Goal-Directed Decision Procedures for Input/Output Logics

Alexander Steen [1]

University of Luxembourg, FSTM
6, Avenue de la Fonte
L-4364 Esch-sur-Alzette, Luxembourg

Abstract

Input/Output (I/O) logics address the abstract study of conditional norms. Here, norms are represented as pairs of formulas instead of statements that themselves carry truth-values. I/O logics have been studied thoroughly in the past, including further applications and refinements. In this paper, a class of automated reasoning procedures is presented that, given a set of norms and a concrete situation, decide whether a specific state of affairs is obligatory according to the output operations of I/O logics. The procedures are parametric in the underlying logical formalism and can be instantiated with different classical objects logics, such as propositional logic or first-order logic. The procedures are shown to be correct, and a proof-of-concept implementation for propositional I/O logics is surveyed.

Keywords: Deontic logic, I/O logics, Automated reasoning, Normative reasoning.

1 Introduction

Input/Output (I/O) logics have been devised by Makinson and van der Torre [8] as a class of formal systems for norm-based deontic reasoning. Intuitively, they formalize the question which obligations can be detached from a given set of conditional norms and a specific situation. I/O logics differ from other deontic logics, such as Standard Deontic Logic (SDL, a modal logic of type **KD**) and Dyadic Deontic Logic (DDL) [1], in the sense that the norms themselves are not part of the object logic and hence do not carry truth values. Furthermore, in SDL and DDL the deontic operators are evaluated with respect to a set of possible words, whereas in I/O logics they are evaluated with respect to a set of norms. An overview of deontic logic formalisms can be found in the literature, see e.g. [7].

The field of automated reasoning studies the conception, implementation, application and evaluation of methods for automating logical inferences on the computer [14]. This includes, among others, methods for deciding satisfiability and tautology, for model generation, and for computer algebra systems. The

[1] E-Mail: `alexander.steen@uni.lu`; ORCID ID: 0000-0001-8781-9462

study of automated deduction systems denotes one of the earliest concerns of artificial intelligence, and is today often referred to as symbolic AI; in contrast to recently successful approaches using statistical and learning-based approaches. One of the core applications is automated theorem proving (ATP): ATP systems are computer programs that, given a set A of axioms and a conjecture C as input, try to prove that C is a logical consequence of A, i.e., that C is true whenever every formula in A holds. In this context, the search for a proof is conducted autonomously so that no intervention or advice from human users is necessary. Unfortunately, most ATP systems focus on classical logics only and hence there are only few systems available for automating logics relevant to deontic reasoning. Notable exceptions are ATP systems for (normal) modal logics, but since these logics suffer from various theoretical drawbacks, their application to normative reasoning in, e.g., legal contexts [5] is limited.

In this paper, a first structured step is taken towards automation of I/O formalisms: Decision procedures for four different deontic operators of (unconstrained) I/O logic are presented that decide whether a formula x can be detached as an obligation given a set of norms and a situation (put in I/O logic terms: the procedures decide whether a formula x is in the output given a certain input). They are shown to be sound and complete, and to be decidable if the underlying logical language is decidable. Furthermore, a prototype implementation of the procedures is presented. This implementation is freely available as a web application and can be used to conduct own experiments.

Related work. I/O logics have also been employed in the context of studying conditional permissions [10]. Also, there exist extensions of I/O logics, called *constrained I/O logics*, that address the classical deontic paradoxes [9] such as contrary-to-duty scenarios. Recent work furthermore addresses weaker notions of I/O logic that allow for a fined-grained control over employed inference principles [12].

From a computational perspective, there are comparably few related approaches available. Complexity aspects of I/O logics have been studied [17]. However, the methods used in the analysis do not yield means for implementing respective decision procedures. Quite recent work focuses on automating other deontic logics via shallow semantical embeddings into classical higher-order logic [3]. However, such an approach is not yet available for all unconstrained I/O logic operations [2], and indeed seems more complex than for other logical systems in the context of deontic reasoning [4]. There are representation results available for expressing I/O logics in modal logic; and there exists an alternative proof-theoretic (dynamic) characterization of I/O logic [16]. However, these results have, up to the author's knowledge, not yet been utilized in the context of automated reasoning systems.

2 I/O Logics

I/O logic is used for studying conditional norms, e.g., obligations under some legal code. Here, conditional codes are represented as pairs of formulas and therefore do not carry truth values themselves, whereas declarative statements are usual Boolean formulas that come from some logical language L.

