Constrained Input/Output Logic in HOL: An Algebraic Embedding

Ali Farjami[1], Luca Pasetto[1]

[1]University of Luxembourg, Esch-sur-Alzette, Luxembourg

Abstract

Input/Output (I/O) logic provides a flexible framework for representing conditional norms without reducing them to truth-functional implications. Among its variants, constrained Input/Output logic refines the applicability of norms by introducing constraints that limit when obligations may be detached, enabling a fine-grained account of defeasible normative reasoning. In this paper, we present a formal embedding of I/O logic in Higher-Order Logic (HOL), building on the algebraic approach based on subordination structures and slanted algebras. The embedding is based on the algebraic characterization of input/output operations with intermediate translations into slanted modal algebras. We formalize constrained input/output operations within the algebraic framework and embed them in HOL. We then implement the translation in the Isabelle/HOL proof assistant, enabling automated reasoning within the system. Our approach generalizes previous embeddings of I/O logics and provides a foundation for the analysis and verification of defeasible normative systems.

Keywords

Input/Output Logic, Constrained I/O Logic, Defeasible Normative Reasoning, Subordination Algebras, Slanted Algebras, Algebraic Semantics, Higher-Order Logic, Isabelle/HOL, LogiKEy

1. Introduction

Normative reasoning plays a central role in the formal analysis of rule-based systems, including legal frameworks, multi-agent systems, and AI-based decision-making [41, 13]. Input/Output (I/O) logic, introduced by Makinson and van der Torre, provides a well-established framework for representing norms as conditional rules that associate inputs with outputs [29, 33]. In contrast to classical implication, norms in I/O logic are not treated as truth-functional statements, but rather as devices for generating obligations, permissions, or recommendations from given factual situations [29, 28]. This non-truth-functional perspective enables a more faithful representation of normative reasoning, particularly in contexts involving defeasibility and context sensitivity.

To address limitations of the basic framework, several variants of Input/Output logic have been proposed that refine the applicability of norms [33, 40, 23]. In particular, *constrained Input/Output logic*, introduced by Makinson and van der Torre [30], extends the standard setting by incorporating constraints that restrict the conditions under which norms may be applied. These constraints restrict the detachment of normative conclusions, thereby preventing inappropriate or unintended outputs. As a result, constrained I/O logic supports a more fine-grained account of normative reasoning, capturing context-sensitive behavior, exceptions, and controlled forms of defeasibility [30, 34].

Despite its conceptual importance, constrained Input/Output logic has largely remained at an abstract and meta-theoretical level. In particular, there is a lack of formal embeddings that support mechanized reasoning and the verification of its properties. This gap limits its applicability in areas such as formal verification, explainable AI, and the design of normative multi-agent systems [10, 13]. Moreover, while existing infrastructures such as the *LogiKEy* framework provide powerful tool support for formalizing and automating normative reasoning in Higher-Order Logic (HOL) [5, 10], constrained variants of I/O logic have not yet been fully integrated into this setting.

In this paper, we address this gap by providing a formal embedding of I/O logic in HOL, based on its algebraic semantics, and subsequently extending this embedding to constrained Input/Output logic. Building

24th International Workshop on Nonmonotonic Reasoning, July 17–19, 2026, Lisbon, Portugal

farjami110@gmail.com (A. Farjami); luca.pasetto@uni.lu (L. Pasetto)

on prior work on the algebraic characterization of I/O logic [18, 16] (see Proposition 2.2) and existing embeddings of standard I/O logic in HOL [6], we first develop a faithful HOL representation of input/output operations. We then extend this formalization to capture constrained output operations, thereby accommodating restrictions on the detachment of normative conclusions. Our approach is developed within the LogiKEy methodology, enabling a uniform and expressive environment for combining logical systems and supporting normative reasoning tasks. The resulting embedding provides a precise, machine-checkable semantics and facilitates automated reasoning in proof assistants such as Isabelle/HOL [31].

A key aspect of our approach is the use of algebraic semantics as an intermediate layer in the embedding. Algebraic structures, such as subordination algebras and related frameworks, provide a uniform and compositional account of input/output operations [18, 16, 15]. This algebraic perspective is crucial in our setting: it allows us to first obtain a modular and structurally transparent embedding of standard I/O logic, and then systematically extend it to the constrained case. In particular, constraints on the applicability of norms can be naturally represented at the algebraic level and subsequently transferred to their HOL counterparts. This layered approach serves as a bridge between the abstract theory of I/O logic and its formal realization in HOL within the LogiKEy framework, ensuring both faithfulness to the underlying semantics and suitability for mechanization and automated reasoning.

The main contributions of this paper are threefold. First, we provide a formal embedding of Input/Output logic in Higher-Order Logic based on its algebraic modal semantics, yielding a faithful and compositional representation of input/output operations. Second, we systematically extend this embedding to *constrained* Input/Output logic [30], formalizing *maxfamilies*, *outfamilies*, and constrained output operators within the same algebraic framework. This extension constitutes the main novelty of the present work, as previous HOL embeddings focused exclusively on unconstrained output operations. Third, we implement the resulting formalization within the LogiKEy methodology [10, 5], thereby enabling mechanized reasoning and verification of constrained normative systems in *Isabelle/HOL* [31].

Together, these results establish a uniform pipeline from algebraic semantics to machine-checked reasoning. In contrast to previous HOL embeddings [6], which relied on separate modal encodings for different output operations, the present approach exploits the algebraic characterization of output operators (Proposition 2.2) as a common semantic layer. This yields a modular architecture in which both unconstrained and constrained I/O systems can be represented and reasoned about within a single framework.

The remainder of the paper is structured as follows. Section 2 reviews unconstrained and constrained I/O logic and recalls the algebraic perspective underlying our approach. Section 3 describes the LogiKEy methodology and classical Higher-Order Logic (HOL). Section 4 presents the shallow semantical embedding in HOL. Sections 5 and 6 describe the Isabelle/HOL implementation of unconstrained and constrained I/O logics, including the Chisholm example. Section 7 discusses related work, and Section 8 concludes.

2. Background

2.1. Input/Output Logic

We recall the standard definition of Input/Output logic, following [29, 33] and our previous work [16]. Let $\mathscr{L} = (\mathrm{Fm}, \vdash)$, s.t. Fm is the language of classical propositional logic over a given (denumerable) set Prop of proposition variables, and $\vdash \subseteq \mathscr{P}(\mathrm{Fm}) \times \mathrm{Fm}$ is the entailment relation of classical propositional logic. Throughout this paper, *formulas*, i.e. elements in Fm will be denoted by lowercase Greek letters, and sets of formulas by uppercase Greek letters. For any $\Gamma \subseteq \mathrm{Fm}$, let $Cn(\Gamma) := \{\varphi \in \mathrm{Fm} \mid \Gamma \vdash \varphi\}$. A *normative system* on $\mathscr{L}$ is a relation $N \subseteq \mathrm{Fm} \times \mathrm{Fm}$, the elements (α, φ) of which are called *conditional norms* (or obligations).

