

Uniform Modeling of Human and Artificial Agents Using Epistemic and Deontic Logic¹

Jamika Burge, Kevin Mosley, and Albert C. Esterline

Department of Computer Science/NASA ACE

North Carolina A&T State University

Greensboro, NC 27411

email: {jb988330, kmosley, esterlin}@ncat.edu

ABSTRACT

We present a framework that uses epistemic logic and deontic to model a multi-agent system consisting of humans and non-human agents. Epistemic logic addresses belief, knowledge, notably common knowledge, and, in many applications in computer science, it addresses knowledge acquired by communication. With deontic logic, we represent obligations, prohibitions, and permissions applying to agents. We are concerned with agents' actions, their attitudes, and what they know. Our framework addresses the ability of agents to maintain ongoing relationships with each other. This framework supports a formal methodology for analysis and design for multi-agent/multi-human collaboration.

KEYWORDS: multiagent systems, epistemic and deontic logic, human-computer interaction

INTRODUCTION

Epistemic logic, the logic of knowledge, and deontic logic, the study of normative concepts, are modal logics originating in philosophy. Epistemic logic has modal operators for various kinds of knowledge (or belief), and deontic logic has operators for obligation, permission, and prohibition. We specify how agents (human and non-human) should cooperate in a system. Our modeling of agents is uniform: human and non-human agents are modeled with the same formalisms. Such modeling entails abstraction: we use a formal methodology to formulate abstract specifications for a system of agents. An abstract approach lets us express logical formulas that let us specify the communication behavior of processes as required by their users. Continual communication, interaction, and responsiveness among the agents ensure that a system maintains its reactive nature; that is, the agents maintain ongoing relationships with each other.

We present a multi-agent/multi-person framework for constructing models to better understand the interaction between human and non-human agents. This framework supports formal methods by providing an abstract specification language, allowing us to

¹ This research was supported by grant NAG 2-1150, "Motion Planning in a Society of Intelligent Mobile Agents," from NASA Ames Research Center.

eliminate ambiguities from expected system behavior. The next section addresses the syntax and intuitive semantics of the framework. We then show the use of modal logics through examples, and we present some specifications using our framework.

SYNTAX AND INTUITIVE SEMANTICS

We refer to [5] in defining the syntactic categories of our language using a sorted first-order action logic. We can think of a sort as denoting a set whose elements are of the same algebraic ‘type’. Predicates and function symbols used in the system are also typed. Agents and actions are the most interesting sorts. Agents perform actions and are the recipients of actions performed. Since the most interesting actions involve communication, we model actions as speech acts. We introduce predicates to describe properties about and relations among entities. An n -ary predicate symbol applied to n terms of the appropriate sorts results in an atomic formula.

Sorted First-Order Action Logic (SFOAL)

We now present our sorted first-order action logic (SFOAL). We begin by introducing the syntactic categories, followed by the formation rules for our language.

Syntactic Categories

- A non-empty finite collection of *sorts* $S = \{Act, Agt, s_1, s_2, \dots\}$, where *Act* is a non-empty action sort, and *Agt* is a non-empty agent sort.
- *Constant symbols* denoting individuals: for each sort $s \in S$, there is a possibly empty set of constants, each said to be of sort s .
- *Agent names*: for all $n > 0$ and each n -tuple, $\langle s_1, \dots, s_n \rangle$ such that $s_1, \dots, s_n \in S$, there is a (possibly empty) set of n -place agent names of sort *Agt* of type $\langle s_1, \dots, s_n \rangle$.
- *Action names*: for all $n > 0$ and each n -tuple, $\langle s_1, \dots, s_n \rangle$ such that $s_1, \dots, s_n \in S$, there is a (possibly empty) set of n -place action names of sort *Act* of type $\langle s_1, \dots, s_n \rangle$.
- *Predicate symbol*: for all $n > 0$ and each n -tuple, $\langle s_1, \dots, s_n \rangle$ such that $s_1, \dots, s_n \in S$, there is a (possibly empty) set of n -place predicate symbols of type $\langle s_1, \dots, s_n \rangle$.
- *Variables*: A non-empty collection of distinct variables for each sort, $s \in S$.
- *Quantifiers*: for each sort $s \in S$, the universal quantifier, \forall_s , and the existential quantifier, \exists_s .
- *Logical operators*: ‘ \neg ’ (negation), ‘ \leftrightarrow ’ (equivalence [iff]), ‘ \rightarrow ’ (implication), ‘ \wedge ’ (conjunction), ‘ \vee ’ (disjunction), ‘ \Box ’ (strong dynamic logic modal operator), and the action combinators ‘;’ (sequential composition), ‘+’ (non-deterministic choice), and ‘ $\bar{}$ ’ (negation).
- *Punctuation*: (,), and ,.

