Coordination of actions in an autonomous robotic system

Erik Sandewall[1]

Department of Computer and Information Science
Linköping University
Linköping, Sweden
erisa@ida.liu.se

Abstract. Robots are autonomous agents whose actions are performed in the real world during a period of time. There are a number of general constraints on such actions, for example that the same action can not have two separate instances during overlapping time intervals, or restrictions that are due to which state variables affect the action or are affected by it. Each process in the robot's cognitive system that is to request the initiation of an action must respect those restrictions. In this article we describe the design and formal characterization of a separate process, called an action coordinator, that manages these restrictions.

1 Topic

The familiar three-level architecture for robotic systems with high-level autonomy is defined in terms of a lower 'process' layer, a middle 'reactive' layer, and an upper 'deliberative' layer. Such an architecture may be natural if the activities of the robotic system are defined in terms of 'actions' with extended duration in time. The deliberative layer will then be in charge of prediction and planning in terms of such actions. It will also be capable of invoking actions through a request to the reactive layer. The current state of the reactive layer at each point in time will specify what are the currently ongoing actions and the current state within the action, at least on a qualitative level. This current state, in turn, defines the operational mode for the control algorithms that are used in the process layer.

In such an architecture there are a number of restrictions on how actions can be performed. The exact nature of these restrictions is an aspect of the semantics for the actions, as represented by a particular logic of actions and change. In the Cognitive Robotics Logic [4], the execution of an action a during and interval of time from s to t is expressed by the formula $\mathsf{D}([s,t],a)$, and one of the restrictions is that if $\mathsf{D}([s,t],a)$ and $\mathsf{D}([s',t'],a)$ hold, then the intervals $[s,t]$ and $[s',t']$ can not overlap in more than one single timepoint, unless they are identical. In other words, we allow $s'=t$ but not $s<s'<t$. There are also a number of other restrictions, for example those having to do with the limitations on when a particular action is executable in itself.

The task of enforcing such restrictions is assigned to the deliberative layer in the classical three-level architecture. For example, this layer must be designed in such a way that it will not invoke an action that is already going on. This means that the deliberative layer must be one coherent entity. The generic three-layer model is therefore not easily compatible with a distributed-AI approach where the deliberative function is organized as a collection of independent agents each of which has the capability of invoking actions. For example, it is not sufficient to just queue the action requests of individual agents, since an agent may wish to retract or add requests if an action can not be executed at the time or in the way that this agent has requested.

In order to accomodate a collection of independent 'agents' that together constitute the deliberative layer of the robotic system, it is natural to separate the enforcement of restrictions on actions into an architectural unit of its own. We shall refer to it as the *action coordinator.* The architecture will now consist of four layers rather than three: the process layer, the reactive layer, the action coordinator, and the swarm of deliberative agents. In procedural terms, every such agent is able to issue action requests to the action coordinator, and the action coordinator will send action initiation commands to the reactive layer if and when it determines that this is appropriate. It will also inform the deliberative agents about whether their requests have been accepted or not, and about the termination of the actions they requested.

The function of the action coordinator is reminiscent of the concept of electronic institutions that has emerged in the field of distributed AI, particularly in the context of auctions and negotiation in a community of agents[1]. One may think of the action coordinator as a kind of electronic institution that is specialized for robotic applications which are characterized by layered architectures and actions with duration.

The procedures surrounding the action coordinator can be understood in terms of message-passing for the purpose of invocation and information. However, in a logicist framework where both the deliberative agents and the actions themselves are characterized in logic, it is appropriate to also use logic for characterizing the action coordinator and its interactions with the agents. This has the advantage that additional, complex behavior can be 'programmed' in a clear and transparent way into the action coordinator. In the present article we shall describe the action coordinator using Cognitive Robotics Logic, CRL [4].

2 Cognitive Robotics Logic

The present section contains the basic definitions for CRL, using the same presentation as in [3] except that the use of composite actions has been removed. We retain the constructs for success and failure of actions for continuity, and in order to lay the ground for future extension of the results presented here, although those constructs will not be used in the present article.

