On the Complexity of Input/Output Logic

Xin Sun*

The John Paul II Catholic University of Lublin, Poland.
Sun Yat-sen University, China
xin.sun.logic@gmail.com

Livio Robaldo†

University of Luxembourg, Luxembourg
livio.robaldo@uni.lu

Abstract

Input/output logic is a formalism in deontic logic and normative reasoning. Unlike deontic logical frameworks based on *possible-world semantics*, input/output logic adopts *norm-based* semantics in the sense of (Hansen, 2014), specifically *operational* semantics. It is well-known in theoretical computer science that complexity is an indispensable component of every logic. So far, previous literature in input/output systems focuses on proof theory and semantics, while neglects complexity. This paper adds the missing component by giving the complexity results of main decision problems in input/output logic. Our results show that input/output logic is `coNP` hard and in the 2nd level of the polynomial hierarchy.

Keywords: input/output logic, decidability, complexity, deontic logic, norm-based semantics

1 Introduction

Deontic logic is the logic of deontic modalities, such as obligation, permission, and prohibition. It has been used since the 1950s as a formal instrument for modeling normative reasoning (von Wright, 1968). Deontic logic has been studied in several research areas, including philosophy, linguistics, and computer science; for a recent survey, we address

*Xin Sun has been supported by the National Science Centre of Poland (BEETHOVEN, UMO-2014/15/G/HS1/04514).

†Livio Robaldo has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 661007 for the project "ProLeMAS: PROcessing LEgal language in normative Multi-Agent Systems".

the reader to (Gabbay et al., 2014). Many logical formalisms have been proposed to model deontic statements. This paper focuses on input/output logic, which has been originally introduced in (Makinson and van der Torre, 2000).

Input/output logic has been proposed with the aim of providing "a theoretical framework for the use of norms and normative reasoning in applications in computer science, law, linguistics, ethics, and other domains" ((Gabbay et al., 2014), pp. 501-502).

The key peculiarity of input/output is its *norm-based semantics*, in the sense of (Hansen, 2014). Specifically, as it will be explained below in section 2, input/output logic adopts a kind of norm-based semantics termed *operational* semantics: an input/output system is conceived as a deductive machine, like a black box which produces deontic statements as output, when we feed it factual statements as input.

Deontic frameworks grounded on norm-based semantics have been proposed as an alternative to standard deontic frameworks grounded on possible-world semantics, such as STIT logic (Horty, 2001) and dynamic deontic logic (Meyer, 1988; van der Meyden, 1996). Two main motivations are assumed to justify such an alternative semantics:

1. Philosophically, it is widely acknowledged that there is a distinction between norms on the one hand, and declarative statements on the other. Declarative statements may have truth-values, which means they are capable of being true or false; but norms are not. They may be complied or violated. But it makes no sense to describe norms as true or as false. Hence, it seems there cannot be a logic of norms: this is the well-known Jørgensen's dilemma (Jørgensen, 1937).

   In input/output logic conditional norms do not bear truth values. They are not embedded in compound formulae using truth-functional connectives. To keep clear of all confusion, norms are not even treated as formulae, but simply as ordered pairs (a, x) of logical formulae. if (a, x) is a *mandatory* norm, then it is read as "given a, x is obligatory". If it is a *permissive* norm, then it is read as "given a, x is permitted". Input/output logic aims at solving Jørgensen's dilemma at its starting line (cf. also (Makinson and van der Torre, 2003b)).

2. Norm-based semantics appears to offer a straightforward and simple way to deal with moral conflicts, i.e. situations when an agent ought to perform two or more actions but he cannot perform them all in that they conflict with each other (cf. (Horty, 2003)). For instance, suppose an agent both ought to make an expensive present to a friend and ought to save money to pay the rent. A standard way to deal with them in norm-based semantics is extending the basic formalism with priorities on norms in order to rank them according to certain preference criteria. In input/output logic, such a solution has been proposed in (Parent, 2011), which develops an extension of (Makinson and van der Torre, 2000)'s basic definitions with prioritized norms in order to handle moral conflicts.

It is well-known in theoretical computer science that complexity is an indispensable component of every logic. However, previous literature in input/output logic focuses on proof theory and semantics, while neglects complexity. This paper adds the missing component, thus completing the formal characterization of the logical framework.

This paper show that most decision problems of input/output logic are `NP` hard and in the 2nd level of the polynomial hierarchy. Our results show that although the complexity

of input/output logic is not low, it is not also astonishingly high. For example modal logic is at least as complex as input/output logic because it is PSPACE complete.

The structure of the paper is quite simple. In section 2, we will formally introduce the basic input/output systems to deal with obligations and permissions. In section 3 we will show our complexity results, on each system presented in section 2. In section 4, we discuss related work. Section 5 concludes this paper.

2 Input/output logic

Input/output logic takes its origin in the study of conditional norms and it has been originally introduced in (Makinson and van der Torre, 2000). Input/output logic is not a single logic but a family of logics, just like modal logic is a family of logics containing systems K, KD, S4, S5, etc. However, unlike modal logic, which usually uses *possible-world* semantics, input/output logic adopts *operational* semantics. An input/output system is conceived as a deductive machine, like a black box which produces deontic statements as output, when we feed it factual statements as input. Figure 1 is a brief visualization of input/output logic.

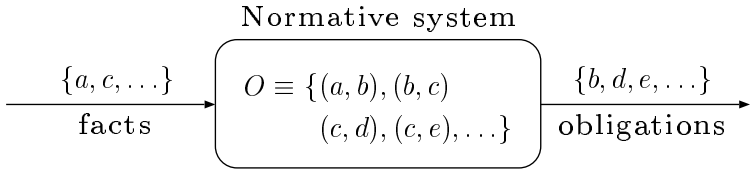

Figure 1: general input/output logic system

The set O represents the set of mandatory norms of a normative system. Formally, it is a set of pairs (x, y), where x and y are formulae in some *object logic*, e.g. standard propositional logic. Each pair (x, y) in O refers to a *mandatory* norm, in that it is read as "given x, y is obligatory". O corresponds to the deduction machine of the input/output system: whenever one of the formulae in the left-hand side elements of the pairs is given in input, the corresponding right-hand side elements are given in output.

As shown in the next section, in order to obtain a family of input/output logics, we add axioms to the basic deductive system exemplified in Fig.1, thus restricting/constraining the set of pairs belonging to O. But the pairs in O are never evaluated with respect to a model, i.e. associated with truth values. The deductive machine simply matches the input with the left-hand side elements of the pairs and returns the corresponding right-hand side elements. In that sense the semantics is *operational*.

This section presents input/output logic along three subsections. Subsection 2.1 introduces the eight unconstrained input/output logic used for modeling mandatory norms. Each system features a different (operational) semantics, given in terms of sets of formulae in the object logic associated with the inputs, which are again sets of formulae in the object logic. The systems are obtained by adding derivation rules to the general deductive machine depicted in Fig.1. The derivation rules are sound and complete with respect to the corresponding desired semantics. In subsection 2.2, we review constrained

input/output logic. In constrained input/output logic, the systems are equipped with the (non-monotonic) capability of determining which obligations are operative in a situation that already violates some of them and with a strategy that allows to solve conflicts, in case of multiple choices. Finally, subsection 2.3 extends unconstrained input/output logic systems fit to handle permissions.

2.1 Unconstrained input/output logic

Let $\mathbb{P} = \{p_0, p_1, \ldots\}$ be a countable set of propositional letters and $L_{\mathbb{P}}$ be the propositional language built upon $\mathbb{P}$. Let $O \subseteq L_{\mathbb{P}} \times L_{\mathbb{P}}$ be a set of ordered pairs of formulae of $L_{\mathbb{P}}$. A pair (a, x) in O refers to a mandatory norm, in that it is read as "given a, x is obligatory".

O can be viewed as a function from $2^{L_{\mathbb{P}}}$ to $2^{L_{\mathbb{P}}}$ such that for a set A of formulae, $O(A) = \{x \in L_{\mathbb{P}} : (a, x) \in O \text{ for some } a \in A\}$. Depending on the pairs included in O, a different output is produced against an input A. (Makinson and van der Torre, 2000) introduce four basic relevant outputs for an input A, called out_1, out_2, out_3, and out_4. These are functions taking O and A as arguments defined as follows:

(1)
- $out_1(O, A) = Cn(O(Cn(A)))$.
- $out_2(O, A) = \bigcap\{Cn(O(V)) : A \subseteq V, V \text{ is complete}\}$.
- $out_3(O, A) = \bigcap\{Cn(O(B)) : A \subseteq B = Cn(B) \supseteq O(B)\}$.
- $out_4(O, A) = \bigcap\{Cn(O(V) : A \subseteq V \supseteq O(V)), V \text{ is complete}\}$.

Here Cn is the classical consequence operator of propositional logic. That is $Cn(A) = \{a \in L_{\mathbb{P}} : A \vdash a\}$. A set of formulae is complete if it is either maximal consistent or equal to $L_{\mathbb{P}}$. These four operators are called 'simple-minded output', 'basic output', 'simple-minded reusable output' and 'basic reusable output' respectively.

For each of these four operators, a throughput version that allows inputs to reappear as outputs is defined as $out_i^+(O, A) = out_i(O_{id}, A)$, where $O_{id} = O \cup \{(a, a) : a \in L_{\mathbb{P}}\}$. When A is a singleton, we write $out_i(O, a)$ for $out_i(O, \{a\})$. Thus, we obtain eight basic input/output logic systems in total.

