y argument. As mentioned above, we expect that normally y would not have any influence on effects, e.g. the SSA

$$Hungry(do(a, s)) \equiv Hungry(s) \wedge \neg\exists y\, a = eat(pizza, y)$$

simply requires such a y (not) to exist (though nothing keeps us from referring to aspects in the SSA by means of the $HasAspect$ predicate, should we so desire).

For our further treatment we introduce a more general class of action expressions by extending a similar definition from (Baader and Zarrieß 2013) as follows:

[2] Note that we can use existential quantifiers to rewrite a formula such as $s' = do(A(\vec{x}; \Gamma), s)$ to $\exists a.\, a = A(\vec{x}; \Gamma) \wedge s' = do(a, s)$.

Definition 2. The *guarded actions* are the least set such that

- any primitive action α is a guarded action and
- if γ is a guarded action, then so are $\pi x.\gamma$ and $\phi?;\gamma$, where x is a variable and ϕ a situation-suppressed formula.

A guarded action hence is a primitive action, possibly preceded by a sequence of pick operators and test conditions. While any action with aspects falls into this category, guarded actions generally may include multiple picks and tests that refer to (rigid and fluent) predicates other than $HasAspect$. An important special case is the "wildcard" action, i.e., the nondeterministic choice of a single action:

$$\top \doteq \pi a.\, a \tag{7}$$

3.2 Joint and Negated Actions

For the purpose of expressing permission and prohibition with regards to actions, we have to include two additional notions into the GOLOG language. On the one hand, we need to be able to say that the agent adheres to two or more courses of action *simultaneously*. On the other hand, we require to express that the agent does *not* follow a certain course of action, i.e. *refrains* from its execution. For this purpose, we extend GOLOG by two additional constructs, namely for *joint* actions and action *negation*.

The definition of joint actions is straightforward. We add the following macro to the ones given in Definition 1:

Definition 3 (Joint Actions)**.**

$$Do(\delta_1 \times \delta_2, s, s') \doteq Do(\delta_1, s, s') \wedge Do(\delta_2, s, s')$$

Executing the two programs δ_1 and δ_2 jointly thus means the agent chooses an action sequence that is admitted by both δ_1 and δ_2 at the same time. For example, if $walk$, $sing$ and $talk$ are primitive actions, joining $(walk + sing)$ and $(sing + talk)$ yields a program only admitting $sing$. The $\times$ operator hence intuitively behaves like intersection, and should not be confused with concurrent execution, where primitive actions such as $walk$ and $sing$ can be performed simultaneously, and where interactions between the simultaneous actions' preconditions and effects have to be taken into consideration. Here, the agent still executes single primitive actions one at a time, only that the space of possible executions may be constrained by multiple GOLOG program expressions. This definition fits in rather nicely as $\times$ corresponds to conjunction in the same way that $+$ corresponds to disjunction, or π to existential quantification.

Unfortunately, it is less straightforward to define a widely applicable notion of what it means to negate complex actions. In analogy to what is said above, one might be tempted to simply use logical negation, i.e., define $Do(\overline{\delta}, s, s')$ as $\neg Do(\delta, s, s')$. However, this would have the undesired effect that $\overline{\delta}$ would suddenly connect pairs of situations s and s' that are otherwise unreachable from one another. An absurd example is that the action of *not* eating a pizza could thus bring the agent back to S_0, i.e., travel back in time! We therefore at least have to restrict the set of situations s' reachable by not doing δ to the ones that can actually be reached through some sequence of primitive actions:

$$Do(\overline{\delta}, s, s') \doteq s \sqsubseteq s' \wedge \neg Do(\delta, s, s') \tag{8}$$

Since situations are merely action sequences, this relativized action negation is much easier to define in the Situation Calculus than in formalisms of dynamic logic (Broersen 2004), where the exact semantics of negation depends on what other constructs are included in the action algebra.

However, (8) causes other problems when we consider sequences of actions. Meyer (1988) argues that it makes sense to understand a negated sequence $\overline{\delta_1;\delta_2}$ as being equivalent to $\overline{\delta_1}+\delta_1;\overline{\delta_2}$, i.e., in order to *not* do δ_1 followed by δ_2, either do something that is not δ_1, or, if you do δ_1, do something different from δ_2 afterwards. This is one of five postulates he presents "must reasonably hold" in the sense that they alone entail many important theorems for his formalism, for example the fact that a sequence $\langle\alpha,\beta\rangle$ is obligatory just in case α is obligatory, and β is obligatory after doing α. Sadly, with (8) we obtain that one way of not doing a primitive action α is to do the *sequence* $\langle\alpha,\beta\rangle$, and so his postulate does not apply here. One option would be to adopt the action semantics proposed in (Meyer 1988). It is however quite involved, as every complex action is identified with the set of infinite traces that have a finite prefix admitted by that action. Another possibility would be to take inspiration from another proposal (Wansing 2004) where refraining from a sequence $\delta_1;\delta_2$ is understood as refraining from all its parts, i.e., equating $\overline{\delta_1;\delta_2}$ with $\overline{\delta_1};\overline{\delta_2}$. While this is technically pleasing, Broersen (2004) argues that this may be too strong of an assumption in many contexts.

For the purpose of this paper, we only really need to apply negation to program expressions whose "duration" is exactly one action, meaning programs that only admit action sequences of length one (if any). Intuitively, we want that refraining from a program δ of this kind means executing a *single* action that is *not* among those admitted by δ. We therefore use the following, restricted variant of (8):

Definition 4 (Negated Actions)**.**

$$Do(\overline{\delta}, s, s') \doteq \exists a.\, s' = do(a, s) \wedge \neg Do(\delta, s, s')$$

This lets us identify a subset of programs for which it is "safe" to apply this limited type of negation:

Definition 5. We define the *guarded-action fragment* as the set of expressions admitted by the grammar

$$\delta \;::=\; \gamma \mid \overline{\delta} \mid \delta + \delta \mid \delta \times \delta$$

where γ denotes a guarded action.

