# Automating Access Control Logics in Simple Type Theory with LEO-II [*]

Christoph Benzmüller

**Abstract** Garg and Abadi recently proved that prominent access control logics can be translated in a sound and complete way into modal logic S4. We have previously outlined how normal multimodal logics, including monomodal logics K and S4, can be embedded in simple type theory and we have demonstrated that the higher-order theorem prover LEO-II can automate reasoning in and about them. In this paper we combine these results and describe a sound (and complete) embedding of different access control logics in simple type theory. Employing this framework we show that the off the shelf theorem prover LEO-II can be applied to automate reasoning in and about prominent access control logics.

## 1 Introduction

The provision of effective and reliable control mechanisms for accessing resources is an important issue in many areas. In computer systems, for example, it is important to effectively control the access to personalized or security critical files.

A prominent and successful approach to implement access control relies on logic based ideas and tools. Abadi's article [2] provides a brief overview on the frameworks and systems that have been developed under this approach. Garg and Abadi recently showed that several prominent access control logics can be translated into modal logic S4 [18]. They proved that this translation is sound and complete.

We have previously shown [10] how multimodal logics can be elegantly embedded in simple type theory (*STT*) [15, 5]. We have also demonstrated that proof prob-

Christoph Benzmüller
International University in Germany, Bruchsal, Germany, e-mail: `c.benzmueller@googlemail.com`

[*] This work was supported by EU grant PIIF-GA-2008-219982 (THFTPTP).

lems in and about multimodal logics can be effectively automated with the higher-order theorem prover LEO-II [12].

In this paper we combine the above results and show that different access control logics can be embedded in *STT*, which has a well understood syntax and semantics [22, 4, 3, 9].

The expressiveness of *STT* furthermore enables the encoding of the entire translation from access control logic input syntax to *STT* in *STT* itself, thus making it as transparent as possible. Our embedding furthermore demonstrates that prominent access control logics as well as prominent multimodal logics can be considered and treated as natural fragments of *STT*.

Using our embedding, reasoning in and about access control logic can be automated in the higher-order theorem prover LEO-II. Since LEO-II generates proof objects, the entire translation and reasoning process is in principle accessible for independent proof checking.

This paper is structured as follows: Section 2 reviews background knowledge and Section 3 outlines the translation of access control logics into modal logic S4 as proposed by Garg and Abadi [18]. Section 4 restricts the general embedding of multimodal logics into *STT* [10] to an embedding of monomodal logics K and S4 into *STT* and proves its soundness. These results are combined in Section 5 in order to obtain a sound (and complete) embedding of access control logics into *STT*. Moreover, we present some first empirical evaluation of the approach with the higher-order automated theorem prover LEO-II. Section 6 concludes the paper.

## 2 Preliminaries

We assume familiarity with the syntax and semantics and of multimodal logics and simple type theory and only briefly review the most important notions.

The multimodal logic language *ML* is defined by

$$s,t ::= p \mid \neg s \mid s \vee t \mid \Box_r s$$

where $p$ denotes atomic primitives and $r$ denotes accessibility relations (distinct from $p$). Other logical connectives can be defined from the chosen ones in the usual way.

A Kripke frame for *ML* is a pair $\langle W, (R_r)_{r\in I:=\{1,\dots,n\}}\rangle$, where $W$ is a non-empty set (called possible worlds), and the $R_r$ are binary relations on $W$ (called accessibility relations). A Kripke model for *ML* is a triple $\langle W, (R_r)_{r\in I}, \models\rangle$, where $\langle W, (R_r)_{r\in I}\rangle$ is a Kripke frame, and $\models$ is a satisfaction relation between nodes of $W$ and formulas of *ML* satisfying: $w \models \neg s$ if and only if $w \not\models s$, $w \models s \vee t$ if and only if $w \models s$ or $w \models t$, $w \models \Box_r s$ if and only if for all $u$ with $R_r(w,u)$ holds $u \models s$. The satisfaction relation $\models$ is uniquely determined by its value on the atomic primitives $p$. A formula $s$ is valid in a Kripke model $\langle W, (R_r)_{r\in I}, \models\rangle$, if $w \models s$ for all $w \in W$. $s$ is valid in a Kripke frame $\langle W, (R_r)_{r\in I}\rangle$ if it is valid in $\langle W, (R_r)_{r\in I}, \models\rangle$ for all possible $\models$. If $s$ is valid for all possible Kripke frames $\langle W, (R_r)_{r\in I}\rangle$, then $s$ is called valid and we write $\models^K s$. $s$ is called $S4$-valid (we write $\models^{S4} s$) if it is valid in all reflexive, transitive Kripke frames $\langle W, (R_r)_{r\in I}\rangle$, that is, Kripke frames with only reflexive and transitive relations $R_r$.

Classical higher-order logic or simple type theory *STT* [5, 15] is a formalism built on top of the simply typed $\lambda$-calculus. The set $\mathscr{T}$ of simple types is usually freely generated from a set of basic types $\{o, \iota\}$ (where $o$ denotes the type of Booleans) using the function type constructor $\rightarrow$.

The simple type theory language *STT* is defined by ($\alpha, \beta, o \in \mathscr{T}$):

$$s,t ::=$$
$$p_\alpha | X_\alpha | (\lambda X_\alpha \mathbin{.} s_\beta)_{\alpha\to\beta} | (s_{\alpha\to\beta}\, t_\alpha)_\beta | (\neg_{o\to o}\, s_o)_o | (s_o \vee_{o\to o\to o} t_o)_o | (\Pi_{(\alpha\to o)\to o}\, s_{\alpha\to o})_o$$

$p_\alpha$ denotes typed constants and $X_\alpha$ typed variables (distinct from $p_\alpha$) . Complex typed terms are constructed via abstraction and application. Our logical connectives of choice are $\neg_{o\to o}$, $\vee_{o\to o\to o}$ and $\Pi_{(\alpha\to o)\to o}$ (for each type $\alpha$). From these connectives, other logical connectives, such as $\Rightarrow, \wedge, \bot$, and $\top$, can be defined in the usual way. We often use binder notation $\forall X_\alpha \mathbin{.} s$ and $\exists X_\alpha \mathbin{.} t$ for $(\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha \mathbin{.} s_o))$ and $\neg(\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha \mathbin{.} \neg t_o))$. We denote substitution of a term $s_\alpha$ for a variable $X_\alpha$ in a term $t_\beta$ by $[s/X]t$. Since we consider $\alpha$-conversion implicitly, we assume the bound variables of $B$ avoid variable capture. Two common relations on terms are given by $\beta$-reduction and $\eta$-reduction. A $\beta$-redex $(\lambda X \mathbin{.} s)t$ $\beta$-reduces to $[t/X]s$. An $\eta$-redex $(\lambda X \mathbin{.} sX)$ where variable $X$ is not free in $s$, $\eta$-reduces to $s$. We write $s{=_\beta}t$ to mean $s$ can be converted to $t$ by a series of $\beta$-reductions and expansions. Similarly, $s{=_{\beta\eta}}t$ means $s$ can be converted to $t$ using both $\beta$ and $\eta$.