Input/output logics are given a proof theoretic characterization. We say that an ordered pair of formulae (a, x) is derivable from a set O iff (a, x) is in the least set that extends $O \cup \{(\top, \top)\}$ and is closed under a number of derivation rules (axioms). The following are the derivation rules we need for out_1 to out_4^+:

(2)
- `SI` (strengthening the input): from (a, x) to (b, x) whenever $a \in Cn(\{b\})$.
- `WO` (weakening the output): from (a, x) to (a, y) whenever $y \in Cn(\{x\})$.
- `AND` (conjunction of output): from (a, x) and (a, y) to $(a, x \wedge y)$.
- `OR` (disjunction of input): from (a, x) and (b, x) to $(a \vee b, x)$.
- `CT` (cumulative transitivity): from (a, x) and $(a \wedge x, y)$ to (a, y).
- `ID` (identity): from nothing to (a, a).

The derivation system based on the rules SI, WO and AND is called $deriv_1$. Adding OR to $deriv_1$ gives $deriv_2$. Adding CT to $deriv_1$ gives $deriv_3$. The five rules together give $deriv_4$. Adding ID to $deriv_i$ gives $deriv_i^+$ for $i \in \{1,2,3,4\}$. $(a,x) \in deriv_i(O)$ is used to denote that the norms (a,x) is derivable from $O \cup \{(\top,\top)\}$ using rules of $deriv_i$.

In (Makinson and van der Torre, 2000), the following soundness and completeness theorems are given, in order to prove that the semantics of the derivation systems generated from the axioms in (2) are exactly the ones in (1):

Theorem 2.1 *Given a set of mandatory norms O and a formula a,*

- $x \in out_i(O,a)$ *iff* $(a,x) \in deriv_i(O)$, *for* $i \in \{1,2,3,4\}$.
- $x \in out_i^+(O,a)$ *iff* $(a,x) \in deriv_i^+(O)$, *for* $i \in \{1,2,3,4\}$.

2.2 Constrained input/output logic

An important feature deontic frameworks must have is the ability of determining which obligations are detached in a situation that already violates some among them. In case an agent already violates a norm, we are still interested in knowing if it violates other norms, and so more penalties ought to be applied. In input/output logic, this is handled by introducing the definitions of $maxfamily$ and $outfamily$, which lead to the concept of *constrained* input/output logic (Makinson and van der Torre, 2001):

Definition 2.1 Given a set of mandatory norms O, a set of input $A \subseteq L_{\mathbb{P}}$ and a set of constraints $C \subseteq L_{\mathbb{P}}$, for $i \in \{1,\ldots,4,1^+,\ldots,4^+\}$:

- $maxfamily_i(O,A,C) = \{O' \subseteq O : out_i(O',A) \cup C$ is satisfiable, and $out_i(O'',A) \cup C$ is not satisfiable, for every $O' \subsetneq O''\}$.
- $outfamily_i(O,A,C) = \{out_i(O',A) : O' \in maxfamily_i(O,A,C)\}$

According to the definitions, $maxfamily_i(O,A,C)$ is the family of all maximal subsets O' of O such that $out_i(O',A)$ is consistent with C, while $outfamily_i(O,A,C)$ is the family of all such set $out_i(O',A)$. When $C{=}\emptyset$, $out_i(O',A)$ has to be *internally* consistent, i.e. it must not be possible to derive a contradiction ($\bot$) by conjoining the formulae in $out_i(O',A)$. When $C{=}A$, the output has to be consistent with the input, i.e. it must not be possible to derive $\bot$ by conjoining the formulae occurring either in $out_i(O',A)$ or in A (or in both). $maxfamily$ and $outfamily$ allow to straightforwardly overcome well-known limits of standard deontic logic, above all dealing with contrary-to-duty reasoning, i.e. reasoning about what to do in the face of violations of obligations.

To understand how $maxfamily$ and $outfamily$ work in practice, we report below an example taken from (Makinson and van der Torre, 2003b). Other examples, including the well-known Chisholm's and Forrester's paradoxes, may be found in (Makinson and van der Torre, 2001) and (Gabbay et al., 2014).

Suppose we have the following two norms: "The cottage should not have a fence or a dog" and "if it has a dog it must have both a fence and a warning sign." We may formalize these as:

$$O = \{(\top, \neg(f \vee d)), (d, f \wedge w)\}.$$

Suppose further that we are in the situation that the cottage has a dog, i.e. that our input context is $A = \{d\}$. In this context, the first norm is violated. Still we have to check if the cottage has both a fence and a warning sign. $maxfamily$ and $outfamily$ determine the obligations that are still in effect in the scenario. When $C{=}A$, it holds[1]:

- $maxfamily_i(O, A, \{d\}) = \{\{(d, f \wedge w)\}\}$
- $outfamily_i(O, A, \{d\}) = \{Cn(f \wedge w)\}$

Thus, $maxfamily_i(O, A, \{d\})$ tells us that the second obligation is still operative in the scenario although the first one has been already violated. This agrees with the intuitive assessment of the example: given the norms in O and $A = \{d\}$, our cottage must have both a fence and a warning sign.

Looking at the maximal subsets consistent with a set of constraints is a technique that is well-known in the more specific areas of belief change and non-monotonic reasoning. In the non-monotonic reasoning literature[2], it is likewise well-known that it is possible to identify two kinds of non-monotonic reasoning types: the credulous and the skeptical reasoning types, which allow to deal with non-resolvable conflicts in two different (opposite) ways. In input/output logic, a non-resolvable conflict corresponds to a scenario where $outfamily$ contains more than one set (and so $maxfamily$). The credulous and the skeptical reasoning types are obtained by joining and intersecting the sets in the $outfamily$ respectively, via two additional operators termed "full-join output" ($out_i^{\cup}$) and "full-meet output" ($out_i^{\cap}$):

Definition 2.2 Given a set of mandatory norms O, a set of input $A \subseteq L_{\mathbb{P}}$ and a set of constraints $C \subseteq L_{\mathbb{P}}$, for $i \in \{1, \ldots, 4, 1^+, \ldots, 4^+\}$.

- $out_i^{\cup}(O, A, C) = \bigcup outfamily_i(O, A, C)$ [credulous]
- $out_i^{\cap}(O, A, C) = \bigcap outfamily_i(O, A, C)$ [skeptical]

To understand how $out_i^{\cup}(O, A, C)$ and $out_i^{\cap}(O, A, C)$ work in practice, we report below an example taken from (Makinson and van der Torre, 2001).

Consider the obligations "if Alice is invited to dinner then Bob must be invited too", "if Bob is invited to dinner then Carol must be invited too", and "if Carol is invited to dinner then Alice must be *not* invited". These three obligations are formalized as: $O = \{(a, b), (b, c), (c, \neg a)\}$. Suppose further that Alice is invited to dinner, i.e. $A = \{a\}$. In the input/output system out_3 it holds:

- $maxfamily_3(O, A, A) = \{\{(a, b), (b, c)\}, \{(a, b), (c, \neg a)\}, \{(b, c), (c, \neg a)\}\}$
- $outfamily_3(O, A, A) = \{Cn(b, c), Cn(b), Cn(\emptyset)\}$
- $out_3^{\cup}(O, A, A) = \bigcup outfamily_3(O, A, A) = Cn(b, c)$

[1] On the other hand, if $C{=}\emptyset$, $maxfamily_i(O, A, \emptyset) = \{\{(\top, \neg(f \vee d))\}, \{(d, f \wedge w)\}\}$ and $outfamily_i(O, A, \emptyset) = \{\{Cn(\neg(f \vee d))\}, \{Cn(f \wedge w)\}\}$.

[2] Cf. http://plato.stanford.edu/entries/logic-nonmonotonic.

- $out_3^{\bigcap}(O, A, A) = \bigcap outfamily_3(O, A, A) = Cn(\emptyset)$

In this scenario, under the credulous inference we are still obliged to invite Bob and Carol. On the other hand, under the skeptical inference we are not obliged to invite anyone else.

2.3 Permissive input/output logic

The definitions presented in subsection 2.1 only allow to deal with obligations. On the other hand, in input/output logic also permissions have been studied in depth. In order to handle permissions in input/output systems, we simply define another set $P \subseteq L_{\mathbb{P}} \times L_{\mathbb{P}}$ of ordered pairs of formulae of $L_{\mathbb{P}}$. A pair $(a, x) \in P$, call it a permissive norm, is read as "given a, x is permitted". $N = (O, P)$, where O is a set of mandatory norms and P a set of permissive norms, is called a normative system.

It is common in philosophy to distinguish between two kinds of permissions: negative permission and positive permission. Something is negatively permitted if and only if it is not prohibited by the norms, i.e., if and only if it is *not* explicitly asserted that it is not permitted. On the other hand, something is positively permitted if and only if it can be derived from the norms. Negative permission is straightforward to describe while positive permission is more elusive. In what sense something may be derived from the norms? (Makinson and van der Torre, 2003a) introduce two types of positive permission: static and dynamic permission. Here is an example from (Makinson and van der Torre, 2003a) to distinguish these two kinds of positive permission: assume the normative system includes the obligation "A man is obliged to pay tax on condition of having salary." and the permission "A man is permitted to vote on condition of being older than 18.". Now the question is, according to the given normative system:

Is a man permitted to vote on the condition of having salary?

