ECASL : A Model of Rational Agency for Communicating Agents
(ECASL : Un modèle d’agent rationnel)

Shakil M. Khan  Yves Lespérance
skhan@cs.yorku.ca  lesperan@cs.yorku.ca
Dept. of Computer Science
York University
Toronto, ON, Canada M3J 1P3

Abstract:
The Cognitive Agent Specification Language (CASL) is a framework for specifying and verifying complex communicating multiagent systems. In this paper, we develop an extended version, ECASL, which incorporates a formal model of means-ends reasoning suitable for a multiagent context. In particular, we define a simple model of cooperative ability, give a definition of rational plans, and show how an agent’s intentions play a role in determining her next actions. This bridges the gap between intentions to achieve a goal and intentions to act. We also show that in the absence of interference, an agent that is capable of achieving a goal, intends to do so, and is acting rationally will eventually achieve it.

Keywords: Agent theory, rationality, intentions

1 Introduction

Most agent theories [1, 17] suffer from a similar problem : they axiomatize the relation between the different mental attitudes of the agents and the physical states of the world, but they do not account for how the agents will achieve their goals, how they plan and commit to plans. Ideally, an agent’s intention to achieve a state of affairs in a situation should drive the agent to intend to execute a plan that she thinks is rational in that situation. In other words, an agent’s future directed intentions should lead her to adopt rational plans and eventually achieve her intentions.

Another recent thread in agent theory introduces a procedural component to the framework in an attempt to close the gap between agents’ intentions to achieve a state of affairs and their intentional actions, as well as to support the modeling of complex multiagent systems. One example of this is the Cognitive Agent Specification Language (CASL) [23, 24], which is a framework for specifying and verifying complex communicating multiagent systems. However, it is somewhat restricted in the sense that it requires the modeler to specify agent behavior explicitly, and the program that controls the agent’s actions need not be consistent with the agent’s intentions, or do anything to achieve them.

In this paper, we propose a solution to this problem by developing an extended version of CASL, ECASL. In particular, we define rational plans and ability in a multiagent context, and use these notions to link future and present directed intentions. We introduce a special action, the commit action, that makes the agent commit to a plan, and define a meta-controller BehaveRationallyUntil that has the agent act rationally to achieve a specific goal by choosing and committing to a rational plan, and carrying it out. Then we show that given that an agent has an intention, she will act to achieve it provided that she is able to do so.

The paper is organized as follows : in the next section, we outline previous work on CASL. In Section 3, we develop a simple formalization of cooperative ability for agents working in a multiagent setting. In Section 4, we define rational plans, relate future and present directed intentions, and discuss what it means for an agent to behave rationally. We also state a theorem that links an agent’s intentions and abilities to the eventual achievement of her intentions.

2 CASL

In CASL [23, 24], agents are viewed as entities with mental states, i.e., knowledge and goals, and the specifier can define the behavior of the
agents in terms of these mental states. CASL combines a declarative action theory defined in the situation calculus with a rich programming/process language, ConGolog [3]. Domain dynamics and agents’ mental states are specified declaratively in the theory, while system behavior is specified procedurally in ConGolog.

In CASL, a dynamic domain is represented using an action theory formulated in the situation calculus [15], a (mostly) first order language for representing dynamically changing worlds in which all changes are the result of named actions. CASL uses a theory that includes the following set of axioms:

- action precondition axioms, one per action,
- successor state axioms (SSA), one per fluent, that encode both effect and frame axioms and specify exactly when the fluent changes [18],
- initial state axioms describing what is true initially including the mental states of the agents,
- axioms identifying the agent of each action,
- unique name axioms for actions, and
- domain-independent foundational axioms describing the structure of situations [9].

Within CASL, the behavior of agents is specified using the notation of the logic programming language ConGolog [3]. A typical ConGolog program is composed of a sequence of procedure declarations, followed by a complex action. Complex actions can be composed using constructs that include primitive actions (a), waiting for a condition (φ ?), sequence (δ1; δ2), nondeterministic branch (δ1 | δ2), nondeterministic choice of arguments (πx.δ), conditional branching (If φ Then δ1 Else δ2 EndIf), while loop (While φ Do σ EndWhile), and procedure call (β(γ)). Intuitively, πx.δ nondeterministically picks a binding for the variable x and performs the program δ for this binding of x. ConGolog also supports nondeterministic iteration, concurrent execution with and without priorities, and interrupts. To deal with multiagent processes, primitive actions in CASL take the agent of the action as argument.

The semantics of the ConGolog process description language is defined in terms of transitions. Two special predicates Final and Trans are introduced, and are characterized by defining axioms for each of the above constructs, where Final(δ, s) means that program δ may legally terminate in situation s, and where Trans(δ, s, δ′, s′) means that program δ in situation s may legally execute one step, ending in situation s′ with program δ′ remaining. The overall semantics of a program is specified by the Do relation:

$$\text{Do}(δ, s, s′) \equiv \exists δ′ \cdot (\text{Trans}^*(δ, s, δ′, s′) \land \text{Final}(δ′, s′))$$.

