I/O Logic in HOL

Christoph Benzmüller
Freie Universität Berlin, Germany, and University of Luxembourg, Luxembourg
`c.benzmueller@gmail.com`

Ali Farjami
University of Luxembourg, Luxembourg
`ali.farjami@uni.lu`

Paul Meder
University of Luxembourg, Luxembourg
`paul.meder@uni.lu`

Xavier Parent
University of Luxembourg, Luxembourg
`xavier.parent@uni.lu`

Abstract

A shallow semantical embedding of Input/output logic in classical higher-order logic is presented, and shown to be faithful (sound and complete). This embedding has been implemented in Isabelle/HOL, a proof assistant tool. It is applied to a well-known example in moral philosophy, the example of moral luck.

Keywords: Input/output logic; Classical higher-order logic; Isabelle/HOL; Moral luck.

This work has been supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 690974 – MIREL – MIning and REasoning with Legal texts. Benzmüller has been funded by the Volkswagen Foundation under project CRAP – Consistent Rational Argumentation in Politics.

1 Introduction

Deontic logic is concerned with normative concepts like obligation, permission, and prohibition. On the one hand, we have the family of traditional deontic logics which includes Standard Deontic Logic (SDL), a modal logic of type **KD**, and Dyadic Deontic Logic (DDL) [2, 16, 17]. On the other hand, we have so-called "norm-based" deontic logics. The deontic operators are evaluated not with reference to a set of possible worlds but with reference to a set of norms. A particular framework that falls within this category, is called Input/Output (I/O) logic [19]. It has gained recognition from the AI community, and has a dedicated chapter in the handbook of deontic logic [16]. The framework is expressive enough to support reasoning about constitutive, prescriptive and defeasible rules; these notions play an important role in the legal and ethical domains [14].

Our focus is on two I/O logics called *Basic Output* and *Basic Reusable Output*. We present an embedding of them into classical Higher-Order Logic (HOL), also known as simple type theory [15, 1], and study their automation. The syntax and semantics of HOL are well understood [4] and there exist automated proof tools for it; examples include Isabelle/HOL [22], LEO-II [11] and Leo-III [24]. Our approach is an indirect one. We take advantage of the known possibility of embedding I/O logic into modal logic, and we reuse the *shallow semantical embedding* of modal systems **K** and **KT** in HOL [9]. In related work, Benzmüller et al. [5, 6, 7] developed analogous shallow semantical embeddings for some well-known dyadic deontic logics.

The embeddings presented in this article are faithful (sound and complete). They are also encoded in Isabelle/HOL to enable experiments. As an illustration of the kind of experiments the framework enables, we use a well-known example in moral philosophy, the example of moral luck [21]. This term refers to situations where an agent receives moral praise or blame for an action and its consequences even though he did not have full control over them. In particular, a classical scenario of moral luck, known as the *Drink and Drive* example, is used as an illustration.

The article is structured as follows: Section 2 gives a quick review of modal logic and higher-order logic, and Section 3 introduces I/O logic. The semantical embeddings of *Basic Output* and *Basic Reusable Output* in HOL are then described in Section 4. This section also shows the faithfulness of the embeddings. In Section 5 we apply the framework to the *Drink and Drive* example.

2 Preliminaries

In this section, we recap some important notions from modal logic and HOL.

2.1 Modal logic K

The language of **K** is obtained by supplementing the language of Propositional Logic (PL) with a modal operator $\Box$. It is generated as follows:

$$\varphi ::= p|\neg\varphi|\varphi \vee \varphi|\Box\varphi$$

where p denotes an atomic formula. Other logical connectives such as $\wedge$, $\rightarrow$ and $\Diamond$, are defined in the usual way. The axioms of system **K** consist of those of PL plus $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$, called axiom K. The rules of **K** are *Modus ponens* (from φ and $\varphi \rightarrow \psi$ infer ψ) and *Necessitation* (from φ infer $\Box\varphi$).

A Kripke model for **K** is a triple $M = \langle W, R, V\rangle$, where W is a non-empty set of possible worlds, R is a binary relation on W, called accessibility relation, and V is a function assigning a set of worlds to each atomic formula, that is, $V(p) \subseteq W$.

Truth of a formula φ in a model $M = \langle W, R, V\rangle$ and a world $s \in W$ is written as $M, s \models \varphi$. We define $V(\varphi) = \{s \in W|M, s \models \varphi\}$. The relation $\models$ is defined as follows:

$M, s \models p$	if and only if	$s \in V(p)$
$M, s \models \neg\varphi$	if and only if	$M, s \not\models \varphi$ (that is, not $M, s \models \varphi$)
$M, s \models \varphi \vee \psi$	if and only if	$M, s \models \varphi$ or $M, s \models \psi$
$M, s \models \Box\varphi$	if and only if	for every $t \in W$such that sRt, $M, t \models \varphi$

As usual, a modal formula φ is *true in a Kripke model* $M = \langle W, R, V\rangle$, i.e., $M \models \varphi$, if and only if for all worlds $s \in W$, we have $M, s \models \varphi$. A formula φ is *valid in a class* $\mathcal{C}$ *of Kripke models*, denoted as $\models_{\mathcal{C}} \varphi$, if and only if it is true in every model in class $\mathcal{C}$.

System **K** is determined by (i.e., is sound and complete with respect to) the class of all Kripke models. System **KT** is obtained from system **K** by adding the schema T : $\Box\varphi \rightarrow \varphi$ as an axiom. System **KT** is determined by the class of all Kripke models in which R is reflexive. We denote the class of all Kripke models and the class of Kripke models where R is reflexive as $\mathcal{C}_K$ and $\mathcal{C}_{KT}$, respectively.

Two other axiom schemas that can be added to **K** are 4 : $\Box\varphi \rightarrow \Box\Box\varphi$ and 5 : $\Diamond\varphi \rightarrow \Box\Diamond\varphi$. For instance, **K45** is an extension of **K** obtained by adding 4 and 5 as axioms. The schemas 4 and 5 are valid if R is *transitive* and *euclidean*, respectively.

2.2 Classical higher-order logic

HOL is based on simple typed λ-calculus. We assume that the set $\mathcal{T}$ 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 $\alpha, \beta, o \in \mathcal{T}$, the *language of HOL* is generated as follows:

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

where p_α represents a typed constant symbol (from a possibly infinite set $\mathcal{P}_\alpha$ of such constant symbols) and X_α represents a typed variable symbol (from a possibly infinite set $\mathcal{V}_\alpha$ 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 $\mathcal{P}_\alpha$ 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 $\alpha \in \mathcal{T}$. From the selected set of primitive connectives, other logical connectives can be introduced as abbreviations. Type information as well as brackets may 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 $((\vee_{o\rightarrow o\rightarrow o}\, s_o)\, t_o)_o$. We often write $\forall X_\alpha s_o$ as syntactic sugar for $\Pi_{(\alpha\rightarrow o)\rightarrow o}(\lambda X_\alpha s_o)$.

The notions of *free variables*, *α-conversion*, *$\beta\eta$-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.

The semantics of HOL are well understood and thoroughly documented [4]. In the remainder, the semantics of choice are Henkin's general models [18].