Let L be a logical language that is closed under the truth-functional connectives such as conjunction ($\wedge$) and disjunction ($\vee$). From a semantical perspective, it is assumed that the eligible logical languages considered in the following come with a derivation relation $\vdash$ for which the operation $Cn(A)$, given by $Cn(A) = \{x \in L \mid A \vdash x\}$, is a Tarskian closure operator. A prominent example for L in the context of I/O logics is the language of classical propositional logic with the usual consequence relation (often assumed in the literature). However, also first-order logic or even higher-order logic languages are possible. In the following, it is assumed that L comes from a classical logic.

A normative system $N \subseteq L \times L$ is a set of pairs (a, x) of formulas. The pair (a, x) represents the conditional obligation that *given a, it ought to be x*. By convention, given a norm (a, x) the first element a is also referred to as the body and the second element x is referred to as the head. The image of N, denoted $N(A)$, where A is a set of formulas, is given by $N(A) = \{x \in L \mid (a, x) \in N \textit{ for some } a \in A\}$. Given a normative system N and a set of formulas A (the input set), $out(N, A)$ denotes the output of A under N where out is the respective output operator.

The semantics of I/O logics is operational in the sense that the meaning of normative concepts is given by generated outputs given a set of norms and an input. The four output operators out_i, $i \in \{1, 2, 3, 4\}$, studied in the literature are defined as follows [8]:

$out_1(N, A) = Cn(N(Cn(A)))$
$out_2(N, A) = \bigcap \big\{Cn(N(V)) \mid V \supseteq A,\ V \textit{ complete}\big\}$
$out_3(N, A) = \bigcap \big\{Cn(N(B)) \mid A \subseteq B = Cn(B) \supseteq N(B)\big\}$
$out_4(N, A) = \bigcap \big\{Cn(N(V)) \mid A \subseteq V \supseteq N(V),\ V \textit{ complete}\big\}$

where a set $V \subseteq L$ is called complete iff $V = L$ or V is a maximally consistent set.

A proof-theoretic characterization of the different output operations can be achieved by putting $out_i(N) = \big\{(A, x) \mid x \in out_i(N, A) \text{ for some } A \subseteq L\big\}$, $i \in \{1, 2, 3, 4\}$. The specific inference rules are the following (the first component of each pair are assumed to be singleton sets and the curly braces are omitted):

SI:	From (a, x) to (b, x) if $b \vdash a$
WO:	From (a, x) to (a, y) if $x \vdash y$
AND:	From (a, x), (a, y) to $(a, x \wedge y)$
OR:	From (a, x), (b, x) to $(a \vee b, x)$
CT:	From (a, x), $(a \wedge x, y)$ to (a, y)

In earlier work [8] it is shown that (a, x) is in the respective set $out_i(N)$ if and only if it is contained in the least superset of $N \cup (\top, \top)$ that is closed under the inference rules as follows: $\{SI, WO, AND\}$ for out_1, $\{SI, WO, AND, OR\}$ for out_2, $\{SI, WO, AND, CT\}$ for out_3, and $\{SI, WO, AND, OR, CT\}$ for out_4. For non-singleton sets A, derivability of (A, x) from N is reduced to derivability of (a, x), where $a = a_1 \wedge \ldots \wedge a_n$ is some conjunction of the elements of A.

The empty input is interpreted as an empty conjunction and assumed to be a tautology; hence $(\emptyset, x)$ is reduced to derivability of $(\top, x)$.

Although there is an adequate syntactic characterization for I/O logics that can be used to derive outputs, the above calculi are not *machine-oriented* in the sense that they can be implemented as-is in an effective manner on a computer. This is, in particular, because the rules *SI* and *WO* allow to derive an infinite number of outputs, and it is not immediately clear what intermediate derivations actually contribute to the ultimate proof goal. However, without effective means of automation it is challenging, or even impossible, to apply I/O logics to practical scenarios, e.g., in the context of multi-agent systems or legal reasoning use cases [13].

3 Decision Procedures for I/O logic

In this section, four different decision procedures are presented, one for each output operation, that allow automated reasoning within I/O logics in an effective way. It is implicitly assumed that $\vdash$ is a sound and complete derivation relation for L. Furthermore, it is assumed that inputs to the decision procedures are finite, i.e. N is a finite set of norms and A is a finite set of formulae.

Let $\textsc{In-Out}_i$, $i \in \{1, 2, 3, 4\}$, denote the following decision problem: *Given a formula $x \in L$, a set of norms N and an input A, is it the case that $x \in out_i(N, A)$?* The decision procedure that addresses the respective decision problem $\textsc{In-Out}_i$ is denoted $\mathsf{IO}_i^{\vdash}$. By convention, $(N, A, x) \in \mathsf{IO}_i^{\vdash}$ is written iff $\mathsf{IO}_i^{\vdash}$ gives "yes" for a set of norms N, input A and prospective output x; i.e., it is identified with the subset of parameter tuples for which it decides positively.