In one sense the answer is 'no'. If we stick to the straight derivation, a man's having salary does not imply that he is older than 18. Therefore we cannot derive he is permitted to vote. This is one notion of positive permission, the static permission. In another sense the answer is 'yes'. The reason is: suppose we add "A man is not permitted to vote on the condition of having salary" to the normative system. We will make the normative system incoherent in the sense that when the normative system is applied to a man who has salary and is older than 18, then he is permitted to vote meanwhile not permitted to vote. This is another notion of positive permission, the dynamic permission.

Formal definitions of the three type of permission are introduced in (Makinson and van der Torre, 2003a).

Definition 2.3 (negative permission) Given a normative system $N = (O, P)$ and a set of formulae A:

$$NegPerm_i(N, A) = \{x \in L_{\mathbb{P}} : \neg x \notin out_i(O, A)\}.$$

Intuitively, x is negatively permitted iff x is not forbidden. Since a formula is forbidden iff its negation is obligatory, x is not forbidden is equivalent to $\neg x$ is not obligatory.

Definition 2.4 (positive-static permission) Given a normative system $N = (O, P)$ and a set of formulae A,

- If $P \neq \emptyset$, then $StaPerm_i(N, A) = \{x \in L_{\mathbb{P}} : x \in out_i(O \cup \{(a', x')\}, A)$, for some $(a', x') \in P\}$.
- If $P = \emptyset$, then $StaPerm_i(N, A) = out_i(O, A)$.

Intuitively, positive-static permissive norms are treated like weak mandatory norms, the basic difference is that while the latter may be used jointly, the former may only be applied one by one.

Definition 2.5 (positive-dynamic permission) Given a normative system $N = (O, P)$ and a finite set of formulae A:

$$x \in DyPerm_i(N, A) \text{ iff there is a consistent set of formulae } C \text{ such that}$$
$$StaPerm_i(N, C) \cup out_i(O \cup \{(\bigwedge A, \neg x)\}, C) \text{ is inconsistent.}$$

Intuitively, x is dynamically permitted given condition A iff the prohibition of x under condition A, that is, taking $(\bigwedge A, \neg\, x)$ as a mandatory norm, will create inconsistency of the normative system with respect to some consistent input C.

In order to clarify the difference between static and dynamic permission, let's consider again the example above. In that scenario, two norms hold: "a man is obliged to pay tax on condition of having salary" and "a man is permitted to vote on condition of being older than 18". We want to know if a man is permitted to vote on condition of having salary. The normative system is formalized as follows:

$$O = \{(s,\, t)\} \qquad P = \{(> 18,\, v)\} \qquad A{=}\{s\}$$

In this normative system, a man is *not* statically permitted to vote, in that v does not belong to $StaPerm_i(N, A)$, for all $i = \{1, 2, 3, 4\}$:

$$StaPerm_i(N, A) = \{x \in L_{\mathbb{P}} : x \in out_i(\{(s,\, t)\} \cup \{(> 18,\, v)\},\, \{s\}) = \{t\}$$

On the other hand, a man is dynamically permitted to vote. In line with the example, let's consider a man who has salary and is older than 18, i.e., let's consider a consistent set $C{=}\{s, > 18\}$. For all $i = \{1, 2, 3, 4\}$, we obtain:

$$StaPerm_i(N,\, \{s, > 18\}) \cup out_i(\{(s,\, t)\} \cup \{(s, \neg v)\},\, \{s, > 18\}\ ) =$$
$$\{t, v\} \cup \{t, \neg v\} = \{t, v, t, \neg v\}$$

Since $\{t, v, t, \neg v\}$ is inconsistent, $v \in DyPerm_i(N, A)$, i.e., a man is permitted to vote.

3 On the complexity of input/output logic

The present section, after a rapid excursion on the background ingredients we are going to use in our proofs, in subsection 3.1, presents theorems on the complexity of the basic computational problems in the input/output logic systems illustrated in the previous section. To enhance comprehension, we analyze (the complexity of) the input/output logic systems in the same order and with the same structure into subsections.

3.1 Background - complexity theory

We assume the readers are familiar with notions like Turing machine and the complexity class P, NP and coNP. More comprehensive introduction of complexity theory can be found in (Arora and Barak, 2009). The following complexity results of a fragment of modal logic will be used in this paper.

Theorem 3.1 *(Halpern, 1995)*

1. *The satisfiability problem of modal logic K of model depth 1 is NP-complete.*
2. *The satisfiability problem of modal logic T of model depth 1 is NP-complete.*

The boolean hierarchy is the hierarchy of boolean combinations (intersection, union and complementation) of NP classes. BH_1 is the same as NP. BH_2 is the class of languages which are the intersection of a language in NP and a language in coNP. (Wagner, 1986) shows that the following 2-parity SAT problem is complete for BH_2:

Given two propositional formulae x_1 and x_2 such that if x_2 is satisfiable then x_1 is satisfiable, is it true that x_1 is satisfiable while x_2 is not?

Oracle Turing machine and two complexity classes related to oracle Turing machine will be used in this paper.

Definition 3.1 (oracle Turing machine (Arora and Barak, 2009)) An oracle for a language L is a device that is capable of reporting whether any string w is a member of L. An oracle Turing machine M^L is a modified Turing machine that has the additional capability of querying an oracle. Whenever M^L writes a string on a special oracle tape it is informed whether that string is a member of L, in a single computation step.

$\mathtt{P}^{\mathtt{NP}}$ is the class of problems solvable by a deterministic polynomial time Turing machine with an NP oracle. $\mathtt{NP}^{\mathtt{NP}}$ is the class of problems solvable by a non-deterministic polynomial time Turing machine with an NP oracle.

$\mathtt{NP}^{\mathtt{NP}}$ and $\mathtt{coNP}^{\mathtt{NP}}$ are also termed as Σ_2^p and Π_2^p respectively, to mark the fact that $\mathtt{NP}^{\mathtt{NP}}$ and $\mathtt{coNP}^{\mathtt{NP}}$ belong to the *2nd* level of the polynomial hierarchy. However, for the sake of uniform terminology, in what follows we will continue to use the symbols $\mathtt{NP}^{\mathtt{NP}}$ and $\mathtt{coNP}^{\mathtt{NP}}$ to refer to the two classes of problems.

3.2 Complexity of unconstrained input/output logic

The complexity of input/output logic has been sparsely studied in the past. Although the reversibility of derivation rules as a proof re-writing mechanism has been studied for the input/output logic framework (Makinson and van der Torre, 2000), the length or complexity of such proofs have not been developed. We approach the complexity of unconstrained input/output logic from a semantic point of view. We focus on the following *fulfillment problem*:

Given a finite set of mandatory norms O, a finite set of formulae A and a formula x, is $x \in out(O, A)$?

3.2.1 Complexity of out_1 and out_1^+

We start with the complexity of out_1. The following theorem shows that out_1 has the same complexity as propositional logic in the sense that both the fulfillment problem of out_1 and the validity problem of propositional logic is coNP-complete.

Theorem 3.2 *The fulfillment problem of simple-minded input/output logic out_1 is coNP-complete.*

Proof Concerning the coNP hardness, we prove by reducing the validity problem of propositional logic to the fulfillment problem of simple- minded input/output logic: given an arbitrary $x \in L_{\mathbb{P}}$, $\vdash x$ iff $x \in Cn(\top)$ iff $x \in Cn(O(Cn(A)))$ where $O = \emptyset$ iff $x \in out_1(O, A)$ where $O = \emptyset$.

Now we prove the coNP membership. We provide the following non-deterministic Turing machine to solve the complement of our problem. Let $O = \{(a_1, x_1), \ldots, (a_n, x_n)\}$, A be a finite set of formulae and x be a formula.

1. Guess a sequence of valuations $V_1, \ldots, V_n$ and V' on the propositional letters appearing in $A \cup \{a_1, \ldots, a_n\} \cup \{x_1, \ldots, x_n\} \cup \{x\}$.
2. Let $O' \subseteq O$ be the set of norms which contains all (a_i, x_i) such that $V_i(A) = 1$ and $V_i(a_i) = 0$.
3. Let $X = \{x : (a, x) \in O - O'\}$.
4. If $V'(X) = 1$ and $V'(x) = 0$. Then return "accept" on this branch. Otherwise return "reject" on this branch.

The main intuition of the proof is: O' collects all norms which *cannot* be triggered[3] by A. On some branches we must have that O' contains exactly those norms which are not triggered by A. In those lucky branches X is the same as $O(Cn(A))$. If there is a valuation V' such that $V'(X) = 1$ and $V'(x) = 0$, then we know $x \notin Cn(X) = Cn(O(Cn(A)))$.

It can be verified that $x \notin Cn(O(Cn(A)))$ iff the algorithm returns "accept" on some branches and the time complexity of the Turing machine is polynomial. ⊣

Now we move the the complexity of out_1^+. The following lemma will be used.

Lemma 3.3 $out_1^+(O, A) = Cn(A \cup O(Cn(A)))$.

Proof Assume $x \in out_1^+(O, A) = out_1(O_{id}, A)$. Then $x \in Cn(O_{id}(Cn(A))) = Cn((O \cup \{(a, a) : a \in L_{\mathbb{P}}\})(Cn(A))) = Cn(O(Cn(A)) \cup (\{(a, a) : a \in L_{\mathbb{P}}\})(Cn(A))) = Cn(O(Cn(A)) \cup Cn(A))$. Therefore there are some $x_1, \ldots, x_n \in O(Cn(A)) \cup Cn(A)$ such that $x_1 \wedge \ldots \wedge x_n \vdash x$. Without loss of generality, assume $x_1, \ldots, x_{n-1} \in O(Cn(A))$ and $x_n \in Cn(A)$. Then there are some $y_1, \ldots, y_m$ such that $y_1 \wedge \ldots \wedge y_m \vdash x_n$. Then we know $x_1, \ldots, x_{n-1}, y_1, \ldots, y_m \in O(Cn(A)) \cup A$ and $x_1 \wedge \ldots, x_{n-1} \wedge y_1 \wedge \ldots \wedge y_m \vdash x$. Therefore $x \in Cn(A \cup O(Cn(A)))$.