Semantics of *STT* is well understood and thoroughly documented in the literature [9, 3, 4, 22]; our summary below is adapted from Andrews [6].

A frame is a collection $\{D_\alpha\}_{\alpha\in\mathscr{T}}$ of nonempty domains (sets) $D_\alpha$, such that $D_o = \{T, F\}$ (where $T$ represents truth and $F$ represents falsehood). The $D_{\alpha\to\beta}$ are collections of functions mapping $D_\alpha$ into $D_\beta$. The members of $D_\iota$ are called individuals. An interpretation is a tuple $\langle\{D_\alpha\}_{\alpha\in\mathscr{T}}, I\rangle$ where function $I$ maps each typed constant $c_\alpha$ to an appropriate element of $D_\alpha$, which is called the denotation of $c_\alpha$ (the denotations of $\neg$, $\vee$ and $\Pi$ are always chosen as intended). A variable assignment $\phi$ maps variables $X_\alpha$ to elements in $D_\alpha$. An interpretation $\langle\{D_\alpha\}_{\alpha\in\mathscr{T}}, I\rangle$ is a Henkin model (general model) if and only if there is a binary function $\mathscr{V}$ such that $\mathscr{V}_\phi\, s_\alpha \in D_\alpha$ for each variable assignment $\phi$ and term $s_\alpha \in L$, and the following conditions are satisfied for all $\phi$ and all $s, t \in L$: (a) $\mathscr{V}_\phi X_\alpha = \phi X_\alpha$, (b) $\mathscr{V}_\phi\, p_\alpha = I p_\alpha$, (c) $\mathscr{V}_\phi(s_{\alpha\to\beta}\, t_\alpha) = (\mathscr{V}_\phi\, s_{\alpha\to\beta})(\mathscr{V}_\phi t_\alpha)$, and (d) $\mathscr{V}_\phi(\lambda X_\alpha \mathbin{.} s_\beta)$ is that function from $D_\alpha$ into $D_\beta$ whose value for each argument $z \in D_\alpha$ is $\mathscr{V}_{[z/X_\alpha],\phi} s_\beta$, where $[z/X_\alpha], \phi$ is that variable assignment such that $([z/X_\alpha], \phi)X_\alpha = z$ and $([z/X_\alpha], \phi)Y_\beta = \phi Y_\beta$ if $Y_\beta \neq X_\alpha$.[2]

[2] Since $I\neg$, $I\vee$, and $I\Pi$ are always chosen as intended, we have $\mathscr{V}_\phi(\neg s) = T$ iff $\mathscr{V}_\phi\, s = F$ , $\mathscr{V}_\phi(s \vee t) = T$ iff $\mathscr{V}_\phi\, s = T$ or $\mathscr{V}_\phi\, t = T$, and $\mathscr{V}_\phi(\forall X_\alpha \mathbin{.} s_o) = \mathscr{V}_\phi(\Pi^\alpha(\lambda X_\alpha \mathbin{.} s_o)) = T$ iff for all $z \in D_\alpha$ we have $\mathscr{V}_{[z/X_\alpha],\phi}\, s_o = T$. Moreover, we have $\mathscr{V}_\phi\, s = \mathscr{V}_\phi\, t$ whenever $s{=_{\beta\eta}}t$.

If an interpretation $\langle \{D_\alpha\}_{\alpha\in\mathscr{T}}, I\rangle$ is a Henkin model, the function $\mathscr{V}_\phi$ is uniquely determined. An interpretation $\langle \{D_\alpha\}_{\alpha\in\mathscr{T}}, I\rangle$ is a standard model if and only if for all $\alpha$ and $\beta$, $D_{\alpha\rightarrow\beta}$ is the set of all functions from $D_\alpha$ into $D_\beta$. Each standard model is also a Henkin model.

We say that formula $A \in L$ is valid in a model $\langle \{D_\alpha\}_{\alpha\in\mathscr{T}}, I\rangle$ if an only if $\mathscr{V}_\phi A = T$ for every variable assignment $\phi$. A model for a set of formulas $H$ is a model in which each formula of $H$ is valid.

A formula $A$ is Henkin-valid (standard-valid) if and only if $A$ is valid in every Henkin (standard) model. Clearly each formula which is Henkin-valid is also standard-valid, but the converse of this statement is false. We write $\models^{STT} A$ if $A$ is Henkin-valid and we write $\Gamma \models^{STT} A$ if $A$ is valid in all Henkin models in which all formulas of $\Gamma$ are valid.

## 3 Translating Access Control Logic to Modal Logic

The access control logic *ICL* studied by Garg and Abadi [18] is defined by

$$s ::= p \,|\, s_1 \wedge s_2 \,|\, s_1 \vee s_2 \,|\, s_1 \supset s_2 \,|\, \bot \,|\, \top \,|\, A \texttt{ says } s$$

$p$ denotes atomic propositions, $\wedge$, $\vee$, $\supset$, $\bot$ and $\top$ denote the standard logical connectives, and $A$ denotes principals, which are atomic and distinct from the atomic propositions $p$. Expressions of the form `A says s`, intuitively mean that $A$ `asserts` (or `supports`) $s$. *ICL* inherits all inference rules of intuitionistic propositional logic. The logical connective `says` satisfies the following axioms:

$$\vdash s \supset (A \texttt{ says } s) \qquad \text{(unit)}$$
$$\vdash (A \texttt{ says } (s \supset t)) \supset (A \texttt{ says } s) \supset (A \texttt{ says } t) \qquad \text{(cuc)}$$
$$\vdash (A \texttt{ says } A \texttt{ says } s) \supset (A \texttt{ says } s) \qquad \text{(idem)}$$

*Example 1 (from [18]).* We consider a file-access scenario with an administrating principal `admin`, a user `Bob`, one file `file1`, and the following policy:

1. If `admin` says that `file1` should be deleted, then this must be the case.
2. `admin` trusts Bob to decide whether `file1` should be deleted.
3. `Bob` wants to delete `file1`.