For any $\Gamma \subseteq \mathrm{Fm}$, let $N(\Gamma) := \{\psi \mid \exists \alpha(\alpha \in \Gamma \;\&\; (\alpha, \psi) \in N)\}$. An *Input/Output logic* is a tuple $\mathbb{L} = (\mathscr{L}, N)$ s.t. $\mathscr{L}$ is a classical propositional logic, and N is a normative system on $\mathscr{L}$.

For any Input/Output logic $\mathbb{L} = (\mathscr{L}, N)$, and each $1 \leq i \leq 4$, the output operation $out_i(N)$ is defined as follows: for any $\Gamma \subseteq \mathrm{Fm}$,

$$out_i(N, \Gamma) := N_i(\Gamma) = \{\psi \in \mathrm{Fm} \mid \exists \alpha(\alpha \in \Gamma \;\&\; (\alpha, \psi) \in N_i)\}$$

where $N_i \subseteq \mathrm{Fm}\times\mathrm{Fm}$ is the *closure* of N under (i.e. the smallest extension of N satisfying) the inference rules below, as specified in Table 1.

$$\frac{}{(\top,\top)}\ (\top) \qquad \frac{}{(\bot,\bot)}\ (\bot) \qquad \frac{(\alpha,\varphi) \quad \beta\vdash\alpha}{(\beta,\varphi)}\ (\mathrm{SI}) \qquad \frac{(\alpha,\varphi) \quad \varphi\vdash\psi}{(\alpha,\psi)}\ (\mathrm{WO})$$

$$\frac{(\alpha,\varphi) \quad (\alpha,\psi)}{(\alpha,\varphi\wedge\psi)}\ (\mathrm{AND}) \qquad \frac{(\alpha,\varphi) \quad (\beta,\varphi)}{(\alpha\vee\beta,\varphi)}\ (\mathrm{OR}) \qquad \frac{(\alpha,\varphi) \quad (\alpha\wedge\varphi,\psi)}{(\alpha,\psi)}\ (\mathrm{CT})$$

N_i	Rules
N_1	$(\top)$,(SI),(WO),(AND)
N_2	$(\top)$,(SI),(WO),(AND),(OR)
N_3	$(\top)$,(SI),(WO),(AND),(CT)
N_4	$(\top)$,(SI),(WO),(AND),(OR),(CT)

Table 1: Closures of normative systems

2.2. Constrained Input/Output Logic

In the original approach proposed by Makinson and van der Torre [30], the contrary-to-duty problem is resolved according to the maxfamily of norms in which the output is consistent with some constraint. A set of formulas A is *consistent with* C iff $Cn(A\cup C)\neq\mathrm{Fm}$.

Definition 2.1 (Constrained I/O logic)**.** *Fix an output operation out_i. For a normative system N, input family A, and constraint family C, define*

$$\begin{aligned} maxfamily_i(N,A,C) &= \{N'\subseteq N \mid out_i(N',A) \text{ is consistent with } C \\ &\qquad \text{and for every } N'' \text{ with } N'\subset N''\subseteq N,\ out_i(N'',A) \text{ is not consistent with } C\}, \\ outfamily_i(N,A,C) &= \{out_i(N',A) \mid N'\in maxfamily_i(N,A,C)\}. \end{aligned}$$

The skeptical and credulous constrained outputs are respectively

$$out^{\cap}_{i,c}(N,A,C)=\bigcap outfamily_i(N,A,C), \qquad out^{\cup}_{i,c}(N,A,C)=\bigcup outfamily_i(N,A,C).$$

When the constraints are taken to be the factual input itself, we write $out^{\cap}_{i,c}(N,A)$ and $out^{\cup}_{i,c}(N,A)$ for the special case $C=A$.

A standard example of contrary-to-duty reasoning is Chisholm's scenario:

(1) It ought to be that Jones helps his neighbours.
(2) If Jones helps them, he ought to tell them he is coming.
(3) If Jones does not help them, he ought not tell them he is coming.
(4) Jones does not help them.

Using atoms h and t, this can be represented by

$$N_{Ch}=\{(\top,h),(h,t),(\neg h,\neg t)\}, \qquad A=\{\neg h\}.$$

For the unconstrained operation $out_3(N_{Ch})$, both t and $\neg t$ become derivable in the violation context. In the constrained setting with $C=A$, the primary norm $(\top,h)$ is blocked by the constraint $\neg h$, and the intended contrary-to-duty conclusion $\neg t$ is recovered from the maximal consistent subfamily.

The *outfamily* operations in the constrained version of I/O logic [30] can be used for reasoning about contrary-to-duties. There are syntactical ([30], Section 6) and proof-theoretical ([40], Section 3) characterizations for constrained I/O logic.

2.3. Algebras of Input/Output Logic

Before introducing the algebraic construction underlying our framework, we briefly recall the semantical structures on which it is based. In recent work, slanted algebras have been proposed as a generalization of modal algebras, where modal operations are interpreted as maps from an algebra into its canonical extension [17]. These operations are not necessarily endomaps on the base algebra, but instead target the lattice of open or closed elements of the canonical extension, thereby enabling a refined treatment of modality that captures asymmetries such as obligation, permission, or temporal orientation [18]. The origin of slanted algebras lies in relational structures known as (proto-)subordination algebras, which consist of an ordered algebra equipped with a binary relation encoding a notion of normative or informational dependency [15]. These structures provide a natural semantical environment for modelling normative systems, where the relation captures how certain states or conditions give rise to others [16]. Importantly, under suitable conditions, subordination relations give rise to slanted modal operators via algebraic constructions on the canonical extension, thus bridging relational and algebraic perspectives [15]. This correspondence allows us to interpret normative transformations—such as the generation of outputs from inputs—in purely algebraic terms, paving the way for their integration into higher-order logical frameworks.

A normative system $N \subseteq \mathrm{Fm} \times \mathrm{Fm}$ can be naturally viewed as inducing a binary relation on formulas, where each conditional norm (α, β) is interpreted as a dependency $\alpha \prec \beta$. This relational perspective provides the bridge between Input/Output logic and the algebraic structures introduced below, where such dependencies give rise to operators capturing output generation.

A *proto-subordination algebra* is a structure $\mathbb{S} = (A, \prec)$, where A is an ordered algebra (e.g. a lattice or Boolean algebra) and $\prec \subseteq A \times A$ is a binary relation encoding a form of dependency or norm activation. Intuitively, $a \prec b$ expresses that the condition a gives rise to, or supports, the outcome b. Different closure properties of the relation $\prec$ correspond to different variants of input/output operations.