Since $\top$ is a guarded action, $\bot \doteq \overline{\top}$ ("failure") is in the guarded-action fragment. For any two programs δ_1, δ_2, let

$$\delta_1 \equiv \delta_2 \;\doteq\; \mathcal{D}_{fnd} \models Do(\delta_1, s, s') \equiv Do(\delta_2, s, s').$$

Then, not to much of a surprise, we get:

Proposition 6. *The guarded-action fragment is a Boolean algebra:*

1. $(\delta_1 + \delta_2) + \delta_3 \equiv \delta_1 + (\delta_2 + \delta_3)$ *($+$ is associative)*
2. $\delta_1 + \delta_2 \equiv \delta_2 + \delta_1$ *($+$ is commutative)*
3. $\delta_1 + \bot \equiv \delta_1$ *($\bot$ is the neutral element wrt $+$)*
4. $\delta_1 + (\delta_2 \times \delta_3) \equiv (\delta_1 + \delta_2) \times (\delta_1 + \delta_3)$ *($+$ distributivity)*

5. $\delta_1 + \delta_1 \equiv \delta_1$ *(+ is idempotent)*
6. $\delta_1 + \overline{\delta_1} \equiv \top$ *(complement wrt +)*
7. $(\delta_1 \times \delta_2) \times \delta_3 \equiv \delta_1 \times (\delta_2 \times \delta_3)$ *(× is associative)*
8. $\delta_1 \times \delta_2 \equiv \delta_2 \times \delta_1$ *(× is commutative)*
9. $\delta_1 \times \top \equiv \delta_1$ *(⊤ is the neutral element wrt ×)*
10. $\delta_1 \times (\delta_2 + \delta_3) \equiv (\delta_1 \times \delta_2) + (\delta_1 \times \delta_3)$ *(× distributivity)*
11. $\delta_1 \times \delta_1 \equiv \delta_1$ *(× is idempotent)*
12. $\delta_1 \times \overline{\delta_1} \equiv \bot$ *(complement wrt ×)*

As a direct consequence, we get that the usual rules for double negation and De Morgan's laws apply:

Proposition 7. *For δ_1, δ_2 from the guarded-action fragment,*

1. $\overline{\overline{\delta_1}} \equiv \delta_1$ *2.* $\overline{\delta_1 \times \delta_2} \equiv \overline{\delta_1} + \overline{\delta_2}$ *3.* $\overline{\delta_1 + \delta_2} \equiv \overline{\delta_1} \times \overline{\delta_2}$

As GOLOG was originally inspired by, among other things, dynamic logic, the following is perhaps even less surprising:

Proposition 8. *The fragment of* GOLOG *consisting of $+$ (nondeterministic choice), ; (sequence), and * (iteration) make up a Kleene algebra, where $\bot$ is again the neutral element wrt $+$, and where the neutral element wrt ; is the empty program $nil \doteq$ TRUE?. In particular, we have that*

$$\bot; \delta \equiv \bot.$$

Note that Propositions 6 and 8 talk about *fragments*, and that GOLOG is still more than just a combination of these two algebras. Through incorporating first-order aspects in the form of test conditions, pick operators, and successor state axioms, it allows for defining dynamic domains at a much finer level of granularity. In particular, programs and even actions have an internal structure, and so it is useful to have the following comparison operator:

$$\delta_1 \rhd \delta_2 \;\doteq\; \mathcal{D}_{fnd} \models Do(\delta_1, s, s') \supset Do(\delta_2, s, s')$$

$\delta_1 \rhd \delta_2$ thus expresses that δ_1 is more specific than δ_2 in the sense that every action sequence admitted by δ_1 is also admitted by δ_2. An important special case is:

Proposition 9. *If Γ_1 and Γ_2 are finite sets of action aspects such that $\Gamma_1 \supseteq \Gamma_2$, then $A(\vec{x}; \Gamma_1) \rhd A(\vec{x}; \Gamma_2)$.*

Moreover, if one complex action is more specific than another, the following simplifications are possible:

Proposition 10. *Let $\delta_1 \rhd \delta_2$. Then*

1. $\delta_1 \times \delta_2 \equiv \delta_1$ *2.* $\overline{\delta_1} \times \overline{\delta_2} \equiv \overline{\delta_2}$ *3.* $\delta_1 \times \overline{\delta_2} \equiv \bot$

Finally, for single-action programs, joint execution distributes over sequence:

Proposition 11. *Let $\delta_1, \ldots, \delta_4$ be of the guarded-action fragment. Then* $(\delta_1; \delta_2) \times (\delta_3; \delta_4) \equiv (\delta_1 \times \delta_3); (\delta_2 \times \delta_4)$.

4 The Main Framework

Our goal is to define a deontic preference relation $\prec$ over pairs of situations. Intuitively, $s \prec s'$ means that the situation described by s is preferred, or more ideal, than the one denoted by s'. When the agent is deliberating and considering different alternative courses of action, it can choose the one that will lead it to the most ideal (i.e., minimal) situation in terms of $\prec$. While there exist many conceivable ways of specifying such a relation, we will describe one with some desirable properties below. For now, all we require is that $\prec$ is asymmetric. As usual, we understand $\preceq$ as the reflexive closure of $\prec$. Once a deontic preference relation is fixed, we can use it to define dyadic notions of obligation, permission and prohibition. For this purpose, let the *minimal* situations σ' wrt $\prec$ reachable by means of a program δ from a starting situation σ be given through the following:

$$\begin{aligned}&\mathrm{MinDo}(\prec, \delta, \sigma, \sigma') \;\doteq\\ &\quad Do(\delta, \sigma, \sigma') \wedge \neg\exists s.\, Do(\delta, \sigma, s) \wedge (s \prec \sigma')\end{aligned} \tag{9}$$

Then we define:

Definition 12. Given a deontic preference relation $\prec$ over situations, two programs δ and γ, and a situation term σ, the fact that γ is *obligatory* given δ in σ is defined by

$$\mathbf{obl}(\gamma \,|\, \delta)[\sigma] \doteq \forall s.\, Do(\gamma, \sigma, s) \equiv \mathrm{MinDo}(\prec, \delta, \sigma, s).$$