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. $D_{\alpha\rightarrow\beta}$ 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:

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

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 $t_o \in \Sigma$.

3 Input/Output Logic

Input/output logic was initially introduced by Makinson and van der Torre [19]. There are various I/O operations. In this paper we focus on two of them, called "*Basic Output*" and "*Basic Reusable Output*".

3.1 Syntax

$G \subseteq \mathcal{L} \times \mathcal{L}$ is called a normative system, with $\mathcal{L}$ representing the set of all the formulas of propositional logic. A pair $(a, x) \in G$ is referred to as a conditional norm or obligation, where a and x are formulas of propositional logic. The pair (a, x) is read as "given a, it is obligatory that x". a is called the body and represents some situation or condition, whereas x is called the head and represents what is obligatory or desirable in that situation.

3.2 Semantics

For a set of formulas A, we define $G(A) = \{x \mid (a,x) \in G \text{ for some } a \in A\}$ and $Cn(A) = \{x \mid A \vdash x\}$ with $\vdash$ denoting the classical consequence relation. A set of formulas V is *maximal consistent* if it is consistent, and no proper extension of V is consistent. A set of formulas V is said to be *complete* if it is either *maximal consistent* or equal to $\mathcal{L}$.

Definition 1 (Basic Output)**.** *Given a set of conditional norms G and an input set A of propositional formulas,*

$$out_2(G,A) = \bigcap\{Cn(G(V)) \mid A \subseteq V, V \ \ complete\}$$

Definition 2 (Basic Reusable Output)**.** *Given a set of conditional norms G and an input set A of propositional formulas,*

$$out_4(G,A) = \bigcap\{Cn(G(V)) \mid A \subseteq V \supseteq G(V), V complete\}$$

Besides those traditional formulations of the operations, the paper [19] documents modal formulations for out_2 and out_4.

Theorem 1. *$x \in out_2(G,A)$ if and only if $x \in Cn(G(\mathcal{L}))$ and $G^{\Box} \cup A \vdash_{\mathbf{S}} \Box x$ for any modal logic **S** with $\boldsymbol{K_0} \subseteq \boldsymbol{S} \subseteq \boldsymbol{K45}$.*

Theorem 2. *$x \in out_4(G,A)$ if and only if $x \in Cn(G(\mathcal{L}))$ and $G^{\Box} \cup A \vdash_{\mathbf{S}} \Box x$ for any modal logic **S** with $\boldsymbol{K_0T} \subseteq \boldsymbol{S} \subseteq \boldsymbol{KT45}$.*

$\mathbf{K_0}$ is a subsystem of system $\mathbf{K}$ with axiom K, *modus ponens* and the inference rule “from ψ, infer $\Box\psi$, for all tautologies in propositional logic”. $G^{\Box}$ denotes the set containing all modal formulas of the form $b \to \Box y$, such that $(b,y) \in G$. We have that $G^{\Box} \cup A \vdash_{\mathbf{S}} \Box x$ if for a finite subset Y of $G^{\Box} \cup A$, it holds that $(\bigwedge Y \to \Box x) \in \mathbf{S}$. The notation $\bigwedge Y$ stands for the conjunction of all the elements $y_1, y_2, \ldots, y_n$ in Y, i.e., $y_1 \wedge y_2 \wedge \cdots \wedge y_n$.

3.3 Proof theory

The proof theory of an I/O logic is specified via a number of derivation rules acting on pairs (a,x) of formulas. Given a set G of pairs, we write $(a,x) \in deriv_i(G)$ to say that (a,x) can be derived from G using those rules.

- (SI) Strengthening of the input: from (a,x) and $\vdash b \to a$, infer (b,x)
- (WO) Weakening of the output: from (a,x) and $\vdash x \to y$, infer (a,y)

- (AND) Conjunction of the output: from (a, x) and (a, y), infer $(a, x \wedge y)$
- (OR) Disjunction of the input: from (a, x) and (b, x), infer $(a \vee b, x)$
- (CT) Cumulative transitivity: from (a, x) and $(a \wedge x, y)$, infer (a, y)

The *Basic Output* is syntactically characterized by $deriv_2(G)$ that is closed under rules SI, WO, AND and OR. The *Basic Reusable Output* is determined by $deriv_4(G)$ that is closed under all of the five rules.

4 Shallow Semantical Embedding

The *shallow semantical embedding* approach proposed by Benzmüller [3] uses HOL as a meta-logic in order to represent and model the syntactic and semantical elements of a specific target logic. This methodology is documented and studied for Kripke semantics in [9] and for neighborhood semantics in [6]. This section presents shallow semantical embeddings of the I/O operations out_2 and out_4 in HOL and provides proofs for the soundness and completeness of both operations. To realize these embeddings, we below use the provided modal formulations of the operations; an alternative approach is studied in [8].

4.1 Semantical embedding of K and KT in HOL

We start by describing the semantical embeddings of **K** and **KT** in HOL. This material is taken from [9, 10].

By introducing a new type i to denote possible worlds, the formulas of **K** are identified with certain HOL terms (predicates) of type $i \to o$. The type $i \to o$ is abbreviated as τ in the remainder. This allows us to represent the formulas of **K** as functions from possible worlds to truth values in HOL and therefore the truth of a formula can explicitly be evaluated in a particular world. The HOL signature is assumed to further contain the constant symbol $r_{i \to i \to o}$. Moreover, for each atomic propositional symbol p^j of **K**, the HOL signature must contain the corresponding constant symbol p^j_τ. Without loss of generality, we assume that besides those symbols and the primitive logical connectives of HOL, no other constant symbols are given in the signature of HOL.

The mapping $\lfloor \cdot \rfloor$ translates a formula φ of **K** into a term $\lfloor \varphi \rfloor$ of HOL of type τ.

The mapping is defined recursively:

$$\begin{array}{lcl} \lfloor p^j \rfloor & = & p^j_\tau \\ \lfloor \neg\varphi \rfloor & = & \neg_{\tau\to\tau} \lfloor\varphi\rfloor \\ \lfloor \varphi \vee \psi \rfloor & = & \vee_{\tau\to\tau\to\tau} \lfloor\varphi\rfloor \lfloor\psi\rfloor \\ \lfloor \Box\varphi \rfloor & = & \Box_{\tau\to\tau} \lfloor\varphi\rfloor \end{array}$$

$\neg_{\tau\to\tau}$, $\vee_{\tau\to\tau\to\tau}$ and $\Box_{\tau\to\tau}$ abbreviate the following terms of HOL:

$$\begin{array}{ll} \neg_{\tau\to\tau} & = \lambda A_\tau \lambda X_i \neg(A\, X) \\ \vee_{\tau\to\tau\to\tau} & = \lambda A_\tau \lambda B_\tau \lambda X_i (A\, X \vee B\, X) \\ \Box_{\tau\to\tau} & = \lambda A_\tau \lambda X_i \forall Y_i (\neg(r_{i\to i\to o} X\, Y) \vee A\, Y) \end{array}$$

Analyzing the truth of formula φ, represented by the HOL term $\lfloor\varphi\rfloor$, in a particular world w, represented by the term w_i, corresponds to evaluating the application $(\lfloor\varphi\rfloor\, w_i)$. In line with the previous work [10], we define $vld_{\tau\to o} = \lambda A_\tau \forall S_i (A\, S)$. With this definition, validity of a formula s in **K** corresponds to the validity of the formula $(vld\, \lfloor\varphi\rfloor)$ in HOL, and vice versa.