Formation Rules

Terms

- For each sort $s \in S$, a variable or constant of sort s is a term (most basic).
- If t_1, \dots, t_n are terms of sorts s_1, \dots, s_n (resp.), and s_1, \dots, s_n are all taken from S , and α is an action symbol of type $\langle s_1, \dots, s_n \rangle$, then $\alpha(t_1, \dots, t_n)$ is a term of sort *Act*.
- If t_1, \dots, t_n are terms of sorts s_1, \dots, s_n (respectively), and s_1, \dots, s_n are all taken from S , and i is an agent symbol of type $\langle s_1, \dots, s_n \rangle$, then $i(t_1, \dots, t_n)$ is a term of sort *Agt*.

- Nothing else is a term.

Atoms

- If t_1, \dots, t_n are terms of sorts s_1, \dots, s_n , respectively, and s_1, \dots, s_n are all taken from S , and p is a predicate symbol of type $\langle s_1, \dots, s_n \rangle$, then $p(t_1, \dots, t_n)$ is an atom.
- For each sort $s \in S$, given two terms of s , t_i and t_{ii} , $t_i =_s t_{ii}$ is an atom, provided that $=_s$ is in the language.
- Nothing else is an atom.

Formulas

- An atom is a formula.
- If ϕ is a formula, so is $\neg\phi$.
- If ϕ and ψ are formulas, so are:
 $\phi \wedge \psi$, $\phi \rightarrow \psi$, $\phi \vee \psi$, and $\phi \leftrightarrow \psi$
- For each sort, $s \in S$, if x is a variable of sort s and ϕ is a formula, then $\forall_s x \phi$ and $\exists_s x \phi$ are also formulas.
- Nothing else is a formula.

The sentential connectives, punctuation, connectives, and quantifiers are standard. Our constants, predicates, and variable names are categorized above. The action combinators provide our framework with the ability to combine actions to make more complex actions. In dynamic logic, $[\alpha]\phi$ means that every possible execution of α leads to a situation in which formula ϕ is true [4]. Our action operators have the form $[i, \alpha]\phi$, so ϕ is true when agent i performs action α . For example, if agent i buys a Ford Explorer, then that explorer is sold: $[i, \text{BUY}(\text{explorer})] \text{SOLD}(\text{explorer})$.

Temporal Extensions of Sorted First-Order Action Logic (TSFOAL)

We extend SFOAL to include temporal logic operators, using a linear, discrete time. These operators are \Box (“always”), \Diamond (“eventually”), N (“next time”), and U (“until”) [3]. Intuitively, $\Box\phi$ is true if ϕ is true currently and at all subsequent time points; $\Diamond\phi$ is true if ϕ is true at some point now or in the future; $N\phi$ is true if ϕ is true at the next time point; and $\phi U \psi$ is true if ϕ is true until ψ is true (and ψ is eventually true).

To our existing formation rules, we add that, if ϕ and ψ are formulas, then so are $\Box\phi$, $\Diamond\phi$, $N\phi$, and $\phi U \psi$. We also add the operators \Box^p , \Diamond^p , N^p , and U^p for past time. The semantics for these temporal operators are mirror images of their future-time counterparts. We can express, for example, that agent i performed action α by $\Diamond^p(i, \alpha)$.

Epistemic Extensions of TSFOAL (ETSFOAL)

Epistemic logic (ETSFOAL) introduces a new category denoting finite groups of agents. We augment our language with the operator ‘ K_i ’ for each i of sort **Agnt**, and ‘ E_G ’ and ‘ C_G ’ for each group of agents, G . We also have additional formation rules. If i is of sort **Agnt** and ϕ is a formula, then $K_i\phi$ is a formula, and, if G is a group name, then $E_G\phi$ and $C_G\phi$ are formulas. $K_i\phi$ means that agent i knows that ϕ , $E_G\phi$ that every agent in group G knows that ϕ , and $C_G\phi$ that it is common knowledge (see below) in G that ϕ .

We define the modal operator $E_G\phi$ (‘everyone in the group G knows ϕ ’) as

$$E_G \varphi \stackrel{\text{def}}{=} \bigwedge_{i \in G} K_i \varphi,$$

(The “ \wedge ” operator is a meta-linguistic symbol for iterated conjunction.) Iterated applications of the E_G operator are expressed with the familiar superscript notation:

$$E_G^1 \varphi \stackrel{\text{def}}{=} E_G \varphi \text{ (base case),}$$

$$E_G^k \varphi \stackrel{\text{def}}{=} E E_G^{k-1} \varphi \text{ for } k > 1.$$

We establish Common knowledge by the *common-knowledge induction schema* [2]:

i and *i*² have common knowledge that φ iff

(a) *i* and *i*² know that φ and know that (a).