Do(δ, s, s′) holds if and only if s′ can be reached by performing a sequence of transitions starting with program δ in s, and the remaining program δ′ may legally terminate in s′. Here, Trans* is the reflexive transitive closure of the transition relation Trans.

CASL incorporates a branching time temporal logic, where each situation has a linear past and a branching future. In this framework, one can write both state formulas and path formulas. A state formula φ(s) takes a single situation as argument and is evaluated with respect to that situation. On the other hand, a path formula ψ(s1, s2) takes two situations as arguments and is evaluated with respect to the interval (finite path) [s1, s2]. We often use φ (and ψ) to denote a formula whose fluents may contain a placeholder constant now (now and then, resp.) that stands for the situation in which φ (ψ, resp.) must hold. φ(s) (and ψ(s1, s2)) is the formula that results from replacing now with s (now and then with s1 and s2, resp.). Where the intended meaning is clear, we sometimes suppress the placeholder(s).

CASL allows the specifier to model agents in terms of their mental states by including operators to specify agents’ information (i.e., their knowledge), and motivation (i.e., their goals or intentions). We usually use state formulas within the scope of knowledge, and path formulas within the scope of intentions. Following [16, 21], CASL models knowledge using a possible worlds account adapted to the situation calculus. K(agt, s′, s) is used to denote that in situation s, agt thinks that she could be in situation s′. s′ is called a K-alternative situation for agt in s. Using K, the knowledge or belief of an agent, Know(agt, φ, s), is defined as ∀s′(K(agt, s′, s) ⊃ φ(s′)), i.e. agt knows φ in s if φ holds in all of agt’s K-accessible situations in s. In CASL, K is constrained to be reflexive, transitive, and euclidean in the initial situation to capture the fact that agents’ knowledge is true, and that agents have positive and negative introspection. As shown in [21], these constraints then continue to hold after any sequence of actions since they are preserved by the successor state axiom for K.

Scherl and Levesque [21] showed how to cap-

---

1Since we have predicates that take programs as arguments, we need to encode programs as first-order terms as in [3]. For notational simplicity, we suppress this encoding and use formulae as terms directly.
ture the changes in beliefs of agents that result from actions in the successor state axiom for $K$. These include knowledge-producing actions that can be either binary sensing actions or non-binary sensing actions. Following [12], the information provided by a binary sensing action is specified using the predicate $SF(a, s)$, which holds if the action $a$ returns the binary sensing result 1 in situation $s$. Similarly for non-binary sensing actions, the term $sff(a, s)$ is used to denote the sensing value returned by the action.

Lespérance [11] extends the SSA of $K$ in [21] to support two variants of the $inform$ communicative action, namely $informWhether$ and $informRef$. Here, $inform(inf, agt, φ)$, $informWhether(inf, agt, ψ)$, and $informRef(inf, agt, θ)$ mean that $inf$ informs $agt$ that $φ$ currently holds, $inf$ informs $agt$ about the current truth value of $ψ$, and $inf$ informs $agt$ of who/what $θ$ is, respectively. The preconditions of $inform$ are as follows:

$$Poss(inform(inf, agt, φ), s) ≡ Know(inf, φ, s) \land ¬Know(inf, Know(agt, φ, now), s).$$

In other words, the agent $inf$ can inform $agt$ that $φ$, iff $inf$ knows that $φ$ currently holds, and does not believe that $agt$ currently knows that $φ$. The preconditions of $informWhether$ and $informRef$ are similar to that of $inform$. The SSA for $K$ is defined as follows :

$$K(agt, s^*, do(a, s)) \equiv ∃s'. [K(agt, s', s) ∧ s^* = do(a, s') \land Poss(a, s') \land ((BinarySensingAction(a) ∧ Agent(a) = agt) \lor (SF(a, s') ≡ SF(a, s))) \land ((NonBinarySensingAction(a) ∧ Agent(a) = agt) \lor (sff(a, s') = sff(a, s))) \land inf, φ, (a = inform(inf, agt, φ) ⊃ φ(s')) \land inf, ψ, (a = informWhether(inf, agt, ψ) \lor (ψ(s') ≡ ψ(s))) \land inf, θ, (a = informRef(inf, agt, θ) \lor (θ(s') = θ(s))).$$

This says that after an action happens, every agent learns that it has happened. Moreover, if the action is a sensing action, the agent performing it acquires knowledge of the associated proposition or term. Furthermore, if the action involves someone informing $agt$ that $φ$ holds, then $agt$ knows this afterwards, and similarly for $informWhether$ and $informRef$. Note that this axiom only handles knowledge expansion, not revision.

