Toward Ethical Robots via Mechanized Deontic Logic*

Konstantine Arkoudas and **Selmer Bringsjord** and **Paul Bello**
Rensselaer AI & Reasoning (RAIR) Lab
Department of Cognitive Science
Department of Computer Science
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
{arkouk,selmer,bellop}@rpi.edu

Abstract

We suggest that mechanized multi-agent deontic logics might be appropriate vehicles for engineering trustworthy robots. Mechanically checked proofs in such logics can serve to establish the permissibility (or obligatoriness) of agent actions, and such proofs, when translated into English, can also explain the rationale behind those actions. We use the logical framework Athena to encode a natural deduction system for a deontic logic recently proposed by Horty for reasoning about what agents ought to do. We present the syntax and semantics of the logic, discuss its encoding in Athena, and illustrate with an example of a mechanized proof.

Introduction

As machines assume an increasingly prominent role in our lives, there is little doubt that they will eventually be called upon to make important, ethically charged decisions. How can we trust that such decisions will be made on sound ethical principles? Some have claimed that such trust is impossible and that, inevitably, AI will produce robots that both have tremendous power and behave immorally (Joy 2000). These predictions certainly have some traction, particularly among a public that seems bent on paying good money to see films depicting such dark futures. But our outlook is a good deal more optimistic. We see no reason why the future, at least in principle, can't be engineered to preclude doomsday scenarios of malicious robots taking over the world.

One approach to the task of building well-behaved robots emphasizes careful ethical reasoning based on mechanized formal logics of action, obligation, and permissibility; that is the approach we explore in this paper. It is a line of research in the spirit of Leibniz's famous dream of a universal moral calculus (Leibniz 1984):

> When controversies arise, there will be no more need for a disputation between two philosophers than there would be between two accountants [computistas]. It would be enough for them to pick up their pens and sit at their abacuses, and say to each other (perhaps having summoned a mutual friend): 'Let us calculate.'

In the future we envisage, Leibniz's "calculation" would boil down to formal proof and/or model generation in rigorously defined, machine-implemented logics of action and obligation.

Such logics would allow for *proofs* establishing that:

1. Robots only take permissible actions; and
2. all actions that are obligatory for robots are actually performed by them (subject to ties and conflicts among available actions).

Moreover, such proofs would be highly reliable (i.e., have a very small "trusted base"), and explained in ordinary English.

Clearly, this remains largely a vision. There are many thorny issues, not least among which are criticisms regarding the practical relevance of such formal logics, efficiency issues in their mechanization, etc.; we will discuss some of these points shortly. Nevertheless, mechanized ethical reasoning remains an intriguing vision worth investigating.

Of course one could also object to the wisdom of logic-based AI in general. While other ways of pursuing AI may well be preferable in certain contexts, we believe that in this case a logic-based approach (Bringsjord & Ferrucci 1998a; 1998b; Genesereth & Nilsson 1987; Nilsson 1991; Bringsjord, Arkoudas, & Schimanski forthcoming) is promising because one of the central issues here is that of trust—and mechanized formal proofs are perhaps the single most effective tool at our disposal for establishing trust.

Deontic logic, agency, and action

In standard deontic logic (Chellas 1980; Hilpinen 2001; Aqvist 1984), or just SDL, the formula $\bigcirc P$ can be interpreted as saying that *it ought to be the case that P*, where P denotes some state of affairs or proposition. Notice that there is no agent in the picture, nor are there actions that an agent might perform. This is a direct consequence of the fact that SDL is derived directly from standard modal logic, which applies the possibility and necessity operators $\Diamond$ and $\Box$ to formulae standing for propositions or states of affairs. For example, the deontic logic D^* has one rule of inference, viz.,

$$\frac{P \rightarrow Q}{\bigcirc P \rightarrow \bigcirc Q}$$

*We gratefully acknowledge that this research was in part supported by Air Force Research Labs (AFRL), Rome.
Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

and three axiom schemas:

- $(\bigcirc P \wedge \bigcirc Q) \rightarrow \bigcirc(P \wedge Q)$
- $\bigcirc \top$ ($\approx$ "That which must be is obligatory.")
- $\neg \bigcirc \bot$ ($\approx$ "Nothing impossible is obligatory.")