The fact that γ is *permitted* given δ in σ is expressed as

$$\mathbf{per}(\gamma \,|\, \delta)[\sigma] \doteq \forall s.\, Do(\gamma, \sigma, s) \supset \mathrm{MinDo}(\prec, \delta, \sigma, s).$$

Here we deviate from the standard definition where obligation is a normal (KD) modality, which in our framework would translate to saying that every minimal δ-situation is also a γ-situation. Instead we require that γ represents *all and only* the minimal δ-situations for it to be obligatory. Furthermore, our definition of permission is not the dual of obligation, but a variant of *strong permission*, where γ being permitted means all its reachable situations are among the ideal ones. We thus follow ideas similar to those underlying *minimal* deontic (action) logics (van Benthem 1979; Trypuz and Kulicki 2015), which lets us avoid some of the standard paradoxes of SDL that result from reading logical disjunction as free choice and including the axiom of necessitation (any logical consequence of an obligation is an obligation itself). For example, if m stands for "mail the letter", and b for "burn the letter", then in SDL $\mathbf{O}m$ entails $\mathbf{O}(m \vee b)$, but it is counter-intuitive that an obligation to mail the letter means one has the free choice between mailing or burning it (Ross 1930). On the other hand, in the above definition, the $+$ operator indeed denotes a free choice for the agent, and no necessitation applies. Consequently, $\mathbf{obl}(m \,|\, \top)[\sigma]$ does not entail $\mathbf{obl}(m + b \,|\, \top)[\sigma]$. Similarly, *free choice permission* means that if one is permitted to do a or b, then this should entail both the permission for a and for b, yet $\mathbf{P}(a \vee b) \supset \mathbf{P}a \wedge \mathbf{P}b$ is not a theorem of SDL. On the other hand, Definition 12 yields that $\mathbf{per}(\gamma_1 + \gamma_2 \,|\, \delta)[\sigma]$ in fact implies $\mathbf{per}(\gamma_1 \,|\, \delta)[\sigma] \wedge \mathbf{per}(\gamma_2 \,|\, \delta)[\sigma]$.

A difference is that here we see $\mathbf{obl}(\gamma \,|\, \delta)[\sigma]$ and $\mathbf{per}(\gamma \,|\, \delta)[\sigma]$ merely as emergent notions, derived from the normative system encoded by $\prec$. One could view this as a case of a common distinction made in Deontic Logic: For one, *prima facie* obligations represent general, possibly conflicting, moral principles imposed on the agent by some moral authority, its conscience or the like. For another, *all-things-considered* obligations determine what ought to be done in a particular situation, when all circumstances are taken into consideration, and after conflicting moral principles were weighed against each other (Horty 2003).

$\mathbf{obl}(\gamma \mid \delta)[\sigma]$ then describes the result of the agent's moral deliberation, signifying that if a course of action according to δ is under consideration, γ specifies all and only options leading to ideal outcomes, whereas $\mathbf{per}(\gamma \mid \delta)[\sigma]$ expresses that every choice admitted by γ is guaranteed to be ideal.

4.1 Action Conditionals

Many ways of defining a deontic preference relation are conceivable. Here we propose to employ *conditional* rules that describe deontic constraints in a *defeasible*, declarative and qualitative fashion, and let the combined influence of these constraints induce our overall preference relation.

Already in the case of propositional formulas, defining an appropriate semantics for defeasible conditionals is an issue in and of itself. Various systems have been proposed over the years, all with their individual strengths and weaknesses (Kern-Isberner and Eichhorn 2014). For the sake of simplicity, we here consider the popular and well-understood approach of *rational closure* (Kraus, Lehmann, and Magidor 1990), which was shown to be equivalent to 1-entailment in System Z (Pearl and Goldszmidt 1990). To determine a unique ranking induced by a finite set of conditionals, we follow the construction presented in (Lehmann and Magidor 1992), with the difference that in place of propositional formulas, we consider complex actions. Here, instead of possible worlds, antecedents and consequences of a rule are to be understood in terms of the *situations* they can reach. Consequently, the notion of *consistency* of propositional formulae is replaced by that of *executability* of programs.

Definition 13. A *conditional* is an expression of the form

$$\delta \Rightarrow \gamma$$

where δ and γ are from the guarded-action fragment. We read $\delta \Rightarrow \gamma$ as "if the agent is committed to the course of action described by δ, it ought to do γ." The special case $\top \Rightarrow \gamma$ denotes an unconditional deontic constraint and is to be read as "the agent ought to do γ."

Note that $\Rightarrow$ is a special symbol denoting a defeasible conditional, and not to be confused with material implication. For the latter, because of the fact that our variant of GOLOG includes not only the disjunction, but also negation and conjunction of actions, we can define the following:

Definition 14. For a conditional $r = (\delta \Rightarrow \gamma)$, its *materialized* counterpart is given by

$$\mathfrak{M}(r) \doteq (\overline{\delta} + \gamma)$$

For $R = \{r_1, \ldots, r_k\}$, let $\mathfrak{M}(R) \doteq \mathfrak{M}(r_1) \times \cdots \times \mathfrak{M}(r_k)$.

Definition 15. Given a ground situation term σ, and a finite set of conditionals R, the formula expressing that conditional $r = (\delta \Rightarrow \gamma)$ is *exceptional for* R is defined as

$$Exc(r, R, \sigma) \doteq \neg\exists s.\ Do(\mathfrak{M}(R) \times \delta, \sigma, s).$$

Intuitively, a conditional is exceptional wrt R when its antecedent cannot be executed concurrently with $\mathfrak{M}(R)$.