CASL also incorporates goal expansion and a limited form of goal contraction. Goals or intentions are modeled using an accessibility relation $W$ over possible situations. The $W$-accessible situations for an agent are the ones where she thinks that all her goals are satisfied. $W$-accessible situations may include situations that the agent thinks are impossible, unlike Cohen and Levesque’s [1] $G$-accessible worlds. But intentions are defined in terms of the more primitive $W$ and $K$ relations so that the intention accessible situations are $W$-accessible situations that are also compatible with what the agent knows, in the sense that there is a $K$-accessible situation in their history. This guarantees that agents’ intentions are realistic, that is, agents can only intend things that they believe are possible. Thus we have :

$$Int(agt, ψ, s) ≡ ∀s', s^*. [W(agt, s', s) \land K(agt, s', s) \land s^* ≤ s'] \lor ψ(s', s^*).$$

This means that the intentions of an agent in $s$ are those formulas that are true for all intervals between situations $s'$ and $s^*$ where the situations $s^*$ are $W$-accessible from $s$ and have a $K$-accessible situation $s'$ in their history. Intentions are future oriented, and any goal formula will be evaluated with respect to a finite path defined by a pair of situations, a beginning situation $s'$ and an ending situation $s^*$. This formalization of goals can deal with both achievement goals and maintenance goals. An achievement goal $ψ$ is said to be satisfied if $ψ$ holds between now and then, i.e., if Eventually($ψ, now, then$), which is defined as $∃s'. (now ≤ s' ≤ then \land ψ(s'))$. In [22], Shapiro showed how positive and negative introspection of intentions can be modeled by placing some constraints on $K$ and $W$. To make sure that agents’ wishes and intentions are consistent, $W$ is also constrained to be serial.

The SSA for $W$ which handles intention change in CASL, has the same structure as a SSA for a domain dependent fluent. In the following, $W^+(agt, a, s^*, s)$ ($W^-(agt, a, s^*, s)$, respectively) denotes the conditions under which $s^*$ is added to (dropped from, respectively) $W$ as a result of the action $a$ :

$$W(agt, s^*, do(a, s)) \equiv W^+(agt, a, s^*, s) \lor (W(agt, s^*, s) \land ¬W^-(agt, a, s^*, s)).$$

An agent’s intentions are expanded when it is requested something by another agent. After the request(req, agt, ψ) action, $agt$ adopts the goal that $ψ$, unless she has a conflicting goal or is not willing to serve req for $ψ$. Therefore, this action should cause $agt$ to drop any paths in $W$ where
ψ does not hold. This is handled in $W^-$:

$$W^-(\text{agt, } a, s^*, s) \equiv \text{IncompRequest}(\text{agt, } a, s^*, s),$$

$$\text{IncompRequest}(\text{agt, } a, s^*, s) \equiv$$

$$[\exists \text{req}, \psi. \ a = \text{request}(\text{req, } \text{agt, } \psi)$$

$$\land \text{Serves}(\text{agt, } \text{req, } \psi, s) \land \neg \text{Int}(\text{agt, } \neg \psi, s)$$

$$\land \exists \text{s}' . \ K(a \land s', s) \land s' \leq s^*$$

$$\land \neg \psi(\text{do}(a, s^*, s')).]$$

Here, the request action is considered a primitive action. The preconditions of request are:

$$\text{Poss}(\text{request}(\text{req, } \text{agt, } \phi), s) \equiv \text{Int}(\text{req, } \phi, s).$$

A limited form of intention contraction is also handled in CASL. Suppose that the agent req requests agt that $\psi$ and later decides it no longer wants this. The requester req can perform the action cancelRequest(req,agt,ψ), which causes agt to drop the goal that $\psi$. cancelRequest actions are handled by determining what the W relation would have been if the corresponding request action had never happened. This type of goal contraction is handled in $W^+$, which can be defined as follows:

$$W^+(\text{agt, } a, s^*, s_1) \equiv \exists a_1. W(\text{agt, } s^*, s_1)$$

$$\land \exists a_1, \text{do}(a_1, s_1) \leq s \land \text{Cancels}(a, a_1)$$

$$\land (\forall a^*, s'. \text{do}(a_1, s_1) < \text{do}(a^*, s') \leq s)$$

$$\land \neg W^-(\text{agt, } a^*, s^*, s')).$$

$$\text{Cancels}(a, a') \equiv [\exists \text{req}, \psi. a' = \text{request}(\text{req, } \text{agt, } \psi)$$

$$\land a = \text{cancelRequest}(\text{req, } \text{agt, } \psi)].$$

Suppose that a cancelRequest action occurs in situation s. The W relation is first restored to the way it was before the corresponding request action occurred, i.e., in $s_1$. Then starting just after the request, all the actions $a^*$ that occurred in the history of s (say in situation $s'$) are considered, and any situation $s^*$ in W that satisfies $W^-(\text{agt, } a^*, s^*, s')$ is removed from W. A cancelRequest action can only be executed if a corresponding request action has occurred in the past.