To prove the soundness and completeness, that is, faithfulness, of the above embedding, a mapping from Kripke models into Henkin models is employed.

Lemma 1 (Kripke models ⇒ Henkin models)**.** *For every Kripke model $M = \langle W, R, V\rangle$ there exists a corresponding Henkin model H^M, such that for all formulas δ of* **K***, all assignments g and worlds s it holds:*

$$M, s \models \delta \text{ if and only if } \|\lfloor\delta\rfloor\, S_i\|^{H^M, g[s/S_i]} = T$$

Proof. See [9, 10]. □

Lemma 2 (Henkin models ⇒ Kripke models)**.** *For every Henkin model $H = \langle\{D_\alpha\}_{\alpha\in\mathcal{T}}, I\rangle$ there exists a corresponding Kripke model M_H, such that for all formulas δ of* **K***, all assignments g and worlds s it holds:*

$$\|\lfloor\delta\rfloor S_i\|^{H, g[s/S_i]} = T \text{ if and only if } M_H, s \models \delta$$

Proof. See [9, 10]. □

The following table summarizes the alignment of Kripke models and Henkin models. For the class of Kripke models $\langle W, R, V\rangle$ that satisfies some properties, such as *reflexivity*, the corresponding class of Henkin models needs to satisfy a corresponding property. In system **KT**, for example, the class of Kripke models satisfies the property of *reflexivity*, which corresponds to axiom T. The counterpart of this property is represented as REF in HOL: $\forall X_i(r_{i\to i\to o}X_iX_i)$, where constant symbol $r_{i\to i\to o}$ denotes the accessibility relation.

Kripke model $\langle W, R, V\rangle$	Henkin model $\langle D, I\rangle$
Possible worlds $s \in W$	Set of individuals $s_i \in D_i$
Accessibility relation R sRu	Binary predicates $r_{i\to i\to o}$ $Ir_{i\to i\to o}(s_i, u_i) = T$
Propositional letters p^j Valuation function $s \in V(p^j)$	Unary predicates $p^j_{i\to o}$ Interpretation function $Ip^j_{i\to o}(s_i) = T$

These correspondences between Kripke and Henkin models include the assumptions that have been formulated at the beginning of this section.

Theorem 3 (Faithfulness of the embedding of system **K** in HOL)**.**

$$\models_{\mathcal{C}_K} \varphi \textit{ if and only if } \models^{HOL} vld\lfloor\varphi\rfloor$$

Proof. See [9, 10]. □

Theorem 4 (Faithfulness of the embedding of system **KT** in HOL)**.**

$$\models_{\mathcal{C}_{KT}} \varphi \textit{ if and only if } \{REF\} \models^{HOL} vld\lfloor\varphi\rfloor$$

Proof. See [9, 10]. □

4.2 Semantical embedding of I/O logic in HOL

In order to embed the operations out_2 and out_4 in HOL, we just use the corresponding modal formulations. We apply Theorem 3 and Theorem 4, respectively, to prove the faithfulness of the embeddings.

Theorem 5 (Faithfulness of the embedding of out_2 in HOL)**.**

$$\varphi \in out_2(G, A)$$

if and only if

$$\models^{HOL} vld\lfloor\bigwedge(G^\square \cup A) \to \square\varphi\rfloor \textit{ and } \models^{HOL} vld\lfloor\bigwedge G(\mathcal{L}) \to \varphi\rfloor$$

Proof. We choose $\mathbf{S} = \mathbf{K}$ in Theorem 1 and then apply Theorem 3.

$$\varphi \in out_2(G, A)$$

if and only if

$$G^{\Box} \cup A \vdash_{\mathbf{K}} \Box\varphi \text{ and } \varphi \in Cn(G(\mathcal{L}))$$

if and only if

$$\models_{\mathcal{C}_K} \bigwedge(G^{\Box} \cup A) \to \Box\varphi \text{ and } \bigwedge G(\mathcal{L}) \vdash \varphi$$

if and only if

$$\models_{\mathcal{C}_K} \bigwedge(G^{\Box} \cup A) \to \Box\varphi \text{ and } \models_{\mathcal{C}_K} \bigwedge G(\mathcal{L}) \to \varphi$$

if and only if

$$\models^{HOL} \mathit{vld}\,\lfloor \bigwedge(G^{\Box} \cup A) \to \Box\varphi \rfloor \text{ and } \models^{HOL} \mathit{vld}\lfloor \bigwedge G(\mathcal{L}) \to \varphi \rfloor$$

□

Theorem 6 (Faithfulness of the embedding of out_4 in HOL)**.**

$$\varphi \in out_4(G, A)$$

if and only if

$$\{REF\} \models^{HOL} \mathit{vld}\,\lfloor \bigwedge(G^{\Box} \cup A) \to \Box\varphi \rfloor \text{ and } \{REF\} \models^{HOL} \mathit{vld}\,\lfloor \bigwedge G(\mathcal{L}) \to \varphi \rfloor$$

Proof. We choose $\mathbf{S} = \mathbf{KT}$ in Theorem 2 and then apply Theorem 4.

$$\varphi \in out_4(G, A)$$

if and only if

$$G^{\Box} \cup A \vdash_{\mathbf{KT}} \Box\varphi \text{ and } \varphi \in Cn(G(\mathcal{L}))$$

if and only if

$$\models_{\mathcal{C}_{KT}} \bigwedge(G^{\Box} \cup A) \to \Box\varphi \text{ and } \bigwedge G(\mathcal{L}) \vdash \varphi$$