While D^* has some desirable properties, it and its relatives are plagued by various paradoxes (Hilpinen 2001), and, more importantly given present purposes, these logics aren't targeted at formalizing the concept of *actions* being obligatory (or permissible or forbidden) for an *agent*. Interestingly, deontic logics that have agents and their actions in mind do go back to the very dawn of this subfield of logic (von Wright 1951), but only recently has an "AI-friendly" semantics been proposed (Belnap, Perloff, & Xu 2001; Horty 2001) and corresponding axiomatizations been investigated (Murakami 2004).

We have used the Athena logical framework (briefly discussed in the next section) to encode a natural deduction calculus for a modern logic of agent action and obligation developed by Horty and axiomatized by Murakami in order to investigate mechanical deontic reasoning.

The ideal conditions for building "ethical robots" via a logic-based approach to AI would be as follows: we would have an expressive deontic logic $\mathcal{L}$ of high practical relevance, and an efficient algorithm for determining theoremhood in $\mathcal{L}$. That algorithm could then be built into the robot (perhaps implemented directly on its hardware), and the robot would only take an ethically charged action if it could formally prove that the action is permissible. Unfortunately, there is a legendarily strong tension between expressiveness and efficiency, and so it is certain that these ideal conditions will never obtain. For expressiveness, we will likely need highly hybrid modal and deontic logics that are at least first-order, which means that theoremhood in such logics will be undecidable. Even for decidable logics, such as the zero-order version of Horty's system [1] that we present in this paper, decision procedures are likely to be of inordinate computational complexity.

Therefore, we must reconcile ourselves to the possibility that a robot might not be able by itself to pass judgement on certain actions that it is contemplating; and that instead of a single monolithic decision procedure for deontic theoremhood (or validity, assuming the logic is complete), the robot might instead need to be armed with a knowledge base of lemmas and a panoply of *tactics*, each capable of settling certain restricted subclasses of deontic questions. This might well turn out to be sufficient in practice. If and when a complicated proposition arises that ethically paralyzes the robot, humans could intervene to settle the situation as they see fit. The logical framework that we have used to mechanize Horty's logic, Athena (Arkoudas 2003), facilitates the formulation of lemmas and of highly reliable tactics.

Athena

Athena (Arkoudas 2003) is a new interactive theorem proving system for polymorphic multi-sorted first-order logic that incorporates facilities for model generation, automated theorem proving, and structured proof representation and checking. It also provides a higher-order functional programming language, and a proof abstraction mechanism for expressing arbitrarily complicated inference *methods* in a way that guarantees soundness, akin to the tactics and tacticals of LCF-style systems such as HOL (Gordon & Melham 1993) and Isabelle (Paulson 1994). Proof automation is achieved in two ways: first, through user-formulated proof methods; and second, through the seamless integration of state-of-the-art ATPs such as Vampire (Voronkov 1995) and Spass (Weidenbach 2001) as primitive black boxes for general reasoning. For model generation, Athena integrates Paradox (Claessen & Sorensson 2003), a new highly efficient model finder. For proof representation and checking, Athena uses a block-structured Fitch-style natural deduction calculus (Pelletier 1999) with novel syntactic constructs and a formal semantics based on the abstraction of *assumption bases* (Arkoudas 2000). Most interestingly, a block-structured natural deduction format is used not only for writing proofs, but also for writing tactics (methods). This is a novel feature of Athena. Tactics in this style are considerably easier to write and remarkably useful in making proofs more modular and abstract.

Athena has been used to implement a proof-emitting optimizing compiler (Rinard & Marinov 1999); to integrate model checking and theorem proving for relational reasoning (Arkoudas *et al.* 2003); to implement various "certifying" algorithms (Arkoudas & Rinard 2004); to verify the core operations of a Unix-like file system (Arkoudas *et al.* 2004); to prove the correctness of dataflow analyses (Hao 2002); and to reason about generic software (Musser 2004). A concise presentation of Athena's syntax and semantics can be found elsewhere (Arvizo 2002).

Horty's logic and its Athena encoding

Murakami (Murakami 2004) presents an axiomatization of Horty's utilitarian formulation of multi-agent deontic logic (Horty 2001), and shows it decidable by proving that it has the finite model property. In this section we develop an alternative, sequent-based natural-deduction formulation of Murakami's system. The logic is encoded in Athena, which is then used as a metalanguage in order to reason about the encoded object language; we have used this methodology successfully with other intensional logics (Arkoudas & Bringsjord 2005). In what follows we briefly review the abstract syntax and semantics of the logic, and then present our formulation of a natural deduction system for it.