### 3 Simple Cooperative Ability

An agent cannot be expected to eventually achieve an intention just because she has that intention, and she is acting rationally. We also need to make sure that the agent is capable of achieving the goal in the current situation [10]. In a single agent domain, an agent’s ability can roughly be defined as her knowledge of a plan that is physically and epistemically executable and whose execution achieves the goal. However, modeling multiagent ability is a more complex problem, since in this case we need to consider the agents’ knowledge about each other’s knowledge and intentions as well as how they choose actions, behave rationally, etc. In this section, we develop a simple model of cooperative ability of agents suitable for a limited multiagent context in the absence of exogenous actions, i.e., actions whose performance is not intended by the planning agent. In an open multiagent framework, agents’ actions may interfere with each other, possibly perturbing their plans. In some cases, there are multiple strategies to achieve a common goal, and the agents may fail unless they coordinate their choice of strategy by reasoning about each other’s knowledge, ability, and rational choice. Moreover, agents may have conflicting goals or intentions. To simplify, we restrict our framework by only allowing plans where the actions of the other agents must do are fully specified, i.e., action delegation is possible, but (sub)goal delegation is not. The primary agent, who is doing the planning, is constrained to know the whole plan in advance. Thus, the primary agent is allowed to get help from others, but she can only ask them to perform specific actions. Given this, we do not need to model the fact that the other agents behave rationally.

When dealing with ability, it is not enough to say that the agent is able to achieve a goal iff she has a physically executable plan, and any execution of this plan starting in the current situation achieves the goal. We should also take into account the epistemic and intentional feasibility of the plan. This is necessary as physical executability does not guarantee that the executor will not get stuck in a situation where it knows that some step must be performed, but does not know which. For example, consider the plan (a; If $\phi$ Then b Else c EndIf) | d, where actions $a$, $b$, $c$, and $d$ are always possible, but where the agent does not know whether $\phi$ holds after $a$. If the agent follows the branch where the first action is $a$, she will get stuck due to incomplete knowledge. Hence, the result of deliberation should be a kind of plan where the executor will know what to do next at every step, a plan that does not itself require deliberation to interpret. To deal with this, De Giacomo et al. [2] defined the notion of Epistemically Feasible Deterministic Programs (EFDPs) for single agent plans and characterized deliberation in terms of it. Note that EFDPs are deterministic, since they are the result of deliberation and their execution should not require making further choices.
Since we are dealing with cooperative multiagent ability, we also need to make sure that the cooperating agents intend to perform the requested actions when it is their turn to act. We extend the notion of EDFP to handle simple multiagent plans as follows. A program is called an Epistemically and Intentionally Feasible Deterministic Program (EIFDP) in situation $s$ for agent $agt$, if at each step of the program starting at $s$, $agt$ always has enough information to execute the next action in the program, or knows that the executor of the next action is another agent, and that this agent has enough information to execute this action and intends to do it.

Put formally:

$$EIFDP(agt, \delta, s) \equiv$$

$$\forall \delta', s'. \text{Trans}'(\delta, s, \delta', s') \supset LEIFDP(agt, \delta', s'), \text{LEIFDP}(agt, \delta, s) \equiv$$

$$\text{Know}(agt, \text{Final}(\delta, \text{now})) \wedge$$

$$\neg \exists \delta', s'. \text{Trans}(\delta, \text{now}, \delta', s', s)$$

$$\vee \exists \delta', s'. \text{Know}(agt, \sim \text{Final}(\delta, \text{now})) \wedge$$

$$\text{UTrans}(\delta, \text{now}, \delta', s)$$

$$\vee \exists \delta', s'. \text{Know}(agt, \sim \text{Final}(\delta, \text{now})) \wedge$$

$$\text{Agent}(a) = agt \wedge$$

$$\text{UTrans}(\delta, \text{now}, \delta', do(a, \text{now})), s)$$

$$\vee \exists \delta', a. \text{Know}(agt, \sim \text{Final}(\delta, \text{now})) \wedge$$

$$\exists a. \text{UTrans}(\delta, \text{now}, \delta', do(a, \text{now}))) \wedge$$

$$\text{Agent}(a) = agt' \neq agt \wedge$$

$$\text{Int}(agt', \exists s', s' \leq \text{then})$$

$$\wedge \text{Do}(a, \text{now}, s'), (\text{now}), s).$$

Thus to be an EIFDP, a program must be such that all configurations reachable from the initial program and situation, involve a Locally Epistemically and Intentionally Feasible Deterministic Program (LEIFDP). A program is a LEIFDP in a situation with respect to an agent, if the agent knows that the program is currently in its Final configuration and no further transitions are possible, or knows that she is the agent of the next action and knows what unique transition (with or without an action) it can perform next, or knows that someone else $agt'$ is the agent of the next action, that $agt'$ knows what the action is and intends to do it next, and knows what unique transition the program can perform next with this action. Here, $U\text{Trans}(\delta, s, \delta', s')$ means that the program $\delta$ in $s$ can perform a unique transition, which takes the agent to $s'$ with the remaining program $\delta'$. Note that when it is the other agent’s turn, $agt$ does not have to know exactly what the next action is, i.e., know all the parameters of the next action. However, at every step, she must know what the remaining program is.

EIFDPs are suitable results for planning. They can always be executed successfully and since they are deterministic, they do not require further deliberation to execute. Using EIFDP, ability can be defined as follows:

$$\text{Can}(agt, \psi(\text{now}, \text{then}), s) \equiv$$

$$\exists s'. \text{Do}(\delta, \text{now}, s') \wedge$$

$$\forall s'. (\text{Do}(\delta, \text{now}, s') \supset \psi(\text{now}, s')), s).$$

Thus, an agent can achieve a goal in situation $s$, iff she knows of a plan $\delta$ that is an EIFDP, is executable starting at $s$, and any possible execution of the plan starting in the current situation brings about the goal.