Assume $x \in Cn(A \cup O(Cn(A)))$, then $x \in Cn(Cn(A) \cup O(Cn(A))) = Cn(O(Cn(A)) \cup (\{(a, a) : a \in L_{\mathbb{P}}\})(Cn(A))) = Cn(O_{id}(Cn(A))) = out_1^+(O, A)$. ⊣

[3] We say a norm (a, x) is triggered by A if $a \in Cn(A)$).

Theorem 3.4 *The fulfillment problem of simple-minded output throughput input/output logic out_1^+ is* coNP*-complete.*

Proof Concerning the lower bound, we prove by a reduction from the validity problem of propositional logic: given arbitrary $x \in L_{\mathbb{P}}$, $\vdash x$ iff $x \in Cn(\top)$ iff $x \in Cn(A \cup O(Cn(A)))$ where $O = \emptyset = A$ iff $x \in out_1^+(O, A)$ where $O = \emptyset = A$.

Concerning the upper bound, we prove by giving a non-deterministic Turing machine similar to the one in the proof of Theorem 3.2. The only change is now in step 4 we test if $V'(A \cup X) = 1$ and $V'(x) = 0$. It can be verified that $x \notin Cn(A \cup O(Cn(A)))$ iff the non-deterministic Turing machine returns "accept" on some branch. By Lemma 3.3 we know this Turing machine solves our problem. ⊣

3.2.2 Complexity of out_2 and out_2^+

Let $s(O) = \{x : (a, x) \in O\}$ be the projection of O to the second component of its consisting norms. Let $O^{\Box} = \{a \to \Box x : (a, x) \in O\}$. Here $\Box$ is the necessity modality of modal logic K. Let $\vdash_K$ be the consequence relation of modal logic K. The following theorem reveals the relation between basic input/output logic and modal logic, and is useful in the complexity proof.

Theorem 3.5 *(Makinson and van der Torre, 2000)*

$$x \in out_2(O, A) \text{ iff } x \in Cn(s(O)) \text{ and } O^{\Box} \cup A \vdash_K \Box x.$$

Theorem 3.6 *The fulfillment problem of basic input/output logic out_2 is* coNP*-complete.*

Proof Concerning the lower bound, we prove by a reduction from the validity problem of propositional logic: given arbitrary $x \in L_{\mathbb{P}}$, $\vdash x$ iff $x \in Cn(\emptyset)$ iff $x \in out_2(O, A)$ where $O = \emptyset$.

Concerning the upper bound, we prove by polynomially reducing the fulfillment problem of basic input/output logic to the validity problem of modal logic K with modal depth 1. Theorem 3.5 gives us the key idea of the reduction. By Theorem 3.5, $x \in out_2(O, A)$ iff $\vdash_K (\bigwedge s(O) \to x) \wedge ((\bigwedge O^{\Box} \wedge A) \to \Box x)$. Then by Theorem 3.1 we know the upper bound is coNP. ⊣

To study the complexity of out_2^+, we make use of the materialisation of norms introduced by (Makinson and van der Torre, 2000). Let $m(O) = \{a \to x : (a, x) \in O\}$ be the materialisation of O. That is, the operator $m(\bullet)$ transforms a norms (a, x) into a classical implication $a \to x$. The following theorem shows that input/output logic can be reduced to propositional logic via the materialisation of norms.

Theorem 3.7 *(Makinson and van der Torre, 2000)*

$$out_2^+(O, A) = out_4^+(O, A) = Cn(A \cup m(O)).$$

Theorem 3.8

1. *The fulfillment problem of basic throughput input/output logic out_2^+ is* coNP*-complete.*

2. *The fulfillment problem of basic reusable throughput input/output logic* out_4^+ *is* coNP*-complete.*

Proof Given Theorem 3.7, the proof is routine: $x \in out_2^+(O, A)$ iff $x \in Cn(A \cup m(O))$ iff $A \cup m(O) \vdash x$ iff $\vdash \bigwedge(A \cup m(O)) \rightarrow x$. ⊣

3.2.3 Complexity of out_3 and out_3^+

To study the complexity of out_3, we make use of a fixed-point characterization of out_3 developed by Sun (2014). Given a set O of mandatory norms and a set A of formulae, we define a function $f_A^O : 2^{L_{\mathbb{P}}} \rightarrow 2^{L_{\mathbb{P}}}$ such that $f_A^O(X) = Cn(A \cup O(X))$. It can be proved that f_A^O is monotonic with respect to the set theoretical $\subseteq$ relation, and $(2^{L_{\mathbb{P}}}, \subseteq)$ is a complete lattice. Then by Tarski's fixed point theorem there exists a least fixed point of f_A^O. The following proposition shows that the least fixed point can be constructed in an inductive manner.

Proposition 3.9 *(Sun, 2014) Let* B_A^O *be the least fixed point of the function* f_A^O*. Then* $B_A^O = \bigcup_{i=0}^{\infty} B_{A,i}^O$*, where* $B_{A,0}^O = Cn(A), B_{A,i+1}^O = Cn(A \cup O(B_{A,i}^O))$*.*

Using the least fixed point, a more constructive semantics of out_3 and out_3^+ are stated as follows, such semantics gives us insights to develop algorithms to solve the fulfillment problem of reusable input/output logic:

Theorem 3.10 *(Sun, 2014) For a set of norms* O *and a formula* a*,*

1. $(a, x) \in deriv_3(O)$ *iff* $x \in Cn(O(B_{\{a\}}^O))$*.*
2. $(a, x) \in deriv_3^+(O)$ *iff* $x \in Cn(O_{id}(B_{\{a\}}^{O_{id}}))$*.*

Theorem 3.11 *The fulfillment problem of simple-minded reusable input/output logic* out_3 *is between* coNP *and* $\mathtt{P}^{\mathtt{NP}}$*.*

Proof The lower bound can be proved using the same reduction as in the the proof of the lower bound of out_1. Concerning the upper bound, we provide the following algorithm on an oracle Turing machine with oracle SAT.

Let $O = \{(a_1, x_1), \ldots, (a_n, x_n)\}$, A be a finite set of formulae and x be a formula.

1. Let $X = A, Y = Z = O$, $U = \emptyset$.
2. for each $(a_i, x_i) \in Y$, ask the oracle if $\neg(\bigwedge X \rightarrow a_i)$ is satisfiable.
   (a) If "no", then let $X = X \cup \{x_i\}$, $Z = Z - \{(a_i, x_i)\}$.
   (b) Otherwise do nothing.
3. If $Y = Z$, goto 4. Otherwise let $Y = Z$, goto step 2.
4. for each $(a_i, x_i) \in O$, ask the oracle if $\neg(\bigwedge X \rightarrow a_i)$ is satisfiable.

(a) If "no", then let $U = U \cup \{x_i\}$.

(b) Otherwise do nothing

5. Ask the oracle if $\neg(\bigwedge U \to x)$ is satisfiable.

   (a) If "no", then return "accept".

   (b) Otherwise return "reject".

Theorem 3.10 is useful to prove the correctness of the above algorithm. Here we just state some crucial points: from step 1 to step 3, the algorithm generates X as the least fixed point of f_A^O. In step 4, U is generated, which should be understood as $O(X)$.

Concerning the time complexity, the times of loop in step 2 is at most n. Each loop can be finished in polynomial time. Therefore all the loops in step 2 can be done in polynomial time. Step 3 calls for step 2 for at most n times. Therefore it can still be done in polynomial time. The times of loop in step 4 is exactly n. Each loop can be finished in polynomial time. Therefore all the loops in step 4 can be done in polynomial time. Step 5 can be done in polynomial time. Therefore the algorithm is polynomial. $\dashv$

We use the following examples to illustrate how the above algorithm works.

Example 1 *Let p, q, r, s and t be propositional letters. Let $O = \{(p, q), (q, r), (p \wedge r, s), (t, t)\}$, $A = \{p\}$, $t \to s \in out_3(O, A)$ is computed as follows:*

1. *Let $X = A = \{p\}$, $Y = Z = O = \{(p, q), (q, r), (p \wedge r, s), (t, t)\}$, $U = \emptyset$.*
2. *Compute whether $\{p\} \vdash p$, $\{p\} \vdash q$, $\{p\} \vdash p \wedge r$, $\{p\} \vdash t$.*
3. *Let $X = \{p, q\}$, $Z = \{(q, r), (p \wedge r, s), (t, t)\}$.*
4. *Compare if $Y = Z$. The result is negative. So let $Y = \{(q, r), (p \wedge r, s), (t, t)\}$*
5. *Compute whether $\{p, q\} \vdash q$, $\{p, q\} \vdash p \wedge r$, $\{p\} \vdash t$.*
6. *Let $X = \{p, q, r\}$. $Z = \{(p \wedge r, s), (t, t)\}$.*
7. *Compare if $Y = Z$. The result is negative. So let $Y = \{(p \wedge r, s), (t, t)\}$.*
8. *Compute whether $\{p, q, r\} \vdash p \wedge r$, $\{p\} \vdash t$.*
9. *Let $X = \{p, q, r, s\}$. $Z = \{(t, t)\}$.*
10. *Compare if $Y = Z$. The result is negative. So let $Y = \{(t, t)\}$.*
11. *Compute whether $\{p, q, r, x\} \vdash t$.*
12. *Let $X = \{p, q, r, s\}$. $Z = \{(t, t)\}$.*
13. *Compare if $Y = Z$. The result is positive.*
14. *Compute whether $\{p, q, r, s\} \vdash p$, $\{p, q, r, s\} \vdash q$, $\{p, q, r, s\} \vdash p \wedge r$, $\{p, q, r, s\} \vdash t$.*
15. *Let $U = \{q, r, s\}$.*
16. *Compute whether $\{q, r, s\} \vdash t \to s$. The answer is positive, so we conclude $t \to s \in out_3(O, A)$.*

Theorem 3.12 *The fulfillment problem of simple-minded reusable throughput input/output logic out_3^+ is between* coNP *and* $\mathtt{P}^{\mathtt{NP}}$.