Note that because it is not fully specified what logical formalism L is employed, the procedures $\mathsf{IO}_i^{\vdash}$ presented in the following in fact describe a class of parametric decision procedures that can be used in different contexts. As an example, if L is taken as classical propositional logic the implementation of the $\mathsf{IO}_i^{\vdash}$ is straight-forward, and they are guaranteed to terminate and to yield correct results.

It is also possible to apply each $\mathsf{IO}_i^{\vdash}$ to logics that are not decidable: Already existing automated theorem provers can be utilized as oracle for $\vdash$, e.g., in the context of first-order logic or higher-order logic. This way a wide range of input-output logic reasoners can be implemented with comparably low effort. Of course, for logics that are not decidable the procedures $\mathsf{IO}_i^{\vdash}$ might never terminate; as usual in automated reasoning in expressive logics.

The decision procedures are given in pseudo-code in the following.

3.1 Simple-minded output

Listing 1 shows the decision procedure $\mathsf{IO}_1^{\vdash}$ for deciding $\textsc{In-Out}_1$ with respect to a logical language L with underlying derivation relation $\vdash$.

The general idea of $\mathsf{IO}_1^{\vdash}$ is the following: First, the subset N' of norms whose body is satisfied by the input A is calculated. Then, the procedure returns `Yes` if and only if the heads of N' satisfy x. Note that this approach allows to effectively handle the infinite sets $Cn(A)$ and $Cn(N(Cn(A)))$ that cannot be

```
1  Input:   $N = \{(b_1,h_1),(b_2,h_2),\ldots\}$ set of norms
2           $A = \{a_1,\ldots,a_m\}$ set of formulae
3           $x$ formula
4  Output: Yes or No
5
6  $N'$ := $\{(b,h) \in N \mid A \vdash b\}$
7  if $\{h \mid (b,h) \in N'\} \vdash x$ then
8    return Yes
9  else
10   return No
11 endif
```

Listing 1: Decision procedure $\mathsf{IO}_1^{\vdash}$ for In-Out$_1$.

computed exhaustively in an explicit way. The procedure $\vdash$ used here acts as oracle. Depending on the logic L that is assumed, the effort for implementing such a procedure may vary. An example implementation is described in §4.

Adequateness of $\mathsf{IO}_1^{\vdash}$ is ensured as shown in the following:

Theorem 3.1 (Partial correctness of $\mathsf{IO}_1^{\vdash}$) *$\mathsf{IO}_1^{\vdash}$ is sound and complete for* In-Out$_1$*; in particular, $x \in out_1(N,A)$ if and only if $(N,A,x) \in \mathsf{IO}_1^{\vdash}$.*

Proof For the first direction, assume $x \in out_1(N,A)$ for some formula $x \in L$. By definition, $N(Cn(A)) \vdash x$ hence there exists some subset $\{(b_1,h_1),\ldots,(b_m,h_m)\} \subseteq N$ of norms such that $A \vdash \bigwedge_{i=1}^{m} b_i$ and $\bigwedge_{i=1}^{m} h_i \vdash x$. Since N' as computed in line 6 is the largest subset of N for which each body of $(b,h) \in N'$ it holds that $A \vdash b$, by monotony of $\vdash$ it also holds that $\{h \mid (b,h) \in N'\} \vdash x$. As a consequence, the `if` condition in line 7 is true and thus $(N,A,x) \in \mathsf{IO}_1^{\vdash}$.

For the second part, assume $(N,A,x) \in \mathsf{IO}_1^{\vdash}$. Then, by definition, there is a subset N' of norms such that $\{h \mid (b,h) \in N'\} \vdash x$. It furthermore holds that $A \vdash b$ for each $(b,h) \in N'$ by construction. As a consequence, it holds that $x \in Cn(N'(Cn(A)))$. Since $N' \subseteq N$ it follows that $x \in Cn(N(Cn(A)))$ and hence $x \in out_1(N,A)$. $\square$

Theorem 3.2 (Total correctness of $\mathsf{IO}_1^{\vdash}$) *Let $\vdash$ be a decidable derivation relation for L. $\mathsf{IO}_1^{\vdash}$ terminates and is sound and complete for* In-Out$_1$.

Proof Theorem 3.1 already yields soundness and completeness. Termination is straight-forward: As all input is finite and since $\vdash$ is decidable by assumption the set N' can be constructed in finite time. Also, the `if`-condition in line 7 can be evaluated in finite time as $\vdash$ is decidable by assumption. $\square$

3.2 Basic output

Listing 2 presents the decision procedure $\mathsf{IO}_2^{\vdash}$ for deciding In-Out$_2$.