This policy can be encoded in *ICL* as follows:

$$(\texttt{admin says deletefile1}) \supset \texttt{deletefile1} \qquad (1.1)$$
$$\texttt{admin says } ((\texttt{Bob says deletefile1}) \supset \texttt{deletefile1}) \qquad (1.2)$$
$$\texttt{Bob says deletefile1} \qquad (1.3)$$

The question whether `file1` should be deleted in this situation corresponds to proving `deletefile` (1.4), which follows from (1.1)-(1.3), (unit), and (cuc).

Garg and Abadi [18] propose the following mapping $\lceil . \rceil$ of *ICL* formulas into modal logic S4 formulas (similar to Gödels translation from intuitionistic logic to S4 [19] and by providing a mapping for the additional connective `says` ).

$$\begin{array}{rclcrcl}
\lceil p \rceil &=& \Box p & & & & \\
\lceil s \wedge t \rceil &=& \lceil s \rceil \wedge \lceil t \rceil & & \lceil \top \rceil &=& \top \\
\lceil s \vee t \rceil &=& \lceil s \rceil \vee \lceil t \rceil & & \lceil \bot \rceil &=& \bot \\
\lceil s \supset t \rceil &=& \Box(\lceil s \rceil \supset \lceil t \rceil) & & \lceil A \, \mathtt{says} \, s \rceil &=& \Box(A \vee \lceil s \rceil)
\end{array}$$

Logic $ICL^{\Longrightarrow}$ extends $ICL$ by a *speaks-for* operator (represented by $\Longrightarrow$) which satisfies the following axioms:

$$\begin{array}{ll}
\vdash A \Longrightarrow A & \text{(refl)} \\
\vdash (A \Longrightarrow B) \supset (B \Longrightarrow C) \supset (A \Longrightarrow C) & \text{(trans)} \\
\vdash (A \Longrightarrow B) \supset (A \, \mathtt{says} \, s) \supset (B \, \mathtt{says} \, s) & \text{(speaking-for)} \\
\vdash (B \, \mathtt{says} \, (A \Longrightarrow B)) \supset (A \Longrightarrow B) & \text{(handoff)}
\end{array}$$

The use of the new $\Longrightarrow$ operator is illustrated by the following modification of Example 1.

*Example 2 (from [18]).* `Bob` delegates his authority to delete `file1` to `Alice` (see (2.3)), who now wants to delete `file1`.

$$\begin{array}{ll}
(\mathtt{admin} \, \mathtt{says} \, \mathtt{deletefile1}) \supset \mathtt{deletefile1} & (2.1) \\
\mathtt{admin} \, \mathtt{says} \, ((\mathtt{Bob} \, \mathtt{says} \, \mathtt{deletefile1}) \supset \mathtt{deletefile1}) & (2.2) \\
\mathtt{Bob} \, \mathtt{says} \, \mathtt{Alice} \Longrightarrow \mathtt{Bob} & (2.3) \\
\mathtt{Alice} \, \mathtt{says} \, \mathtt{deletefile1} & (2.4)
\end{array}$$

Using these facts and (handoff) and (speaking-for) one can prove `deletefile` (2.5)

The translation of $ICL^{\Longrightarrow}$ into S4 extends the translation from $ICL$ to S4 by

$$\lceil A \Longrightarrow B \rceil = \Box(A \supset B)$$

Logic $ICL^B$ differs from $ICL$ by allowing that principals may contain Boolean connectives ($a$ denotes atomic principals distinct from atomic propositions):

$$A, B ::= a \,|\, A \wedge B \,|\, A \vee B \,|\, A \supset B \,|\, \bot \,|\, \top$$

$ICL^B$ satisfies the following additional axioms:

$$\begin{array}{ll}
\vdash (\bot \, \mathtt{says} \, s) \supset s & \text{(trust)} \\
\text{If } A \equiv \top \text{ then } \vdash A \, \mathtt{says} \, \bot & \text{(untrust)} \\
\vdash ((A \supset B) \, \mathtt{says} \, s) \supset (A \, \mathtt{says} \, s) \supset (B \, \mathtt{says} \, s) & \text{(cuc')}
\end{array}$$

Abadi and Garg show that the `speaks-for` operator from $ICL^{\Longrightarrow}$ is definable in $ICL^B$. The use of $ICL^B$ is illustrated by the following modification of Example 1.

*Example 3 (from [18]).* `admin` is trusted on `deletefile1` and its consequences (3.1). (3.2) says that `admin` further delegates this authority to `Bob`.

$$\begin{array}{ll}
(\mathtt{admin} \, \mathtt{says} \, \bot) \supset \mathtt{deletefile1} & (3.1) \\
\mathtt{admin} \, \mathtt{says} \, ((\mathtt{Bob} \supset \mathtt{admin}) \, \mathtt{says} \, \mathtt{deletefile1}) & (3.2) \\
\mathtt{Bob} \, \mathtt{says} \, \mathtt{deletefile1} & (3.3)
\end{array}$$

Using these facts and the available axioms one can again prove `deletefile` (3.4).

The translation of $ICL^B$ into S4 is the same as the translation from $ICL$ to S4. However, the mapping $\lceil A \texttt{ says } s \rceil = \Box(A \vee \lceil s \rceil)$ now guarantees that Boolean principal expressions $A$ are mapped one-to-one to Boolean expressions in S4.

Garg and Abadi prove their translations sound and complete:

**Theorem 1 (Soundness and Completeness).** *$\vdash s$ in ICL (resp. $ICL^{\Rightarrow}$ and $ICL^B$) if and only if $\vdash \lceil s \rceil$ in S4.*

*Proof.* See Theorem 1 (resp. Theorem 2 and Theorem 3) og Garg and Abadi [18].