(Note that the sentence labeled (a) refers to itself.) This can be generalized in the obvious way for any finite number of group members. Thus, Group G has common knowledge that φ ($C_G \varphi$) if everyone (in G) knows φ , everyone knows that everyone knows φ , and so on, *ad infinitum*. We define the operator C_G as an infinite conjunction:

$$C_G \varphi = \bigwedge_{i \in \omega^+} E_G^i \varphi.$$

Mutual belief closely resembles common knowledge [2]. The important difference is that mutual belief, like belief, in general, lacks the knowledge axiom required for knowledge, namely, $K_i \varphi \rightarrow \varphi$. We use belief operators $B_i \varphi$ (‘agent i believes φ ’), $H_G \varphi$ (‘everyone in group G believes φ ’), and $M_G \varphi$ (‘it is mutually believed in G that φ ’).

The semantics of epistemic logic is generally developed with Kripke structures involving possible worlds and accessibility (possibility) relations among them defined for each agent [3]. Several standard axiomatic systems for epistemic logic have been developed; we assume the axioms and rules of inference of the S5 system [3].

Deontic Extensions of ETSFOAL (DETSFOAL)

Where i is of sort **Agnt** and α is of sort **Act**, $O(i, \alpha)$ states that agent i is obligated to perform action α . We use the same notation for permission, $P(i, \alpha)$, and prohibition, $F(i, \alpha)$. These operators are related by the valid equivalences $O\varphi \leftrightarrow \neg P\neg\varphi$ and $P\varphi \leftrightarrow \neg F\varphi$. We distinguish between conditional obligations, permissions, and prohibitions (dyadic deontic logic) and unconditional ones (monadic deontic logic) [1]. $O(i, \alpha)/\psi$, for example, says that agent i is obligated to perform action α provided ψ holds. The syntax allows nested deontic operators.

The semantics of deontic logic is also generally presented in terms of Kripke structures, and there are several standard axiomatic systems. Our framework is quite standard in these respects. A significant difference between the knowledge operators and the obligation operator is that $O\varphi \rightarrow \varphi$ is not valid. Thus, deontic logic distinguishes between what is actual (e.g., φ) and what is ideal (e.g., $O\varphi$). This is perhaps the most significant feature of deontic logic and makes it appropriate, for example, for specifying fault-tolerant software systems, where non-ideal actual states must be recognized.

AN EXAMPLE

We present an example of the modeling and framework discussed. We shall model aspects of an imaginary car dealership, denoted by a constant, Als_Ford , of sort **D**, the

sort of dealerships. At the top of our chain of command, the supervisor (of sort *Agt* and type $\langle D \rangle$) of Al's Ford (denoted *Als_Ford*) is denoted by *Supervisor(Als_Ford)*. Our two car dealers are in the next level. We denote them with *Representative(Als_Ford, 0)* and *Representative(Als_Ford, 1)* of sort *Agt* and type $\langle Dealership, \mathbb{N} \rangle$. Next, the *Liaison* (or proxy) agent is the link between the customer and the representative(s). This agent is of sort *Agt* and of type $\langle Dealership, \mathbb{N} \rangle$, and is denoted by *Liaison(Als_Ford, 0)*. He forwards information from the customer to the dealer. Customers are denoted by names of the form *Customer(Als_Ford, n)*, of sort *Agt* and type $\langle Dealership, \mathbb{N} \rangle$. Any of these agents could be human or non-human, but we might expect the supervisor to be human.

The agents communicate with one another by speech acts, often about cars. So we introduce another sort, *Car*. Some action names are as follows.

BUY of type $\langle Car, Agt \rangle$;

NOTIFY of type $\langle Agt, Agt, Act \rangle$;

MAKE_APPOINTMENT of type $\langle Agt, Date \rangle$;

ORDER of type $\langle Make \rangle$.

Agents' actions (whether performed jointly or independently) are enabled by their common knowledge. For example, suppose that the supervisor posts a memo stating that ordering new cars is the responsibility of *Representative(Als_Ford, 1)*. Then, provided that it is common knowledge in the dealership group, *G* (which includes the supervisor, liaison, and representative agents), that job assignments are posted in this way, it becomes common knowledge in group *G* that *Representative(Als_Ford, 1)* should order the new inventory. Expressing this common knowledge requires nesting modal operators, as in the following case specialized to *Tempo* (of sort *Make*):

$$C_G(O(\text{Representative}(\text{Als_Ford}, 1), \text{ORDER}(\text{tempo})),$$

Our framework presents certain *general* axioms in addition to the purely logical axioms and rules (such as those of the S5 system for epistemic logic) that we use. The general axioms constrain the meanings of certain ubiquitous action names (such as *NOTIFY*), interrelate the operators from the different modal and temporal logics, and express general principles of common knowledge about actions. For each domain modeled, we formulate certain *proper* axioms that capture the rules governing the agents in that domain. Some of these rules are illustrated below.