Definition 16. A BAT $\mathcal{D}$ induces a ranking over a finite set of conditionals R wrt a ground situation term σ as follows:

$$R_0(\sigma) := R$$
$$R_{i+1}(\sigma) := \{r \in R_i \mid \mathcal{D} \models Exc(r, R_i, \sigma)\}$$

The *rank* of a ground situation term σ' from the point of view of σ, given $\mathcal{D}$ and R is then given by

$$rank(\mathcal{D}, R)[\sigma, \sigma'] \doteq \min\{i \mid \mathcal{D} \models Do(\mathfrak{M}(R_i), \sigma, \sigma')\}$$

Note that each R_{i+1} is a subset of the corresponding R_i, and so the inductive construction will converge after at most $|R|$ steps, when at some point no more rule is removed.

Example 17 (Forrester's Paradox)**.** The aforementioned contrary-to-duty scenario of the "gentle murderer" is expressed by the following set of rules R:

$$\top \Rightarrow \overline{murder(jones; \{\})} \quad (10)$$
$$murder(jones; \{\}) \Rightarrow murder(jones; \{gently\}) \quad (11)$$

saying that generally, Smith (the agent) should not murder Jones, but if he does, he ought to do so gently. Let $\mathcal{D}$ be any BAT that contains the foundational axioms (fluents and action effects are not required for this scenario), and σ be an arbitrary ground situation. In order to determine the ranking for R wrt $\mathcal{D}$ and s, let $R_0 = R = \{(10), (11)\}$. To obtain R_1, we first have to construct the materialized version of R_0. In the following, let m stand for $murder(jones; \{\})$ and g for $murder(jones; \{gently\})$. For rule (10), with Proposition 6 we obtain $\overline{\top} + \overline{m} \equiv \bot + \overline{m} \equiv \overline{m}$, and thus

$$\begin{aligned}\mathfrak{M}(R_0) &\equiv \overline{m} \times (\overline{m} + g)\\ &\equiv (\overline{m} \times \overline{m}) + (\overline{m} \times g) \equiv \overline{m} + \bot \equiv \overline{m}.\end{aligned}$$

The second last simplification step is because $g \rhd m$ by Prop. 9, and so $\overline{m} \times g \equiv \bot$ due to Prop. 10 (it is impossible to refrain from murdering while murdering gently).

To determine which rules are exceptional, note that $\overline{m} \times \top$ admits any action that is not an instance of $murder$, while $\overline{m} \times m \equiv \bot$ does not admit any actions. Therefore,

$$\mathcal{D} \not\models \neg\exists s.\ Do(\overline{m} \times \top, \sigma, s) \Rightarrow \mathcal{D} \not\models Exc((10), R_0, \sigma),$$
$$\mathcal{D} \models \neg\exists s.\ Do(\overline{m} \times m, \sigma, s) \Rightarrow \mathcal{D} \models Exc((11), R_0, \sigma).$$

Hence, $R_1 = \{(11)\}$ and $\mathfrak{M}(R_1) \equiv \overline{m} + g$. With

$$\begin{aligned}(\overline{m} + g) \times m &\equiv (\overline{m} \times m) + (g \times m)\\ &\equiv \bot + (g \times m) \equiv g \times m \equiv g\end{aligned}$$

we get $\mathcal{D} \not\models \neg\exists s.\ Do((\overline{m} + g) \times m, \sigma, s)$, i.e., $\mathcal{D} \not\models Exc((11), R_1, \sigma)$, hence $R_2 = \emptyset$ and $\mathfrak{M}(R_2) = \top$. We thus end up with

$$\mathfrak{M}(R_0) \equiv \overline{m}, \quad \mathfrak{M}(R_1) \equiv \overline{m} + g, \quad \mathfrak{M}(R_2) \equiv \top$$

which induces the following ranking:

$$rank(\mathcal{D}, R)[\sigma, do(a, \sigma)] = \begin{cases} 0, & a \neq murder(jones; \{\}) \\ 1, & a = murder(jones; \{gently\}) \\ 2, & a = murder(jones; \{\overline{gently}\}) \end{cases}$$

Example 18 (The Pizza Example)**.** To demonstrate how our approach deals with specificity, consider the following scenario adapted from (Horty 2014):

$$\pi x.\ eat(x; \{\}) \Rightarrow \overline{\pi x.\ eat(x; \{hands\})} \quad (12)$$
$$eat(pizza; \{\}) \Rightarrow eat(pizza; \{hands\}) \quad (13)$$

The first statement expresses that generally, when eating, one ought not eat with one's hands. The second one says that when eating pizza, one ought to eat with one's hands.

Let e stand for $\pi x.\ eat(x;\{\})$, h for $\pi x.\ eat(x;\{hands\})$, p for $eat(pizza;\{\})$, and ph for $eat(pizza;\{hands\})$. Obviously, $ph \rhd h$, $ph \rhd p$, $h \rhd e$, $p \rhd e$, and $ph \rhd e$, so for $R_0 = \{(12),(13)\}$ we get

$$\begin{aligned}\mathfrak{M}(R_0) &\equiv (\overline{e} + \overline{h}) \times (\overline{p} + ph)\\ &\equiv (\overline{e} \times \overline{p}) + (\overline{e} \times ph) + (\overline{h} \times \overline{p}) + (\overline{h} \times ph)\\ &\equiv \quad \overline{e} \quad + \quad \bot \quad + (\overline{h} \times \overline{p}) + \quad \bot\\ &\equiv \ \overline{e} + (\overline{h} \times \overline{p})\end{aligned}$$

For arbitrary $\mathcal{D}$ and σ, we get $\mathcal{D} \not\models Exc((12), R_0, \sigma)$, but $\mathcal{D} \models Exc((13), R_0, \sigma)$. Hence, $R_1 = \{(13)\}$, $R_2 = \emptyset$, and

$$rank(\mathcal{D}, R)[\sigma, do(a, \sigma)] = \begin{cases} 0, & \forall x, y.\ a \neq eat(x;y) \vee \\ & \exists x.\ a = eat(x;\{\overline{hands}\}) \wedge x \neq pizza \\ 1, & a = eat(pizza;\{hands\}) \\ 2, & a = eat(pizza;\{\overline{hands}\}) \vee \\ & \exists x.\ a = eat(x;\{hands\}) \wedge x \neq pizza \end{cases}$$

This correctly encodes that one ought not eat with one's hands, except when eating pizza. However, another perhaps undesired consequence is that generally, one ought not eat pizza. This is because the system of rational closure, which we adapted for the sake of simplicity, is intended for reasoning about *normality*, and so some of its inferences may be too strong under a *deontic* interpretation. For future work, we want to study how the framework can be adapted to follow different styles of conditional reasoning such as the one described in (Delgrande 2020).