Proof The lower bound can be proved using the same reduction as in the the proof of the lower bound of out_1. Concerning the upper bound, we prove by giving an algorithm similar to the one in the proof of Theorem 3.11. We make the following change:

- In step 2 and 4 we ask the oracle if $\neg(\bigwedge A \wedge \bigwedge X \rightarrow a_i)$ is satisfiable.
- In step 5 we ask the oracle if $\neg(\bigwedge A \wedge \bigwedge U \rightarrow a_i)$ is satisfiable. ⊣

Example 2 *Let p, q, r and s be propositional letters. Let $O = \{(p, q \wedge r), (p \wedge r, s)\}$, $A = \{p\}$, $\neg p \wedge s \in out_3^+(O, A)$ is computed as follows:*

1. *Let $X = A = \{p\}$, $Y = Z = O = \{(p, q \wedge r), (p \wedge r, s)\}$, $U = \emptyset$.*
2. *Compute whether $\{p\} \vdash p$, $\{p\} \vdash p \wedge r$.*
3. *Let $X = \{p, q \wedge r\}$, $Z = \{(p \wedge r, s)\}$.*
4. *Compare if $Y = Z$. The result is negative. So let $Y = \{(p \wedge r, s)\}$.*
5. *Compute whether $\{p, q \wedge r\} \vdash p \wedge r$.*
6. *Let $X = \{p, q \wedge r, s\}$. $Z = \emptyset$.*
7. *Compare if $Y = Z$. The result is negative. So let $Y = \emptyset$.*
8. *Compute whether $\{p, q \wedge r, s\} \vdash p$, $\{p, q \wedge r, s\} \vdash p \wedge r$*
9. *Let $U = \{r, s\}$.*
10. *Compute whether $A \cup U = \{p, r, s\} \vdash \neg p \wedge s$. The answer is negative, so we conclude $\neg p \wedge s \notin out_3^3(O, A)$.*

3.2.4 Complexity of out_4

Similar to out_2, out_4 can also be translated to modal logic.

Theorem 3.13 *(Makinson and van der Torre, 2000) $x \in out_4(O, A)$ iff $x \in Cn(s(O))$ and $O^{\Box} \cup A \vdash_T \Box x$. Here $\vdash_T$ is the consequence relation of modal logic T.*

Theorem 3.14 *The fulfillment problem of basic reusable input/output logic out_4 is* coNP*-complete.*

Proof The lower bound can be proved using the same reduction as in the the proof of the lower bound of out_1. Concerning the upper bound, we prove by polynomially reducing the fulfillment problem of basic reusable input/output logic to the validity problem of modal logic T with modal depth 1. Theorem 3.13 gives us the key idea of the reduction. By Theorem 3.13, $x \in out_4(O, A)$ iff $\vdash_T (\bigwedge s(O) \rightarrow x) \wedge ((\bigwedge O^{\Box} \wedge A) \rightarrow \Box x)$. Then by Theorem 3.1 we know the upper bound is coNP. ⊣

3.3 Complexity of constrained input/output logic

In the constrained setting, a finite set of mandatory norms O and a subset $O' \subseteq O$, a finite set of input A and a finite set of constrains C are given. We study the complexity of the following problems:

- consistency checking: is $out_i(O, A)$ consistent with C?
- maxfamily membership: is $O' \in maxfamily_i(O, A, C)$?
- full-join fulfillment: is $x \in out_i^{\cup}(O, A, C)$?
- full-meet fulfillment: is $x \in out_i^{\cap}(O, A, C)$?

Theorem 3.15

- *For $i \in \{1, 2, 4, 1^+, 2^+, 4^+\}$, the consistency checking problem is* NP *complete.*
- *For $i \in \{3, 3^+\}$, the consistency checking problem is* NP *hard and in* $\mathtt{P}^{\mathtt{NP}}$.

Proof The NP hard problem SAT can be reduced to the consistency checking problem, which gives us the result of the lower bound. The consistency checking problem can be solved by a reduction to the complement of the fulfillment problem, which gives us the result of the upper bound: $out_i(O, A)$ is consistent with C iff $C \cup out_i(O, A)$ is satisfiable iff $out_i(O, A) \nvdash \neg \bigwedge C$ iff $\neg \bigwedge C \notin out_i(O, A)$. ⊣

We show now that the maxfamily membership problem is BH_2 complete, where BH_2 is the class of languages which are the intersection of a language in NP and a language in coNP (cf. section 3.1 above).

Theorem 3.16 *For $i \in \{1, 2, 4, 1^+, 2^+, 4^+\}$, the maxfamily membership problem is BH_2 complete.*

Proof The BH_2 hardness can be proved by a reduction from the 2-Parity SAT problem. Suppose we are given two formulae x_1 and x_2 such that (if x_2 is satisfiable, then x_1 is satisfiable). Our aim is to answer the question: is it true that (x_1 is satisfiable but x_2 is not satisfiable)?

Let $O = \{(\top, x_1 \vee x_2), (\top, x_2)\}$, $A = C = \emptyset$, $O' = \{(\top, x_1 \vee x_2)\}$.

- If $O' \in maxfamily_i(O, A, C)$ then $out_i(O', A) = Cn(x_1 \vee x_2)$ is consistent and $out_i(O, A) = Cn((x_1 \vee x_2) \wedge x_2)$ is inconsistent. Therefore $x_1 \vee x_2$ is satisfiable and $(x_1 \vee x_2) \wedge x_2$ is not satisfiable. Then we know x_2 is not satisfiable. It then follows that x_1 is satisfiable because otherwise $x_1 \vee x_2$ is not satisfiable.
- If x_1 is satisfiable and x_2 is not satisfiable, then $x_1 \vee x_2$ is satisfiable. Therefore $out_i(O', A)$ is consistent and $out_i(O, A)$ is inconsistent. It then follows that $O' \in maxfamily_i(O, A, C)$.

So we have proved x_1 is satisfiable and x_2 is not satisfiable iff $O' \in maxfamily_i(O, A, C)$, which proves the BH_2 hardness.

Concerning the BH_2 membership, let $O = O' \cup \{(a_1, x_1), \ldots, (a_n, x_n)\}$. Then $O' \in maxfamily_i(O, A, C)$ iff C is consistent with $out_i(O', A)$ but not consistent with $out_i(O' \cup \{(a_j, x_j)\}, A)$ for every $j \in \{1, \ldots, n\}$. Since deciding if C is consistent with $out_i(O', A)$ is in NP and deciding if C is not consistent with $out_i(O' \cup \{(a_j, x_j)\}, A)$ is in coNP, we know that deciding if $O' \in maxfamily_i(O, A, C)$ is in BH_2. $\dashv$

Theorem 3.17 *For $i \in \{3, 3^+\}$, the maxfamily membership problem is BH_2 hard and in $\mathtt{P}^{\mathtt{NP}}$.*

Proof The BH_2 hardness can be proved just like other input/output logics.

Concerning the upper bound, let $O = O' \cup \{(a_1,\ x_1), \ldots, (a_n, x_n)\}$. Then $O' \in maxfamily(O, A, C)$ iff C is consistent with $out_i(O', A)$ but not consistent with $out_i(O' \cup \{(a_j, x_j)\}, A)$ for every $j \in \{1, \ldots, n\}$. Since deciding if C is consistent with $out_i(O', A)$ is in $\mathtt{P}^{\mathtt{NP}}$ and deciding if C is not consistent with $out_i(O' \cup \{(a_j, x_j)\}, A)$ is also in $\mathtt{P}^{\mathtt{NP}}$, we know that deciding if $O' \in maxfamily_i(O, A, C)$ is in $\mathtt{P}^{\mathtt{NP}}$. $\dashv$

Theorem 3.18 *For $i \in \{1, 2, 3, 4, 1^+, 2^+, 3^+, 4^+\}$, the full-join fulfillment problem is $\mathtt{NP}^{\mathtt{NP}}$ complete.*

Proof Concerning the $\mathtt{NP}^{\mathtt{NP}}$ membership, we prove by giving the following algorithm on a non-deterministic Turing machine with an $\mathtt{NP}$ oracle to solve our problem.

1. Guess a subset $O' \subseteq O$.
2. Use the NP oracle to test if $O' \in maxfamily_i(O, A, C)$. If no, return "reject" on this branch. Otherwise continue.
3. Use the NP oracle to test if $x \in out_i(O', A)$. If $x \in out_i(O', A)$, then return "accept" on this branch. Otherwise return "reject" on this branch.