$\mathsf{IO}_2^{\vdash}$ is a slightly modified version of the respective procedure for In-Out$_1$, adapted to incorporate the validity of the *OR* rule. If the output of a formula can be established for different inputs, then it is also in the output set for the disjunction of these inputs. From a semantical perspective this amounts to incorporating reasoning by cases (cf. definition of out_2 in §2): If a norm n

```
1  Input:   $N = \{(b_1, h_1), (b_2, h_2), \ldots\}$ set of norms
2           $A = \{a_1, \ldots, a_m\}$ set of formulae
3           $x$ formula
4  Output:  Yes or No
5
6  $N'$ := $\emptyset$
7  for all $(b, h) \in N$ do
8    $C$ := $\{(b', h') \in N \mid h' \vdash h\}$
9    if $A \vdash \bigvee\{b' \mid (b', h') \in C\}$ then
10     $N'$ := $N' \cup \{(b, h)\}$
11   endif
12 endfor
13
14 if $\{h \mid (b, h) \in N'\} \vdash x$ then
15    return Yes
16 else
17    return No
18 endif
```

Listing 2: Decision procedure $\mathsf{IO}_2^{\vdash}$ for IN-OUT$_2$.

is triggered by every complete extension of the input, then the head of n is contained in the output set. In order to reflect this in $\mathsf{IO}_2^{\vdash}$, the calculation of the subset of triggered norms N' is modified as follows. If $n \in N$ is a norm, then let $n' \in N$ denote a *n-compatible norm* if and only if the body of n' is at least as strong as the body of n, i.e., it holds that $h' \vdash h$ where h and h' are the heads of n and n', respectively. Intuitively, every n-compatible norm can be considered a conditional case in which n's head is triggered. In order to check whether the head of a given norm n is triggered in every complete extension of A, the set of n-compatible norms is first collected in a set C (cf. line 8). Subsequently, it is checked whether the disjunction of every body in C is a consequence of A (`if`-condition in line 9). If this is the case, the norm n is added to the set of triggered norms N'. This is iteratively conducted for every norm on N; as soon as the `for`-loop has terminated, the set N' contains every norm that is triggered by A in the *basic* setting. Finally, just like for $\mathsf{IO}_1^{\vdash}$, is it checked whether the prospective output x is entailed by the heads of N' (cf. line 14) and a respective answer is returned.

The following results establish adequateness for IN-OUT$_2$:

Theorem 3.3 (Partial correctness of $\mathsf{IO}_2^{\vdash}$) *$\mathsf{IO}_2^{\vdash}$ is sound and complete for* IN-OUT$_2$*; in particular, $x \in out_2(N, A)$ if and only if $(N, A, x) \in \mathsf{IO}_2^{\vdash}$.*

Proof For the left-to-right direction, the contrapositive is shown. Assume that $(N, A, x) \notin \mathsf{IO}_2^{\vdash}$. By definition, it follows that $\{h \mid (b, h) \in N'\} \nvdash x$, where N' is the set generated in lines 6-12. It remains to be shown that there is no head $h' \in L$ that was incorrectly not detached by $\mathsf{IO}_2^{\vdash}$. Let $n = (b', h') \in N \setminus N'$ be a norm such that $\{h \mid (b, h) \in N'\} \nvdash h'$. By construction it holds that $A \nvdash \bigvee\{b \mid (b, h) \in N \text{ and } h \vdash h'\}$, and hence there exists at least one complete extension $V \supseteq A$ such that $V \vdash \neg b$ for every (b, h) with $h \vdash h'$. It follows that $N(V) \nvdash h'$ for some complete $V \supseteq A$. By generalization, it holds that

```
Input:   $N = \{(b_1, h_1), (b_2, h_2), \ldots\}$ set of norms
         $A = \{a_1, \ldots, a_m\}$ set of formulae
         $x$ formula
Output: Yes or No

$A' := A$
$N'$ := $\{(b, h) \in N \mid A' \vdash b\}$
$\overline{N}$ := $N \setminus N'$

while not $\{h \mid (b, h) \in N'\} \vdash x$ do
  $A'$ := $A' \cup \{h \mid (b, h) \in N'\}$
  $M$ := $\{(b, h) \in \overline{N} \mid A' \vdash b\}$
  if $M = \emptyset$ then
    return No
  else
    $N'$ := $N' \cup M$
    $\overline{N}$ := $\overline{N} \setminus M$
  endif
end while
return Yes
```

Listing 3: Decision procedure $\mathsf{IO}_3^{\vdash}$ for In-Out$_3$.

$N(V) \not\vdash x$ and thus $x \notin out_2(N, A)$.