We use the letters P, Q, $R, \ldots$, to designate arbitrary *propositions*, built according to the following abstract grammar:

$$\begin{array}{rcl} P & ::= & A \mid \top \mid \bot \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \\ & \mid & \Box P \mid \Diamond P \mid [\alpha \textit{ cstit: } P] \mid \odot [\alpha \textit{ cstit: } P] \end{array}$$

where A and α range over a countable set of atomic propositions ("atoms") and a primitive domain of *agents*, respectively. Propositions of the form $[\alpha \textit{ cstit: } P]$ and $\odot [\alpha \textit{ cstit: } P]$ are read as "α sees to it that P" and "α

[1] Proved complete and decidable by Murakami (Murakami 2004).

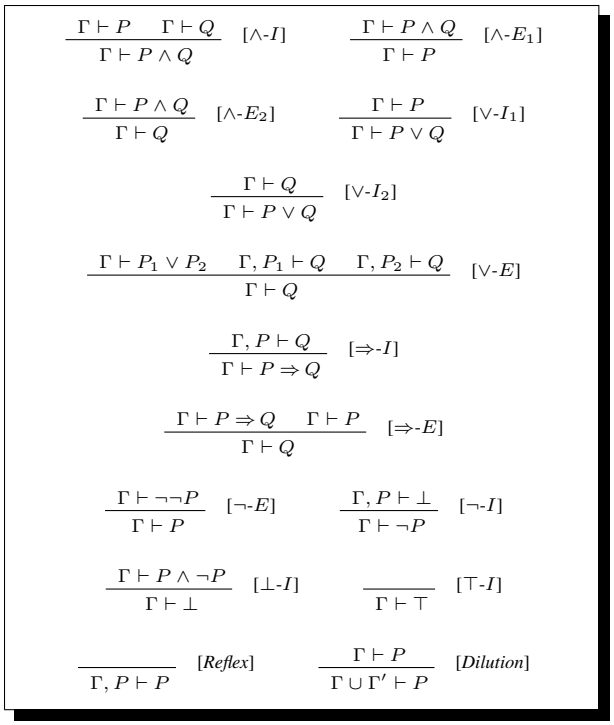

Figure 1: Inference rules for the propositional connectives.

ought to see to it that P," respectively. [2] We stress that $\odot\,[\alpha$ *cstit:* $P]$ is *not* read as "It ought to be the case that α sees to it that P." That is the classic Meinong-Chisholm "ought-to-be" analysis of agency, captured by another formula altogether, $\bigcirc\,[\alpha$ *cstit:* $P]$, where $\bigcirc$ is the non-agent-oriented "ought" operator similar to what is found in SDL. In Horty's semantics, $\bigcirc\,[\alpha$ *cstit:* $P]$ and $\odot\,[\alpha$ *cstit:* $P]$ are not equivalent statements; neither implies the other (Horty 2001). In general, the operator $\bigcirc$, taken over from SDL, applies to P just in case P holds in each of the best worlds. As Horty explains, an analogue to this basic idea is expressed by $\odot\,[\alpha$ *cstit:* $P]$, because this locution holds whenever P is ensured by each of the agent's best actions. (We have little use for the standard obligation operator $\bigcirc$ and hence we omit it from our formulation, although it could be easily included.)

The formal semantics are given on the basis of the theory of indeterminate branching time (Prior 1967; Thomason 1984), augmented with constructs for dealing with agent actions. The usual Kripke frames of modal logic are replaced by *deontic stit frames*. A deontic stit frame has the following components:

- A set of *moments* M, along with a stict partial order $<$ on M (i.e., $<$ is irreflexive and transitive, and hence asymmetric as well). A maximal linearly ordered subset of M is called a *history*. The set of all histories containing a moment $m \in M$ is written as H_m.
- A set A of *agents*.
- A binary function *Choice* that maps any given agent α and moment m into a partition *Choice*(α, m) of H_m. This function must satisfy two constraints: *independence of agents*, and *no choice between undivided histories*; see (Horty 2001) for details.
- For each $m \in M$, a utility function V_m from H_m into some partially ordered set of values (typically the real numbers).