4.2 Temporal Action Conditionals

Many scenarios, such as the Chisholm example, involve some sort of temporal aspect. Specifically, we want to be able to express that the agent ought to do a specific action *after* or *before* doing a certain other action it intends to do. For these cases, we propose the following:

Definition 19. The two types of *temporal conditionals* are

$$\delta \Rightarrow_a \gamma \quad \text{and} \quad \delta \Rightarrow_b \gamma,$$

where again δ and γ are from the guarded-action fragment. We read $\delta \Rightarrow_a \gamma$ as "if committed to doing δ, the agent ought to do γ *afterwards*", and $\delta \Rightarrow_b \gamma$ as "…*before*". Once again, the special case with $\top$ as the antecedent denotes an unconditional, or universal obligation.

For these new types of conditionals, we accordingly adapt the notions of materializing and exceptionality:

Definition 20. For temporal conditionals, let

$$\begin{aligned}\mathfrak{M}(\delta \Rightarrow_a \gamma) &\doteq (\overline{\delta}; \top + \top; \gamma)\\ \mathfrak{M}(\delta \Rightarrow_b \gamma) &\doteq (\top; \overline{\delta} + \gamma; \top)\end{aligned}$$

As before, for a finite set $R = \{r_1, \ldots, r_k\}$ we understand $\mathfrak{M}(R)$ as $\mathfrak{M}(r_1) \times \cdots \times \mathfrak{M}(r_k)$.

Definition 21. Given a ground situation term σ, and a finite set of temporal conditionals R, the formula expressing the fact that temporal conditional $r = (\delta \Rightarrow_x \gamma)$, $x \in \{a, b\}$ is *exceptional for* R is defined as

$$\begin{aligned}Exc(\delta \Rightarrow_a \gamma, R, \sigma) &\doteq \neg\exists s.\ Do(\mathfrak{M}(R) \times (\delta; \top), \sigma, s)\\ Exc(\delta \Rightarrow_b \gamma, R, \sigma) &\doteq \neg\exists s.\ Do(\mathfrak{M}(R) \times (\top; \delta), \sigma, s)\end{aligned}$$

Rankings are defined as before (Definition 16).

Example 22 (The Chisholm Paradox)**.** The first three statements of the Chisholm scenario can be expressed as:

$$\top \Rightarrow_a help \tag{14}$$

$$help \Rightarrow_b tell \tag{15}$$

$$\overline{help} \Rightarrow_b \overline{tell} \tag{16}$$

The first rule states that generally, the agent ought to go help the neighbours. The second one means that when the agent intends to go and help, it should tell the neighbours immediately before. If on the other hand, says the third rule, the agent does not intend to go and help, it ought not tell them.

In the following, let h stand for the action term $help$, and t for $tell$. Materializing $R_0 = \{(14), (15), (16)\}$ then yields:

$$\begin{aligned}\mathfrak{M}((14)) &= (\overline{\top}; \top) + (\top; h) \quad \equiv (\top; h)\\ \mathfrak{M}((15)) &= \qquad\qquad\quad (\top; \overline{h}) + (t; \top)\\ \mathfrak{M}((16)) &= (\top; \overline{\overline{h}}) + (\overline{t}; \top) \quad \equiv (\top; h) + (\overline{t}; \top)\end{aligned}$$

Recall that $\mathfrak{M}(R_0)$ is given by the conjunction of these three expressions. Since according to Proposition 6, the usual distributive laws apply, we can again "multiply" them out. Observe that $(\top; h)$ is incompatible with $(\top; \overline{h})$, and that $(t; \top)$ contradicts with $(\overline{t}; \top)$. The result is hence equivalent to $(\top; h) \times (t; \top)$, which in turn can be simplified to $(t; h)$ using Propositions 11 and 6. Therefore,

$$\begin{aligned}&\mathcal{D} \not\models \neg\exists s. Do((t;h) \times (\top;\top), \sigma, s) \quad (\equiv Exc((14), R_0, \sigma))\\ &\mathcal{D} \not\models \neg\exists s. Do((t;h) \times (\top;h), \sigma, s) \quad (\equiv Exc((15), R_0, \sigma))\\ &\mathcal{D} \models \neg\exists s. Do((t;h) \times (\top;\overline{h}), \sigma, s) \quad (\equiv Exc((16), R_0, \sigma))\end{aligned}$$

Because rule (16) is the only exceptional one, we obtain $R_1 = \{(16)\}$ and $\mathfrak{M}(R_1) \equiv \top; h + \overline{t}; \top$. The rule is obviously not exceptional with itself, so $R_2 = \emptyset$, hence $\mathfrak{M}(R_2) = \top$. We thus end up with

$$\mathfrak{M}(R_0) \equiv t; h,\ \ \mathfrak{M}(R_1) \equiv \top; h + \overline{t}; \top,\ \ \mathfrak{M}(R_2) \equiv \top$$

which induces the following ranking:

$$rank(\mathcal{D}, R)[\sigma, do(\langle a, b\rangle, \sigma)] = \begin{cases} 0, & a = tell \wedge b = help \\ 1, & a \neq tell \\ 2, & a = tell \wedge b \neq help \end{cases}$$

4.3 A Deontic Preference Relation over Situations

What is left now is to define how the rankings induced by conditional rules for single situation terms σ can be combined into one general deontic preference relation $\prec$. Again, various definitions are conceivable, and we only describe one such option here that we believe is useful. Recall that we

want to use deontic conditionals as *constraints* that govern the general behaviour of the agent. That is to say we assume that the agent is deliberating over a course of action to solve a bigger task, and we would like it to choose one whose entire execution violates as few constraints as possible, if any.