For the second part, assume $(N, A, x) \in \mathsf{IO}_2^{\vdash}$. Then, by construction, there exists a set $N' \subseteq N$ such that $\{h \mid (b, h) \in N'\} \vdash x$. Let $(b, h) \in N'$ be some norm. It follows that there exists some set of norms $\{(b'_1, h'_1), \ldots, (b'_m, h'_m)\} \subseteq N$ such that $h'_i \vdash h$, for each $1 \leq i \leq m$, and $A \vdash b'_1 \vee \ldots \vee b'_m$. By monotony, it also holds that $V \vdash b'_1 \vee \ldots \vee b'_m$ for every complete set $V \supseteq A$. Since V is complete, it follows that $V \vdash b'_i$ for some $1 \leq i \leq m$ and hence $N(V) \vdash h$. By generalization, it holds that $N(V) \vdash \bigwedge\{h \mid (b, h) \in N'\}$ and thus $N(V) \vdash x$. Again, by generalization, $N(V) \vdash x$ for every complete $V \supseteq A$ and hence $x \in out_2(N, A)$.

□

Theorem 3.4 (Total correctness of $\mathsf{IO}_2^{\vdash}$) *Let $\vdash$ be a decidable derivation relation for L. $\mathsf{IO}_2^{\vdash}$ terminates and is sound and complete for* In-Out$_2$.

Proof Theorem 3.3 already yields soundness and completeness. The termination argument is analogous to the the $\mathsf{IO}_1^{\vdash}$ case if $\vdash$ is decidable. □

3.3 Reusable output

Listing 3 shows the decision procedure $\mathsf{IO}_3^{\vdash}$ for deciding In-Out$_3$ with respect to a logical language L with underlying derivation relation $\vdash$.

In $\mathsf{IO}_3^{\vdash}$ a more complex *proof search* is conducted in order to accommodate the interdependence between the operators $Cn(.)$ and $N(.)$ in the semantics of out_3. The basic approach is quite similar to the $\mathsf{IO}_1^{\vdash}$ procedure for out_1: A set N' of norms is calculated that is triggered by the input A; then, it is checked whether the heads of these norms N' satisfy the output x. However, since out_3 validates the CT rule, it is also possible that x it not satisfied directly by the heads of N' but rather by some superset of N' which is triggered by $A \cup \{h \mid (b,h) \in N'\}$. Put differently, the output is reused to further strengthen the input and, in turn, to possibly trigger more outputs. As this can be done repeatedly, the $\mathsf{IO}_3^{\vdash}$ procedure iteratively updates the input (called A') by the heads of the triggered norms, and subsequently collects all newly triggered norms (by A') in an updated set N'. If in some iteration N' satisfies the output x, the proof search succeeds; if, however, x is not satisfied and there are no new norms triggered, the process is terminated and a negative answer is returned. The termination condition intuitively reflects that N' is a fixed point with respect to $Cn(.)$ and $N(.)$ and moreover does not satisfy x.

Let A^* be the least superset of A that is closed both under Cn and N. The following results establish adequateness for In-Out$_3$, using so-called *bulk increments* [15]:

Theorem 3.5 (Partial correctness of $\mathsf{IO}_3^{\vdash}$) *$\mathsf{IO}_3^{\vdash}$ is sound and complete for* In-Out$_3$*; in particular, $x \in out_3(N, A)$ if and only if $(N, A, x) \in \mathsf{IO}_3^{\vdash}$.*

Proof Assume $x \in out_3(N, A)$ for some formula $x \in L$. Then, it holds that $x \in Cn(N(A^*))$. This implies that there exists a subset $N' = \{(b_1, h_1), \ldots, (b_m, h_m)\} \subseteq N$ of norms such that $A^* \vdash \bigwedge_{i=1}^m b_i$ and $\bigwedge_{i=1}^m h_i \vdash x$. By construction, in every iteration it holds that $A \subseteq A'$ and either $N(A') \vdash x$ in which case already `Yes` is returned, or A' will eventually reach a fixed point that is A^*. In the latter case $N(A') \vdash x$ iff $N(A^*) \vdash x$, which holds by assumption, and `Yes` is returned. In either case, $(N, A, x) \in \mathsf{IO}_3^{\vdash}$.

For the second part, assume $(N, A, x) \in \mathsf{IO}_3^{\vdash}$. Then, by construction there exists some A' such that $A \subseteq A' \subseteq A^*$ and $N(A') \vdash x$. Since N is monotone, it also holds that $N(A^*) \vdash x$ and hence $x \in out_3(N, A)$. $\square$

Theorem 3.6 (Total correctness of $\mathsf{IO}_3^{\vdash}$) *Let $\vdash$ be a decidable derivation relation for L. $\mathsf{IO}_3^{\vdash}$ terminates and is sound and complete for* In-Out$_3$.