The semantics are given with respect to moment/history pairs. Specifically, a *deontic stit model* is a deontic stit frame along with a truth valuation that maps each pair $\langle m, h\rangle$ with $m \in M, h \in H_m$ into a subset of atomic propositions (intuitively, these are the atoms that are true at the index $\langle m, h\rangle$). Given a deontic stit model $\mathcal{M}$ and a moment/history pair $\langle m, h\rangle$ (with $h \in H_m$), we write $\mathcal{M} \models_{\langle m, h\rangle} P$ to mean that $\mathcal{M}$ satisfies proposition P at index $\langle m, h\rangle$. The definition of $\mathcal{M} \models_{\langle m, h\rangle} P$ is given by induction on the structure of P. The cases of atoms and propositional combinations are standard. Cstit propositions are handled as follows: $\mathcal{M} \models_{\langle m, h\rangle} [\alpha$ *cstit:* $P]$ iff

$$\mathcal{M} \models_{\langle m, h'\rangle} P$$

for every $h' \in$ *block*(m, α, h), where *block*(m, α, h) is the unique block (equivalence class) containing h in the partition *Choice*(α, m) of H_m. We refer the reader to (Horty 2001) for the semantics of $\odot\,[\alpha$ *cstit:* $P]$.

A *sequent* $\Gamma \vdash P$ consists of a context Γ (a finite set of propositions) and a proposition P. Intuitively, this states that P follows from Γ. We write Γ, P (or P, Γ) as an abbreviation for $\Gamma \cup \{P\}$. The sequent calculus that we use consists of a collection of inference rules for deriving judgments of the form $\Gamma \vdash P$. Figure 1 shows the inference rules that deal with the standard propositional connectives. These are the usual introduction and elimination rules for each connective, in addition to reflexivity and dilution (weakening). Further, we have thirteen rules pertaining to the modal and deontic operators, shown in Figure 2. $[R_1]$, $[R_4]$ and $[R_6]$ are sequent formulations of Kripke's "K" axiom for the operators $\Box$, *cstit*, and $\odot$, respectively. $[R_2]$ and $[R_7]$ are the usual "T" axioms of modal logic for $\Box$ and *cstit*. $[R_3]$ is the "axiom 5" for $\Box$. $[R_8]$ and $[R_9]$ express that necessary truths are ensured (if P is necessary then every agent sees to it) and obligatory. $[R_{10}]$ asserts that obligations are possible. $[R_{12}]$ is a necessitation rule ensuring that all tautologies (propositions derivable from the empty context) are necessary. (A similar necessitation rule for *cstit* can be derived from $[R_8]$ in tandem with $[R_{12}]$, so we do not need to take it as primitive.) $[R_{13}]$ says that if α seeing to P strictly implies α seeing to Q, then if α ought to stit P then α also ought to stit Q. Finally, $[R_5]$ is a slightly disguised formulation of the standard "axiom 5" for *cstit*. It is provably equivalent to

$$\neg[\alpha \textit{ cstit: } \neg P] \Rightarrow [\alpha \textit{ cstit: } \neg[\alpha \textit{ cstit: } \neg P]]$$

[2] The 'c' in *cstit* stands for "Chellas." Horty (Horty 2001) attributes the naming to the fact that *cstit* is analogous—though not identical—to an operator introduced by Brian Chellas in his 1969 doctoral dissertation (Chellas 1969). There are other stit operators in the literature, e.g., the achievement stit ("astit"), the deliberative stit ("dstit"), etc.

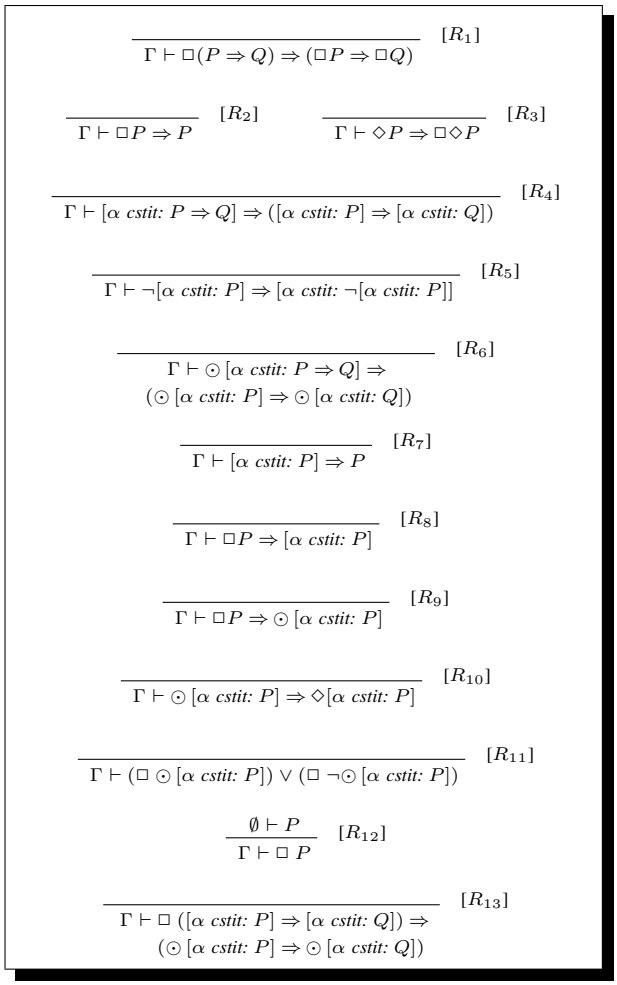

Figure 2: Inference rules for the deontic operators.