It can be verified that $x \in out^{\cup}(O, A, C)$ iff the non-deterministic Turing machine returns "accept" on some branches. Step 2 can be done in polynomial time steps because the maxfamily membership problem is in $\mathtt{P}^{\mathtt{NP}}$. Step 3 can also be done in polynomial time steps because the fulfillment problem is also in $\mathtt{P}^{\mathtt{NP}}$. Therefore the time complexity of this non- deterministic Turing machine is polynomial.

Concerning the $\mathtt{NP}^{\mathtt{NP}}$ hardness, we show that the validity problem of 2-QBF$^{\exists}$ can be reduced to the full-join fulfillment problem.

Let $\exists p_1 \ldots p_m \forall q_1 \ldots q_n \Phi$ be a 2-QBF$^{\exists}$ where Φ is a propositional formula with variables in $\{p_1, \ldots, p_m, q_1, \ldots, q_n\}$. Let $A = C = \emptyset$, $O = \{(\top, p_1), \ldots, (\top, p_m), (\top, \neg p_1), \ldots, (\top, \neg p_m), (\top, \neg\Phi)\}$. Our aim is to show that this 2-QBF$^{\exists}$ is valid iff $\Phi \in out_i^{\cup}(O, A, C)$.

- If $\exists p_1 \ldots p_m \forall q_1 \ldots q_n \Phi$ is valid, then there is a valuation V for $\{p_1, \ldots, p_m\}$ such that for all valuations V' for $\{q_1, \ldots, q_n\}$, $V \cup V'$ gives truth value 1 to Φ and 0 to $\neg\Phi$. Let $O' = \{(\top, p'_1), \ldots, (\top, p'_m)\}$, where each p'_i is p_i if $p_i \in V$ and it is $\neg p_i$ if $p_i \notin V$. Then $O' \in maxfamily_i(O, A, C)$ because $out_i(O', A) = Cn(\{p'_1, \ldots, p'_m\})$ is consistent with C and adding anything from $\{(\top, \neg p_1), \ldots, (\top, \neg p_m), (\top, \neg\Phi)\}$

to O' will destroy the consistency. Note that $\Phi \in Cn(\{p'_1, \ldots, p'_m\})$ by the construction of $\{p'_1, \ldots, p'_m\}$. Therefore $\Phi \in out_i(O', A)$, which further implies that $\Phi \in out_i^{\cup}(O, A, C)$.

- If $\exists p_1 \ldots p_m \forall q_1 \ldots q_n \Phi$ is not valid, then for all valuations V for $\{p_1, \ldots, p_m\}$ there is a valuation V' for $\{q_1, \ldots, q_n\}$ such that $V \cup V'$ gives truth value 0 to Φ and 1 to $\neg\Phi$. Let $O' = \{(\top, p'_1), \ldots, (\top, p'_m), (\top, \neg\Phi)\}$ be an arbitrary set such that each p'_i is either p_i or $\neg p_i$. Then $out_i(O', A) = Cn(\{p'_1 \ldots, p'_m, \neg\Phi\})$, which is consistent. Moreover it can be verified that $O' \in maxfamily_i(O, A, C)$. Therefore $\neg\Phi \in out_i(O', A)$ and $\Phi \notin out_i(O', A)$. By the construction we can further verify that O' ranges over all elements of $maxfamily_i(O, A, C)$. Then we conclude $\Phi \notin out_i^{\cup}(O,\ A, C)$.

So we have reduced the validity problem of 2-QBF$^{\exists}$ to the full-join fulfillment problem, which shows the latter is $\mathtt{NP}^{\mathtt{NP}}$ hard. $\dashv$

In the setting of normative/legal reasoning, the full-meet and full-join fulfillment problems are the two most important decision problems of input/output logic. The full-meet and full-join operators are the operators our artificial moral agents will use to reason on obligations.

Theorem 3.19 *For $i \in \{1, 2, 3, 4, 1^+, 2^+, 3^+, 4^+\}$, the full-meet fulfillment problem is $\mathtt{coNP}^{\mathtt{NP}}$ complete.*

Proof Concerning the $\mathtt{coNP}^{\mathtt{NP}}$ membership, we prove by giving the following algorithm on a non-deterministic Turing machine with an NP oracle to solve the complement of our problem.

1. Guess a subset $O' \subseteq O$.
2. Use the NP oracle to test if $O' \in maxfamily_i(O, A, C)$. If no, return "reject" on this branch. Otherwise continue.
3. Use the NP oracle to test if $x \notin out_i(O', A)$. If $x \notin out_i(O', A)$, then return "accept" on this branch. Otherwise return "reject" on this branch.

It can be verified that $x \notin out^{\cap}(O, A, C)$ iff the non-deterministic Turing machine returns "accept" on some branches. Step 2 can be done in polynomial time steps because the maxfamily membership problem is in $\mathrm{P}^{\mathtt{NP}}$. Step 3 can also be done in polynomial time steps because the fulfillment problem is also in $\mathrm{P}^{\mathtt{NP}}$. Therefore the time complexity of this non-deterministic Turing machine is polynomial.

Concerning the $\mathtt{coNP}^{\mathtt{NP}}$ hardness, we show that the validity problem of 2-QBF$^{\forall}$ can be reduced to the full-meet fulfillment problem.

Let $\forall p_1 \ldots p_m \exists q_1 \ldots q_n \Phi$ be a 2-QBF$^{\forall}$ where Φ is a propositional formula with variables in $\{p_1, \ldots, p_m, q_1, \ldots, q_n\}$. Let $A = C = \emptyset$, $O = \{(\top, p_1), \ldots, (\top, p_m), (\top, \neg p_1), \ldots, (\top, \neg p_m), (\top, \Phi)\}$. Our aim is to show that this 2-QBF$^{\forall}$ is valid iff $\Phi \in out_i^{\cap}(O, A, C)$.

- If $\forall p_1 \dots p_m \exists q_1 \dots q_n \Phi$ is valid, then for all valuations V for $\{p_1, \dots, p_m\}$ there is a valuation V' for $\{q_1, \dots, q_n\}$ such that $V \cup V'$ gives truth value 1 to Φ and 0 to $\neg\Phi$.

  Let $O' = \{(\top, p'_1), \dots, (\top, p'_m), (\top, \Phi)\}$ be an arbitrary set such that each p'_i is either p_i or $\neg p_i$. Then $out_i(O', A) = Cn(\{p'_1 \dots, p'_m, \Phi\})$, which is consistent. Moreover it can be verified that $O' \in maxfamily_i(O, A, C)$. Therefore $\Phi \in out_i(O', A)$. By the construction we can further verify that O' ranges over all elements of $maxfamily_i(O, A, C)$. Then we conclude $\Phi \in out_i^{\cap}(O,\ A, C)$.

- If $\forall p_1 \dots p_m \exists q_1 \dots q_n \Phi$ is not valid, then there is a valuation V for $\{p_1, \dots, p_m\}$ such that for all valuations V' for $\{q_1, \dots, q_n\}$, $V \cup V'$ gives truth value 0 to Φ and 1 to $\neg\Phi$.

  Let $O' = \{(\top, p'_1), \dots, (\top, p'_m)\}$, where each p'_i is p_i if $p_i \in V$ and it is $\neg p_i$ if $p_i \notin V$. Then $O' \in maxfamily_i(O, A, C)$ because $out_i(O', A) = Cn(\{p'_1, \dots, p'_m\})$ is consistent with C and adding anything from $\{(\top, \neg p_1), \dots, (\top, \neg p_m), (\top, \Phi)\}$ to O' will destroy the consistency. Note that $\neg\Phi \in Cn(\{p'_1, \dots, p'_m\})$ by the construction of $\{p'_1, \dots, p'_m\}$. Therefore $\Phi \notin out_i(O', A)$, which further implies that $\Phi \notin out_i^{\cap}(O, A, C)$.

So, we have reduced the validity problem of 2-QBF$^{\forall}$ to the full-meet fulfillment problem, which shows the latter is $\mathtt{coNP}^{\mathtt{NP}}$ hard. ⊣

3.4 Complexity of permissive input/output logic

In this section we study the complexity of the following decision problems about permissive input/output logic: given a finite set of norms $N = (O, P)$, a finite set of input A and a formula x:

- negative permission checking: is $x \in NegPerm_i(N, A)$?
- positive-static permission checking: is $x \in StaPerm_i(N, A)$?
- positive-dynamic permission checking: is $x \in DyPerm_i(N, A)$?

Negative permission checking is relatively easy because it is simply the complement of the fulfillment problem.

Theorem 3.20

1. *For $i \in \{1, 2, 4\}$, the negative permission checking is* NP *complete.*
2. *For $i = 3$, negative permission checking is* NP *hard and in* $\mathtt{P}^{\mathtt{NP}}$.

Proof The negative permission checking is complement to the fulfillment problem. That is, $x \in NegPerm_i(N, A)$ iff $\neg x \notin out_i(O, A)$. Therefore the complexity of the negative permission checking problem belongs to the complement complexity class of the fulfillment problem. ⊣

Positive-static permission checking is no harder than the fulfillment problem because both the class coNP and $\mathtt{P}^{\mathtt{NP}}$ are closed under finite union.

Theorem 3.21

1. *For $i \in \{1, 2, 4\}$, the positive-static permission checking is* coNP *complete.*
2. *For $i = 3$, the positive-static permission checking is* coNP *hard and in* $\mathtt{P}^{\mathtt{NP}}$.