Proof Theorem 3.5 already yields soundness and completeness. As $\vdash$ is decidable and every input is finite, every loop iteration itself terminates. There are only a finite number of loop iterations, as the set $\overline{N}$ is monotonously decreasing and $Cn(.)$ is monotone and idempotent. If no new norms can be triggered, the loop is terminated. $\square$

3.4 Basic reusable output

Listing 4 shows the decision procedure $\mathsf{IO}_4^{\vdash}$ for deciding In-Out$_4$ with respect to a logical language L with underlying derivation relation $\vdash$.

The procedure $\mathsf{IO}_4^{\vdash}$ combines the incremental proof search approach of $\mathsf{IO}_3^{\vdash}$ with the method for calculating the triggered norms in the basic output scenario as incorporated by $\mathsf{IO}_2^{\vdash}$: In the resulting procedure, the set N' of (basic) triggered norms is generated first (cf. lines 7-13). The remaining norms that have not been triggered (yet) are collected in the set $\overline{N}$. If the heads of N' do not entail the prospective output x, the set N' is incrementally augmented with the previous output (cf. line 17) and, subsequently, the freshly triggered norms are calculated in an auxiliary set M (cf. lines 18-24). If, at some point

```
1  Input:  $N = \{(b_1, h_1), (b_2, h_2), \ldots\}$ set of norms
2          $A = \{a_1, \ldots, a_m\}$ set of formulae
3          $x$ formula
4  Output: Yes or No
5
6  $A' := A$
7  $N'$ := $\emptyset$
8  for all $(b, h) \in N$ do
9    $C$ := $\{(b', h') \in N \mid h' \vdash h\}$
10   if $A' \vdash \bigvee\{b' \mid (b', h') \in C\}$ then
11     $N'$ := $N' \cup \{(b, h)\}$
12   endif
13 endfor
14 $\overline{N}$ := $N \setminus N'$
15
16 while not $\{h \mid (b, h) \in N'\} \vdash x$ do
17   $A'$ := $A' \cup \{h \mid (b, h) \in N'\}$
18   $M$ := $\emptyset$
19   for all $(b, h) \in \overline{N}$ do
20     $C$ := $\{(b', h') \in N \mid h' \vdash h\}$
21     if $A' \vdash \bigvee\{b' \mid (b', h') \in C\}$ then
22       $M$ := $M \cup \{(b, h)\}$
23     endif
24   endfor
25   if $M = \emptyset$ then
26     return No
27   else
28     $N'$ := $N' \cup M$
29     $\overline{N}$ := $\overline{N} \setminus M$
30   endif
31 end while
32 return Yes
```

Listing 4: Decision procedure $\mathsf{IO}_4^{\vdash}$ for IN-OUT$_4$.

in the iterative process, no new norms have been triggered (i.e. M is empty), the procedure returns `No`. In the opposite case, the loop terminates with return value `Yes` as soon as the set of triggered norms N' entails the output x (cf. `while` condition in line 16).

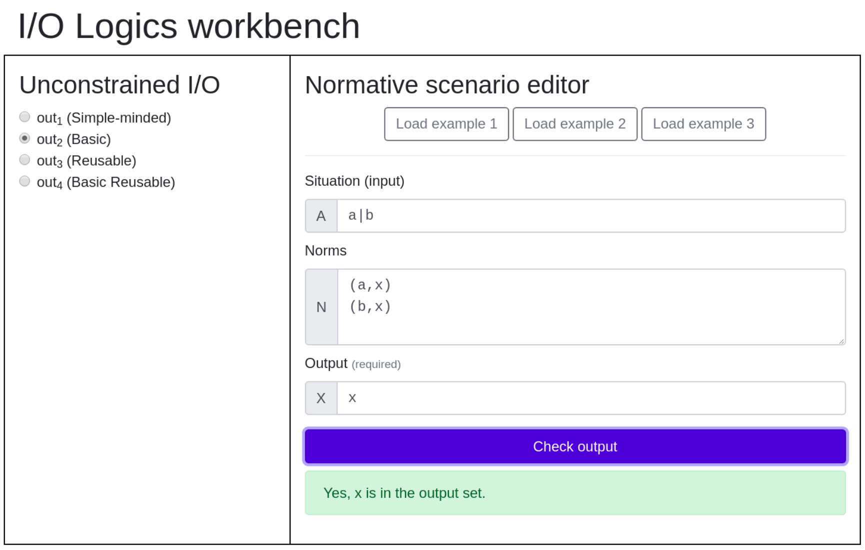
The following results establish adequateness for IN-OUT$_4$:

Theorem 3.7 (Partial correctness of $\mathsf{IO}_4^{\vdash}$) *$\mathsf{IO}_4^{\vdash}$ is sound and complete for* IN-OUT$_4$*; in particular, $x \in out_4(N, A)$ if and only if $(N, A, x) \in \mathsf{IO}_4^{\vdash}$.*

Proof The argument is analogous to the proof of Theorem 3.5. However, in every incremental step, the set of triggered norms corresponds to the basic output, hence accommodating the principle of reasoning by cases analogously to Theorem 3.3. □

Theorem 3.8 (Total correctness of $\mathsf{IO}_4^{\vdash}$) *Let $\vdash$ be a decidable derivation relation for L. $\mathsf{IO}_4^{\vdash}$ terminates and is sound and complete for* IN-OUT$_4$*.*

Proof Theorem 3.7 already yields soundness and completeness. As $\vdash$ is decidable and every input is finite, every loop iteration itself terminates. There



○ out_1 (Simple-minded)
● out_2 (Basic)
○ out_3 (Reusable)
○ out_4 (Basic Reusable)

Situation (input)
A a|b

Norms
N (a,x)
(b,x)

Output (required)
X x

Yes, x is in the output set.

Figure 1. The I/O Logic Workbench: An open-source implementation of the $\mathsf{IO}_i^{\vdash}$ procedures as a browser-based application.

is only a finite number of iterations as the set M is monotonously decreasing and $Cn(.)$ is monotone and idempotent. If no new norms can be triggered, the loop is terminated. $\square$

4 Implementation

A prototype implementation of the decision procedures presented in this paper is freely available as an open-source software library at GitHub. [2]

This above library constitutes the basis for the I/O Logics Workbench (IOLW) that provides graphical means for reasoning in I/O logics. IOLW is a browser-based application and is implemented in JavaScript. There is no need for any backend server infrastructure, as IOLW is implemented purely as a client-side application. Hence, it runs in every reasonably current browser, ready-to-use for conducting own experiments without any installation or set-up. An instance of IOLW is hosted at the author's personal web site. [3]

The user interface of IOLW is presented in Fig. 1. In the left menu panel, a user can choose which *out* operation should be used for the reasoning process. On the right side, the input A, the set of norms N and a prospective output x can be entered. The input language is an ASCII representation of propositional logic, where `|`, `&` and `~` denote disjunction, conjunction and negation, respectively. The input A is a comma separated list of formulas, whereas the set of norms N is, as usual, represented as a set of pairs. Every norm is entered

[2] See `github.com/I-O-Logic` for the source code files and further information.

[3] See `alexandersteen.de/iol` for details.

as a separate line in the text area. Additionally, some example scenarios can be loaded using the respective buttons at the top.

The implementation of the decision procedure library and the IOLW will be extended with further I/O operations and input logics, cf. further work in §5 below for more details.

5 Conclusion

In this paper four decision procedures are presented, one of each out_i operation, $i \in \{1, 2, 3, 4\}$, that abstract from the underlying classical logical language L. These procedures are designed to decide whether a given formula $x \in L$ is in the output $out_i(N, A)$, given a set of norms N and an input A. They are shown to be correct (sound and complete) and to be decidable if the derivation relation $\vdash$ of the underlying logic L is decidable.

Instead of deciding for every prospective output $x \in L$ individually, the ideas underlying the decision procedures can also be used to calculate a finite base $\{x_1, \ldots, x_n\} \subset L$ of the output set $out_i(N, A)$ itself. Such a set can be constructed by modifying the presented procedures in such a way that all triggered norms are collected in a result set. Deciding $\textsc{In-Out}_i$ can then be reduced to checking entailment with respect to $\{x_1, \ldots, x_n\}$.

The output operators with so-called *throughput* [8], denoted out_i^+ for $i \in \{1, 2, 3, 4\}$, can easily be covered by the procedures presented in this paper: Intuitively, these operators behave similar to the respective operators without throughput with the exception that the input A is incorporated into the output set (in addition to the generated output). It is known that out_2^+ and out_4^+ coincide and that they collapse to classical consequence (cf. [9] for details). Moreover, the operators out_1^+ and out_3^+ can be expressed in terms of their non-throughput counterpart [9]. As a consequence, the decision procedures for all the out_i^+ operators can simply be reduced to the underlying routines for $\vdash$ (in case of out_2^+ and out_4^+) and to the routines for out_1 and out_3 (in case of out_1^+ and out_3^+, respectively).

On the practical side the procedures are quite simple to implement, since already existing implementations of decision procedures for $\vdash$ can be used as a black box. A prototypical browser-based implementation for classical propositional logic as underlying logical language is presented. The implementation is open-source, publicly available at GitHub, and can be used for conducting (small) independent normative experiments. The browser-based graphical user interface is primarily intended to serve as a pedagogical tool, e.g., to be used in university teaching for a more interactive exposure to logical reasoning. However, the decision procedures themselves can easily be used as general components in larger software systems.