which is of the exact same form as the "axiom 5":

$$\Diamond P \Rightarrow \Box \Diamond P$$

once we realize that $\Diamond P$ stands for $\neg\Box \neg P$.

Our Athena formalization introduces a domain of agents and a datatype that captures the abstract syntax of the propositions of the logic:

```
(datatype Prop
   False
   True
  (Atom Boolean)
  (If Prop Prop)
  (Not Prop)
  (And Prop Prop)
  (Or Prop Prop)
  (Stit Agent Prop)
  (OughtToStit Agent Prop)
  (Nec Prop)
  (Pos Prop))
```

We proceed to introduce a binary relation `sequent` that may obtain between a finite set of propositions and a single proposition:

```
(declare sequent
   (-> ((FSet-Of Prop) Prop)
       Boolean))
```

Here `FSet-Of` is a unary sort constructor: for any sort `T`, `(FSet-Of T)` is a new sort representing the set of all finite sets of elements of `T`. Finite sets are built with two polymorphic constructors: the constant `null`, representing the empty set; and the binary constructor `insert`, which takes an element x of sort `T` and a finite set S (of sort `(FSet-Of T)`) and returns the set $\{x\} \cup S$. We also have all the usual set-theoretic operations available (`union`, `intersection`, etc.).

The intended interpretation is that if `(sequent` S P`)` holds for a set of propositions S and a proposition P, then the sequent $S \vdash P$ is derivable in the logic via the above rules. Accordingly, we introduce axioms capturing those rules. For instance, the conjunction introduction rule and rule $[R_{10}]$ are represented, respectively, by the following two axioms:

```
(define And-I
  (forall ?Gamma ?P ?Q
    (if (and (sequent ?Gamma ?P)
             (sequent ?Gamma ?Q))
        (sequent ?Gamma (And ?P ?Q)))))

(define R10
  (forall ?Gamma ?a ?P
    (sequent ?Gamma
       (If (OughtToStit ?a ?P)
           (Pos (Stit ?a ?P))))))
```

Note the object/meta-level distinction between, e.g., `and` and `And`. The former is a native Athena propositional constructor, i.e., part of the metalogic, whereas the latter is a propositional constructor of the encoded object logic.

As we have argued elsewhere (Arkoudas & Bringsjord 2005), such a direct proof-theoretic encoding of a modal logic in a first-order logical framework such as Athena carries several advantages:

- The proofs are in natural deduction format and hence easier to read, write, and translate into English.
- Theorem proving is facilitated because we are able to leverage state-of-the-art automated theorem provers (ATPs) such as Vampire (Voronkov 1995) and Spass (Weidenbach 2001) that are integrated with Athena. Tactics can be programmed at a fairly high level of abstraction, with tedious details outsourced to the ATPs.
- Because we have explicitly encoded the abstract syntax of the logic, we are able to quantify over agents, propositions, and sequents. This provides us with the generalization benefits of higher-order logic, even though we are working in a first-order system.

```
pick-any P a
 begin
   S1 := (sequent
           null
           (If (Not (Stit a (Not (Not P))))
               (Stit
                 a
                 (Not (Stit
                        a (Not (Not P)))))))
         from R5;
    S2 := (sequent
            null
            (If (Not (Stit
                       a
                       (Not (Stit
                              a
                              (Not (Not P))))))
                (Not (Not (Stit
                            a
                            (Not (Not P)))))))
          from S1, contrapositive;
    S3 := prove (sequent
                  null
                  (If (Not
                        (Not (Stit
                               a
                               (Not (Not P)))))
                      (Stit a (Not (Not P))))));
    S4 := (sequent
            null
            (If (Not (Stit
                       a
                       (Not (Stit
                              a
                              (Not (Not P))))))
                (Stit a (Not (Not P)))))
          from S2, S3, transitivity;
    S5 := prove (sequent
                  null (Iff P (Not (Not P))));
    S6 := (sequent null
            (Iff
              (Not (Stit a (Not (Stit a P))))
              (Not (Stit
                     a
                     (Not (Stit
                            a
                            (Not (Not P))))))))
          from S5, lemma-1.7;
```