From such relational structures, one can systematically construct algebraic operators via canonical extensions [21]. This leads to the notion of *slanted algebras*, in which modal operators are interpreted as maps from the base algebra A into its canonical extension A^δ [17]. In particular, a slanted algebra is a structure $\mathbb{A} = (A, \Diamond)$ where $\Diamond : A \to A^\delta$ captures the algebraic counterpart of output generation, typically defined from the relation $\prec$ via suitable meet or join constructions. Each conditional norm (α, β) can be captured algebraically as a constraint on the operator $\Diamond$, by interpreting $\alpha \preceq \beta$ as the inequality $\Diamond\alpha \leq \beta$. In the algebraic setting, each output system $out_i(N)$ of Input/Output logic corresponds to a class of $\Diamond$-algebras within the hierarchy of proto-subordination algebras . The correspondence highlights how the closure properties of the $\prec$-relation translate into structural requirements on the slanted operator $\Diamond$. In particular:

- $out_1(N)$ captures the case where $\Diamond$ is *monotone*, i.e. $\Diamond$-defined: $a \leq b \Rightarrow \Diamond a \leq \Diamond b$
- $out_2(N)$ corresponds to $\Diamond$ being *regular*, i.e. monotone plus : $\Diamond(a \vee b) = \Diamond a \vee \Diamond b$.
- $out_3(N)$ assumes $\Diamond$ is *monotone*, and requires in addition the *cumulative condition* $\Diamond a \leq \Diamond(a \wedge \Diamond a)$.
- $out_4(N)$ assumes $\Diamond$ is *regular*, together with the same cumulative condition.

I/O system	$\Diamond$-algebra type	Extra condition
$out_1(N)$	$\Diamond$-monotone	–
$out_2(N)$	$\Diamond$-regular	–
$out_3(N)$	$\Diamond$-monotone	$\Diamond a \leq \Diamond(a \wedge \Diamond a)$
$out_4(N)$	$\Diamond$-regular	$\Diamond a \leq \Diamond(a \wedge \Diamond a)$

Table 2: Correspondence between I/O systems and $\Diamond$-algebra classes

The output operators $out_i(N)$ for $1 \leq i \leq 4$ associated with a given Input/Output logic $\mathbb{L} = (\mathscr{L}, N)$ can be given semantical counterparts in the environment of proto-subordination algebras as follows: for every proto-subordination algebra $\mathbb{S} = (A, \prec)$, we let $\mathbb{S}_i := (A, \prec_i)$ where $\prec_i \subseteq A \times A$ is the smallest extension of $\prec$ which satisfies the properties indicated in outputs 1 to 4 in I/O systems.

Proposition 2.2. *For any defined*[1] *proto-subordination algebra* $\mathbb{S}=(A,\prec)$,

1. $\Diamond_1$ *is the largest monotone map dominated by* $\Diamond$ *(i.e. pointwise-smaller than or equal to* $\Diamond$*).*
2. $\Diamond_2$ *is the largest regular map dominated by* $\Diamond$*.*
3. $\Diamond_3$ *is the largest monotone map satisfying* $\Diamond_3 a \leq \Diamond_3(a \wedge \Diamond_3 a)$ *dominated by* $\Diamond$*.*
4. $\Diamond_4$ *is the largest regular map satisfying* $\Diamond_4 a \leq \Diamond_4(a \wedge \Diamond_4 a)$ *dominated by* $\Diamond$*.*

See Proposition 6.1 in [15]. □

Proposition 2.2 is the key result underlying our embedding. It characterizes the output operations of I/O logic as largest operators satisfying suitable modal conditions, enabling their direct representation in HOL.

3. LogiKEy: Implementation Methodology

Our implementation is based on the *LogiKEy* methodology [10], which supports semantical embeddings of a wide variety of logical systems into Higher-Order Logic (HOL). These embeddings preserve the semantics of each target logic within a unified formal meta-language, allowing them to re-use infrastructure such as proof assistants (e.g., Isabelle/HOL [31]), model finders (e.g., Nitpick [11]), and theorem provers (e.g., Sledgehammer [12], Leo-III [39]). The LogiKEy framework currently supports a range of logics relevant to normative reasoning and beyond:

- **Modal Logic KD**: Captures that obligations imply permissions (axiom D) and supports basic modal reasoning [3, 4].
- **Conditional Logic (System E)**: Supports reasoning about conditional norms, exceptions, and contrary-to-duty obligations [7, 8, 32].
- **Input/Output Logic (I/O Logic)**: Models conditional norms as input/output operations, supporting factual and deontic detachment [6, 22].

Further applications of the LogiKEy methodology can be found in [9, 26, 35, 36, 37].

3.1. Classical Higher-Order Logic

Types and Syntax. HOL is based on simple typed λ-calculus. We follow standard presentations of Higher-Order Logic [2, 25]. We assume that the set $\mathscr{T}$ of simple types is freely generated from a set of basic types $\{o,i\}$ using the function type constructor $\rightarrow$. Type o denotes the set of Booleans whereas type i refers to a non-empty set of individuals.

For $\alpha,\beta,o \in \mathscr{T}$, the *language of HOL* is generated as follows:

$$s,t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha s_\beta)_{\alpha\rightarrow\beta} \mid (s_{\alpha\rightarrow\beta} t_\alpha)_\beta$$

where p_α represents a typed constant symbol (from a possibly infinite set $\mathscr{P}_\alpha$ of such constant symbols) and X_α represents a typed variable symbol (from a possibly infinite set $\mathscr{V}_\alpha$ of such symbols). $(\lambda X_\alpha s_\beta)_{\alpha\rightarrow\beta}$ and $(s_{\alpha\rightarrow\beta} t_\alpha)_\beta$ are called *abstraction* and *application*, respectively. HOL is a logic of terms in the sense that the *formulas of HOL* are given as terms of type o.

Moreover, we require a sufficient number of primitive logical connectives in the signature of HOL, i.e., these logical connectives must be contained in the sets $\mathscr{P}_\alpha$ of constant symbols. The primitive logical connectives of choice in this paper are $\neg_{o\rightarrow o}$, $\vee_{o\rightarrow o\rightarrow o}$, $\Pi_{(\alpha\rightarrow o)\rightarrow o}$ and $=_{\alpha\rightarrow\alpha\rightarrow o}$. The symbols $\Pi_{(\alpha\rightarrow o)\rightarrow o}$ and $=_{\alpha\rightarrow\alpha\rightarrow o}$ generally assumed for each type $\alpha \in \mathscr{T}$. From the selected set of primitive connectives, other logical connectives can be introduced as abbreviations. Type information as well as brackets may

[1] A defined proto-subordination algebra satisfies the following conditions: (SF) $\forall a \exists x(a \prec x)$ and (DD) $\forall a \forall x_1 \forall x_2\big((a \prec x_1 \,\&\, a \prec x_2) \Rightarrow \exists x(a \prec x \,\&\, x \leq x_1 \,\&\, x \leq x_2)\big)$. In the Boolean setting considered here, (SF) and (DD) are implied by the stronger closure conditions ($\top$), (SI), and (AND), and therefore need not be assumed explicitly.

be omitted if obvious from the context, and we may also use infix notation to improve readability. For example, we may write $(s \vee t)$ instead of $((\vee_{o\to o\to o} s_o)t_o)_o$. We often write $\forall X_\alpha s_o$ as syntactic sugar for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha s_o)$. Here, Π denotes universal quantification over all objects of the given type.