Here we propose a simple additive model where the total rank of a situation is determined by adding all ranks assigned to preceding situations by preceding situations:

$$drank(\mathcal{D}, R)[\sigma] \doteq \sum_{\sigma' \sqsubseteq \sigma'' \sqsubseteq \sigma} rank(\mathcal{D}, R)[\sigma', \sigma'']$$

The intuition here is that once a "bad" action (e.g., murdering) has been performed, all subsequent situations will be ranked as less ideal. There is no way of undoing a bad act, and any further bad deed makes the situation less and less ideal. The deontic preference relation then simply compares the total ranks of both situations:

$$\sigma \prec \sigma' \;\doteq\; drank(\mathcal{D}, R)[\sigma] < drank(\mathcal{D}, R)[\sigma']$$

5 Compiling Deontic Constraints into BATs

So far, we defined rankings and preference relations as metatheoretic notions. For practical application, it would be useful to be able to express them within the standard framework of BATs, without the need for additional reasoning machinery. Thus, they could be easily integrated into existing solutions based on the Situation Calculus and GOLOG, whose close relation (in terms of inter-compilability) to planning languages such as STRIPS (Lin and Reiter 1997) and PDDL (Baier, Fritz, and McIlraith 2007; Röger, Helmert, and Nebel 2008) moreover would similarly allow to incorporate the encoded norms and guidelines into planning tasks.

In this section, we therefore present a simple approach for compiling deontic constraints, from a restricted fragment, directly into a classical BAT. The result of this preprocessing step will contain an additional numeric fluent whose value corresponds to the rank of the current situation, and thus can serve as a metric to be minimized by the planner when searching for an action execution sequence. For this purpose, we identify sufficient conditions for when such a compilation is possible. First, notice that in all examples discussed so far, the ground situation σ didn't actually matter when determining the ranking. Formally:

Definition 23. We say that a ranking is *situation independent* iff for all ground situations terms σ and σ' and all ground action sequences $\vec{\alpha}$,

$$rank(\mathcal{D}, R)[\sigma, do(\vec{\alpha}, \sigma)] = rank(\mathcal{D}, R)[\sigma', do(\vec{\alpha}, \sigma')]$$

The following is easy to see:

Proposition 24. *If the rules in a set R of (temporal) conditionals do not mention any fluents (inside test conditions $\phi?$), the induced ranking is situation independent.*

Next, notice that our ranking definitions required a rule to be *entailed to be* exceptional by the BAT in order to be included in the next rank. If a rule is not included, it may be because it is actually not exceptional, or simply because the BAT does not contain sufficient information to decide whether the program expression in question is executable, e.g. if it starts with a test condition $\phi?$ about whose truth value the agent is ignorant. Since including such epistemic notions would be beyond the scope of this paper, we require the following:

Definition 25. We say that a BAT $\mathcal{D}$ and a set of conditionals R is *exceptionality determinate* iff for every $r \in R$, every subset $R' \subseteq R$, and every ground situation term σ,

$$\mathcal{D} \not\models Exc(r, R', \sigma) \;\text{ iff }\; \mathcal{D} \models \neg Exc(r, R', \sigma) \qquad (17)$$

One way to ensure this property is the following:

Proposition 26. *If $\mathcal{D}$ is consistent, and the only test conditions occurring in rules in R are due to action aspects, $\mathcal{D}$ and R are exceptionality determinate.*

Note that Propositions 24 and 26 apply to all examples in this paper. For representing ordinal ranks in BATs, we now introduce a new functional fluent $ideal(s)$, expressing the "degree of ideality" situation s has wrt the action sequence it represents, where higher values stand for less ideal situations and 0 is optimal. Let the initial theory $\mathcal{D}_0$ contain the axiom

$$ideal(S_0) = 0, \qquad (18)$$

The value may increase due to actions, as per the SSA

$$ideal(do(a, s)) = ideal(s) + bad(a, s). \qquad (19)$$

The potential increase is given by means of another functional fluent $bad(a, s)$, describing how "bad" it is to do action a in situation s, again expressed by a natural number.

5.1 The Non-Temporal Case

For defining the badness of actions in the non-temporal case, suppose we determined a ranking for an arbitrary ground situation σ (e.g., S_0). The result consists of a finite number of rule sets $R_0, \ldots, R_k$ whose materialized counterparts

$$\mathfrak{M}(R_0) \equiv \delta_0, \quad \mathfrak{M}(R_1) \equiv \delta_1, \quad \ldots \quad \mathfrak{M}(R_k) \equiv \delta_k$$

are program expressions from the guarded-action fragment. Under the assumption of situation-independence, for every action a and situation s, whatever program δ_i with minimal index i allows to execute a in s determines that i is the rank s assigns to $do(a, s)$. To compile this information into an SSA, we have to express it in terms of a formula uniform in s. For this purpose, we define the following operator that takes a program δ from the guarded-action fragment, an action variable a, and a situation variable s:

Definition 27.