We use the following as our running example (adapted from [16]) throughout the paper. Consider a world in which there is a safe with a combination lock. If the safe is locked and the correct combination is dialed, then the safe becomes unlocked. However, dialing the incorrect combination will cause the safe to explode. The agent can only dial a combination if the safe is intact, and it is not possible to change the combination of the safe. Initially, the agent $Agt_1$ has the intention to open the safe, but does not know the combination. However, she knows that $Agt_2$ knows it. She also knows that $Agt_2$ is willing to serve/help her, and that $Agt_2$ does not have the intention of informing her of the combination of the safe. Here are some of the axioms that we use to model this domain:

$$sf_1) \text{Poss}(a, s) \supset [\text{Exploded}(\text{do}(a, s)) \equiv$$

$$\exists c, \text{agt}. (a = \text{dial}(\text{agt}, c) \wedge \text{Comb}(s) \neq c) \vee \text{Exploded}(s)].$$

$$sf_2) \text{Poss}(\text{dial}(\text{agt}, c), s) \equiv \sim \text{Exploded}(s).$$

$$sf_3) \text{Agent}(\text{dial}(\text{agt}, c)) = \text{agt}.$$

$$sf_4) \sim \text{Exploded}(\text{Safe}).$$

The first axiom, a successor state axiom, states that the safe has exploded after doing action $a$ iff $a$ denotes the action of dialing the wrong combination, or if the safe has already exploded. The second axiom, a precondition axiom, states that it is possible to dial a combination for the safe in situation $s$ iff the safe is intact in $s$. The third axiom is an agent axiom and defines the agent of the dial action. The last axiom is an initial situation axiom, and states that the safe is initially intact. From now on, we will use $D_{\text{safe}}$ to denote the set of axioms that we use to model this safe domain (see [8] for the complete axiomatization).
Now, consider the following plan:

\[ \sigma_{\text{saf e}} = \text{requestAct}(Agt_1, Agt_2, informRef(Agt_2, Agt_1, Comb(s))); \]
\[ \text{informRef}(Agt_2, Agt_1, Comb(s)); \]
\[ \text{dial}(Agt_1, Comb(s)). \]

So, the plan is that Agt_1 will request Agt_2 to inform her of the combination of the safe. Agt_2 will inform Agt_1 of the combination of the safe, and finally, Agt_1 will dial the combination to open the safe. We claim that \( \sigma_{\text{saf e}} \) is an EIFDP in the initial situation for Agt_1, and that Agt_1 is able to achieve her intention of opening the safe in the initial situation \( S_0 \):

**Theorem 1**

a. \( D_{\text{saf e}} \models \text{EIFDP}(Agt_1, \sigma_{\text{saf e}}, S_0). \)

b. \( D_{\text{saf e}} \models \text{Can}(Agt_1, \text{Eventually}(-\text{Locked}), S_0). \)

(a) holds as all configurations reached by \( \sigma_{\text{saf e}} \) starting in \( S_0 \) are LEIFDP. (b) holds as Agt_1 knows of a plan (i.e., \( \sigma_{\text{saf e}} \)), which she knows is an EIFDP and is executable, and knows that any execution of this plan ends up in a situation where the safe is unlocked.

**4 From Intentions That to Intentions to Act**

In this section, we define rational plans and extend CASL to model the role of intention and rationality in determining an agent’s actions. This bridges the gap between future directed intentions and present directed ones. We also present a theorem that relates intention and ability to the eventual achievement of intended goals.

Before going further, let us discuss the communication actions that we will use in ECASL. Like in CASL, we use three primitive informative communication actions, namely, inform, informWhether, and informRef. However, unlike in CASL, we provide two intention transfer communication actions, request and requestAct, and these are defined in terms of inform.\(^3\)

The request action can be used by an agent to request another agent to achieve some state of affairs, whereas requestAct involves an agent’s request to another agent to perform some particular complex action starting in the next situation. Formally,

\[
\text{request}(req, agt, \phi) \triangleq \begin{align*}
\text{inform}(req, agt, \text{Int}(req, \phi, now)), \\
\text{requestAct}(req, agt, \delta) \triangleq \begin{align*}
\text{request}(req, agt, \exists s', a. \text{Do}(\delta, do(a, now), s')) \\
\wedge \text{now} < s' \leq \text{then} \wedge \text{Agent}(\delta) = agt.
\end{align*}
\end{align*}
\]

Here \( \text{Agent}(\delta) = agt \) means that the agent of all actions in \( \delta \) is \( agt \). In our specification, we only allow sincere requests. That is, an agent can perform a request if the request is not contradictory to her current intentions. So defining requests as informing of intentions is reasonable. However, since requests are modeled in terms of inform, and since we are using true belief, the account seems to be overly strict. For instance, in the safe domain, \( \sigma_{\text{saf e}} \) is a rational plan for Agt_1 in the initial situation. However, initially Agt_1 does not have the intention that Agt_2 informs her the combination of the safe. So we cannot show that \( \sigma_{\text{saf e}} \) is rational, since it requires Agt_1 to know that she has the intention before she can inform about it. One way to solve this is to relax the preconditions of inform. However, this can have problematic consequences, as someone could inform of something without knowing it, and this might require belief revision by the addressee. Later, we will discuss another way to avoid this problem by building commitment into plans. For now, we just assume that initially Agt_1 has the intention that Agt_2 informs her of the combination of the safe.