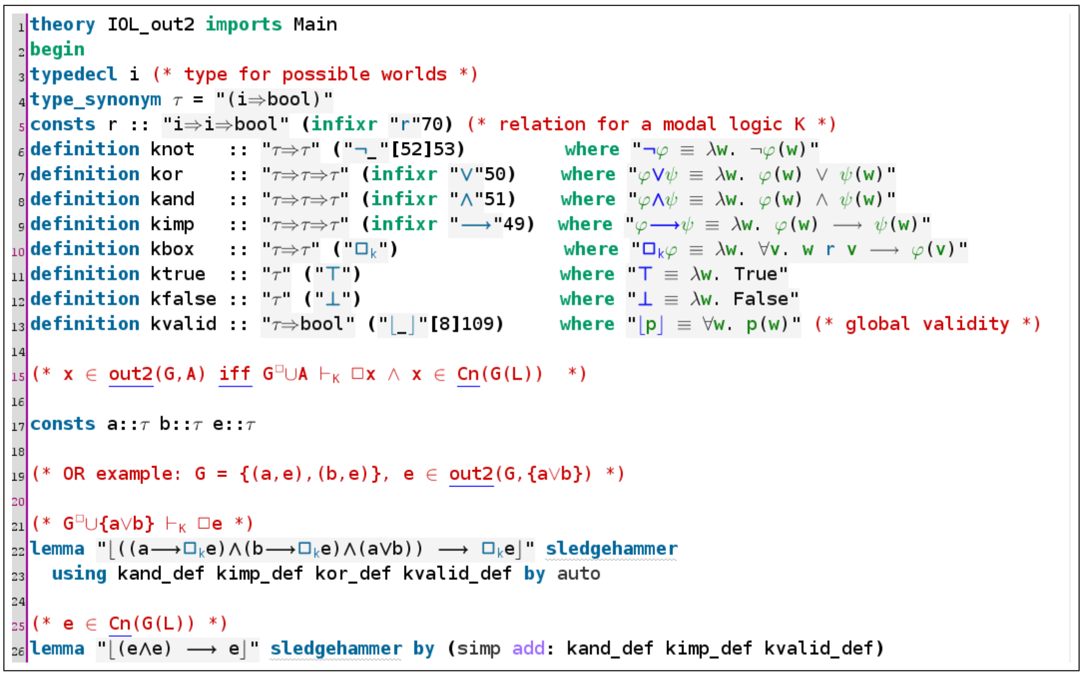
if and only if

$$\models_{\mathcal{C}_{KT}} \bigwedge(G^{\Box} \cup A) \to \Box\varphi \text{ and } \models_{\mathcal{C}_{KT}} \bigwedge G(\mathcal{L}) \to \varphi$$

if and only if

$$\{REF\} \models^{HOL} \mathit{vld}\,\lfloor \bigwedge(G^{\Box} \cup A) \to \Box\varphi \rfloor \text{ and } \{REF\} \models^{HOL} \mathit{vld}\,\lfloor \bigwedge G(\mathcal{L}) \to \varphi \rfloor$$

□



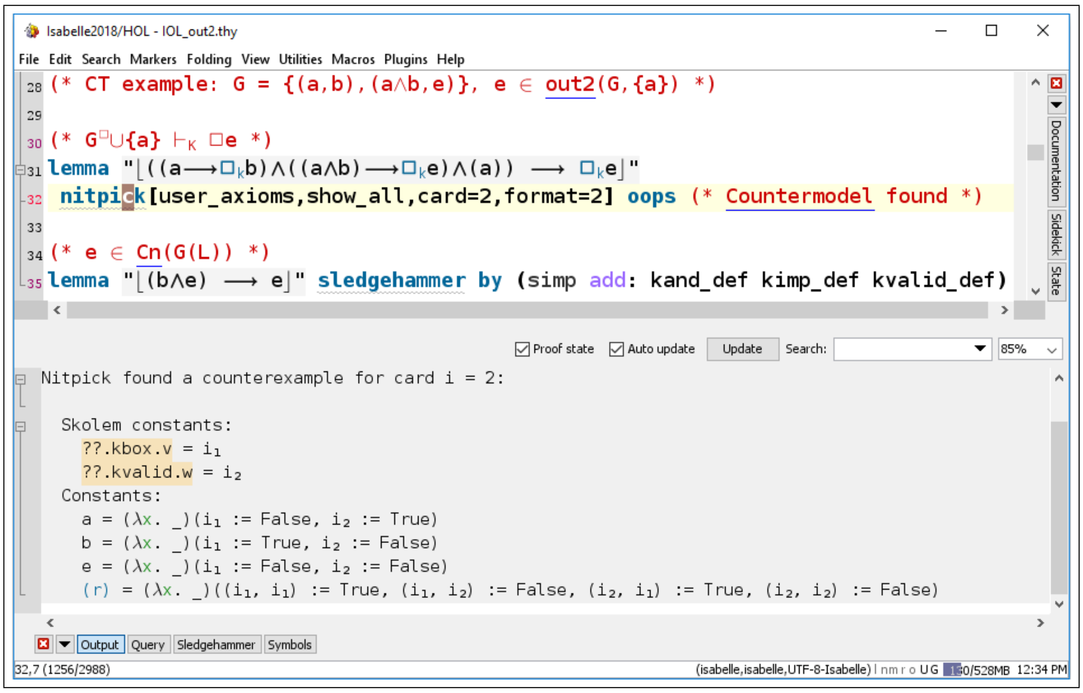
```
1  theory IOL_out2 imports Main
2  begin
3  typedecl i (* type for possible worlds *)
4  type_synonym τ = "(i⇒bool)"
5  consts r :: "i⇒i⇒bool" (infixr "r"70) (* relation for a modal logic K *)
6  definition knot   :: "τ⇒τ" ("¬_"[52]53)         where "¬φ ≡ λw. ¬φ(w)"
7  definition kor    :: "τ⇒τ⇒τ" (infixr "∨"50)     where "φ∨ψ ≡ λw. φ(w) ∨ ψ(w)"
8  definition kand   :: "τ⇒τ⇒τ" (infixr "∧"51)     where "φ∧ψ ≡ λw. φ(w) ∧ ψ(w)"
9  definition kimp   :: "τ⇒τ⇒τ" (infixr "⟶"49)    where "φ⟶ψ ≡ λw. φ(w) ⟶ ψ(w)"
10 definition kbox   :: "τ⇒τ" ("□k")                where "□kφ ≡ λw. ∀v. w r v ⟶ φ(v)"
11 definition ktrue  :: "τ" ("⊤")                   where "⊤ ≡ λw. True"
12 definition kfalse :: "τ" ("⊥")                   where "⊥ ≡ λw. False"
13 definition kvalid :: "τ⇒bool" ("⌊_⌋"[8]109)      where "⌊p⌋ ≡ ∀w. p(w)" (* global validity *)
14
15 (* x ∈ out2(G,A) iff G□∪A ⊢K □x ∧ x ∈ Cn(G(L))  *)
16
17 consts a::τ b::τ e::τ
18
19 (* OR example: G = {(a,e),(b,e)}, e ∈ out2(G,{a∨b}) *)
20
21 (* G□∪{a∨b} ⊢K □e *)
22 lemma "⌊((a⟶□ke)∧(b⟶□ke)∧(a∨b)) ⟶ □ke⌋" sledgehammer
23   using kand_def kimp_def kor_def kvalid_def by auto
24
25 (* e ∈ Cn(G(L)) *)
26 lemma "⌊(e∧e) ⟶ e⌋" sledgehammer by (simp add: kand_def kimp_def kvalid_def)
```

Figure 1: Semantical embedding of out_2 in Isabelle/HOL

4.3 Implementation of I/O logic in Isabelle/HOL

The semantical embeddings of the operations out_2 and out_4 in HOL as devised in the previous section have been implemented in the higher-order proof assistant tool Isabelle/HOL [22], see Fig. 1. We declare the type i to denote possible worlds and introduce the relevant connectives in lines 6–12.