## 4 Embedding Modal Logic in Simple Type Theory

Embeddings of modal logics into higher-order logic have not yet been widely studied, although multimodal logic can be regarded as a natural fragment of $STT$. Gallin [16] appears to mention the idea first. He presents an embedding of modal logic into a 2-sorted type theory. This idea is picked up by Gamut [17] and a related embedding has recently been studied by Hardt and Smolka [20]. Carpenter [14] proposes to use lifted connectives, an idea that is also underlying the embeddings presented by Merz [24], Brown [13], Harrison [21, Chap. 20], and Kaminski and Smolka [23].

In our previous work [10] we pick up and extend the embedding of multimodal logics into $STT$ as studied by Brown [13]. The starting point is a characterization of multimodal logic formulas as particular $\lambda$-terms in $STT$. A distinctive characteristic of the encoding is that the definiens of the $\Box_R$ operator $\lambda$-abstracts over the accessibility relation $R$. As we have shown this supports the formulation of meta properties of encoded multimodal logics such as the correspondence between certain axioms and properties of the accessibility relation $R$. And some of these meta properties can be efficiently automated within our higher-order theorem prover LEO-II.

The general idea of this encoding is very simple: Choose base type $\iota$ and let this type denote the set of all possible worlds. Certain formulas of type $\iota \to o$ then correspond to multimodal logic expressions, whereas the modal operators $\neg$, $\vee$, and $\Box_r$ itself become $\lambda$-terms of type $(\iota \to o) \to (\iota \to o)$, $(\iota \to o) \to (\iota \to o) \to (\iota \to o)$, and $(\iota \to \iota \to o) \to (\iota \to o) \to (\iota \to o)$ respectively.

The mapping $\lfloor . \rfloor$ translates formulas of multimodal logic $ML$ into terms of type $\iota \to o$ in $STT$:

$$\begin{aligned}
\lfloor p \rfloor &= p_{\iota\to o} & |p| &= p_{\iota\to o}\\
\lfloor r \rfloor &= r_{\iota\to\iota\to o} & |r| &= r_{\iota\to\iota\to o}\\
\lfloor \neg\, s \rfloor &= \lambda X_\iota\text{.}\,\neg(\lfloor s \rfloor X) & |\neg| &= \lambda A_{\iota\to o}\text{.}\,\lambda X_\iota\text{.}\,\neg(A\,X)\\
\lfloor s \vee t \rfloor &= \lambda X_\iota\text{.}\,(\lfloor s \rfloor X) \vee (\lfloor t \rfloor X) & |\vee| &= \lambda A_{\iota\to o}\text{.}\,\lambda B_{\iota\to o}\text{.}\,\lambda X_\iota\text{.}\,(A\,X) \vee (B\,X)\\
\lfloor \Box_r\, s \rfloor &= \lambda X_\iota\text{.}\,\forall Y_\iota\text{.}\,(\lfloor r \rfloor X\,Y) \Rightarrow (\lfloor s \rfloor Y) & |\Box| &= \lambda R_{\iota\to\iota\to o}\text{.}\,\lambda A_{\iota\to o}\text{.}\\
&& &\qquad \lambda X_\iota\text{.}\,\forall Y_\iota\text{.}\,(R\,X\,Y) \Rightarrow (A\,Y)
\end{aligned}$$

The expressiveness of *STT* (in particular the use of $\lambda$-abstraction and $\beta\eta$-conversion) allows us to replace mapping $\lfloor . \rfloor$ by mapping $|.|$ which works locally and is not recursive.[3]

It is easy to check that this local mapping works as intended. For example,

$$|\Box_r\, p \vee \Box_r\, q)| := |\vee|\,(|\Box|\,|r|\,|p|)\,(|\Box|\,|r|\,|q|) {=}_{\beta\eta} \lfloor \Box_r\, p \vee \Box_r\, q) \rfloor$$

Further local definitions for other multimodal logic operators can be introduced this way. For example, $|\supset| = \lambda A_{\iota\to o}. \lambda B_{\iota\to o}. \lambda X_\iota. (A\,X) \Rightarrow (B\,X)$, $|\bot| = \lambda A_{\iota\to o}. \bot$, $|\top| = \lambda A_{\iota\to o}. \top$, and $|\wedge| = \lambda A_{\iota\to o}. \lambda B_{\iota\to o}. \lambda X_\iota. (A\,X) \wedge (B\,X)$.

A notion of validity for the $\lambda$-terms (of type $\iota \to o$) which we obtain via definition expansion is still missing: We want $A_{\iota\to o}$ to be valid if and only if for all possible worlds $w_\iota$ we have $(A_{\iota\to o}\, w_\iota)$, that is, $w \in A$. This notion of validity is again introduced as a local definition:

$$|\mathtt{Mval}| := \lambda A_{\iota\to o}. \forall W_\iota. A\, W$$

Garg and Abadi's translation of access control into modal logic as presented in Section 3 is monomodal and does not require different $\Box_r$-operators. Thus, for the purpose of this paper we restrict the outlined general embedding of multimodal logics into *STT* to an embedding of monomodal logic into *STT*. Hence, for the remainder of the paper we assume that *ML* provides exactly one $\Box_r$-operator, that is, a single relation constant $r$.

We next study soundness of this embedding. Our soundness proof below employs the following mapping of Kripke frames into Henkin models.

**Definition 1 (Henkin model $M^K$ for Kripke Model $K$).** Given a Kripke model $K = \langle W, (R_r), \models \rangle$. Henkin model $M^K = \langle \{D_\alpha\}_{\alpha\in\mathscr{T}}, I \rangle$ for $K$ is defined as follows: We choose the set of individuals $D_\iota$ as the set of worlds $W$ and we choose the $D_{\alpha\to\beta}$ as the set of all functions from $D_\alpha$ to $D_\beta$. Let $p^1, \dots, p^m$ for $m \geq 1$ be the atomic primitives occuring in modal language *ML*. Remember that $\Box_r$ is the only box operator of *ML*. Note that $|p^j| = p^j_{\iota\to o}$ and $|r| = r_{\iota\to\iota\to o}$. Thus, for $1 \leq i \leq m$ we choose $Ip^j_{\iota\to o} \in D_{\iota\to o}$ such that $(Ip^j_{\iota\to o})(w) = T$ for all $w \in D_\iota$ with $w \models p^j$ in Kripke model $K$ and $(Ip^j_{\iota\to o})(w) = F$ otherwise. Similarly, we choose $Ir_{\iota\to\iota\to o} \in D_{\iota\to\iota\to o}$ such that $(Ir_{\iota\to\iota\to o})(w, w') = T$ if $R_r(w, w')$ in Kripke model $K$ and $(Ir_{\iota\to\iota\to o})(w, w') = F$ otherwise. Clearly, if $R_r$ is reflexive and transitive then, by construction, $Ir_{\iota\to\iota\to o}$ is so as well. It is easy to check that $M^K = \langle \{D_\alpha\}_{\alpha\in\mathscr{T}}, I \rangle$ is a Henkin model. In fact it is a standard model since the function spaces are full.