To facilitate the cancellation of requests, we also provide two actions, namely, cancelRequest, and cancelReqAct. Unlike CASL where cancelRequest is primitive, we define it using inform. These two actions are defined as follows:

\[
\text{cancelRequest}(req, agt, \psi) \triangleq \begin{align*}
\text{inform}(req, agt, \neg\text{Int}(req, \psi, now)), \\
\text{cancelReqAct}(req, agt, \delta) \triangleq \begin{align*}
\text{cancelRequest}(req, agt, \exists s^*, s^+, \text{prev.} \\
\text{prev} = \text{do(requestAct(req, agt, \delta), s^*)} \\
\wedge s^+ < \text{now} \leq s^* \leq \text{then} \wedge \text{Do}(\delta, \text{prev}, s^*) \\
\wedge \text{Agent}(\delta) = agt.
\end{align*}
\end{align*}
\]

Now let us look at what plans are rational for an agent. An agent that is acting rationally, should prefer some plans to others. To this end, we define an ordering on plans:

\[ \succeq (agt, \delta_1, \delta_2, s) \triangleq \]
∀ s′. K(agt, s′, s) ∧ ∃ s′′. Do(δ, s′, s′′) ∧ W(agt, s′′, s) \\
≥ [∃ s′′. Do(δ, s′, s′′) ∧ W(agt, s′′, s)].

That is, a plan δ₁ is as good as another plan δ₂ in situation s for an agent agt iff for all Waccessible situations that can be reached by following δ₂ from a situation that is K-accessible from s, say s′, there exists a W-accessible situation that can be reached from s′ by following δ₁. In other words, δ₁ is at least as good as δ₂ if it achieves the agent’s goals in all the possible situations where δ₂ does.

Using EIFDP and the ≥ relation, we next define rational plans. A plan δ is said to be rational in situation s for an agent agt if the following holds:

Rational(agt, δ, s) \equiv \\
∀ s′. ≥ (agt, δ', δ, s) ≥ (agt, δ, δ', s) \land EIFDP(agt, δ, s).

Thus, a rational plan in a situation s, is a plan that is as good as any other plan in s and is an EIFDP in s.

For example, consider the plan σsafe. We claim that σsafe is as good as any other plan available to Agt₁ in the initial situation, and that σsafe is rational in the initial situation.

**Theorem 2**

a. Dsafe |= ∀ σ. ≥ (Agt₁, σsafe, σ, S₀).

b. Dsafe |= Rational(Agt₁, σsafe, S₀).

Since this plan achieves Agt₁’s intention of opening the safe starting in any situation that is K-accessible to S₀, (a) holds. (b) follows from the fact that σsafe is as good as any other plan in S₀ and is an EIFDP in S₀.

In most cases, there are many rational plans (i.e., ways of achieving as many goals as possible). The decision of which plan the agent commits to is made based on pragmatic/logical grounds. We do not model this here. Instead, we introduce a commit(agt, δ) action that will model the agent’s committing to a particular plan δ, more specifically, committing to executing δ next. The action precondition axiom for the commit action is as follows:

Poss(commit(agt, δ), s) \equiv \\
¬ Int(agt, ¬ s, s ≤ s ≤ then ∧ Do(δ, now, s), s).

That is, the agent agt can commit to a plan δ in situation s, iff the agent currently does not have the intention that the actions in the plan do not happen next.

Next, we extend the SSA for W seen earlier to handle intention revision as a result of the agent’s commitment to a rational plan. We modify W⁻ as follows:

W⁻(agt, s, s*) \equiv IncompRequest(agt, a, s*, s) ∨ IncompCommit(agt, a, s*, s).

Here, IncompCommit handles the expansion of the agent’s intentions that occur when a commit action occurs. We define IncompCommit as follows:

IncompCommit(agt, a, s*, s) \equiv \\
[∃ δ. a = commit(agt, δ) \land ∃ s′. s′ ≤ s ∧ K(agt, s′, s) \land \neg ∃ s''. (s′ < s'' ≤ s ∧ Do(δ, do(a, s'), s''))].

So, after the performance of a commit action in s, a W-accessible situation s' in s will be dropped from agt’s new set of W-accessible situations if the committed to action does not happen next over the interval between the W-accessible situation s'' and its predecessor s' that is K-accessible from the current situation s.

The definition of W⁺ remains unchanged. Note that if exogenous actions are allowed, agents need to revise their commitments when an exogenous action occurs by uncommitting from the currently committed plan, and committing to a new rational plan. We return to this issue in Section 5.