Future work. The presented procedures only address *unconstrained* input/output operations. While they are interesting operations for different applications, it is pointed out in the literature that they are not fully fit for usage in normative and deontic context [9]; e.g. due to lack of robustness to the usual deontic paradoxes. Further work thus focuses on generalizing the procedures to *constrained* input/output logics [9] that address these aspects.

It is planned to investigate whether the presented decision procedures may contribute to the practical employment of so-called *logical input/output nets* [6], *lions* for short, which combine different normative systems and output operators in a graph structure. Each node of a lion could be implemented by an independent instance of some appropriate $\mathsf{IO}_i^{\vdash}$ procedure.

Furthermore, a prototypical implementation for first-order logic as an underlying formalism is ongoing work. Also, empirical studies have to be conducted for assessing the practical effectiveness of the proposed approach for larger normative systems, e.g., in the context of reasoning with large legal knowledge bases [13].

Finally, the computational approach presented in this paper may be generalized to allow for the employment of non-classical logics as underlying logical formalism, e.g., for intuitionistic I/O logics [11] and further variants.

Acknowledgements

The author would like to thank the anonymous reviewers for their valuable feedback that led to significant improvements of the paper.

References

[1] Åqvist, L.: Deontic logic. In: Handbook of philosophical logic, pp. 147–264. Springer (2002)

[2] Benzmüller, C., Farjami, A., Meder, P., Parent, X.: I/O logic in HOL. FLAP **6**(5), 715–732 (2019)

[3] Benzmüller, C., Farjami, A., Parent, X.: A dyadic deontic logic in HOL. In: Broersen, J.M., Condoravdi, C., Shyam, N., Pigozzi, G. (eds.) DEON. pp. 33–49. College Publications (2018)

[4] Benzmüller, C., Parent, X.: I/O logic in HOL – First Steps. CoRR **abs/1803.09681** (2018), `http://arxiv.org/abs/1803.09681`

[5] Boella, G., van der Torre, L.W.N.: Regulative and constitutive norms in normative multiagent systems. In: KR. pp. 255–266. AAAI Press (2004)

[6] Boella, G., van der Torre, L.W.N.: A logical architecture of a normative system. In: DEON. Lecture Notes in Computer Science, vol. 4048, pp. 24–35. Springer (2006). https://doi.org/10.1007/11786849_5

[7] Gabbay, D., et al. (eds.): Handbook of Deontic Logic and Normative Systems. College Publications (2013)

[8] Makinson, D., van der Torre, L.W.N.: Input/Output Logics. J. Philosophical Logic **29**(4), 383–408 (2000). https://doi.org/10.1023/A:1004748624537

[9] Makinson, D., van der Torre, L.W.N.: Constraints for Input/Output Logics. J. Philosophical Logic **30**(2), 155–185 (2001). https://doi.org/10.1023/A:1017599526096

[10] Makinson, D., van der Torre, L.W.N.: Permission from an input/output perspective. J. Philosophical Logic **32**(4), 391–416 (2003). https://doi.org/10.1023/A:1024806529939

[11] Parent, X., Gabbay, D., Torre, L.v.d.: Intuitionistic basis for input/output logic. In: Hansson, S.O. (ed.) David Makinson on Classical Methods for Non-Classical Problems, pp. 263–286. Springer Netherlands, Dordrecht (2014). https://doi.org/10.1007/978-94-007-7759-0_13

[12] Parent, X., van der Torre, L.W.N.: I/O logics with a consistency check. In: Broersen, J.M., Condoravdi, C., Shyam, N., Pigozzi, G. (eds.) DEON. pp. 285–299. College Publications (2018)

[13] Robaldo, L., et al.: Formalizing GDPR provisions in reified I/O logic: the DAPRECO knowledge base. Journal of Logic, Language and Information (2019). https://doi.org/10.1007/s10849-019-09309-z

[14] Robinson, J.A., Voronkov, A. (eds.): Handbook of Automated Reasoning (in 2 volumes). Elsevier and MIT Press (2001)

[15] Stolpe, A.: Norms and Norm-System Dynamics. Ph.D. thesis, University of Bergen, Norway (2008)

[16] Straßer, C., Beirlaen, M., Putte, F.V.D.: Adaptive logic characterizations of input/output logic. Studia Logica **104**(5), 869–916 (2016). https://doi.org/10.1007/s11225-016-9656-1

[17] Sun, X., Robaldo, L.: On the complexity of input/output logic. Journal of Applied Logic **25**, 69 – 88 (2017). https://doi.org/10.1016/j.jal.2017.03.002