1. $\mathfrak{C}[\alpha, a, s] = (a = \alpha)$
2. $\mathfrak{C}[\phi?; \delta, a, s] = \phi[s] \wedge \mathfrak{C}[\delta, a, s]$
3. $\mathfrak{C}[\pi v.\delta, a, s] = \exists v.\, \mathfrak{C}[\delta, a, s]$
4. $\mathfrak{C}[\overline{\delta}, a, s] = \neg\mathfrak{C}[\delta, a, s]$
5. $\mathfrak{C}[\delta_1 + \delta_2, a, s] = \mathfrak{C}[\delta_1, a, s] \vee \mathfrak{C}[\delta_2, a, s]$
6. $\mathfrak{C}[\delta_1 \times \delta_2, a, s] = \mathfrak{C}[\delta_1, a, s] \wedge \mathfrak{C}[\delta_2, a, s]$

$\mathfrak{C}[\delta, a, s]$ yields a formula that correctly describes the executability of a in s according to δ as follows:

Lemma 28. *Let δ be from the guarded-action fragment. Then $\mathfrak{C}[\delta, a, s]$ is uniform in s and*

$$\mathcal{D} \models \forall a, s.\, \mathfrak{C}[\delta, a, s] \;\equiv\; Do(\delta, s, do(a, s))$$

Note that the formula $Do(\delta, s, do(a, s))$ almost already has the right form, except for the fact that we have to apply foundational axiom (1) to replace expressions of the form $(do(\alpha_1, s) = do(\alpha_2, s))$ by $(\alpha_1 = \alpha_2)$ and $s = s$ by TRUE. With this operator, we can now define an axiom for bad:

$$bad(a, s) = b \equiv \bigvee_{i=0}^{k} (b = i) \wedge \mathfrak{C}[\delta_i, a, s] \ \wedge \bigwedge_{j=0}^{i-1} \neg\mathfrak{C}[\delta_j, a, s] \tag{20}$$

Theorem 29. *Let $\mathcal{D}$ be a BAT, R a set of non-temporal deontic constraints, and $\mathcal{D}_R$ be the result of extending $\mathcal{D}$ with axioms (18) – (20). Then for every ground situation term σ,*

$$drank(\mathcal{D}, R)[\sigma] = d \ \ \textit{iff} \ \ \mathcal{D}_R \models ideal(\sigma) = d$$

For example, in the Forrester scenario, after simplification, we get the following axiom:

$$\begin{aligned} bad(a, s) = b \ \equiv \ & b = 0 \wedge a \neq murder(jones; \{\}) \vee \\ & b = 1 \wedge a = murder(jones; \{gently\}) \vee \\ & b = 2 \wedge a = murder(jones; \{\overline{gently}\}) \end{aligned}$$

In practice, it will be useful to simplify action expressions early on as we did in Examples 17, 18, and 22, using properties from Propositions 6–11 as rewriting rules. Since the guarded-action fragment is a Boolean algebra, knowledge compilation techniques for propositional logic are applicable, e.g. ordered binary decision diagrams (Bryant 1986).

5.2 The Temporal Case

In the temporal case, instead of single actions, we have to consider sequences of two actions. Formally, we are dealing with $\{\times, +\}$-combinations of expressions of the form $\gamma; \delta$, where γ and δ are from the guarded-action fragment. To devise a bad axiom of the appropriate form, we need to be able to "remember" which ones of these γs on the left-hand side of a ";" occurred just before the current situation. We do so by introducing finitely many additional fluent predicates $Did[\gamma](s)$, where for each one $\mathcal{D}_0$ contains the axiom

$$\neg Did[\gamma](S_0) \tag{21}$$

and $\mathcal{D}_{post}$ contains the SSA

$$Did[\gamma](do(a, s)) \equiv \mathfrak{C}[\gamma, a, s]. \tag{22}$$

We then use these when encoding program executability into our axioms:

Definition 30. For the temporal case, we extend Definition 27 to include the following inductive rule:

7. $\mathfrak{C}[\gamma; \delta, a, s] = Did[\gamma](s) \wedge \mathfrak{C}[\delta, a, s]$

We obtain a similar lemma as in the non-temporal case:

Lemma 31. *If δ is a program expression obtained in the temporal case, $\mathfrak{C}[\delta, a, s]$ is uniform in s and*

$$\mathcal{D} \models \forall a, s.\ \mathfrak{C}[\delta, a, s] \ \equiv \ \exists s'.\ Do(\delta, s', do(a, s))$$

In words, $\mathfrak{C}[\delta, a, s]$ is true just in case doing a in s is an action that *completes* the execution of δ, which started in some previous situation s'. Then we have:

Theorem 32. *Let $\mathcal{D}$ be a BAT, R a set of temporal deontic constraints, and $\mathcal{D}_R$ be the result of extending $\mathcal{D}$ with axioms (18) – (22). Then for every ground situation term σ,*

$$drank(\mathcal{D}, R)[\sigma] = d \ \ \textit{iff} \ \ \mathcal{D}_R \models ideal(\sigma) = d$$

For example, in the Chisholm scenario, we get:

$$\begin{aligned} bad(a, s) = b \ \equiv \ & b = 0 \wedge Did[tell](s) \wedge a = help \vee \\ & b = 1 \wedge \neg Did[tell](s) \vee \\ & b = 2 \wedge Did[tell](s) \wedge a \neq help \end{aligned}$$

6 Discussion

While many early researchers applied deontic operators to propositions, in his seminal article, von Wright (1951) originally introduced deontic modalities as applying to action types. He argued that a suitable deontic logic needs to be built upon the foundation of a more general theory of action (von Wright 1963). Deontic action logic is still an active area of research, and more recent contributions include work due to Trypuz and Kulicki (2015), who study deontic notions over expressions of a Boolean algebra of actions. Moreover, there is increasing interest in integrating normative reasoning into (multi-) agent systems (Chopra et al. 2018).

One prominent approach is to use *stit* ("see to it that") semantics (Horty 2001), which employs a branching time structure similar to Situation Calculus, and so defeasible dyadic obligations can be used to define preference relations over histories (Bartha 1999). However, actions in stit do not have proper names or types, but are described purely through their effects, making it difficult to deal with constraints over complex actions. It has been proposed to reintroduce action types into stit, for example to for representing epistemic obligations (Horty 2019), but this raises the question whether a formalism that is based on action types in the first place would not be more appropriate.

Another line of research uses variants of *dynamic logic* (Segerberg 2012; Meyer 2019). Meyer's deontic dynamic logic PDeL applies deontic operators to (complex) action types, where a special proposition V denotes a violation, and prohibition ($\mathbf{F}\alpha \doteq [\alpha]V$), permission ($\mathbf{P}\alpha \doteq \neg\mathbf{F}\alpha$) and obligation ($\mathbf{O}\alpha \doteq \mathbf{F}\overline{\alpha}$) of an action α are defined in terms of whether α brings about V. While the formalism is elegant and could address many of the classical deontic paradoxes, contrary-to-duty scenarios proved to be more difficult and required additional machinery in the form of multiple violation atoms $V_1, V_2, \ldots$ as well as accordingly indexed modal operators $\mathbf{O}_1$,$\mathbf{F}_2$,... (Dignum, Meyer, and Wieringa 1994).