The notions of *free variables*, *α-conversion*, *βη-equality* and *substitution* of a term s_α for a variable X_α in a term t_β, denoted as $[s/X]t$, are defined as usual.

Semantics. The semantics of HOL are well understood and thoroughly documented [2]. In the remainder, the semantics of choice is Henkin's general models [25].

A *frame* D is a collection $\{D_\alpha\}_{\alpha\in\mathcal{T}}$ of nonempty sets D_α, such that $D_o = \{T, F\}$, denoting truth and falsehood, respectively. $D_{\alpha\to\beta}$ represents a collection of functions mapping D_α into D_β.

A *model* for HOL is a tuple $M = \langle D, I\rangle$, where D is a frame and I is a family of typed interpretation functions mapping constant symbols p_α to appropriate elements of D_α, called the *denotation of* p_α. The logical connectives $\neg$, $\vee$, Π and $=$ are always given in their expected standard denotations. A *variable assignment* g maps variables X_α to elements in D_α. $g[d/W]$ denotes the assignment that is identical to g, except for the variable W, which is now mapped to d. The *denotation* $\|s_\alpha\|^{M,g}$ of a HOL term s_α on a model $M = \langle D, I\rangle$ under assignment g is an element $d \in D_\alpha$ defined in the following way:

$$\begin{aligned}
\|p_\alpha\|^{M,g} &= I(p_\alpha)\\
\|X_\alpha\|^{M,g} &= g(X_\alpha)\\
\|(s_{\alpha\to\beta}t_\alpha)_\beta\|^{M,g} &= \|s_{\alpha\to\beta}\|^{M,g}(\|t_\alpha\|^{M,g})\\
\|(\lambda X_\alpha s_\beta)_{\alpha\to\beta}\|^{M,g} &= \text{the function } f \text{ from } D_\alpha \text{ to } D_\beta \text{ such that}\\
&\quad f(d) = \|s_\beta\|^{M,g[d/X_\alpha]} \text{ for all } d \in D_\alpha
\end{aligned}$$

Since $I(\neg_{o\to o})$, $I(\vee_{o\to o\to o})$, $I(\Pi_{(\alpha\to o)\to o})$ and $I(=_{\alpha\to\alpha\to o})$ always denote the standard truth functions, we have:

1. $\|(\neg_{o\to o} s_o)_o\|^{M,g} = T$ iff $\|s_o\|^{M,g} = F$.
2. $\|((\vee_{o\to o\to o} s_o)t_o)_o\|^{M,g} = T$ iff $\|s_o\|^{M,g} = T$ or $\|t_o\|^{M,g} = T$.
3. $\|(\forall X_\alpha s_o)_o\|^{M,g} = \|(\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha s_o))_o\|^{M,g} = T$ iff for all $d \in D_\alpha$ we have $\|s_o\|^{M,g[d/X_\alpha]} = T$.
4. $\|((=_{\alpha\to\alpha\to o} s_\alpha)t_\alpha)_o\|^{M,g} = T$ iff $\|s_\alpha\|^{M,g} = \|t_\alpha\|^{M,g}$.

A HOL formula s_o is *true* in a Henkin model M under the assignment g if and only if $\|s_o\|^{M,g} = T$. This is also expressed by the notation $M, g \models^{HOL} s_o$. A HOL formula s_o is called *valid* in M, denoted as $M \models^{HOL} s_o$, if and only if $M, g \models^{HOL} s_o$ for all assignments g. Moreover, a formula s_o is called *valid*, denoted as $\models^{HOL} s_o$, if and only if s_o is valid in all Henkin models M. Finally, we define $\Sigma \models^{HOL} s_o$ for a set of HOL formulas Σ if and only if $M \models^{HOL} s_o$ for all Henkin models M with $M \models^{HOL} t_o$ for all $t_o \in \Sigma$.

4. Shallow Semantical Embedding of Slanted Algebras in HOL

The implementation adopts a shallow semantical embedding: object-level formulas, norms, and output operators are represented directly as HOL terms, while their intended behaviour is captured by HOL definitions and axioms. Hence the embedding does not introduce a separate syntactic proof calculus inside HOL; instead, it works directly with semantical objects that can be manipulated in Isabelle/HOL.

Let i be a type of algebra points and let $\tau := i \to o$. At the implementation level, where the HOL truth type o is realized by Isabelle/HOL's type `bool`, this becomes $\tau = i \Rightarrow bool$. Thus formulas are represented as predicates of type τ. We consider the basic language of slanted algebras

$$\varphi ::= p^j \mid \neg\varphi \mid \varphi\vee\psi \mid \varphi\wedge\psi \mid \top \mid \bot \mid \Diamond\varphi.$$

Its translation into HOL is given by the mapping $\lfloor\cdot\rfloor$, defined recursively as follows:

$$\begin{array}{rcl} \lfloor p^j \rfloor & = & p^j_\tau, \\ \lfloor \neg\varphi \rfloor & = & \lambda w_i.\neg(\lfloor\varphi\rfloor w), \\ \lfloor \varphi\vee\psi \rfloor & = & \lambda w_i.(\lfloor\varphi\rfloor w\vee\lfloor\psi\rfloor w), \\ \lfloor \varphi\wedge\psi \rfloor & = & \lambda w_i.(\lfloor\varphi\rfloor w\wedge\lfloor\psi\rfloor w), \\ \lfloor \top \rfloor & = & \lambda w_i.True, \\ \lfloor \bot \rfloor & = & \lambda w_i.False. \end{array}$$

If $\varphi\leq\psi$ is an inequality of the algebraic language, its translation is the HOL formula

$$\lfloor\varphi\leq\psi\rfloor = \forall w_i.(\lfloor\varphi\rfloor w\rightarrow\lfloor\psi\rfloor w).$$

Hence the algebraic order is represented by pointwise inclusion of predicates. Families of formulas are represented by predicates $A:\tau\Rightarrow bool$, and their infimum is defined by

$$\bigwedge A := \lambda w.\forall X_\tau.(A(X)\rightarrow X(w)).$$

This yields the HOL definitions of consequence closure and consistency of formula families:

$$Cn(A)(\varphi) := (\bigwedge A\leq\varphi), \qquad fconsistent(A) := (\bigwedge A\neq\bot).$$

The HOL signature is assumed to contain the constant symbol $\preceq_{\tau\rightarrow\tau\rightarrow o}$ representing a normative system, in addition to the logical constants. The constant $\preceq$ represents the underlying subordination relation from which the translation of the slanted operator $\diamond$ is defined as

$$\lfloor\diamond\varphi\rfloor \quad = \quad \bigwedge\{\lfloor x\rfloor\,|\,\lfloor\varphi\rfloor\preceq\lfloor x\rfloor\}.$$