Figure 3: Athena proof of Lemma 1.8, part 1.

```
    S7 := (sequent
            null
            (If (Not (Stit a (Not (Stit a P))))
                (Not (Stit
                       a
                       (Not (Stit
                              a
                              (Not (Not P))))))))
          from S6, Iff-Elim-1;
    S8 := prove
            (sequent null (If (Not (Not P)) P));
    S9 := (sequent
            null
            (If (Stit a (Not (Not P)))
                (Stit a P)))
          from S8, lemma-1.6;
  (sequent null
            (If (Not (Stit a (Not (Stit a P))))
                (Stit a P)))
  from S4, S7, S9, lemma-1.5
 end
```

Figure 4: Athena proof of Lemma 1.8, part 2.

Example

As a simple but non-trivial example, we present the Athena proof of the following "iterated *cstit*" result:

$$[\alpha \textit{ cstit: } P] \Rightarrow [\alpha \textit{ cstit: } [\alpha \textit{ cstit: } P]] \qquad (1)$$

The proof is easily turned into a tactic that can be applied to any given agent and proposition.

A number of lemmas are used in the proof. Most of them express straightforward propositional logic tautologies and are proved automatically by outsourcing them to the ATPs that are integrated with Athena. For instance, the first four lemmas below respectively express the transitivity of logical implication, the contrapositive law, the cut, and disjunctive syllogism.

Lemma 1.1 *If* $\Gamma \vdash P \Rightarrow Q$ *and* $\Gamma \vdash Q \Rightarrow R$ *then* $\Gamma \vdash P \Rightarrow R$.

Lemma 1.2 *If* $\Gamma \vdash P \Rightarrow Q$ *then* $\Gamma \vdash \neg Q \Rightarrow \neg P$.

Lemma 1.3 *If* $\Gamma_1 \vdash P$ *and* $\Gamma_2, P \vdash Q$ *then* $\Gamma_1 \cup \Gamma_2 \vdash Q$.

Lemma 1.4 $\Gamma \vdash (P_1 \vee P_2) \Rightarrow (\neg P_2 \Rightarrow P1)$.

Lemma 1.5 *If* $\Gamma \vdash P' \Rightarrow Q'$, $\Gamma \vdash P \Rightarrow P'$, *and* $\Gamma \vdash Q' \Rightarrow Q$ *then* $\Gamma \vdash P \Rightarrow Q$.

A few properly deontic lemmas are also necessary:

Lemma 1.6 *For all agents* α *and propositions* P *and* Q, *if* $\emptyset \vdash P \Rightarrow Q$ *then* $\emptyset \vdash [\alpha \textit{ cstit: } P] \Rightarrow [\alpha \textit{ cstit: } Q]$.

Lemma 1.7 *For all agents* α *and propositions* P *and* Q, *if* $\emptyset \vdash P \Leftrightarrow Q$ *then* $\emptyset \vdash \neg[\alpha \textit{ cstit: } P] \Leftrightarrow \neg[\alpha \textit{ cstit: } Q]$.

Lemma 1.8 $\emptyset \vdash \neg[\alpha \textit{ cstit: } \neg[\alpha \textit{ cstit: } P]] \Rightarrow [\alpha \textit{ cstit: } P]$ *for all* α *and* P.

Lemma 1.9 $\emptyset \vdash P \Rightarrow \neg[\alpha \textit{ cstit: } \neg P]$.

Lemma 1.6, Lemma 1.7, and Lemma 1.9 are proved automatically. Lemma 1.8 is more challenging and requires user guidance. Its proof, in Athena's natural deduction system, is shown in two parts in Figure 3 and in Figure 4. Very brief explanations of the pertinent Athena constructs are given below to help the reader follow the code. For a more thorough treatment we refer the reader to the Athena Web site.

An Athena deduction D is always evaluated in a given *assumption base*—a finite set of propositions that are assumed to hold for the purposes of D. An assumption base thus represents our "axiom set" or "knowledge base." Athena starts

out with the empty assumption base, which then gets incrementally augmented with the conclusions of the deductions that the user successfully evaluates. Propositions can also be explicitly added into the global assumption base with the top-level directive `assert`.