We now show that our formalization of intentions has some desirable properties:

**Theorem 3**

a. \models ¬ Int(agt, φ, s) ∧ Serves(agt, req, φ, s) ∧ Poss(request(req, agt, φ, s)) ∨ \\
Int(agt, φ, do(request(req, agt, φ, s))).

b. \models Poss(commit(agt, δ), s) ∨ \\
Int(agt, s', Do(δ, now, s') ∧ now ≤ s' ≤ then, do(commit(agt, δ), s)).

(a) says that if an agent agt does not have the intention that not φ in s, then she will have the intention that φ in the situation resulting from another agent req’s request to agt that φ in s, provided that she is willing to serve req on φ, and that the request action is possible in s. (b) states that if an agent agt does not have the intention of not performing a complex action δ in s (i.e. if commit(agt, δ) is possible in s), then
she will have the intention of performing it after she commits to it.

As mentioned earlier, the problem that arises as a result of defining requests as informing of intentions can be solved by building commitment into plans. Consider the safe example; we assumed earlier that initially $Agt_1$ has the intention that $Agt_2$ informs her of the combination of the safe. We can now relax this constraint by considering $\sigma_{safe}^*$ to be our new rational plan, where $\sigma_{safe}^* = commit(Agt_1, \sigma_{safe}); \sigma_{safe}$, i.e., the new plan is that $Agt_1$ commits to $\sigma_{safe}$ and then $\sigma_{safe}$ is performed. Since $Agt_2$ commits to $\sigma_{safe}$ before she executes it, the SSA for $W$ will make her adopt the intention that $Agt_2$ informs her the combination of the safe after she requests $Agt_2$ to do so, and thus we do not need to assume that this holds.

commit provides a way to link future directed intentions and present directed ones. We next specify a generic meta-controller for an agent that arbitrarily chooses a rational plan, commits to it, and executes it. Then we can prove a theorem about the relationship between intention, ability, and the eventual achievement of an intended goal. This theorem serves as a proof of soundness of our agent theory.

The following meta-controller allows us to refer to the future histories of actions that may occur for an agent who is behaving rationally until $\psi$ holds. Rational behavior until $\psi$ can be defined as follows (we assume that there are no exogenous actions):

\[
\text{BehaveRationallyUntil}(agt, \psi(\text{now})) \triangleq \\
\pi. \text{Rational}(agt, \delta, \text{now}); \text{commit}(agt, \delta);
\]

While $\neg \psi(\text{now})$ Do

If $\exists a. \text{Int}(agt, do(a, \text{now}) \leq \text{then}, \text{now}) \land \\
\text{Agent}(a) = agt$ Then

$[\pi. \text{Int}(agt, do(a, \text{now}) \leq \text{then}, \text{now}) \land \\
\text{Agent}(a) = agt)?; a]$ Else

$\pi. \text{Int}(agt, \exists a. do(a, \text{now}) \leq \text{then} \land \\
\text{Agent}(a) \neq agt \land \\
\text{Agent}(a) = agt', \text{now})?; \\
(\pi. \text{Int}(agt', do(a', \text{now}) \leq \text{then}, \text{now})?; a')]
$ EndIf EndWhile.

That is, rational behavior until $\psi$ can be defined as arbitrarily choosing a rational plan, committing to it, and then executing it as long as $\psi$ does not hold. A rational plan can have actions by the planning agent and by other agents. When it is the planning agent’s turn to act, she should perform the action that she intends to perform next; otherwise, she should wait for the other agent to act. When it is the other agent’s turn, it will perform the action that it is supposed to perform, because rational plans are EIFDP, and thus the other agent must intend to do the action required by the plan. Note that we only deal with achievement goals here.

One problem with CASL is that the execution of plans is viewed from the system’s perspective rather than from the agents’ perspective. So, although CASL includes operators that model agents’ knowledge and goals, the system behavior is simply specified as a set of concurrent processes. To deal with this problem, Lespérance [11] proposed an account of subjective plan execution in CASL; the program construct Subj$(agt, \delta)$ ensures that $\delta$ can be executed by $agt$ based on her own knowledge state. We have extended this notion to deal with multiagent plans (i.e., plans with actions by agents other than the executor) and consider other agents’ intentions; see [8] for the formal details.

Next, we present our “success theorem”:

**Theorem 4** From Commitment and Ability to Eventuality

$$
\models [\text{OInt}(agt, \text{Eventually}((\gamma, \text{now}, \text{then}), s) \\
\land \text{Can}(agt, \text{Eventually}((\gamma, \text{now}, \text{then}), s) \\
\land \text{Int}(agt, \text{Eventually}((\psi, \text{now}, \text{then}), s)) \supset \\
\text{AllDo}(\text{Subj}(agt, \text{BehaveRationallyUntil}(agt, \psi)), s]].
$$

Intuitively, if in some situation, an agent intends to achieve some goal and is able to achieve all its intentions, then the agent will eventually achieve the goal in all rational histories from that situation. OInt$(agt, \psi, s)$ means that $\psi$ is all the intentions that $agt$ has in $s$. This construct must be used as we have to assume that the agent is able to achieve all her intentions. If this is not the case, the agent may have to choose between some of its goals and the BehaveRationallyUntil operator will not guarantee that a specific goal (i.e., $\psi$) will be achieved. If there are exogenous actions, then a more generic meta-controller can be defined. We discuss this in the next section.