5. Implementation: I/O Logics

The implementation for I/O logics is arranged as a hierarchy of theory files mirroring the semantic layers of the embedding.[2] The common foundation is `IO_LogicalBase.thy`, which imports `Main` and defines the lifted Boolean connectives on τ, the order relation, infimum, closure and consistency operators, the basic norm relation $\preceq$, and the generic slanted operator $\diamond$. On top of this base sit the four independent output theories `IO1.thy`–`IO4.thy`; each of them introduces exactly one operator $\diamond^i_o$ and proves the lemmas corresponding to the rules of the matching output system. Finally, the auxiliary example file `IO3_Ex_C.thy` imports `IO3.thy` and instantiates the generic output-3 theory with concrete normative systems. The dependency pattern is

$$\texttt{IO\_LogicalBase}\longrightarrow\{\texttt{IO1},\texttt{IO2},\texttt{IO3},\texttt{IO4}\}, \qquad \texttt{IO3}\longrightarrow\texttt{IO3\_Ex\_C}.$$

5.1. Logical Base

In `IO_LogicalBase.thy`, formulas are represented by predicates of type $\tau = i\Rightarrow bool$, Boolean connectives are lifted pointwise, and the order $\leq$ is defined extensionally. The theory also defines the infimum operator $\bigwedge$, consequence closure Cn, and consistency of formula families. A normative system is represented by a relation $\preceq$ on formulas ($\preceq\subseteq\tau\times\tau$) , and the associated slanted operator is defined as $\diamond\varphi := \bigwedge\{x\,|\,\varphi\preceq x\}$.

At this level the embedding already validates the fundamental bridge between norms and modal outputs: if $\alpha\preceq\beta$, then $\diamond\alpha\leq\beta$. In addition, the logical base defines the semantical properties used to characterize the four output operators: monotonicity, $\forall\varphi\forall\psi(\varphi\leq\psi\rightarrow op(\varphi)\leq op(\psi))$; regularity, $op(\varphi\vee\psi)=op(\varphi)\vee op(\psi)$; and, for outputs 3 and 4, the cumulative inequality $op(\varphi)\leq op(\varphi\wedge op(\varphi))$.

On top of this generic layer, the concrete output operators $\diamond^1_o,\ldots,\diamond^4_o$ are introduced as the largest operators satisfying the corresponding algebraic conditions of Proposition 2.2. In this sense the embedding is shallow: semantical constraints on the object logic are represented directly at the HOL level, and Isabelle/HOL can reason both within the logic and about its meta-properties.

[2]The full sources of our implementation, including the constrained case, can be found at http://logikey.org/tree/master/Deontic-Logics/IOL-algebraic.

```
theory IO3
  imports IO_LogicalBase
begin

definition out3_admissible :: "(τ ⇒ τ) ⇒ bool"
  where
  "out3_admissible op ≡
      monotone op
    ∧ (∀φ. op φ ≤ op (φ ∧ op φ))
    ∧ (∀φ. op φ ≤ (◇ φ))"

definition largest_out3 :: "(τ ⇒ τ) ⇒ bool"
  where
  "largest_out3 op ≡
      out3_admissible op
    ∧ (∀op1. out3_admissible op1 ⟶ (∀φ. op1 φ ≤ op φ))"

consts IO3 :: "τ ⇒ τ" ("◇³ₒ")

axiomatization where
  ax_IO3: "largest_out3 (◇³ₒ)"
```

Figure 1: Core definitions of `IO3.thy`.

5.2. Input/Output Operations

Each theory `IOi.thy` introduces a HOL constant $\Diamond_o^i : \tau \Rightarrow \tau$ and constrains it by an admissibility condition and a maximality principle. For example, in the case of output 3 the admissible operators are those which are monotone, satisfy the cumulative inequality, and are pointwise dominated by the base slanted operator. The constant $\Diamond_o^3$ is then postulated to be the *largest* such operator. Outputs 1, 2, and 4 are treated analogously, replacing monotonicity by regularity where required and adding the cumulative condition only for outputs 3 and 4. See Figure 1 for the core definitions of output 3 in the implementation.

This design directly implements Proposition 2.2. Once the corresponding maximality axiom is declared, the familiar proof rules of I/O logic become ordinary HOL lemmas. More precisely, monotonicity yields strengthening of the input (SI), pointwise order yields weakening of the output (WO), the pointwise encoding of conjunction yields (AND), regularity yields (OR), and the cumulative inequality yields (CT), and hence also the derived rule (T). The theories `IO3.thy` and `IO4.thy` show in particular that (CT) is derivable for outputs 3 and 4, while `IO3.thy` also uses `Nitpick` to confirm that (OR) is not derivable for output 3, as expected.

5.3. Example: Chisholm's Paradox

The theory `IO3_Ex_C.thy` illustrates the embedding with Chisholm's paradox. Using atoms h and t, the normative system is represented by the three norms $(\top, h)$, (h, t), and $(\neg h, \neg t)$. The first norm expresses the primary obligation to help, the second the obligation to tell if one helps, and the third the contrary-to-duty obligation not to tell if one does not help.

For output 3, we derive $\Diamond_o^3 \top \leq h$, $\Diamond_o^3 \top \leq t$, and $\Diamond_o^3 \neg h \leq \neg t$. Since $\neg h \leq \top$, monotonicity also yields $\Diamond_o^3 \neg h \leq \Diamond_o^3 \top \leq t$. Hence $\Diamond_o^3 \neg h \leq t \wedge \neg t$, and therefore $\Diamond_o^3 \neg h \leq \bot$. This is the expected collapse of the unconstrained contrary-to-duty scenario: the ordinary output operation validates incompatible obligations in the violation context. The constrained extension introduced next is designed precisely to block this effect. See Figure 2 for an excerpt of the encoded example.

6. Implementation: Constrained I/O Logics

The constrained implementation mirrors the unconstrained one, but inserts an additional shared layer for parameterization on a normative system. The file `IO_CON.thy` imports `IO_LogicalBase.thy` and

```
theory IO3_Ex_C
  imports IO3
begin
(* Chisholm *)

consts h :: "τ"
consts t :: "τ"

context
  assumes ax0: "⊤ ≼ h"
      and ax1: "h ≼ t"
      and ax2: "(¬h) ≼ (¬t)"
begin

lemma ch1: "◇³₀ ⊤ ≤ h"
  using IO3_from_norm[OF ax0] by simp

lemma ch2: "◇³₀ ⊤ ≤ t"
  using IO3T IO3_from_norm ax1 ch1 by blast

lemma ch3: "◇³₀ (¬h) ≤ (¬t)"
  using IO3_from_norm[OF ax2] by simp

lemma ch4: "◇³₀ (¬h) ≤ ⊥"
  by (metis IO3_mono IO_LogicalBase.monotone_def
      ch2 ch3 setnot_def settrue_def)
```

Figure 2: Chisholm's paradox in `IO3_Ex_C.thy`.

introduces the type `normsys` of normative systems, the inclusion relation on such systems, and the norm-indexed base slanted operator $\Diamond_c$. Each constrained output theory `IO1CON.thy`–`IO4CON.thy` then imports its unconstrained counterpart together with `IO_CON.thy`; these files are self-contained and define the constrained operator $\Diamond_c^i$, the corresponding admissibility and maximality conditions, a gross-output predicate (`out1`, ..., `out4`, depending on the file), and the output-dependent notions `out_consistent`, `maxfamily`, `outfamily`, `skepoutfamily`, and `credoutfamily`.

