On Ability to Autonomously Execute Agent Programs with Sensing — Extended Abstract

Sebastian Sardiña
Dept. of Computer Science
University of Toronto
Toronto, Canada
ssardina@cs.toronto.edu

Giuseppe De Giacomo
Dip. Informatica e Sistemistica
Univer. di Roma "La Sapienza"
Roma, Italy
degiacomo@dis.uniroma1.it

Yves Lespérance
Dept. of Computer Science
York University
Toronto, Canada
lesperan@cs.yorku.ca

Hector J. Levesque
Dept. of Computer Science
University of Toronto
Toronto, Canada
hector@cs.toronto.edu

There has been much work already on formal models of deliberation/planning under incomplete information, where an agent can perform sensing actions to acquire additional information. But most of it has been set in epistemic logic-based frameworks and is hard to relate to work on agent programming languages (e.g. 3APL, AgentSpeak(L)). Here, we develop new non-epistemic formalizations of deliberation that are much easier to relate to standard agent programming language semantics based on transition systems.

When doing deliberation/planning under incomplete information, one typically searches over a set of states, each of which is associated with a knowledge base (KB) or theory that represents what is known in the state. To evaluate tests in the program and to determine what transitions/actions are possible, one looks at what is *entailed* by the current KB. To allow for future sensing results, one looks at which of these are *consistent* with the current KB. We call this type of approach to deliberation "entailment and consistency-based" (EC-based). In this paper, we argue that EC-based approaches do not always work, and propose an alternative. Our accounts are formalized within the situation calculus and use a simple programming language based on ConGolog to specify agent programs, but we claim that the results generalize to most proposed agent programming languages/frameworks.

Our accounts rely on a semantics for online executions of programs with sensing. A configuration is a pair (δ, σ) involving a program δ and a history σ specifying the actions performed so far and the sensing results obtained. In the full paper, we define a transition relation for this, i.e. when a configuration (δ, σ) *may evolve* to configuration (δ', σ') w.r.t. a model M (relative to an underlying theory of action $\mathcal{D}$). The definition requires that the theory $\mathcal{D}$, augmented with the sensing results in σ, entail that the transition is possible. The model M is used to represent a possible environment and generate sensing results. We also define when a configuration is *final*, i.e. may legally terminate.

Perhaps the first approach to come to mind for defining when an agent knows how/is able to execute a deterministic program δ in a history σ goes as follows: the agent must always know what the next action prescribed by the program is and be able to perform it such that no matter what sensing output is obtained as a result of doing the action, she can continue this process with what remains of the program and, eventually, reach a configuration where she knows she can legally terminate. We can formalize this idea as follows.

We define $\mathit{KHow}_{EC}(\delta, \sigma)$ to be the smallest relation $\mathcal{R}(\delta, \sigma)$ such that:

(E1) if (δ, σ) is *final*, then $\mathcal{R}(\delta, \sigma)$;

(E2) if (δ, σ) *may evolve* to configurations $(\delta', \sigma \cdot (a, \mu_i))$ w.r.t. some models M_i of theory $\mathcal{D}$ with $i = 1..k$ for some $k \geq 1$, and $\mathcal{R}(\delta', \sigma \cdot (a, \mu_i))$ holds for all $i = 1..k$, then $\mathcal{R}(\delta, \sigma)$.

The first condition states that every terminating configuration is in the relation KHow_{EC}. The second condition states that if a configuration performs an action transition and for every consistent sensing result, the resulting configuration is in KHow_{EC}, then this configuration is also in KHow_{EC}.

Note that the agent's lack of complete knowledge in a history σ is modeled by the theory $\mathcal{D}$ augmented with the sensing results in σ being incomplete and having many different models. KHow_{EC} uses entailment to ensure that the information available is sufficient to determine which transition should be performed next. KHow_{EC} uses consistency to determine which sensing results can occur, for which the agent needs to have a subplan that leads to a final configuration. So we say that KHow_{EC} is an *entailment and consistency-based* (EC-based) account of knowing how.

This EC-based account of knowing how seems quite intuitive and attractive. However it has a fundamental limitation: it fails on programs involving indefinite iteration. The following simple example shows the problem. Consider a situation in which an agent wants to cut down a tree. Assume that the agent has a primitive action $chop$ to chop at the tree, and also assume that she can always

AAMAS'04, July 19-23, 2004, New York, New York, USA.

find out whether the tree is down by doing the (binary) sensing action $look$. If the sensing result is 1, then the tree is down; otherwise the tree remains up. There is also a fluent $RemainingChops(s)$, which we assume ranges over the natural numbers $\mathbb{N}$ and whose value is unknown to the agent, and which is meant to represent how many $chop$ actions are still required in s to bring the tree down. The agent's goal is to bring the tree down, i.e., bringing about a situation s such that $Down(s)$ holds, where $Down(s) \stackrel{\text{def}}{=} RemainingChops(s) = 0$. The example can be specified by an action theory $\mathcal{D}_{tc}$ (see the full paper). There exists some $n \in \mathbb{N}$, though unknown and unbounded, such that the tree will fall after n chops. Because of this, intuitively, we should have that the agent can bring the tree down, since if the agent keeps chopping, the tree will eventually come down, and the agent can find out whether it has come down by looking. Thus, for the program $\delta_{tc} =$ **while** $\neg Down$ **do** $chop; look$ **endWhile** we should have that $\mathit{KHow}_{EC}(\delta_{tc}, \epsilon)$ (note that δ_{tc} is deterministic). However, this is not the case:

Theorem 1 *$\mathit{KHow}_{EC}(\delta_{tc}, \epsilon)$ does not hold.*

Thus, the simple EC-based formalization of knowing how gives the wrong result for this example. Why is this so? Consider the hypothetical scenario in which an agent keeps chopping and looking forever, always seeing that the tree is not down. There is no model of $\mathcal{D}_{tc}$ where δ_{tc} yields this scenario, as the tree is guaranteed to come down after a finite number of chops. However, KHow_{EC} is, in some way, taking this case into account in determining whether the agent knows how to execute δ_{tc}. This happens because every finite prefix of this never-ending execution is indeed *consistent* with $\mathcal{D}_{tc}$. The problem is that the set of *all* of them together is not. This is why KHow_{EC} fails. In the full paper, we show that more generally, KHow_{EC} fails on problems that require unbounded iteration.

KHow_{EC} fails on the tree chopping example because "local consistency" is used to construct the configuration tree, which keeps switching which model of $\mathcal{D}$ (representing the environment) is used to generate the next sensing result, postponing the observation that the tree has come down forever. It seems reasonable to try to correct the problem by fixing the model of $\mathcal{D}$ used in generating possible configurations. Let's do this.

We define when an agent knows how to execute a program δ in a history σ and a model M (which represents the environment), $\mathit{KHowInM}(\delta, \sigma, M)$, as the smallest relation $\mathcal{R}(\delta, \sigma)$ such that:

(T1) if (δ, σ) is *final*, then $\mathcal{R}(\delta, \sigma)$;
(T2) if (δ, σ) *may evolve* to $(\delta', \sigma \cdot (a, \mu))$ w.r.t. M and $\mathcal{R}(\delta', \sigma \cdot (a, \mu))$, then $\mathcal{R}(\delta, \sigma)$;

The only difference between this and KHow_{EC} is that the sensing results come from the fixed model M. Given this, we obtain the following formalization of when an agent knows how to execute a program δ in a history σ:

$\mathit{KHow}_{ET}(\delta, \sigma)$ iff for every model M such that $M \models \mathcal{D} \cup \mathcal{C} \cup \{Sensed[\sigma]\}$, $\mathit{KHowInM}(\delta, \sigma, M)$.

We call this type of formalization *entailment and truth-based,* since it uses entailment to ensure that the agent knows what transitions she can do, and *truth in a model* to obtain possible sensing results.

We claim that KHow_{ET} is actually correct for programs δ that are deterministic. For instance, it handles the tree chopping example correctly:

Theorem 2 *$\mathit{KHow}_{ET}(\delta_{tc}, \epsilon)$ holds w.r.t. theory $\mathcal{D}_{tc}$.*

We have also shown that whenever $\mathit{KHow}_{EC}(\delta, \sigma)$ holds, so does $\mathit{KHow}_{ET}(\delta, \sigma)$, but not vice versa. In the full paper, we show how one can define the notion of ability to achieve a goal in terms of our notions of knowing how, and also account for knowing how to execute a *nondeterministic* program.

Our non-epistemic accounts of knowing how are easily related to models of agent programming language semantics and our results have important implications for this area. While most work on agent programming languages (e.g. 3APL, AgentSpeak(L), etc.) has focused on reactive execution, sensing is acknowledged to be important and there has been interest in providing mechanisms for run-time planning/deliberation. Executing a program on-line without deliberation/lookahead just involves repeatedly selecting some transition allowed in the current configuration and performing it. However, one natural view is that *deliberation can simply be taken as a different control regime involving search over the agent program's transition tree*. In this view, a deliberating interpreter could first lookahead and search the program's transition tree to find a sequence of transitions that leads to successful termination and later execute this sequence. Clearly, in the presence of sensing, this idea needs to be refined. One must find more than just a path to a final configuration in the transition tree; one needs to find some sort of conditional plan or subtree where the agent has chosen some transition among those allowed, but must have branches for all possible sensing results. The natural way of determining which sensing results are possible is checking their consistency with the current belief base. Thus, what is considered here is essentially an EC-based approach. Our results show that this view of deliberation is fundamentally flawed when sensing is present. It produces an account that only handles problems that can be solved in a bounded number of actions. As an approach to implementing deliberation, this may be perfectly fine. But as a semantics or specification, it is wrong. What is required is a much different kind of account, like our ET-based one.
See `www.cs.toronto.edu/cogrobo` for the full paper.

AAMAS'04, July 19-23, 2004, New York, New York, USA.