Let the set of conditional norms G be composed of the elements (a, e) and (b, e), where a, b and e are propositional symbols, and let the input set A correspond to the singleton set containing $a \vee b$. By the rule of disjunction (OR), we should have that $e \in out_2(G, A)$. According to the provided translation, $e \in out_2(G, A)$ if and only if $G^{\Box} \cup A \vdash_{\mathbf{K}} \Box e$ and $e \in Cn(G(\mathcal{L}))$. Theorem 5 provides us now with higher-order formulations for both of these statements, i.e., $\models^{HOL} vld\lfloor\bigwedge(G^{\Box} \cup A) \rightarrow \Box e\rfloor$ and $\models^{HOL} vld\lfloor\bigwedge G(\mathcal{L}) \rightarrow e\rfloor$, respectively. Regarding the implementation, the propositional symbols a, b and e have to be declared as constants of type τ. The framework's integrated automatic theorem provers (ATPs), called via the Sledgehammer tool [13], are able to prove both statements. This is shown in Fig. 1, lines 22–23 and 26.


Isabelle2018/HOL - IOL_out2.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Proof state
Auto update
Update
Search:
85%
Output
Query
Sledgehammer
Symbols
Documentation
Sidekick
State
32,7 (1256/2988)
(isabelle,isabelle,UTF-8-Isabelle) I nm r o UG 150/528MB 12:34 PM

```
(* CT example: G = {(a,b),(a∧b,e)}, e ∈ out2(G,{a}) *)

(* G□∪{a} ⊢K □e *)
lemma "⌊((a⟶□Kb)∧((a∧b)⟶□Ke)∧(a)) ⟶ □Ke⌋"
 nitpick[user_axioms,show_all,card=2,format=2] oops (* Countermodel found *)

(* e ∈ Cn(G(L)) *)
lemma "⌊(b∧e) ⟶ e⌋" sledgehammer by (simp add: kand_def kimp_def kvalid_def)
```

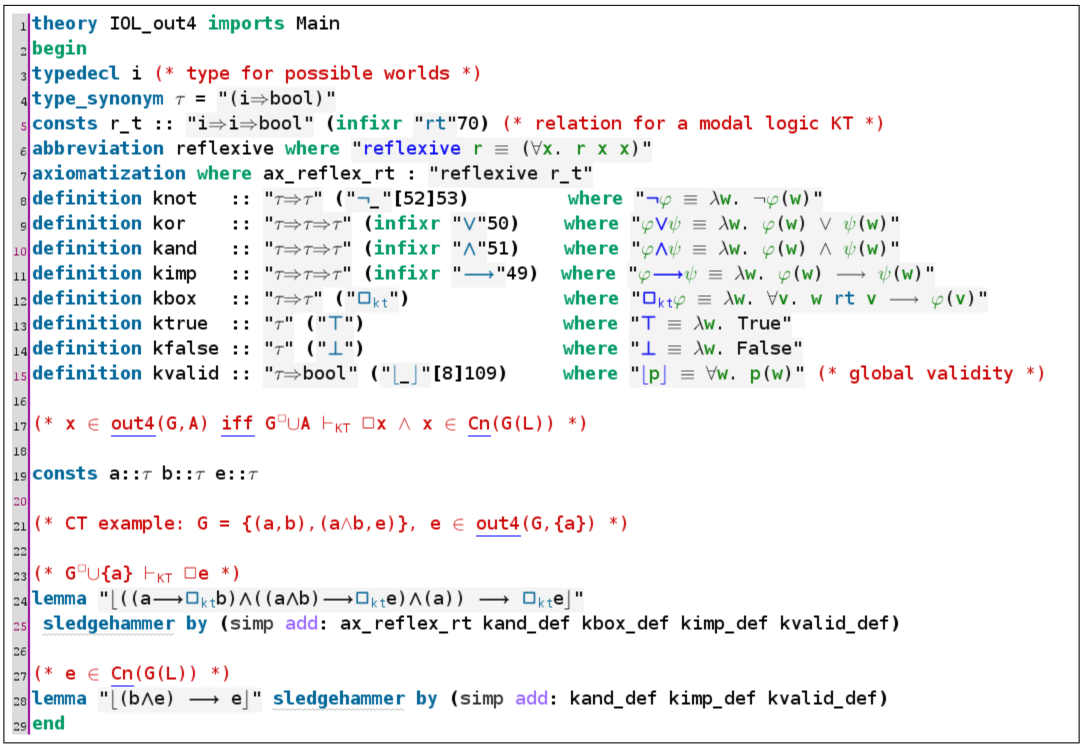
```
Nitpick found a counterexample for card i = 2:

  Skolem constants:
    ??.kbox.v = i₁
    ??.kvalid.w = i₂
  Constants:
    a = (λx. _)(i₁ := False, i₂ := True)
    b = (λx. _)(i₁ := True, i₂ := False)
    e = (λx. _)(i₁ := False, i₂ := False)
    (r) = (λx. _)((i₁, i₁) := True, (i₁, i₂) := False, (i₂, i₁) := True, (i₂, i₂) := False)
```

Figure 2: Failure of CT for out_2

Consider the set of conditional norms $G = \{(a,b),(a \wedge b, e)\}$ with the input set $A = \{a\}$. The rule of cumulative transitivity (CT) is not satisfied by the operation out_2. This can also be verified with our implementation. The model finder Nitpick [12] is able to generate a countermodel for the statement $G^{\Box} \cup A \vdash_{\mathbf{K}} \Box e$ and therefore we were able to show that $e \notin out_2(G,A)$. In particular, Nitpick came up with a model M consisting of two possible worlds i_1 and i_2. We have that $V(a) = \{i_2\}$, $V(b) = \{i_1\}$ and $V(e) = \emptyset$. And $R = \{(i_1,i_1),(i_2,i_1)\}$. The formula $((a \to \Box b) \wedge ((a \wedge b) \to \Box e) \wedge a) \to \Box e$ is not valid in this model. The formulation of the example and the generation of the countermodel is illustrated in Fig. 2.

The embedding of the operation out_4 refers to system **KT** which means that the corresponding class of Kripke models satisfies the property of reflexivity. In our implementation, the accessibility relation for this system is denoted by the constant r_t which we declare as reflexive. Due to this property, the Sledgehammer tool is able to prove the statement $G^{\Box} \cup A \vdash_{\mathbf{KT}} \Box e$ and thus we can verify that $e \in out_4(G,A)$. Fig. 3 shows the encoding of the operation out_4 in Isabelle/HOL and the verification of the CT example.



```
theory IOL_out4 imports Main
begin
typedecl i (* type for possible worlds *)
type_synonym τ = "(i⇒bool)"
consts r_t :: "i⇒i⇒bool" (infixr "rt"70) (* relation for a modal logic KT *)
abbreviation reflexive where "reflexive r ≡ (∀x. r x x)"
axiomatization where ax_reflex_rt : "reflexive r_t"
definition knot   :: "τ⇒τ" ("¬_"[52]53)         where "¬φ ≡ λw. ¬φ(w)"
definition kor    :: "τ⇒τ⇒τ" (infixr "∨"50)     where "φ∨ψ ≡ λw. φ(w) ∨ ψ(w)"
definition kand   :: "τ⇒τ⇒τ" (infixr "∧"51)     where "φ∧ψ ≡ λw. φ(w) ∧ ψ(w)"
definition kimp   :: "τ⇒τ⇒τ" (infixr "⟶"49)    where "φ⟶ψ ≡ λw. φ(w) ⟶ ψ(w)"
definition kbox   :: "τ⇒τ" ("□kt")              where "□ktφ ≡ λw. ∀v. w rt v ⟶ φ(v)"
definition ktrue  :: "τ" ("⊤")                  where "⊤ ≡ λw. True"
definition kfalse :: "τ" ("⊥")                  where "⊥ ≡ λw. False"
definition kvalid :: "τ⇒bool" ("⌊_⌋"[8]109)     where "⌊p⌋ ≡ ∀w. p(w)" (* global validity *)

(* x ∈ out4(G,A) iff G□∪A ⊢KT □x ∧ x ∈ Cn(G(L)) *)

consts a::τ b::τ e::τ

(* CT example: G = {(a,b),(a∧b,e)}, e ∈ out4(G,{a}) *)

(* G□∪{a} ⊢KT □e *)
lemma "⌊((a⟶□ktb)∧((a∧b)⟶□kte)∧(a)) ⟶ □kte⌋"
 sledgehammer by (simp add: ax_reflex_rt kand_def kbox_def kimp_def kvalid_def)

(* e ∈ Cn(G(L)) *)
lemma "⌊(b∧e) ⟶ e⌋" sledgehammer by (simp add: kand_def kimp_def kvalid_def)
end
```