[3] Note that the encoding of the modal operators $\Box_r$ is chosen to explicitly depend on an accessibility relation $r$ of type $\iota \to \iota \to o$ given as first argument to it. Hence, we basically introduce a generic framework for modeling multimodal logics. This idea is due to Brown and it is this aspect where the encoding differs from the LTL encoding of Harrison. The latter chooses the interpreted type *num* of numerals and then uses the predefined relation $\leq$ over numerals as fixed accessibility relation in the definitions of $\Box$ and $\Diamond$. By making the dependency of $\Box_r$ and $\Diamond_r$ on the accessibility relation $r$ explicit, we cannot only formalize but also automatically prove some meta properties of multimodal logics as we have previously demonstrated [10].

**Lemma 1.** *Let $M^K = \langle \{D_\alpha\}_{\alpha\in\mathscr{T}}, I\rangle$ be a Henkin model for Kripke model $K = \langle W, (R_i)_{i\in I}, \models\rangle$. For all $q \in L$, $w \in W$ and variable assignments $\phi$ the following are equivalent: (i) $w \models q$, (ii) $\mathscr{V}_{[w/Z_\iota],\phi}(\lfloor q\rfloor Z) = T$, and (iii) $\mathscr{V}_{[w/Z_\iota],\phi}(|q| Z) = T$.*

*Proof.* We prove (i) if and only if (ii) by induction on the structure of $q$. Let $q = p$ for some atomic primitive $p \in L$. By construction of $M^K$, we have $\mathscr{V}_{[w/Z_\iota],\phi}(\lfloor p\rfloor Z) = \mathscr{V}_{[w/Z_\iota],\phi}(p_{\iota\to o} Z) = (I\, p_{\iota\to o})(w) = T$ if and only if $w \models p$. Let $p = \neg s$. We have $w \models \neg s$ if and only $w \not\models s$. By induction we get $\mathscr{V}_{[w/Z_\iota],\phi}(\lfloor s\rfloor Z) = F$ and hence $\mathscr{V}_{[w/Z_\iota],\phi}\, \neg(\lfloor s\rfloor Z) =_{\beta\eta} \mathscr{V}_{[w/Z_\iota],\phi}(\lfloor \neg s\rfloor Z) = T$. Case $p = (s \vee t)$ is similar. Let $q = \Box_r s$. We have $w \models \Box_r s$ if and only if for all $u$ with $R_r(w,u)$ we have $u \models s$. By induction, for all $u$ with $R_r(w,u)$ we have $\mathscr{V}_{[u/V_\iota],\phi}(\lfloor s\rfloor V) = T$. Hence, $\mathscr{V}_{[u/V_\iota],[w/Z_\iota],\phi}((\lfloor r\rfloor Z V) \Rightarrow (\lfloor s\rfloor V)) = T$ and $\mathscr{V}_{[w/Z_\iota],\phi}(\forall Y_\iota\text{.}((\lfloor r\rfloor Z Y) \Rightarrow (\lfloor s\rfloor Y))) =_{\beta\eta} \mathscr{V}_{[w/Z_\iota],\phi}(\lfloor \Box_r s\rfloor Z) = T$.

We leave it to the reader to prove (ii) if and only if (iii).

We now prove soundness of the embedding of normal monomodal logics $K$ and $S4$ into *STT*. In the case of $S4$ we add axioms that correspond to modal logic axioms $T$ (reflexivity) and 4 (transitivity).[4] Here we call these axiom `R` and `T`.

**Theorem 2 (Soundness of the Embedding of $K$ and $S4$ into *STT*).** *Let $s \in ML$ be a monomodal logic proposition.*

1. *If $\models^{STT} |\texttt{Mval}\, s|$ then $\models^K s$.*
2. *If $\{\texttt{R}, \texttt{T}\} \models^{STT} |\texttt{Mval}\, s|$ then $\models^{S4} s$, where `R` and `T` are shorthands for $\forall X_{\iota\to o}\text{.} |\texttt{Mval}\, \Box_r X \supset X|$ and $\forall X_{\iota\to o}\text{.} |\texttt{Mval}\, \Box_r X \supset \Box_r \Box_r X|$ respectively.*

*Proof.* (1) The proof is by contraposition. For this, assume $\not\models^K s$, that is, there is a Kripke model $K = \langle W, (R_r), \models\rangle$ with $w \not\models s$ for some $w \in W$. By Lemma 1, for arbitrary $\phi$ we have $\mathscr{V}_{[w/W_\iota],\phi}(|s| W) = F$ in Henkin model $M^K$ for $K$. Thus, $\mathscr{V}_\phi(\forall W_\iota\text{.}(|s| W) = \mathscr{V}_\phi |\texttt{Mval}\, s| = F$. Hence, $\not\models^{STT} |\texttt{Mval}\, s|$.

(2) The proof is by contraposition. From $\not\models^{S4} s$ we get by Lemma 1 that $|\texttt{Mval}\, s|$ is not valid in Henkin model $M^K = \langle \{D_\alpha\}_{\alpha\in\mathscr{T}}, I\rangle$ for Kripke model $K = \langle W, (R_r)\rangle$. $R_r$ in $K$ is reflexive and transitive, hence, the relation $(I r) \in D_{\iota\to\iota\to o}$ is so as well. We leave it to the reader to verify that axioms `R` and `T` are valid in $M^K$. Hence, $\{\texttt{R}, \texttt{T}\} \not\models^{STT} |\texttt{Mval}\, s|$.