Proof

1. Let $P = \{(a_1, x_1), \ldots, (a_n, x_n)\}$. Then $x \in StaPerm_i(N, A)$ iff $x \in out_i(O \cup \{(a_1, x_1)\}, A) \cup \ldots \cup out_i(O \cup \{(a_n, x_n)\}, A)$ iff $x \in out_i(O \cup \{(a_1, x_1)\}, A)$ or $x \in out_i(O \cup \{(a_2, x_2)\}, A)$ or $\ldots$ or $x \in out_i(O \cup \{(a_n, x_n)\}, A)$. Since the coNP class is closed under finite union, we know that the positive-static permission checking problem is in coNP. The coNP hardness can be proved by setting $P = \emptyset$ and reduce the fulfillment problem to the static permission checking problem.
2. Similar to the above item. The $\mathtt{P}^{\mathtt{NP}}$ membership follows from the fact that the $\mathtt{P}^{\mathtt{NP}}$ class is closed under finite union. ⊣

Positive-dynamic permission checking is harder than other permission checking as the following theorem shows. The main source of complexity is that in positive-dynamic permission checking we have to first guess a consistent input and then check if it produced some inconsistency.

Theorem 3.22 *For $i \in \{1, 2, 3, 4\}$, the positive-dynamic permission checking is* $\mathtt{NP}^{\mathtt{NP}}$ *complete.*

Proof The $\mathtt{NP}^{\mathtt{NP}}$ membership follows by guessing a consistent set of formulae $C \subseteq f(O) \cup f(P) \cup A$, where $f(O) = \{a : (a, x) \in O\}$ and $f(P) = \{b : (b, y) \in P\}$, then using an NP oracle to check if $\bot \in StaPerm_i(N, C) \cup out_i(O \cup \{(\bigwedge A, \neg x)\}, C)$.

Concerning the $\mathtt{NP}^{\mathtt{NP}}$ hardness, we show that the validity problem of 2-$\mathtt{QBF}^{\exists}$ can be reduced to the positive-dynamic permission checking problem.

Let $\exists p_1 \ldots p_m \forall q_1 \ldots q_n \Phi$ be a 2-QBF$^{\exists}$ where Φ is a propositional formula contains variables only in $\{p_1, \ldots p_m, q_1, \ldots, q_n\}$.

Let $N = (O, P)$ where $O = \{(p_1, p_1), \ldots, (p_m, p_m), (\neg p_1, \neg p_1), \ldots, (\neg p_m, \neg p_m), (\Phi, \bot), (\neg\Phi, \top)\}$, $P = \emptyset$, $A = \emptyset$, $x = \bot$. Our aim is to prove that $\exists p_1 \ldots p_m \forall q_1 \ldots q_n \Phi$ is valid iff $x \in DyPerm_i(N, A)$.

Note that $x \in DyPerm_i(N, A)$ iff there is a consistent set C such that $StaPerm_i(N, C) \cup out_i(O \cup \{(\bigwedge A, \neg x)\}, C)$ is inconsistent, which means there is a consistent set C such that $out_i(O, C) \cup out_i(O \cup \{(\top, \neg\bot)\}, C)$ is inconsistent. This is equivalent to say that there is a consistent set C such that $out_i(O, C)$ is inconsistent. Moreover, the following are equivalent:

- There is a consistent set C such that $out_i(O, C)$ is inconsistent.
- There is a consistent C which is a subset of $f(O)$ such that $out_i(O, C)$ is inconsistent.

- There is a set C which is a maximal consistent subset of $f(O)$ such that $out_i(O, C)$ is inconsistent.

Therefore $x \in DyPerm_i(N, A)$ iff there is a set C which is a maximal consistent subset of $f(O)$ such that $out_i(O, C)$ is inconsistent. We now show that $\exists p_1 \dots p_m \forall q_1 \dots q_n \Phi$ is valid iff there is a set C which is a maximal consistent subset of $f(O)$ such that $out_i(O, C)$ is inconsistent.

- If $\exists p_1 \dots p_m \forall q_1 \dots q_n \Phi$ is valid, then there is a valuation V for $\{p_1, \dots, p_m\}$ such that for all valuations V' for $\{q_1, \dots, q_n\}$, $V \cup V'$ gives truth value 1 to Φ.

  Let $C = \{p'_1, \dots, p'_m, \Phi\}$, where each p'_i is p_i if $p_i \in V$ and it is $\neg p_i$ if $p_i \notin V$. Then C is a maximal consistent subset of $f(O)$. Moreover $\bot \in out_i(O, C)$ because $\Phi \in C$ and $(\Phi, \bot) \in O$.

- If $\exists p_1 \dots p_m \forall q_1 \dots q_n \Phi$ is not valid, then for all valuations V for $\{p_1, \dots, p_m\}$ there is a valuation V' for $\{q_1, \dots, q_n\}$ such that $V \cup V'$ gives truth value 0 to Φ and 1 to $\neg\phi$. Let $C = \{p'_1, \dots, p'_m, \neg\Phi\}$, where each p'_i is p_i if $p_i \in V$ and it is $\neg p_i$ if $p_i \notin V$. Then C is a maximal consistent subset of $f(O)$. Therefore $out_i(O, C) = Cn(\{p'_1, \dots, p'_m, \top\})$ which is consistent.

So we have reduced the validity problem of 2-QBF$^{\exists}$ to the dynamic permission checking problem, which shows the latter is $\mathtt{NP}^{\mathtt{NP}}$ hard. $\dashv$

4 Related work

The complexity results shown above are not so comforting with respect to the goal of using input/output logic in practical applications in computer science, e.g., applications in legal informatics (Boella, Di Caro, and Robaldo, 2013), (Boella et al., 2015) and machine ethics (Anderson and Anderson, 2011), (Sun and Robaldo, 2016). But we are still optimistic about the future of input/output logic in that sense because it seems the competitors of input/output logic face no less problems than input/output logic.

Furthermore, it must be noticed that nowadays there are automatic theorem provers such as systems of answer set programming (Gebser et al., 2012) which gives empirically good performance on NP hard problems.

As stated in the Introduction, input/output logic is a deontic framework for normative reasoning grounded on norm-based semantics. Norm-based semantics have been proposed as an alternative to standard deontic frameworks grounded on possible-world semantics, such as STIT logic (Horty, 2001) and dynamic deontic logic (Meyer, 1988; van der Meyden, 1996).

Possible-world semantics adds an extra machinery that makes the overall computational complexity worse than the one we found for input/output logic. For instance, (Schwarzentruber and Semmling, 2014) proved that group-STIT, which has been used to build deontic STIT (Horty, 2001), is undecidable. On the other hand, (Fischer and Ladner, 1979) and (Pratt, 1980) proved that the computational complexity of dynamic logic is EXPTIME-complete.

For these reasons, we believe norm-based semantics is superior to possible-world semantics from the computational complexity point of view.

However, input/output logic is not the only logic grounded non possible-world semantics. Other three relevant deontic logics based on non possible-world semantics are imperative logic (Hansen, 2008), prioritized default logic (Horty, 2012), and defeasible deontic logic (Governatori et al., 2013).

Hansen's imperative logic aims at properly representing and enabling reasoning on imperatives, e.g. "close the door!", thus it has an objective indeed different from the one of input/output logic. However, imperatives and norms are strongly related, as explained in (Gabbay et al., 2014) (pp. 137-191) first of all because they are both non-declarative prescriptive statements, which do not bear truth values. Hansen's imperative logic is built upon unconstrained input/output logic; the added value is a non possible-world semantics where imperatives are assigned priorities and ranked accordingly.

Priorities are also the basic mechanism of Horty's prioritized default logic, which extends standard Reiter's default logic (Reiter, 1987) by ranking default rules in order to handle moral conflicts. Many examples of how these moral conflicts are modeled in Horty's logic may be found in (Horty, 2012). According to (Parent, 2011), prioritized default logic is particularly suitable for normative reasoning in that it takes reasons as the basic normative concept. However, it fails to properly address more "realistic" scenarios in which multiple norms conflict of one another. In light of these counterexamples, (Parent, 2011), drawing from (Boella and van der Torre, 2003), develops an extension of input/output logic with prioritized norms in order to handle further moral conflicts that Horty's logic is unable to solve.

The complexity of Hansen's imperative logic and the one of Horty's prioritized default logic have not been studied yet. Of course, adding priorities affects the complexity. We hope the present work can serve at least as a starting point to provide an answer to the complexity problem of Hansen's and Horty's logic. On the other hand, extending our complexity results to prioritized input/output logic will be the object of our future work.

Complexity results are shown for defeasible deontic logic. In particular, in (Governatori et al., 2013), it is shown that the approach is computationally tractable. However, it must be noticed that the object logic used in (Governatori et al., 2013) has a quite reduced expressivity, in that it only allows (modal) literals. The object logic is built by taking a small fragment of propositional logic that only includes propositional symbols and their negations. Other two modal operators, which respectively model obligatory and permissive modalities, may be applied to the propositional literals in order to obtain modal literals. Finally, further operators to model priorities are introduced and wrapped around the formulae in the object logic. Given the simplicity of the object logic, tractability comes out easily.

Another final recent approach worth mentioning is the one of (Straßer, Beirlaenz, and van de Puttey, 2015), which proposes a translation of constrained input/output logic into adaptive logic (Batens, 2007). At the end of their paper, the authors consider the complexity of constrained input/output logic as their future work. They tend to investigate whether the complexity results of adaptive logic can be transferred to constrained input/output logic. While adaptive logic is undecidable (Verdee, 2009), our results show that constrained input/output logic is computationally more efficient than adaptive logic. On the other hand, it seems the translation of constrained input/output logic gives us a

decidable fragment of adaptive logic.