Figure 3: Semantical embedding of out_4 in Isabelle/HOL

5 Application: Moral Luck

The literature on moral luck [21] is addressing the question whether luck can ever make a moral difference or not. Examples involving moral luck are typical scenarios in which an agent is held accountable for his actions and its consequences even though it is clear that the agent was neither in full control of his actions nor its consequences. These examples are thus in conflict with the ethical principle that agents are not morally responsible for actions that they are unable to control.

The *Drink and Drive* [20] example highlights a classical scenario of moral luck. There exist many different variations of this example and a possible variant can be formulated as follows:

> Assume a situation where two persons, Ali and Paul, go out for a drink in the evening. Both of them go to the same bar, consume the same amount of alcoholic drinks and end up pretty drunk. At one point during the night, they both decide to leave the place. So they go to their own individual vehicles and hit the road in order to drive home. The roads are pretty deserted at that

time and Ali manages to drive home safely even with the high percentage of alcohol in his blood. Paul, in contrast, is facing something unexpected. Out of nowhere, a child appears in front of his car. Since he had a few drinks too much, his reaction time is impaired by the alcohol and it makes it impossible for him to stop and swerve to avoid hitting and killing the child.

Both *Ali* and *Paul*, made the blameworthy decision of driving while being drunk. But neither one of them had the intention to hit and kill anyone. Nevertheless, most people would tend to judge *Paul* more guilty than *Ali* simply because in his case a child got killed. However, both of them violated the same obligation, namely that one should not drive while being intoxicated and it was only a matter of luck that nobody got harmed or killed in the case of *Ali*. Therefore we say that *Ali* got morally lucky.

To formulate the *Drink and Drive* example in Isabelle/HOL, we first import the Isabelle/HOL file containing the implementation of the operation out_2. This can be done using the Isabelle/HOL command *imports* (cf. Fig. 4, line 1). Next, we have to declare three individuals, namely *Ali* and *Paul*, representing the two drivers, and *Child*, representing the *child* in the scenario (cf. Fig. 4, line 3). In lines 4–5, we define the constant symbols for the relevant propositions (state of affairs or action).

One associates with each driver a set of *Norms* and an *Input*. For the individual *Paul*, the set of *Norms G* is defined as follows: (cf. Fig. 4, lines 9–11)

- $(\top, \neg Kill\,Child \wedge \neg Hurt\,Child)$
  This norm states that it is forbidden to kill or even hurt the *child*.

- $(\top, Drive\_carefully\,Paul)$
  This norm states that *Paul* is obligated to drive carefully in any situation.

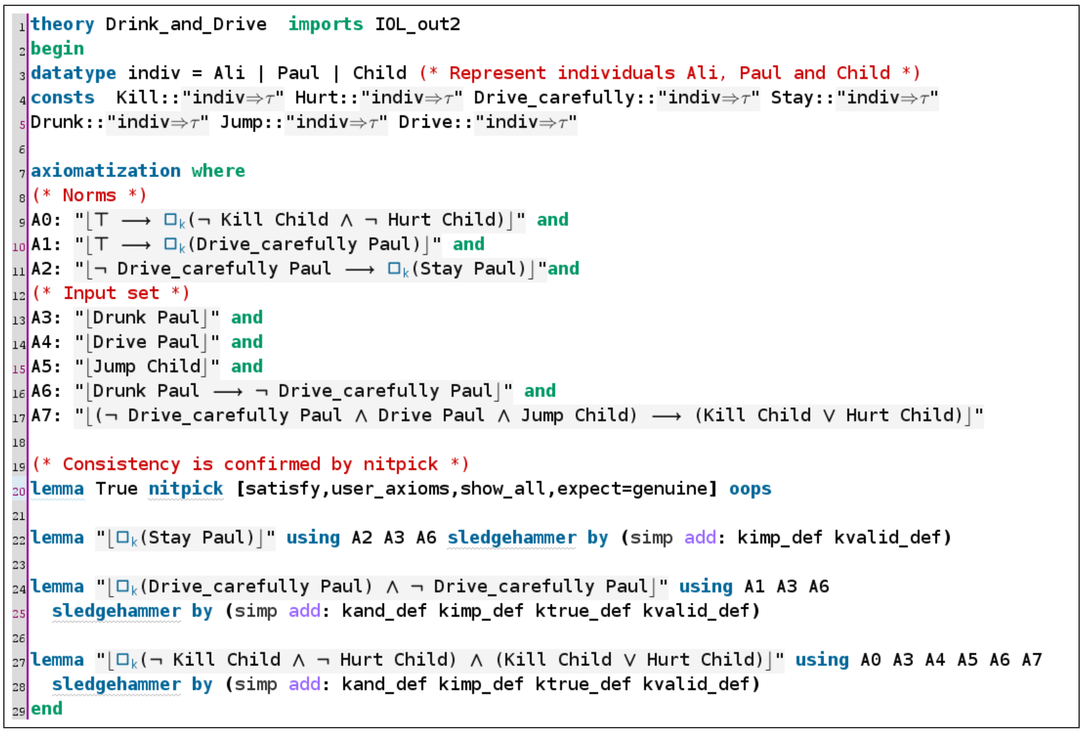
- $(\neg Drive\_carefully\,Paul, StayPaul)$
  This norm states that if *Paul* does not drive carefully, he should stay (at his current location).

To complete the formalization for the individual *Paul*, we need to add the following facts to the *Input* set A: (cf. Fig. 4, lines 13–17)

- $Drunk\,Paul$; $Drive\,Paul$; $Jump\,Child$
  Paul is actually drunk; *Paul* drives home; The *child* jumps.

- $Drunk\,Paul \rightarrow \neg Drive\_carefully\,Paul$
  If *Paul* is drunk then he drives not carefully.