Reasoning problems in modal logics $K$ and $S4$ can thus be considered as reasoning problems in *STT*. Hence, any off the shelf theorem prover that is sound for *STT*, such as our LEO-II, can be applied to them. For example, $\models^{STT} |\texttt{Mval}\, \Box_r \top|$, $\models^{STT} |\texttt{Mval}\, \Box_r a \supset \Box_r a|$, and $\models^{STT} |\texttt{Mval}\, \Diamond_r (a \supset b) \vee (\Box_r a \supset \Box_r b)|$ are automatically proved by LEO-II in 0.024 seconds, 0.026 seconds, and 0.035 seconds respectively. All experiments with LEO-II reported in this paper were conducted with LEO-II version v0.98 [5] on a notebook computer with a Intel Pentium 1.60GHz processor with 1GB memory running Linux.

[4] Note that $T = (\Box_r s \supset s)$ and $4 = (\Box_r s \supset \Box_r \Box_r s)$ are actually axiom schemata in modal logic. As we show here, their counterparts in *STT* actually become proper axioms.

[5] LEO-II is available from `http://www.ags.uni-sb.de/~leo/`.

More impressive example problems illustrating LEO-II's performance for reasoning in and *about* multimodal logic can be found in our previous work [10]. Amongst these problems is also the equivalence between axioms $\Box_r s \supset s$ and $\Box_r s \supset \Box_r \Box_r s$ and the reflexivity and transitivity properties of the accessibility relation $r$:

*Example 4.* $\models^{STT} (\mathtt{R} \wedge \mathtt{T}) \Leftrightarrow (\mathtt{refl}\, r \wedge \mathtt{trans}\, r)$ where $\mathtt{R}$ and $\mathtt{T}$ are the abbreviations as introduced in Theorem 2 and $\mathtt{refl}$ and $\mathtt{trans}$ abbreviations for $\lambda R_{\iota\to\iota\to o}.\forall X_\iota.R X X$ and $\lambda R_{\iota\to\iota\to o}.\forall X_\iota.\forall Y_\iota.\forall Z_\iota.R X Y \wedge R Y Z \Rightarrow R X Z$. LEO-II can solve this modal logic meta-level problem in 2.329 seconds.

## 5 Embedding Access Control Logic in Simple Type Theory

We combine the results from Sections 3 and 4 and obtain the following mapping $\|.\|$ from access control logic *ICL* into *STT*:

$$
\begin{aligned}
\|p\| &= |\Box_r\, p| = \lambda X_\iota.\forall Y_\iota.r_{\iota\to\iota\to o} X\, Y \Rightarrow p_{\iota\to o} Y \\
\|A\| &= |A| = a_{\iota\to o} \ \texttt{(distinct from the } p_{\iota\to o}\texttt{)} \\
\|\wedge\| &= \lambda S.\lambda T.|S \wedge T| = \lambda S_{\iota\to o}.\lambda T_{\iota\to o}.\lambda X_\iota.S X \wedge T X \\
\|\vee\| &= \lambda S.\lambda T.|S \vee T| = \lambda S_{\iota\to o}.\lambda T_{\iota\to o}.\lambda X_\iota.S X \vee T X \\
\|\supset\| &= \lambda S.\lambda T.|\Box_r (S \supset T)| \\
&= \lambda S_{\iota\to o}.\lambda T_{\iota\to o}.\lambda X_\iota.\forall Y_\iota.r_{\iota\to\iota\to o} X\, Y \Rightarrow (S\, Y \Rightarrow T\, Y) \\
\|\top\| &= |\top| = \lambda S_{\iota\to o}.\top \\
\|\bot\| &= |\bot| = \lambda S_{\iota\to o}.\bot \\
\|\mathtt{says}\| &= \lambda A.\lambda S.|\Box_r (A \vee S)| \\
&= \lambda A_{\iota\to o}.\lambda S_{\iota\to o}.\lambda X_\iota.\forall Y_\iota.r_{\iota\to\iota\to o} X\, Y \Rightarrow (A\, Y \vee S\, Y)
\end{aligned}
$$

It is easy to verify that this mapping works as intended. For example:

$$
\begin{aligned}
\|\mathtt{admin\ says}\ \bot\| \ &:= \ \|\mathtt{says}\|\|\mathtt{admin}\|\|\bot\| \\
&=_{\beta\eta} \lambda X_\iota.\forall Y_\iota.r_{\iota\to\iota\to o} X\, Y \Rightarrow (\mathtt{admin}_{\iota\to o} Y \vee \bot) \\
&=_{\beta\eta} |\Box_r (\mathtt{admin} \vee \bot)| =_{\beta\eta} \lfloor \Box_r (\mathtt{admin} \vee \bot) \rfloor \\
&= \lfloor \lceil \mathtt{admin\ says}\ \bot \rceil \rfloor
\end{aligned}
$$

We extend this mapping to logic $ICL^{\Rightarrow}$ by adding a clause for the speaks-for connective $\Longrightarrow$:

$$
\|\Longrightarrow\| = \lambda A.\lambda B.|\Box_r (A \supset B)| = \lambda A_{\iota\to o}.\lambda B_{\iota\to o}.\lambda X_\iota.\forall Y_\iota.r_{\iota\to\iota\to o} X\, Y \Rightarrow (A\, Y \Rightarrow B\, Y)
$$

For the translation of $ICL^B$ we simply allow that the ICL connectives can be applied to principals. Our mapping $\|.\|$ needs not to be modified and is applicable as is.

**Table 1** Performance of LEO-II when applied to problems in access control logic *ICL*

| Name | Problem | LEO (s) |
|---|---|---|
| unit | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, s \supset (A\ \mathtt{says}\ s)\rVert$ | 0.031 |
| cuc | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, (A\ \mathtt{says}\ (s \supset t)) \supset (A\ \mathtt{says}\ s) \supset (A\ \mathtt{says}\ t)\rVert$ | 0.083 |
| idem | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, (A\ \mathtt{says}\ A\ \mathtt{says}\ s) \supset (A\ \mathtt{says}\ s)\rVert$ | 0.037 |
| Ex1 | $\{\mathtt{R},\mathtt{T},\lVert \mathtt{ICLval}\,(1.1)\rVert,\dots,\lVert \mathtt{ICLval}\,(1.3)\rVert\} \models^{STT} \lVert \mathtt{ICLval}\,(1.4)\rVert$ | 3.494 |
| unit$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, s \supset (A\ \mathtt{says}\ s)\rVert$ | – |
| cuc$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, (A\ \mathtt{says}\ (s \supset t)) \supset (A\ \mathtt{says}\ s) \supset (A\ \mathtt{says}\ t)\rVert$ | – |
| idem$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, (A\ \mathtt{says}\ A\ \mathtt{says}\ s) \supset (A\ \mathtt{says}\ s)\rVert$ | – |
| Ex1$^K$ | $\{\lVert \mathtt{ICLval}\,(1.1)\rVert,\dots,\lVert \mathtt{ICLval}\,(1.3)\rVert\} \models^{STT} \lVert \mathtt{ICLval}\,(1.4)\rVert$ | – |