We also have the following corollary for the safe domain:

---

*The construct AllDo is a strict version of Do that requires that all possible executions of a program terminate successfully; see [11] for a formal definition.*
Corollary 1

\
\[ D_{\text{safe}} \models \text{AllDo}(\text{Subj}(\text{Agt}_1, \\
\text{BehaveRationallyUntil}(\text{Agt}_1, \neg \text{Locked}), S_0)). \]

We have shown in Theorem 1(b) that $\text{Agt}_1$ can achieve her intention of opening the safe in the initial situation. Moreover, the only intention of $\text{Agt}_1$ is to open the safe. It follows from Theorem 4 that $\text{Agt}_1$ will eventually open the safe if she behaves rationally starting in $S_0$ (see [8] for a complete proof).

5 Discussion and Future Work

In this paper, we have presented a formal theory of agency that deals with simple multiagent cooperation and shows how future directed intentions and present directed ones can be related. An agent’s current rational plans depend on her current intentions. The commit action models how the agent’s intentions can be updated to include a commitment to a rational plan. Using this, we have formulated a planning framework for multiple cooperating and communicating agents in CASL. We specified how an agent’s future directed intentions will lead the agent to adopt a rational plan and then carry it out using the meta-controller BehaveRationallyUntil.

To relate agents’ intentions with their actions, Cohen and Levesque [1] required that agents eventually drop all their intentions either because they had been achieved or because they were viewed as impossible to achieve (AKA the no infinite deferral assumption). A similar account was presented by Rao and Georgeff [17]. We believe that this no infinite deferral assumption should be a consequence of an agent behaving rationally as specified by other axioms of the theory, rather than be imposed separately. A more intuitive account was presented in [25], where Singh showed that rather than having it as an assumption, the no infinite deferral principle can be derived from the theory. However, he does not explicitly address the interaction between knowledge and action and its relationship with ability.

In [20], Sadek introduces some axioms to incorporate explicit principles of rational behaviour in his adaptation of the Cohen and Levesque framework. The application of these axioms makes it possible for an agent to build rational plans in a deductive way by inference of causal chains of intention, without needing to resort to a separate planner. From an operational point of view, agents in this framework generate plans using a backward chaining planning mechanism. Sadek uses the rational effect of actions as an integral part of his specification. These rational effects express the reasons which lead an agent to select an action, and are related to perlocutionary effects. However, it is not specified under what conditions the rational effects become actual effects, and one cannot reason about these conditions. Moreover, the planning mechanism in [20] is incomplete and many rational plans cannot be inferred. In [19], Sadek et al. describe how this theory is used to develop an implemented rational agent engine called ARTIMIS. This technology has been used to build natural language dialogue systems and multiagent applications. Louis [14] recently extended ARTIMIS [19] to incorporate a more general model of planning (state space planning by regression and hierarchical planning) and plan adoption. His framework is more complex than ours and uses defaults (as does Sadek’s). The approach supports multiagent plans and has been implemented. But there is no formalization of epistemically feasible plans, and no success theorem. Commitment to a plan is modelled using a special predicate rather than using the intention attitude.

Although independently motivated, our account closely resembles the one in [13], where a similar notion of commitment to actions was introduced to relate intentions and actions. However, that framework does not model rationality or provide a success theorem. There has also been related work that tries to extend agent programming languages to support declarative goals (e.g. [7]).

Our semantics of communication acts is mentalistic, in contrast to recent social commitment semantics (e.g. [4]). The public social commitment level is obviously important, but we don’t think that communication can be reduced to it. The reason agents communicate is that this serves their private goals. One must usually reason about these goals and the associated beliefs to really understand the agents’ behavior.

The theory presented here is a part of our ongoing research on the semantics of speech acts and communication in the situation calculus. In [8], we present an extended version of our framework where we allow exogenous actions. To deal with these unintended actions, an agent needs to revise the plan it is committed to whenever an exogenous action occurs. In other words, she needs to un-commit from the pre-
We handle the un-commiting part in the SSA of rational plans, and commit to one of them. We consider the new set of previously committed plans, and commit to one of them. We handle the un-committing part in the SSA for one agent is assumed to do planning. In future work, we will try to relax this restriction and to develop tools to support multiagent programming that conform to ECASL.

Our current agent theory is overly simplistic in many ways. One strict constraint that we have is that we do not allow cooperating agents to choose how they will achieve the goals delegated to them by assuming that the planning agent knows the whole plan in advance. Only one agent is assumed to do planning. In future work, we will try to relax this restriction and to model some interaction protocols that involve multiple planning agents. It would also be interesting to try to use this formalization to implement flexible communication agents as in [19] and to develop tools to support multiagent programming that conform to ECASL.

Références