For space reasons, in the following we focus on output 3, which is the natural setting for contrary-to-duty reasoning. The example file `IO3CON_Ex20_22.thy` imports `IO3CON.thy` and instantiates the constrained framework with the Chisholm contrary-to-duty example. Thus the output-3 constrained fragment has the import pattern

$$\texttt{IO\_LogicalBase} \longrightarrow \texttt{IO\_CON}, \qquad (\texttt{IO3},\texttt{IO\_CON}) \longrightarrow \texttt{IO3CON}, \qquad \texttt{IO3CON} \longrightarrow \texttt{IO3CON\_Ex20\_22}.$$

6.1. Maxfamily and Outfamily

For a family of factual inputs $A : \tau \Rightarrow bool$, gross output is represented directly at the HOL level. For output 3:

$$out_3(N,A)(\psi) := \Diamond_c^3(N, \bigwedge A) \leq \psi.$$

The definition is analogous for outputs 1, 2, and 4. Consistency with a family of constraints C is encoded by

$$out\_consistent(N,A,C) := \neg\big((\Diamond_c^3(N, \bigwedge A) \wedge \bigwedge C) = \bot\big).$$

Thus the output of a candidate sub-system N_0 is admissible exactly when it does not collapse together with the constraints.

Maximal consistent subfamilies are represented by the predicate $max\,family(N,A,C,N_0)$, which states that $N_0 \subseteq N$, that N_0 is output-consistent with respect to A and C, and that no proper extension of N_0 inside N preserves this consistency. The associated family of outputs is then defined by

$$out\,family(N,A,C,B) := \exists N_0\big(max\,family(N,A,C,N_0) \wedge B = out_3(N_0,A)\big).$$

```
(* Gross output on a family of factual inputs A. *)
definition out3 :: "normsys ⇒ (τ ⇒ bool) ⇒ τ ⇒ bool"
  where "out3 N A ψ ≡ (◇³c N (⋀A)) ≤ ψ"

(* Consistency of out3(N,A) with the constraint set C. *)
definition out_consistent :: "normsys ⇒ (τ ⇒ bool) ⇒ (τ ⇒ bool) ⇒ bool"
  where "out_consistent N A C ≡ ¬ (((◇³c N (⋀A)) ∧ (⋀C)) = ⊥)"

definition maxfamily :: "normsys ⇒ (τ ⇒ bool) ⇒ (τ ⇒ bool) ⇒ normsys ⇒ bool"
  where
  "maxfamily N A C N0 ≡
       N0 ⊑ N
     ∧ out_consistent N0 A C
     ∧ (∀N1. N0 ⊑ N1 ∧ N1 ⊑ N ∧ out_consistent N1 A C ⟶ N1 ⊑ N0)"

definition outfamily :: "normsys ⇒ (τ ⇒ bool) ⇒ (τ ⇒ bool) ⇒ (τ ⇒ bool) ⇒ bool"
  where "outfamily N A C B ≡ ∃N0. maxfamily N A C N0 ∧ B = out3 N0 A"

definition skepoutfamily :: "normsys ⇒ (τ ⇒ bool) ⇒ (τ ⇒ bool) ⇒ τ ⇒ bool"
  where "skepoutfamily N A C ψ ≡ ∀B. outfamily N A C B ⟶ B ψ"

definition credoutfamily :: "normsys ⇒ (τ ⇒ bool) ⇒ (τ ⇒ bool) ⇒ τ ⇒ bool"
  where "credoutfamily N A C ψ ≡ ∃B. outfamily N A C B ∧ B ψ"
```

Figure 3: Core definitions of `IO3CON.thy`.

Finally, skeptical and credulous constrained outputs are obtained by universal and existential quantification over the members of *out family*:

$$skepout\,family(N,A,C)(\psi) := \forall B.(out\,family(N,A,C,B) \rightarrow B(\psi)),$$

$$credout\,family(N,A,C)(\psi) := \exists B.(out\,family(N,A,C,B) \wedge B(\psi)).$$

See Figure 3 for the core definitions of constrained output 3 in the implementation. The contrary-to-duty operators used in the examples are the special cases obtained by setting the constraint family equal to the factual input family, i.e. $out^{\cap}_{i,c}(N,A)$ and $out^{\cup}_{i,c}(N,A)$.

6.2. Example: Chisholm Repaired by Constraints

The theory `IO3CON_Ex20_22.thy` revisits Chisholm in the constrained setting, using atoms h and t. The normative system is

$$N_{Ch} = \{(\top,h),(h,t),(\neg h,\neg t)\},$$

the violation input is $A = \{\neg h\}$, and the standard contrary-to-duty choice of constraints is $C = A$.

The theory first shows that every candidate subfamily that still contains the primary norm $(\top,h)$ conflicts with the constraint $\neg h$ and is therefore not output-consistent. This isolates the repaired family

$$N'_{Ch} = \{(h,t),(\neg h,\neg t)\}$$

as the intended maximal consistent subfamily in the violation context. More precisely, in the example we prove the corresponding maxfamily and uniqueness lemmas under two explicit consistency assumptions: that N'_{Ch} is output-consistent for the violation case, and that the full system N_{Ch} is output-consistent for the compliant case. Under these assumptions, the file shows that N'_{Ch} is the unique maximal family for input $\{\neg h\}$, and therefore both skeptical and credulous constrained outputs derive $\neg t$.

The same file also treats the compliant case $A = \{h\}$. Under the corresponding consistency assumption, the full normative system remains the unique maximal family, and the constrained output contains both h and t. Hence the constrained encoding reproduces the standard reading of Chisholm's scenario: in the compliant case the primary and secondary obligations are preserved, whereas in the violation case the primary norm is blocked and only the contrary-to-duty obligation survives. See Figure 4 for an excerpt of the encoded example for the constrained case.

```
lemma chisholm22_repair_maxfamily:
  "maxfamily N_ch A_not_h A_not_h N_ch_repair"
  by (smt (verit) N_ch_repair_subset_N_ch ch_A_h_infimum
      chisholm22_cons chisholm_primary_conflicts_with_constraint
      maxfamily_def subset_N_ch_without_primary_subset_repair)

lemma chisholm22_unique_maxfamily:
  "maxfamily N_ch A_not_h A_not_h N0 ⟷ N0 = N_ch_repair"
proof [18 lines]
next [3 lines]
qed
```

Figure 4: Chisholm's paradox in `IO3CON_Ex20_22.thy`.