Evaluating a deduction D in an assumption base β will either produce a proposition F (the "conclusion" of D in β), or else it will generate an error or will diverge. If D does produce a conclusion F, Athena's semantics guarantee $\beta \models F$, i.e., that F is a logical consequence of β. There are several syntactic forms that can be used for deductions; they form a complete proof system for polymorphic multi-sorted first-order logic.

The form `pick-any` introduces universal generalizations: `pick-any` $I_1 \cdots I_n$ `begin` D `end` (for arbitrary subdeduction D) binds the names $I_1 \cdots I_n$ to fresh variables $v_1, \ldots, v_n$ and evaluates D. If and when D yields a conclusion F, the result returned by the entire `pick-any` is $\forall\, v_1, \ldots, v_n\, .\, F$.

The body of the proof is a semicolon-separated sequence of steps of the form

$$I_1 := D_1 ; \cdots ; I_n := D_n$$

where I_j is a name (identifier) and D_j an arbitrary subproof. The sequence is evaluated by recursively evaluating each D_j in turn, $j = 1, 2, \ldots$, obtaining a conclusion F_j, binding the name I_j to F_j, inserting F_j in the assumption base, and then proceeding with the next step, $I_{j+1} := D_{j+1}$. The conclusion of the entire sequence is the conclusion of the last step, D_n. Note that the last step is not named.

A common proof step is of the form F `from` $F_1, \ldots, F_k$. This instructs Athena to try to automatically derive the conclusion F from the given premises $F_1, \ldots, F_k$ (all k of which must be in the assumption base). After performing some internal translations, Athena outsources this step to an ATP. If the ATP manages to solve the problem within a certain time limit (currently preset to a maximum of 60 seconds), then F is returned as the result of the step; otherwise an error message appears.

A similar step is of the form `prove` F. This attempts to automatically derive F from all the elements of the current assumption base. This is therefore equivalent to F `from` $F_1, \ldots, F_k$, where $F_1, \ldots, F_k$ are all and only the members of the current assumption base.

With the above lemmas at hand, the original goal can be proved as shown in Figure 5.

```
pick-any P a
 begin
  S1 := (sequent null
                 (If (Stit a P)
                     (Not (Stit a (Not (Stit a P))))))
        from lemma-1.9;
  S2 := (sequent null
                 (If (Not (Stit a (Not (Stit a P))))
                     (Stit a (Not
                               (Stit a (Not
                                        (Stit a P))))))
        from R5;
  S3 := (sequent
          null
          (If (Stit a P)
              (Stit a (Not
                        (Stit a (Not (Stit a P)))))))
          from S1, S2, transitivity;
  S4 :=  (sequent null
                  (If (Not (Stit a (Not (Stit a P))))
                      (Stit a P)))
         from lemma-1.8;
  S5 :=  (sequent
           null
           (If (Stit a (Not (Stit a (Not (Stit a P)))))
               (Stit a (Stit a P))))
         from S4, lemma-1.6;
 (sequent null (If (Stit a P)
                   (Stit a (Stit a P))))
 from S3, S5, transitivity
end
```

Figure 5: Athena proof of (1).

Conclusions

We have reported ongoing work on the mechanization of multi-agent logics of action and obligation. It is reasonable to believe that such logics might prove useful in engineering machines that can reason about what they ought to do. We presented an Athena implementation of a natural deduction calculus for a recently developed deontic logic of agency based on indeterminate branching-time semantics augmented with dominance utilitarianism, and presented an example of a mechanized proof in that system. We are currently using mechanized deontic logics to represent wargaming scenarios and to implement wargame agents capable of reasoning about their own ethical codes as well as those of their adversaries. In that direction, we plan to investigate the mechanization of defeasible deontic logics that allow for explicit modeling of contrary-to-duty obligations and violations (Van Der Torre 1997).

References

Aqvist, E. 1984. Deontic logic. In Gabbay, D., and Guenthner, F., eds., *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*. Dordrecht, The Netherlands: D. Reidel. 605–714.

Arkoudas, K., and Bringsjord, S. 2005. Metareasoning for multi-agent epistemic logics. In *Fifth International Conference on Computational Logic In Multi-Agent Systems (CLIMA 2004)*, volume 3487 of *Lecture Notes in Artificial Intelligence (LNAI)*. New York: Springer-Verlag. 111–125.

Arkoudas, K., and Rinard, M. 2004. Deductive runtime certification. In *Proceedings of the 2004 Workshop on Runtime Verification*, 39–56.