- $(\neg Drive\_carefullyPaul \wedge DrivePaul \wedge JumpChild)$
  $\rightarrow (KillChild \vee HurtChild)$
  If *Paul* drives, but does not do it carefully, and the *child* jumps in front of his car then *Paul* will kill or hurt the *child*.



```
1  theory Drink_and_Drive  imports IOL_out2
2  begin
3  datatype indiv = Ali | Paul | Child (* Represent individuals Ali, Paul and Child *)
4  consts  Kill::"indiv⇒τ" Hurt::"indiv⇒τ" Drive_carefully::"indiv⇒τ" Stay::"indiv⇒τ"
5  Drunk::"indiv⇒τ" Jump::"indiv⇒τ" Drive::"indiv⇒τ"
6
7  axiomatization where
8  (* Norms *)
9  A0: "⌊⊤ ⟶ □ₖ(¬ Kill Child ∧ ¬ Hurt Child)⌋" and
10 A1: "⌊⊤ ⟶ □ₖ(Drive_carefully Paul)⌋" and
11 A2: "⌊¬ Drive_carefully Paul ⟶ □ₖ(Stay Paul)⌋"and
12 (* Input set *)
13 A3: "⌊Drunk Paul⌋" and
14 A4: "⌊Drive Paul⌋" and
15 A5: "⌊Jump Child⌋" and
16 A6: "⌊Drunk Paul ⟶ ¬ Drive_carefully Paul⌋" and
17 A7: "⌊(¬ Drive_carefully Paul ∧ Drive Paul ∧ Jump Child) ⟶ (Kill Child ∨ Hurt Child)⌋"
18
19 (* Consistency is confirmed by nitpick *)
20 lemma True nitpick [satisfy,user_axioms,show_all,expect=genuine] oops
21
22 lemma "⌊□ₖ(Stay Paul)⌋" using A2 A3 A6 sledgehammer by (simp add: kimp_def kvalid_def)
23
24 lemma "⌊□ₖ(Drive_carefully Paul) ∧ ¬ Drive_carefully Paul⌋" using A1 A3 A6
25   sledgehammer by (simp add: kand_def kimp_def ktrue_def kvalid_def)
26
27 lemma "⌊□ₖ(¬ Kill Child ∧ ¬ Hurt Child) ∧ (Kill Child ∨ Hurt Child)⌋" using A0 A3 A4 A5 A6 A7
28   sledgehammer by (simp add: kand_def kimp_def ktrue_def kvalid_def)
29 end
```

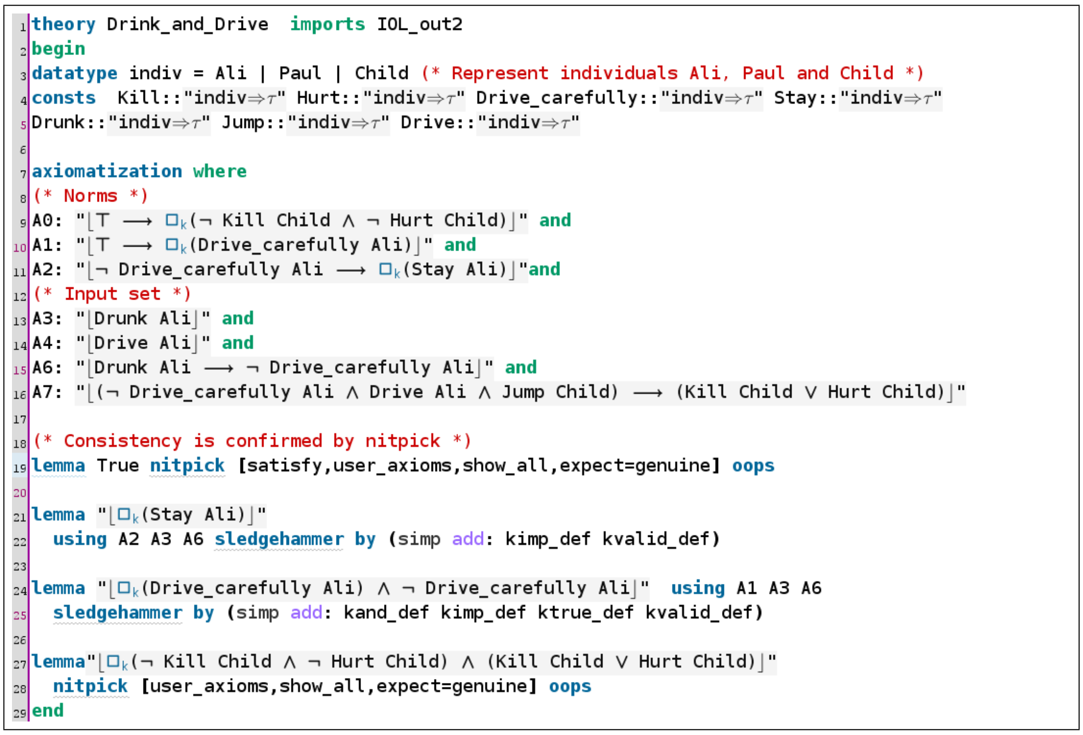
Figure 4: *Drink and Drive* scenario for *Paul* in Isabelle/HOL

Since Nitpick finds a model satisfying our statements, the formalization of the *Drink and Drive* is consistent; cf. Fig. 4, line 20.

Actually, we are able to derive the obligation that *Paul* should stay (at his current position) by using the norm A2 and the facts A3 and A6, meaning that we can derive $G^{\Box} \cup A \vdash_{\mathbf{K}} \Box Stay\_Paul$ and $Stay\_Paul \in Cn(G(\mathcal{L}))$. The first statement is proven by Sledgehammer tool; cf. Fig. 4, line 22. In this example, we skip checking the following (trivial) statements $X \in Cn(G(\mathcal{L}))$ for $X \in \{Stay\_Paul, Drive\_carefully\_Paul, \neg KillChild \wedge \neg HurtChild\}$ in Isabelle/HOL.

Furthermore, our implementation is capable of recognizing violations to norms, formally written as $\alpha \in out_2(G, A)$ and $\neg\alpha \in Cn(A)$. In particular, *Paul* violated the norms A0 and A1. For instance, the violation to A1 is proven by Sledgehammer;