7. Related Work

Algebraic and HOL-based approaches to I/O logic. Previous implementations of input/output logic within the LogiKEy framework have relied on translations into modal logic [6]. Existing translations of I/O logic into modal logic capture only part of its semantical behavior, typically focusing on specific output operations or restricted classes of norms. To overcome these limitations, two main directions have been explored. The first builds on the operational semantics of I/O logic and introduces algebraic operators to obtain sound and faithful embeddings into Higher-Order Logic [22]. The second direction starts from the proof theory of I/O logic—via derivation rules such as AND, OR, and CT—and aligns it with subordination algebras [18] and slanted (co-)Heyting algebras [20], yielding algebraic semantics that support direct implementations of modal operators for obligations and permissions.

Our approach follows this second, algebraic direction and exploits the correspondence between subordination structures and modal operators as an intermediate semantic layer. Compared with previous modal embeddings, the algebraic approach provides a uniform characterization of output operations through Proposition 2.2, which characterizes them as maximal operators satisfying suitable algebraic conditions. This yields a modular framework in which both unconstrained and constrained output operations are treated uniformly. The resulting algebraic semantics induces a translation of I/O reasoning into modal structures, which can then be faithfully embedded into HOL. This intermediate layer enables a computationally tractable pathway—via translations into first-order logic—while supporting a faithful and extensible implementation in HOL [19].

Connections to modal logic and proof-theoretic frameworks. Since its inception, I/O logic has been closely connected to modal and deontic logic, as such connections enable the transfer of powerful model-theoretic and proof-theoretic techniques. Modal characterizations make available tools such as tableaux methods, decidability results, and proof search procedures [27, 14], as well as adaptive and defeasible reasoning frameworks [40]. In contrast to syntactic and proof-theoretic characterizations of constrained I/O logic [30, 40], our approach is purely semantical and algebraic. This perspective is particularly useful for constrained input/output operations, where restrictions on norm applicability can be naturally captured by considering suitable subalgebras or refinements of the underlying subordination structures corresponding to a given input/output relation. This provides a flexible basis for integration with automated reasoning tools.

Alternative implementations and computational perspectives. From a computational standpoint, several alternative approaches to automating I/O logic have been proposed. SAT-based encodings [38] provide efficient implementations by reducing normative reasoning tasks to propositional satisfiability, offering strong performance but typically at the cost of reduced expressivity and limited support for meta-reasoning. In contrast, HOL-based encodings [10] are more expressive and enable reasoning not only within normative systems but also about them, supporting tasks such as meta-logical analysis and ethical theory comparison, albeit with higher computational overhead.

Related work on conditional logics, such as small model constructions for Åqvist systems, further contributes to understanding the computational behavior of normative reasoning frameworks. In parallel, other logics from the tradition of normative systems, such as defeasible deontic logic [24], provide alternative approaches for handling exceptions, conflicts, and priorities, which are central features of real-world normative reasoning. These directions are increasingly relevant in emerging settings such as hybrid symbolic–statistical systems and large language models (LLMs), where defeasibility and context sensitivity play a key role [23].

8. Conclusion

In this paper, we have presented a formal embedding of Input/Output logic in Higher-Order Logic (HOL), based on its algebraic semantics, and extended this embedding to capture constrained Input/Output logic. By leveraging algebraic structures as an intermediate layer, we obtained a modular and faithful representation of input/output operations and their constrained variants. This approach enabled a systematic translation into HOL and provided a precise, machine-checkable semantics for constrained normative reasoning.

The implementation of our framework within the LogiKEy methodology demonstrates the practical applicability of our approach. In particular, it supports automated reasoning and verification of constrained normative systems in Isabelle/HOL, thereby bridging the gap between abstract logical theory and mechanized reasoning tools. Our results show that algebraic semantics offers a robust foundation for integrating expressive variants of I/O logic into existing reasoning infrastructures.

Several directions for future work remain. On the theoretical side, further investigation of additional variants of I/O logic and their algebraic counterparts may enrich the expressiveness of the framework. On the practical side, extending the implementation to support more complex normative scenarios, including dynamic and multi-agent settings or inconsistency measures [1], would enhance its applicability. More broadly, we envisage the integration of this approach into applications in explainable AI and normative system design, where transparent and verifiable reasoning about norms is essential [23].

Acknowledgments

We thank the anonymous reviewers for their constructive comments. We are grateful to all LogiKEy collaborators, and in particular to Christoph Benzmüller for pioneering the research line on shallow embeddings of non-classical logics in HOL. We also thank Alessandra Palmigiano for her helpful feedback on earlier versions of this work. This work was supported by the Luxembourg National Research Fund (FNR) through the project Logical methods for Deontic Explanations (INTER/DFG/23/17415164/LODEX).

Declaration on Generative AI

Generative AI tools were used for language editing and proofreading. All research contributions, formalizations, proofs, and conclusions are the work of the authors.

References

[1] O. Arieli, K. van Berkel, B. Raddaoui, and C. Strasser. "Deontic Reasoning Based on Inconsistency Measures". In: *KR 2024*. 2024, pp. 71–81. DOI: 10.24963/kr.2024/7.

[2] C. Benzmüller, C. Brown, and M. Kohlhase. "Higher-Order Semantics and Extensionality". In: *Journal of Symbolic Logic* 69.4 (2004), pp. 1027–1088.

[3] C. Benzmüller and L. Paulson. "Multimodal and Intuitionistic Logics in Simple Type Theory". In: *Logic Journal of the IGPL* 18.6 (2010), pp. 881–892. DOI: 10.1093/jigpal/jzp080.

[4] C. Benzmüller and L. Paulson. "Quantified Multimodal Logics in Simple Type Theory". In: *Logica Universalis (Special Issue on Multimodal Logics)* 7.1 (2013), pp. 7–20.

[5] C. Benzmüller, A. Farjami, D. Fuenmayor, P. Meder, X. Parent, A. Steen, L. van der Torre, and V. Zahoransky. "LogiKEy Workbench: Deontic Logics, Logic Combinations and Expressive Ethical and Legal Reasoning (Isabelle/HOL Dataset)". In: *Data in Brief* 33 (2020), p. 106409.

[6] C. Benzmüller, A. Farjami, P. Meder, and X. Parent. "I/O Logic in HOL". In: *Journal of Applied Logics – IfCoLoG Journal of Logics and their Applications* 6.5 (2019), pp. 715–732.

[7] C. Benzmüller, A. Farjami, and X. Parent. "A Dyadic Deontic Logic in HOL". In: *Proceedings of DEON 2018*. 2018, pp. 33–50.

[8] C. Benzmüller, A. Farjami, and X. Parent. "Åqvist's Dyadic Deontic Logic E in HOL". In: *Journal of Applied Logics – IfCoLoG Journal of Logics and their Applications* 6.5 (2019), pp. 733–755.

[9] C. Benzmüller, D. Fuenmayor, and B. Lomfeld. "Modelling Value-oriented Legal Reasoning in LogiKEy". In: *Logics* 2.1 (2024), pp. 31–78. DOI: 10.3390/logics2010003.