In this paper, we presented an alternative approach where contrary-to-duty statements are instead represented through dyadic deontic constraints over complex actions. This not only allows to express ought-to-do constraints in a straightforward manner, but also to compile them directly into the agent's action theory, thus making them immediately accessible to standard Situation Calculus reasoning and planning systems. For future work, we want to study how the fragment our approach is currently able to handle can be extended, especially with regards to larger subsets of GOLOG that include iteration.

Acknowledgements

We gratefully acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada and thank the referees for their valuable comments.

References

Baader, F., and Zarrieß, B. 2013. Verification of Golog programs over description logic actions. In *FroCoS*, volume 8152 of *LNAI*. Springer.

Baier, J. A.; Fritz, C.; and McIlraith, S. A. 2007. Exploiting procedural domain control knowledge in state-of-the-art planners. In *ICAPS*, 26–33. AAAI Press.

Bartha, P. 1999. Moral preference, contrary-to-duty obligation and defeasible oughts. *Norms, logics and information systems: new studies in deontic logic and computer science* 93–108.

Broersen, J. M. 2004. Action negation and alternative reductions for dynamic deontic logics. *Journal of Applied Logic* 2(1):153–168.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* 35(8):677–691.

Burgard, W.; Cremers, A. B.; Fox, D.; Hähnel, D.; Lakemeyer, G.; Schulz, D.; Steiner, W.; and Thrun, S. 1999. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence* 114(1–2):3–55.

Chellas, B. F. 1980. *Modal Logic: An Introduction*. Cambridge University Press.

Chisholm, R. M. 1963. Contrary-to-duty imperatives and deontic logic. *Analysis* 24(2):33–36.

Chopra, A.; van der Torre, L.; Verhagen, H.; and Villata, S. 2018. *Handbook of Normative Multiagent Systems*. College Publications.

Delgrande, J. 2020. A preference-based approach to defeasible deontic inference. In *KR*. AAAI Press.

Demolombe, R., and del Pilar Pozos Parra, M. 2005. The Chisholm paradox and the situation calculus. In *ISMIS*, volume 3488 of *LNCS*, 425–434. Springer.

Demolombe, R., and del Pilar Pozos Parra, M. 2009. Integrating state constraints and obligations in situation calculus. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial* 13(41):54–63.

Dignum, F.; Meyer, J. C.; and Wieringa, R. J. 1994. A dynamic logic for reasoning about sub-ideal states. In *Proceedings of the ECAI 1994 Workshop on Artificial Normative Reasoning*, 79–92.

Ferrein, A., and Lakemeyer, G. 2008. Logic-based robot control in highly dynamic domains. *Robotics and Autonomous Systems*.

Forrester, J. W. 1984. Gentle murder, or the adverbial samaritan. *Journal of Philosophy* 81(4):193–197.

Gabbay, D.; Horty, J.; Parent, X.; van der Meyden, R.; and van der Torre, L. 2013. *Handbook of Deontic Logic and Normative Systems*. College Publications.

Goble, L. 2003. Preference semantics for deontic logic part I — simple models. *Logique et Analyse* 46(183/184):383–418.

Hansson, B. 1969. An analysis of some deontic logics. *Noûs* 3(4):373–398.

Horty, J. F. 2001. *Agency and Deontic Logic*. Oxford University Press.

Horty, J. F. 2003. Reasoning with moral conflicts. *Noûs* 37(4):557–605.

Horty, J. F. 2014. Deontic modals: Why abandon the classical semantics? *Pacific Philosophical Quarterly* 95(4):424–460.

Horty, J. 2019. Epistemic oughts in stit semantics. *Ergo – An Open Access Journal of Philosophy* 6:71–120.

Kern-Isberner, G., and Eichhorn, C. 2014. Structural inference from conditional knowledge bases. *Studia Logica* 102(4):751–769.

Kraus, S.; Lehmann, D. J.; and Magidor, M. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44(1-2):167–207.

Lehmann, D. J., and Magidor, M. 1992. What does a conditional knowledge base entail? *Artificial Intelligence* 55(1):1–60.

Levesque, H. J.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming* 31(1–3):59–83.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1–2):131–167.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*. New York: American Elsevier. 463–502.

Meyer, J. C. 1988. A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic* 29(1):109–136.

Meyer, J.-J. 2019. Deontic dynamic logic: a retrospective. *Filosofiska Notiser* 6(1):63–76.

Pearl, J., and Goldszmidt, M. 1990. On the relation between rational closure and System-Z. In *NMR*.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Röger, G.; Helmert, M.; and Nebel, B. 2008. On the relative expressiveness of ADL and Golog: The last piece in the puzzle. In *KR*, 544–550. AAAI Press.

Ross, W. D. 1930. *The Right and the Good*. Oxford University Press.

Segerberg, K. 2012. DΔl: A dynamic deontic logic. *Synthese* 185(Supplement-1):1–17.

Trypuz, R., and Kulicki, P. 2015. On deontic action logics based on boolean algebra. *Journal of Logic and Computation* 25(5):1241–1260.

van Benthem, J. 1979. Minimal deontic logics. *Bulletin of the Section of Logic* 8(1):36–40.

von Wright, G. H. 1951. Deontic logic. *Mind* 60(237):1–15.

von Wright, G. H. 1963. *Norm and action: a logical enquiry*. Routledge and Kegan Paul.

Wansing, H. 2004. On the negation of action types: Constructive concurrent PDL. In *Logic Methodology and Philosophy of Science: Proceedings of the Twelfth International Congress*, 207–225. College Publications.