cf. Fig. 4, lines 24–25. *Paul* did not drive carefully even though there is an obligation to do so, meaning that we have $G^{\Box} \cup A \vdash_{\mathbf{K}} \Box Drive\_carefully\_Paul$ (using A1), $Drive\_carefully\_Paul \in Cn(G(\mathcal{L}))$ and $\neg Drive\_carefully\_Paul \in Cn(A)$ (using A3 and A6).



```
 1 theory Drink_and_Drive  imports IOL_out2
 2 begin
 3 datatype indiv = Ali | Paul | Child (* Represent individuals Ali, Paul and Child *)
 4 consts  Kill::"indiv⇒τ" Hurt::"indiv⇒τ" Drive_carefully::"indiv⇒τ" Stay::"indiv⇒τ"
 5 Drunk::"indiv⇒τ" Jump::"indiv⇒τ" Drive::"indiv⇒τ"
 6 
 7 axiomatization where
 8 (* Norms *)
 9 A0: "⌊⊤  ⟶  □k(¬ Kill Child ∧ ¬ Hurt Child)⌋" and
10 A1: "⌊⊤  ⟶  □k(Drive_carefully Ali)⌋" and
11 A2: "⌊¬ Drive_carefully Ali  ⟶  □k(Stay Ali)⌋"and
12 (* Input set *)
13 A3: "⌊Drunk Ali⌋" and
14 A4: "⌊Drive Ali⌋" and
15 A6: "⌊Drunk Ali  ⟶  ¬ Drive_carefully Ali⌋" and
16 A7: "⌊(¬ Drive_carefully Ali ∧ Drive Ali ∧ Jump Child)  ⟶  (Kill Child ∨ Hurt Child)⌋"
17 
18 (* Consistency is confirmed by nitpick *)
19 lemma True nitpick [satisfy,user_axioms,show_all,expect=genuine] oops
20 
21 lemma "⌊□k(Stay Ali)⌋"
22   using A2 A3 A6 sledgehammer by (simp add: kimp_def kvalid_def)
23 
24 lemma "⌊□k(Drive_carefully Ali) ∧ ¬ Drive_carefully Ali⌋"  using A1 A3 A6
25   sledgehammer by (simp add: kand_def kimp_def ktrue_def kvalid_def)
26 
27 lemma"⌊□k(¬ Kill Child ∧ ¬ Hurt Child) ∧ (Kill Child ∨ Hurt Child)⌋"
28   nitpick [user_axioms,show_all,expect=genuine] oops
29 end
```

Figure 5: *Drink and Drive* scenario for *Ali* in Isabelle/HOL

For the individual *Ali*, the set of *Norms* remains the same except that we adapted the name of the individual accordingly. However, in *Ali's* case, the *child* was not involved. Therefore, the *Input* set only consists of our facts: A3, A4, A6 and A7 (cf. Fig. 5, lines 13–16).

The formalization of *Ali's* scenario is consistent, again proven by Nitpick (cf. Figure 5, line 19). In contrast to *Paul*, *Ali* did not violated the norm A0 as Nitpick find a counter model for the corresponding statement (cf. Fig. 5, lines 27–28).

6 Conclusion

We have presented an embedding of two I/O operations in HOL and we have shown that each embedding is faithful, i.e., sound and complete. The work presented here continues a project started in Benzmüller et al. [6], and aiming at providing the theoretical foundation for the implementation and automation of deontic logic within existing theorem provers and proof assistants for HOL. Future research should investigate whether the provided implementation already supports non-trivial applications in practical normative reasoning such as legal reasoning or multi-agent systems, or whether further improvements are required. We could also employ our implementation to systematically study some meta-logical properties of I/O logic within Isabelle/HOL. Moreover, we could analogously implement intuitionistic I/O logic [23].

Acknowledgements

We thank an anonymous reviewer for valuable comments.

References

[1] Andrews, P.B.: Church's type theory. In: Zalta, E.N. editor, The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Summer 2018 Edition (2018)

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

[3] Benzmüller, C.: Universal (meta-)logical reasoning: Recent successes. Science of Computer Programming, **172**, 48–62 (2019)

[4] Benzmüller, C., Brown, C., Kohlhase, M.: Higher-order semantics and extensionality. Journal of Symbolic Logic, **69**(4), 1027–1088 (2004)

[5] Benzmüller, C., Farjami, A., Parent., X.: Faithful semantical embedding of a dyadic deontic logic in HOL. arXiv preprint, arXiv:1802.08454 [cs.AI] (2018)

[6] Benzmüller, C., Farjami, A., Parent, X.: A dyadic deontic logic in HOL. In: Broersen, J., Condoravdi, C., Nair, S., Pigozzi, G. (eds.) Deontic Logic and Normative Systems — 14th International Conference, DEON 2018, Utrecht, The Netherlands, 3-6 July, 2018, pp. 33–50, College Publications, UK (2018). (John-Jules Meyer Best Paper Award).

[7] Benzmüller, C., Farjami, A., Parent, X.: Åqvist's dyadic deontic logic E in HOL. Journal of Applied Logics – IfCoLoG Journal of Logics and their Applications (2019). (To appear, URL (preprint): `http://orbilu.uni.lu/handle/10993/37014`)

[8] Benzmüller, C., Parent, X.: I/O logic in HOL – First steps. arXiv preprint, arXiv:1803.09681 [cs.AI] (2018)

[9] Benzmüller, C., Paulson, L.: Multimodal and intuitionistic logics in simple type theory. The Logic Journal of the IGPL, **18**(6), 881–892 (2010)

[10] Benzmüller, C., Paulson, L.C.: Quantified multimodal logics in simple type theory. Logica Universalis (Special Issue on Multimodal Logics), **7**(1), 7–20 (2013)

[11] Benzmüller, C., Sultana, N., Paulson, L. C., Theiß, F.: The higher-order prover LEO-II. Journal of Automated Reasoning, **55**(4), 389–404 (2015)

[12] Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: Kaufmann, M., Paulson, L. C. (eds.) International Conference on Interactive Theorem Proving 2010, LNCS, vol. 6172 pp. 131–146, Springer (2010)

[13] Blanchette, J. C., Paulson, L. C.: Hammering away – A user's guide to Sledgehammer for Isabelle/HOL (2017)

[14] Boella, G., van der Torre, L.: Regulative and constitutive norms in normative multiagent systems. In: Dubois, D., Welty, C., Williams, M. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004, pp. 255–266, AAAI Press, USA (2004)

[15] Church, A.: A formulation of the simple theory of types. Journal of Symbolic Logic, **5**(2), 56–68 (1940)

[16] Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L.: Handbook of deontic logic and normative systems. Volume 1. College Publications, UK (2013)

[17] Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L.: Handbook of deontic logic and normative systems. Volume 2. College Publications, UK. (To appear)

[18] Henkin, L.: Completeness in the theory of types. Journal of Symbolic Logic, **5**(2), 81–91 (1950)

[19] Makinson, D., van der Torre, L.: Input/output logics. Journal of Philosophical Logic, **29**(4), 383–408 (2000)

[20] Nagel, T.: Moral luck. Chapter in Mortal Questions, Cambridge University Press, New York (1979)

[21] Nelkin, K.: Moral luck. In: Zalta, E.N. editor, The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University (2013)

[22] Nipkow, T., Paulson, L.C., Wenzel., M.: Isabelle/HOL — A proof assistant for higher-order logic, volume 2283 of Lecture Notes in Computer Science. Springer (2002)

[23] Parent, X.: A modal translation of an intuitionistic I/O operation. Presented at the 7th Workshop on Intuitionistic Modal Logic and Applications (IMLA 2017), organized by V. de Paiva and S. Artemov at the University of Toulouse (France), 17-28 July, 2017 (satellite workshop of ESSLI17). Post-conference version of the paper under review.

[24] Steen, A., Benzmüller, C.: The higher-order prover Leo-III. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) Automated Reasoning. IJCAR 2018, LNCS, vol. 10900, pp. 108-116, Springer (2018)

Received \jreceived