The notion of validity for the terms we obtain after translations is chosen identical to before

$$\lVert \mathtt{ICLval} \rVert = \lambda A_{\iota\to o}\centerdot |\mathtt{Mval}\, A| = \lambda A_{\iota\to o}\centerdot \forall W_\iota\centerdot A\, W$$

**Theorem 3 (Soundness of the Embeddings of *ICL*, *ICL*$^{\Rightarrow}$, and *ICL*$^B$ in *STT*).** *Let $s \in ICL$ (resp. $s \in ICL^{\Rightarrow}$, $s \in ICL^B$) and let $\mathtt{R}$ and $\mathtt{T}$ be as before. If $\{\mathtt{R},\mathtt{T}\} \models^{STT}$ $\lVert \mathtt{ICLval}\, s\rVert$ then $\vdash s$ in access control logic ICL (resp. $ICL^{\Rightarrow}$, $ICL^B$).*

*Proof.* If $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, s\rVert$ then $\models^{S4} s$ by Theorem 2 since $\lVert \mathtt{ICLval}\, s\rVert = |\mathtt{Mval}\, s|$. This implies that $\vdash \lceil s \rceil$ for the sound and complete Hilbert System for S4 studied by Garg and Abadi [18].[6] By Theorem 1 we conclude that $\vdash s$ in access control logic *ICL* (resp. $ICL^{\Rightarrow}$, $ICL^B$).

Completeness of our embeddings of *ICL*, $ICL^{\Rightarrow}$, and $ICL^B$ into *STT* can be shown by similar means [8]. This also implies soundness and completeness for the entailed embedding of intuitionistic logic into *STT*.

We can thus safely exploit our framework to map problems formulated in control logics *ICL*, $ICL^{\Rightarrow}$, or $ICL^B$ to problems in *STT* and we can apply the off the shelf higher-order theorem prover LEO-II (which itself cooperates with the first-order theorem prover E [25]) to solve them. Times are given in seconds.

Table 1 shows that LEO-II can effectively prove that the axioms unit, cuc and idem hold as expected in our embedding of *ICL* in *STT*. This provides additional evidence for the correctness of our approach. Example 1 can also be quickly solved by LEO-II. Problems unit$^K$, cuc$^K$, idem$^K$, and Ex1$^K$ modify their counterparts by omitting the axioms R and T. Thus, they essentially test whether these problems can already be proven via a mapping to modal logic $K$ instead of $S4$. LEO-II answers this questions positively for the cases of cuc$^K$, and Ex1$^K$.

Tables 2 and 3 extend this experiment to the other access control logics, axioms and examples presented in Section 3.

In a separate technical report [8] we present the concrete encoding or our embedding together with the problems unit, cuc, idem, and Ex1 in the new TPTP THF syntax [11], which is also the input syntax of LEO-II.

[6] See Theorem 8 of Garg and Abadi [18] which is only given in the full version of the paper available from `http://www.cs.cmu.edu/~dg/publications.html`.

**Table 2** Performance of LEO-II when applied to problems in access control logic $ICL^{\Rightarrow}$

| **Name** | **Problem** | **LEO (s)** |
|---|---|---|
| refl | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, A \Longrightarrow A \rVert$ | 0.052 |
| trans | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, (A \Longrightarrow B) \supset (B \Longrightarrow C) \supset (A \Longrightarrow C) \rVert$ | 0.105 |
| sp.-for | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, (A \Longrightarrow B) \supset (A\ \mathtt{says}\ s) \supset (B\ \mathtt{says}\ s) \rVert$ | 0.062 |
| handoff | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, (B\ \mathtt{says}\ (A \Longrightarrow B)) \supset (A \Longrightarrow B) \rVert$ | 0.036 |
| Ex2 | $\{\mathtt{R},\mathtt{T},\lVert \mathtt{ICLval}\, (2.1) \rVert,\ldots,\lVert \mathtt{ICLval}\, (2.4) \rVert\} \models^{STT} \lVert \mathtt{ICLval}\, (2.5) \rVert$ | 0.698 |
| refl$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, A \Longrightarrow A \rVert$ | 0.031 |
| trans$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, (A \Longrightarrow B) \supset (B \Longrightarrow C) \supset (A \Longrightarrow C) \rVert$ | – |
| sp.-for$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, (A \Longrightarrow B) \supset (A\ \mathtt{says}\ s) \supset (B\ \mathtt{says}\ s) \rVert$ | – |
| handoff$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, (B\ \mathtt{says}\ (A \Longrightarrow B)) \supset (A \Longrightarrow B) \rVert$ | – |
| Ex2$^K$ | $\{\lVert \mathtt{ICLval}\, (2.1) \rVert,\ldots,\lVert \mathtt{ICLval}\, (2.4) \rVert\} \models^{STT} \lVert \mathtt{ICLval}\, (2.5) \rVert$ | – |