Arkoudas, K.; Khurshid, S.; Marinov, D.; and Rinard, M. 2003. Integrating model checking and theorem proving for relational reasoning. In *Seventh International Seminar on Relational Methods in Computer Science (RelMiCS 2003)*, volume 3015 of *Lecture Notes in Computer Science (LNCS)*, 21–33.

Arkoudas, K.; Zee, K.; Kuncak, V.; and Rinard, M. 2004. Verifying a file system implementation. In *Sixth International Conference on Formal Engineering Methods (ICFEM'04)*, volume 3308 of *Lecture Notes in Computer Science (LNCS)*, 373–390.

Arkoudas, K. 2000. *Denotational Proof Languages*. Ph.D. Dissertation, MIT, Department of Computer Science, Cambridge, USA.

Arkoudas, K. 2003. Athena. `http://www.pac.csail.mit.edu/athena`.

Arvizo, T. 2002. A virtual machine for a type-ω denotational proof language. Masters thesis, MIT, June 2002.

Belnap, N.; Perloff, M.; and Xu, M. 2001. *Facing the future*. Oxford University Press.

Bringsjord, S., and Ferrucci, D. 1998a. Logic and artificial intelligence: Divorced, still married, separated...? *Minds and Machines* 8:273–308.

Bringsjord, S., and Ferrucci, D. 1998b. Reply to Thayse and Glymour on logic and artificial intelligence. *Minds and Machines* 8:313–315.

Bringsjord, S.; Arkoudas, K.; and Schimanski, B. forthcoming. Logic-based AI for the new millennium. *AI Magazine*.

Chellas, B. 1969. The logical form of imperatives. PhD dissertation, Stanford Philosophy Department.

Chellas, B. F. 1980. *Modal Logic: an Introduction*. Cambridge University Press.

Claessen, K., and Sorensson, N. 2003. New techniques that improve Mace-style finite model building. In *Model Computation—principles, algorithms, applications*.

Genesereth, M., and Nilsson, N. 1987. *Logical Foundations of Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann.

Gordon, M. J. C., and Melham, T. F. 1993. *Introduction to HOL, a theorem proving environment for higher-order logic*. Cambridge, England: Cambridge University Press.

Hao, M. 2002. Using a denotational proof language to verify dataflow analyses. Masters thesis, MIT, September 2002.

Hilpinen, R. 2001. Deontic Logic. In Goble, L., ed., *Philosophical Logic*. Oxford, UK: Blackwell. 159–182.

Horty, J. 2001. *Agency and Deontic Logic*. Oxford University Press.

Joy, W. 2000. Why the Future Doesn't Need Us. *Wired* 8(4).

Leibniz. 1984. *Notes on analysis*. Past Masters: Leibniz. Oxford University Press. Translated by George MacDonald Ross.

Murakami, Y. 2004. Utilitarian deontic logic. In *Proceedings of the Fifth International Conference on Advances in Modal Logic (AiML 2004)*, 288–302.

Musser, D. 2004. Generic Software Design. `www.cs.rpi.edu/~musser/gsd`.

Nilsson, N. 1991. Logic and Artificial Intelligence. *Artificial Intelligence* 47:31–56.

Paulson, L. 1994. *Isabelle, A Generic Theorem Prover*. Lecture Notes in Computer Science. Springer-Verlag.

Pelletier, F. J. 1999. A Brief History of Natural Deduction. *History and Philosophy of Logic* 20:1–31.

Prior, A. 1967. *Past, present, and future*. Oxford University Press.

Rinard, M., and Marinov, D. 1999. Credible compilation with pointers. In *Proceedings of the 1999 Workshop on Run-Time Result Verification*.

Thomason, R. 1984. Combinations of tense and modality. In Gabbay, D., and Guenthner, F., eds., *Extensions of classical logic*, volume 2 of *Handbook of Philosophical Logic*. Dordrecht, The Netherlands: D. Reidel.

Van Der Torre, L. 1997. *Reasoning about obligations*. Ph.D. Dissertation, Erasmus University, Rotterdam, Netherlands.

von Wright, G. 1951. Deontic logic. *Mind* 60:1–15.

Voronkov, A. 1995. The anatomy of Vampire: implementing bottom-up procedures with code trees. *Journal of Automated Reasoning* 15(2).

Weidenbach, C. 2001. Combining superposition, sorts, and splitting. In Robinson, A., and Voronkov, A., eds., *Handbook of Automated Reasoning*, volume 2. North-Holland.