Similarly, this paper enables other directions of further research for some non-standard input/output logic systems that have been subsequently proposed in the literature, and for which decidability/complexity results are unknown. For instance, several different unconstrained input/output logics have been developed in (Stolpe, 2008) and (Parent and van der Torre, 2014).

And, other notions of permission have been studied in (Stolpe, 2010), who showed convincingly that (Makinson and van der Torre, 2003a)'s notion of positive permission is problematic, and proposed an alternative account in input/output logic centered on the notion of permission as exception. The approach of (Stolpe, 2010) has been later criticized in (Hansen, 2014), who proposes a unifying framework in which norms are assigned with priorities such that both (Makinson and van der Torre, 2003a)'s dynamic permission and (Stolpe, 2010)'s permission as exception can be characterized.

One limitation of (Hansen, 2014) is that it only applies to unconditional norms. In our future works, we aim at extending (Hansen, 2014)'s framework to conditional norms in order to obtain a powerful logic of permission using prioritized input/output logic, drawing from the solution we recently proposed in (Sun, Zhao, and Robaldo, 2017). The complexity results presented in section 2.3 will provide a basis for the study of the complexity of our future account of permission.

Other future works concern the extension of input/output logic beyond the propositional level, in order to make the framework suitable for practical applications (cf. (Robaldo and Sun, 2017), which integrates in input/output logic flat semantic representations for natural language semantics similar to those advocated in (Robaldo, 2010) and (Robaldo, Szymanik, and Meijering, 2014)). Since standard first-order logic is already semi-decidable, we expect the overall complexity to be intractable. In order to keep the complexity under control, we will consider fragments of first-order logic and/or reducing the set of axioms used in the input/output systems.

5 Conclusions

The present paper is the first relevant work that gives a full characterization of basic computational tasks in input/output logic, namely: (1) fulfillment problem of unconstrained input/output logic (2) consistency checking, (3) $maxfamily$ membership, (4) skeptical/credulous non-monotonic reasoning, and (5) negative, positive-static, and positive-dynamic permission checking. Our main finding is that the computational tasks from (1) to (5) are `coNP/NP` hard and in the 2nd level of the polynomial hierarchy.

It is well-known in theoretical computer science that complexity is an indispensable component of every logic. Past literature on input/output logic focuses on proof theory and semantics only, and neglects complexity. The present paper fills the missing gap.

The complexity results are not so comforting with respect to the goal of using input/output logic in practical applications in computer science. Nevertheless, a survey of main existing relevant frameworks in normative reasoning reveals that none of them appears to be a viable alternative to input/output logic. In our future works we will investigate specific ad-hoc solutions to control complexity on each practical application where we want to employ the formalism or its extensions.

References

Anderson, Michael and Susan L. Anderson. 2011. *Machine Ethics*. Cambridge University Press.

Arora, Sanjeev and Boaz Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, USA.

Batens, Diderik. 2007. A universal logic approach to adaptive logics. *Logica Universalis*, 1(1):221–242.

Boella, Guido, Luigi Di Caro, Michele Graziadei, Loredana Cupi, Carlo Emilio Salaroglio, Llio Humphreys, Hristo Konstantinov, Kornel Marko, Livio Robaldo, Claudio Ruffini, Kiril Simov, Andrea Violato, and Veli Stroetmann. 2015. Linking legal open data: Breaking the accessibility and language barrier in european legislation and case law. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, pages 171–175, New York, NY, USA. ACM.

Boella, Guido, Luigi Di Caro, and Livio Robaldo. 2013. Semantic relation extraction from legislative text using generalized syntactic dependencies and support vector machines. In *Theory, practice, and applications of rules on the web*. Springer Berlin Heidelberg, pages 218–225.

Boella, Guido and Leendert van der Torre. 2003. Permissions and obligations in hierarchical normative systems. In *Proc. of the 9th International Conference on Artificial Intelligence and Law, ICAIL 2003, Edinburgh, Scotland, UK, June 24-28, 2003*, pages 109–118.

Fischer, Michael and Richard Ladner. 1979. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, (18):194–211.

Gabbay, Dov, John Horty, Xavier Parent, Ron van der Meyden, and Leon van der Torre. 2014. *Handbook of Deontic Logic and Normative Systems*. College Publications, London.

Gebser, Martin, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2012. *Answer Set Solving in Practice*. Morgan and Claypool Publishers.

Governatori, Guido, Francesco Olivieri, Antonino Rotolo, and Simone Scannapieco. 2013. Computing strong and weak permissions in defeasible logic. *Journal of Philosophical Logic*, 42(6):799–829.

Halpern, Joseph Y. 1995. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75:361–372.

Hansen, Jörg. 2008. Prioritized conditional imperatives: problems and a new proposal. *Autonomous Agents and Multi-Agent Systems*, 17(1):11–35.

Hansen, Jörg. 2014. Reasoning about permission and obligation. In S. O. Hansson, editor, *David Makinson on Classical Methods for Non-Classical Problems*, pages 287–333. Outstanding Contributions to Logic Volume 3, Springer.

Horty, John. 2001. *Agency and Deontic Logic*. Oxford University Press, New York.

Horty, John. 2003. Reasoning with moral conflicts. *Noûs*, 37:557–605.

Horty, John. 2012. *Reasons as Defaults*. Oxford University Press.

Jørgensen, Jorgen. 1937. Imperatives and logic. *Erkenntnis*, 7:288–296.

Makinson, David and Leendert van der Torre. 2001. Constraints for input/output logics. *Journal of Philosophical Logic*, 30(2):155–185.

Makinson, David and Leendert van der Torre. 2003a. Permission from an input/output perspective. *Journal of Philosophical Logic*, 32:391–416.

Makinson, David and Leendert van der Torre. 2003b. What is input/output logic? In B. Lowe, W. Malzkorn, and T. Rasch, editors, *Foundations of the Formal Sciences II: Applications of Mathematical Logic in Philosophy and Linguistics*, pages 163–174.

Makinson, David and Leendert W. N. van der Torre. 2000. Input/output logics. *Journal of Philosophical Logic*, 29(4):383–408.

Meyer, John Jule. 1988. A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, pages 109–136.

Parent, Xavier. 2011. Moral particularism in the light of deontic logic. *Artificial Intelligence and Law*, 19(2-3):75–98.

Parent, Xavier and Leendert W. N. van der Torre. 2014. "sing and dance!" - input/output logics without weakening. In Fabrizio Cariani, Davide Grossi, Joke Meheus, and Xavier Parent, editors, *Deontic Logic and Normative Systems - 12th International Conference, DEON 2014, Ghent, Belgium, July 12-15, 2014. Proceedings*, volume 8554 of *Lecture Notes in Computer Science*, pages 149–165. Springer.

Pratt, Vaughan R. 1980. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20(2):231–254.

Reiter, Raymond. 1987. A logic for default reasoning. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*. Kaufmann, Los Altos, CA, pages 68–93.

Robaldo, Livio. 2010. Interpretation and inference with maximal referential terms. *The Journal of Computer and System Sciences*, 76(5):373–388.

Robaldo, Livio and Xin Sun. 2017. Reified input/output logic: Combining input/output logic and reification to represent norms coming from existing legislation. *Journal of Logic and Computation*, to appear.

Robaldo, Livio, Jakub Szymanik, and Ben Meijering. 2014. On the identification of quantifiers' witness sets: a study of multi-quantifier sentences. *The Journal of Logic, Language, and Information.*, 23(1).

Schwarzentruber, François and Caroline Semmling. 2014. STIT is dangerously undecidable. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Stolpe, Audun. 2008. Normative consequence: The problem of keeping it whilst giving it up. In Ron van der Meyden and Leendert van der Torre, editors, *Proceedings of the 9th international conference on Deontic Logic in Computer Science*, DEON '08, pages 174–188, Berlin, Heidelberg. Springer-Verlag.

Stolpe, Audun. 2010. A theory of permission based on the notion of derogation. *J. Applied Logic*, 8(1):97–113.

Straßer, Christian, Mathieu Beirlaenz, and Frederik van de Puttey. 2015. Adaptive logic characterizations of input/output logic. *Studia Logica*, 104(5):869–916.

Sun, Xin. 2014. How to build input/output logic. In Nils Bulling, Leendert W. N. van der Torre, Serena Villata, Wojtek Jamroga, and Wamberto Vasconcelos, editors, *Computational Logic in Multi-Agent Systems - 15th International Workshop, CLIMA XV, Prague, Czech Republic, August 18-19, 2014. Proceedings*, volume 8624 of *Lecture Notes in Computer Science*, pages 123–137. Springer.

Sun, Xin and Livio Robaldo, 2016. *Logic and Games for Ethical Agents in Normative Multi-agent Systems*, pages 367–375. Springer International Publishing, Cham.

Sun, Xin, Xishun Zhao, and Livio Robaldo. 2017. Norm-based deontic logic for access control, some computational results. *Future Generation Computer Systems*.

van der Meyden, Ron. 1996. The dynamic logic of permission. *Journal of Logic and Computation*, 6:465–479.

Verdee, Peter. 2009. Adaptive logics using the minimal abnormality strategy are π^1_1-complex. *Synthese*, 167(1):93–104.

von Wright, Georg. 1968. *An Essay in Deontic Logic and the General Theory of Action*. Acta Philosophica Fennica, Fasc. 21, North-Holland.

Wagner, Klaus. 1986. More complicated questions about maxima and minima, and some closures of np. In Laurent Kott, editor, *Automata, Languages and Programming*, volume 226 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pages 434–443.