[10] C. Benzmüller, X. Parent, and L. van der Torre. "Designing Normative Theories for Ethical and Legal Reasoning: LogiKEy Framework, Methodology, and Tool Support". In: *Artificial Intelligence* 237 (2020), p. 103348.

[11] J. C. Blanchette and T. Nipkow. "Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder". In: *ITP 2010*. Vol. 6172. LNCS. 2010, pp. 131–146. DOI: 10.1007/978-3-642-14052-5_11.

[12] J. C. Blanchette, S. Böhme, and L. C. Paulson. "Extending Sledgehammer with SMT solvers". In: *Journal of Automated Reasoning* 51.1 (2013), pp. 109–128.

[13] G. Boella, L. van der Torre, and H. Verhagen. "Introduction to Normative Multiagent Systems". In: *Computation and Mathematical Organizational Theory, Special issue on Normative Multiagent Systems* 12.2-3 (2006), pp. 71–79.

[14] A. Ciabattoni and D. Rozplokhas. "Streamlining Input/Output Logics with Sequent Calculi". In: *Proceedings of KR 2023*. 2023, pp. 146–155.

[15] A. De Domenico, A. Farjami, K. Manoorkar, A. Palmigiano, M. Panettiere, and X. Wang. "Obligations and Permissions, Algebraically". In: *arXiv preprint arXiv:2403.03148* (2024).

[16] A. De Domenico, A. Farjami, K. Manoorkar, A. Palmigiano, M. Panettiere, and X. Wang. "Obligations and Permissions on Selfextensional Logics". In: *Synthese* 206.3 (2025), p. 123.

[17] L. De Rudder and A. Palmigiano. "Slanted Canonicity of Analytic Inductive Inequalities". In: *ACM Transactions on Computational Logic* 22.3 (2021), pp. 1–41.

[18] A. D. Domenico, A. Farjami, K. Manoorkar, A. Palmigiano, M. Panettiere, and X. Wang. "Subordination Algebras as Semantic Environment of Input/Output Logic". In: *WoLLIC 2022*. Vol. 13468. LNCS. 2022, pp. 326–343.

[19] A. D. Domenico, A. Farjami, K. Manoorkar, A. Palmigiano, M. Panettiere, and X. Wang. "Correspondence and Inverse Correspondence for Input/Output Logic and Region-Based Theories of Space". In: *arXiv preprint arXiv:2412.01722* (2024).

[20] A. D. Domenico, A. Farjami, K. B. Manoorkar, A. Palmigiano, M. Panettiere, A. Tzimoulis, and X. Wang. "Normative Implications". In: *Proceedings of Logics for New-Generation AI 2025*. College Publications, 2025.

[21] J. M. Dunn, M. Gehrke, and A. Palmigiano. "Canonical Extensions and Relational Completeness of Some Substructural Logics". In: *Journal of symbolic logic* 70.3 (2005), pp. 713–740. DOI: 10.2178/jsl/1122038911.

[22] A. Farjami. "Normative Reasoning under Uncertainty: Algebraic Extensions of Input/Output Logic with Preferences". In: *Annals of Mathematics and Artificial Intelligence* (2026). DOI: 10.1007/s10472-026-10010-8.

[23] A. Farjami, L. Redondi, and M. Valentino. “Logic-Parametric Neuro-Symbolic NLI: Controlling Logical Formalisms for Verifiable LLM Reasoning”. In: *arXiv preprint arXiv:2601.05705* (2026).

[24] G. Governatori. “Practical Normative Reasoning with Defeasible Deontic Logic”. In: *Reasoning Web 2018*. Ed. by C. d’Amato and M. Theobald. LNCS 11078. Springer International Publishing, 2018, pp. 1–25. DOI: 10.1007/978-3-030-00338-8_1.

[25] L. Henkin. “Completeness in the Theory of Types”. In: *Journal of Symbolic Logic* 15.2 (1950), pp. 81–91.

[26] L. Lawniczak, L. Pasetto, C. Benzmüller, X. Li, and R. Markovich. “Reasoning with Epistemic Rights and Duties: Automating a Dynamic Logic of the Right to Know in LogiKEy”. In: *Proceedings of ECAI 2025*. 2025, pp. 1623–1630.

[27] B. Lellmann. “From Input/Output Logics to Conditional Logics via Sequents–with Provers”. In: *TABLEAUX 2021*. LNCS. 2021, pp. 147–164.

[28] D. Makinson. *Bridges from Classical to Nonmonotonic Logic*. King’s College, 2005.

[29] D. Makinson and L. van der Torre. “Input/Output Logics”. In: *Journal of Philosophical Logic* 29.4 (2000), pp. 383–408.

[30] D. Makinson and L. van der Torre. “Constraints for Input/Output Logics”. In: *Journal of Philosophical Logic* 30.2 (2001), pp. 155–185.

[31] T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Vol. 2283. Lecture Notes in Computer Science. Berlin: Springer, 2002.

[32] X. Parent and C. Benzmüller. “Conditional Normative Reasoning as a Fragment of HOL”. In: *Journal of Applied Non-Classical Logics* 34.4 (2024), pp. 561–592. DOI: 10.1080/11663081.2024.2386917.

[33] X. Parent and L. van der Torre. “Input/Output Logic”. In: *Handbook of Deontic Logic*. Ed. by D. Gabbay, J. Horty, X. Parent, R. van der Meyden, and L. van der Torre. Vol. 1. London: College Publications, 2013, pp. 499–544.

[34] X. Parent and L. van der Torre. “Input/Output Logics without Weakening”. In: *Filosofiska Notiser* 6.1 (2019), pp. 189–208.

[35] L. Pasetto and C. Benzmüller. “Implementing the Fatio Protocol for Multi-Agent Argumentation in LogiKEy”. In: *Proceedings of ARQNL 2024*. 2024, pp. 38–47.

[36] L. Pasetto and C. Benzmüller. “Visualizing Kripke Models in LogiKEy: The Case of SDL”. In: *Joint Proceedings of the Workshops and Doctoral Consortium of ICLP 2025*. 2025, pp. 1–9.

[37] L. Pasetto, C. Benzmüller, and R. Markovich. “Formalizing Mental Privacy in LogiKEy”. In: *Proceedings of AAMAS 2026*. 2026, pp. 3643–3645. DOI: 10.65109/PTSF2244.

[38] A. Steen. “A Reduction of Input/Output Logics to SAT”. In: *Journal of Applied Non-Classical Logics* (2026). DOI: 10.1080/11663081.2026.2658659.

[39] A. Steen. “Extensional Paramodulation for Higher-Order Logic and its Effective Implementation Leo-III”. In: *KI-Künstliche Intelligenz* 34.1 (2020), pp. 105–108.

[40] C. Straßer, M. Beirlaen, and F. van de Putte. “Adaptive Logic Characterizations of Input/Output Logic”. In: *Studia Logica* 104.5 (2016), pp. 869–916.

[41] L. van der Torre and Y.-H. Tan. “Diagnosis and Decision Making in Normative Reasoning”. In: *Artificial Intelligence and Law* 7.1 (1999), pp. 51–67.