**Table 3** Performance of LEO-II when applied to problems in access control logic $ICL^B$

| **Name** | **Problem** | **LEO (s)** |
|---|---|---|
| trust | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, (\bot\ \mathtt{says}\ s) \supset s \rVert$ | 0.049 |
| untrust | $\{\mathtt{R},\mathtt{T},\lVert \mathtt{ICLval}\, A \equiv \top \rVert\} \models^{STT} \lVert \mathtt{ICLval}\, A\ \mathtt{says}\ \bot \rVert$ | 0.053 |
| cuc’ | $\{\mathtt{R},\mathtt{T}\} \models^{STT} \lVert \mathtt{ICLval}\, ((A \supset B)\ \mathtt{says}\ s) \supset (A\ \mathtt{says}\ s) \supset (B\ \mathtt{says}\ s) \rVert$ | 0.131 |
| Ex3 | $\{\mathtt{R},\mathtt{T},\lVert \mathtt{ICLval}\, (3.1) \rVert,\ldots,\lVert \mathtt{ICLval}\, (3.3) \rVert\} \models^{STT} \lVert \mathtt{ICLval}\, (3.4) \rVert$ | 0.076 |
| trust$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, (\bot\ \mathtt{says}\ s) \supset s \rVert$ | – |
| untrust$^K$ | $\{\lVert \mathtt{ICLval}\, A \equiv \top \rVert\} \models^{STT} \lVert \mathtt{ICLval}\, A\ \mathtt{says}\ \bot \rVert$ | 0.041 |
| cuc’$^K$ | $\models^{STT} \lVert \mathtt{ICLval}\, ((A \supset B)\ \mathtt{says}\ s) \supset (A\ \mathtt{says}\ s) \supset (B\ \mathtt{says}\ s) \rVert$ | – |
| Ex3$^K$ | $\{\lVert \mathtt{ICLval}\, (3.1) \rVert,\ldots,\lVert \mathtt{ICLval}\, (3.3) \rVert\} \models^{STT} \lVert \mathtt{ICLval}\, (3.4) \rVert$ | – |

## 6 Conclusion and Future Work

We have outlined a framework for the automation of reasoning in and about different access control logics in simple type theory. Using our framework off the shelf higher-order theorem provers and proof assistants can be applied for the purpose. Our embedding of access control logics in simple type theory and a selection of example problems have been encoded in the new TPTP THF syntax and our higher-order theorem prover LEO-II has been applied to them yielding promising initial results. Our problem encodings have been submitted to the higher-order TPTP library [1] under development in the EU project THFTPTP and are available there for comparison and competition with other TPTP compliant theorem provers such as TPS [7].

Recent experiments have shown that the scalability of our approach for reasoning within access control logics still poses a challenge to LEO-II. However, more promising is the application of LEO-II to meta-properties of access control logics analogous to Example 4 and its use for the exploration of new access control logics.

**Acknowledgments:** Catalin Hritcu inspired this work and pointed to the paper by Garg and Abadi. Chad Brown, Larry Paulson, and Claus-Peter Wirth detected some problems and typos in an earlier version of this paper. Also Deepak Garg provided very valuable feedback to this work.

## References

1. The TPTP THF library is available at `http://www.tptp.org`.
2. Martín Abadi. Logic in access control. In *18th IEEE Symposium on Logic in Computer Science, 22-25 June 2003, Ottawa, Canada, Proceedings*. IEEE Computer Society, 2003.
3. Peter B. Andrews. General models and extensionality. *J. of Symbolic Logic*, 37:395–397, 1972.
4. Peter B. Andrews. General models, descriptions, and choice in type theory. *J. of Symbolic Logic*, 37:385–394, 1972.
5. Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer Academic Publishers, second edition, 2002.
6. Peter B. Andrews. Church's type theory. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2008. http://plato.stanford.edu/archives/fall2008/entries/type-theory-church/.
7. Peter B. Andrews and Chad E. Brown. Tps: A hybrid automatic-interactive system for developing proofs. *J. Applied Logic*, 4(4):367–395, 2006.
8. Christoph Benzmüller. Automating access control logics in simple type theory with LEO-II. SEKI Technical Report SR-2008-01, FB Informatik, U. des Saarlandes, Germany, 2008.
9. Christoph Benzmüller, Chad E. Brown, and Michael Kohlhase. Higher order semantics and extensionality. *J. of Symbolic Logic*, 69:1027–1088, 2004.
10. Christoph Benzmüller and Lawrence Paulson. Festschrift in honour of Peter B. Andrews on his 70th birthday. Studies in Logic and the Foundations of Mathematics, chapter Exploring Properties of Normal Multimodal Logics in Simple Type Theory with LEO-II. IFCoLog, 2009.
11. Christoph Benzmüller, Florian Rabe, and Geoff Sutcliffe. The core TPTP language for classical higher-order logic. In *Fourth International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 5195 of *LNAI*. Springer, 2008.
12. Christoph Benzmüller, Frank Theiss, Larry Paulson, and Arnaud Fietzke. LEO-II - a cooperative automatic theorem prover for higher-order logic. In *Fourth International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 5195 of *LNAI*. Springer, 2008.
13. Chad E. Brown. Encoding hybrid logic in higher-order logic. Unpublished slides from an invited talk presented at Loria Nancy, France, April 2005. `http://mathgate.info/cebrown/papers/hybrid-hol.pdf`.
14. Bob Carpenter. *Type-logical semantics*. MIT Press, Cambridge, MA, USA, 1998.
15. Alonzo Church. A Formulation of the Simple Theory of Types. *J. of Symbolic Logic*, 5:56–68, 1940.
16. Daniel Gallin. *Intensional and Higher-Order Modal Logic*, volume 19 of *North-Holland Mathematics Studies*. North-Holland, Amsterdam, 1975.
17. L. T. F. Gamut. *Logic, Language, and Meaning. Volume II. Intensional Logic and Logical Grammar*, volume 2. The University of Chicago Press, 1991.
18. Deepak Garg and Martín Abadi. A modal deconstruction of access control logics. In R. M. Amadio, editor, *FoSSaCS*, volume 4962 of *LNCS*, pages 216–230. Springer, 2008.
19. Kurt Gödel. Eine interpretation des intuitionistischen aussagenkalküls. *Ergebnisse eines Mathematischen Kolloquiums*, 8:39–40, 1933.
20. Moritz Hardt and Gert Smolka. Higher-order syntax and saturation algorithms for hybrid logic. *Electr. Notes Theor. Comput. Sci.*, 174(6):15–27, 2007.
21. John Harrison. *HOL Light Tutorial (for version 2.20)*. Intel JF1-13, September 2006. `http://www.cl.cam.ac.uk/~jrh13/hol-light/tutorial\_220.pdf`.
22. Leon Henkin. Completeness in the theory of types. *J. of Symbolic Logic*, 15:81–91, 1950.
23. Mark Kaminski and Gert Smolka. Terminating tableaux for hybrid logic with the difference modality and converse. In *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *LNCS*, pages 210–225. Springer, 2008.
24. Stephan Merz. Yet another encoding of TLA in isabelle. Available on the Internet: `http://www.loria.fr/~merz/projects/isabelle-tla/doc/design.ps.gz`, 1999.
25. Stephan Schulz. E – A Brainiac Theorem Prover. *Journal of AI Communications*, 15(2/3):111–126, 2002.