Suppose that *Representative(Als_Ford, 1)* has ordered the new inventory and has notified the supervisor of the order. (We have instances for this domain of a general axiom governing the *NOTIFY* action.) Upon knowing that the new inventory is on its way, the supervisor can begin to make room for the new cars in the showroom, or the liaison could call a customer who asked when the new inventory would arrive. In general, when an agent is notified of something, that agent acquires new knowledge and can perform actions accordingly. So, when *Representative(Als_Ford, 1)* notifies *Supervisor(Als_Ford)* that he has ordered the new inventory, the supervisor has the knowledge that the representative agent ordered the inventory:

$$\begin{aligned} &[(\text{Representative}(\text{Als_Ford}, 1), \text{NOTIFY}(\text{Supervisor}(\text{Als_Ford}), \\ &\quad ((\text{Representative}(\text{Als_Ford}, 1), \text{ORDER}(\text{tempo}))) \\ &\quad K_{\text{Supervisor}(\text{Als_Ford})}(\text{Representative}(\text{Als_Ford}, 1), \text{ORDER}(\text{tempo})). \end{aligned}$$

Performance of some actions requires that some condition hold prior to the action being performed. For example, *Representative(Als_Ford, 1)* can order the new inventory

only if there is ample space for the new cars. Where the predicate *Lot_Space* means there is ample space on the lot for the new cars, we write

$$O(\text{Representative}(\text{Als_Ford}, 1), \text{ORDER}(m) / \text{Lot_Space}(\text{Ford_Lot})),$$

where *Ford_Lot* denotes car lot and is of sort **Lot**. It may be important that an agent has known that he had a certain obligation. To state that *Representative(Als_Ford, 1)* has known that he was supposed to order the new inventory, we write

$$\diamond^p(K_{\text{Representative}(\text{Als_Ford}, 1)} O(\text{Representative}(\text{Als_Ford}, 1), \text{ORDER}(\text{tempo}))).$$

Finally, since deontic logic allows us to model the actual versus the ideal, our framework allows us to capture fault-tolerance. We use a proper axiom to illustrate. If some incident prevents (*Representative(Als_Ford, 1)*) from ordering the new inventory (e.g., he forgets), the responsibility then may become that of another agent, say (*Representative(Als_Ford, 0)*). We express this rule as follows:

$$\begin{aligned} & (O(\text{Representative}(\text{Als_Ford}, 1), \text{ORDER}(\text{tempo})) \wedge \\ & (\text{Representative}(\text{Als_Ford}, 1), \overline{\text{ORDER}(\text{tempo})})) \rightarrow \\ & O(\text{Representative}(\text{Als_Ford}, 0), \text{ORDER}(\text{tempo})). \end{aligned}$$

CONCLUSION

Our framework relates artificial and human agents to formalisms involving concepts from epistemic logic and deontic logic, which capture critical aspects of how agents interact. Thus, content is given to the formalisms specifying the agents, and a formal analysis and design methodology can be based on this framework. Proof techniques can be used to show that a formal specification is internally consistent and complete with respect to the requirements, and a formal design can be proved correct relative to a formal specification. Our framework can address the needs of the human user and thus should help in designing systems that aid human users in completing tasks efficiently [6].

We are specifically interested in agents sharing context and understanding in a relevant way. Thus, by developing this framework, we have gained a better understanding of the interaction between humans and non-humans, user modeling and system usability issues involved in modeling such systems, and the context in which these specifications can be applied. We have used our framework to specify and design a prototype of a small part of the above dealership example. The prototype was implemented using AgentBuilder.

REFERENCES

1. Åqvist, L., "Deontic Logic", in (eds.), *Handbook of Philosophical Logic*, Vol. II. Edited by D. M. Gabbay & F. Guenther. Reidel, Boston (1984), 607-704.
2. Clark, H. H., and Carlson, T. B. "Speech Acts and Hearer's Beliefs", in (ed.), *Mutual Knowledge* Edited by N. V. Smith. Academic Press, New York (1982), 1-35.
3. Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, Y. *Reasoning About Knowledge*. Cambridge, MA: The MIT Press (1995).
4. Harel, D. "Dynamic Logic." *Handbook of Philosophical Logic*, Vol. II. Edited by D. M. Gabbay & F. Guenther, Reidel, Boston. (1984), 497-604.
5. Khosla, S., and Maibaum, T. S. E., "The Prescription and Description of State Based Systems." *Temporal Logic in Specification*. Edited by B. Banieqbal, H. Barringer and A. Pnueli: Springer, Berlin (1987), 243-294.
6. Dix, A. Finlay, J., Abowd, G., and Beale, R. *Human-Computer Interaction* (second edition). Prentice Hall, Hertfordshire, UK (1998).