2.1 Standard constructs in logics with explicit time

The following is the basic notation that is generally used, with minor variations, when time and action is represented using an explicit time domain. Three primary predicates are used. The predicates H for *Holds* and D for *Do* are defined as follows. $\mathsf{H}(t, p)$ says that the "propositional fluent" ([1]) p holds at time t. In other words, p is reified and $\mathsf{H}(t, p)$ is the same as $p(t)$ in the case where p is atomic. $\mathsf{D}([s, t], a)$ says that the action a is performed over exactly the closed temporal interval $[s, t]$. Open and semiopen intervals are denoted (s, t), $[s, t)$, and $(s, t]$ as usual.

Non-propositional fluents are also admitted, using the notation $\mathsf{H}(t, f : v)$ where $f : v$ is the propositional fluent saying that the fluent f has the value v. Fluent-valued functions are allowed, and one of their uses is to define fluents for properties of objects. For example, $ageof(p)$ may be the fluent for the age of the person p, used as in $\mathsf{H}(1998, ageof(john) : 36)$.

In the full notation there is also a third predicate, X, that is pronounced *occludes* and is used for characterizing exceptions from the assumption of continuity of the value of fluents. Continuity includes persistence as a special case, for discrete-valued fluents. $\mathsf{X}(s, f)$ expresses that at time s, the value of the fluent f is not required to be continuous or to persist.

In all cases, s and t are timepoints (usually s for starting time and t for termination time) and a is an action. We assume that time is discrete and linear, and let θt represent the timepoint that precedes t. $s < t$ is defined as $s = \theta^n t$ for some $n > 0$.

Logics with explicit time are usually used in such a way that each model of the axioms characterizes one possible history in the world, not a tree of possible histories. Alternative histories are represented by different models([2]). Therefore, a timepoint t is sufficient for identifying the state of the world at time t in the present model.

2.2 Ontology for invocation and success

For each kind of action, in many applications, there are certain conditions that must be satisfied in order to be able to say that the action can be initiated at all. Once initiated, it can either succeed or fail. The distinctions between inapplicability, failure, and success depend both on the application as such, and

[1] We have previously tried to maintain a terminological distinction between *fluent* as a *function* from timepoints to corresponding values, and a *feature* as a formal object that designates a fluent. With that terminology, the p and f that occur in the second argument of H are features, not fluents. Similarly, the functions *inv*, *app*, and *fail* that will be introduced later in this section, are functions from actions to features. However, since it is so common to use the word 'fluent' both for the function and its designator, we follow that practice here.

[2] However, it was shown in [2] that it is straightforward to generalize the time domain so that it also accounts for the case of branching time.

on how one chooses to model it. In this subsection we specify the ontology ([3]) for these concepts which will be used in the formalization in the next section.

An *invocation* of an action can cause it to begin its *execution*, which ends with either *success* or *failure*. The matter is complicated by the requirement to represent that it is sometimes impossible to execute an action. In our approach, invocation of an action is possible at any time, but the invocation does not necessarily lead to the execution of the action. In particular, it does not if the action is inapplicable by definition (for example, turning on the light in a room where there is no light) or if the action is already executing.

When an action begins to execute, it is said to *initiate*. Once an action has (been) initiated, it must ultimately either *succeed* or *fail*. The distinction between success and failure is done on the following pragmatic grounds: planning goal achievement is done using the assumption that actions succeed, and using knowledge about their results when they do succeed. The case where an action fails is dealt with on a case-by-case basis once the failure has occurred.

For the same reasons, we assume that applicability is defined in such a way that it can be determined at planning time. Those conditions that prevent an action from having its effect and that can in general not be detected until execution time, must be modelled as failure and not as inapplicability([4]).

Each action has a temporal duration, which must be an interval that is greater than a single point except for some specific cases defined below. Note, in particular, that when an action is not applicable, it is considered not to execute, it is not considered to fail instantly.

2.3 Syntax for invocation and success

Two representations will be used for the expression of success, failure, and applicability of actions. In one, we use specially constructed fluents, in the other, variants of the D predicate that distinguish between action success and action failure. The former representation is considered as the basic one, and the latter is introduced as abbreviations or 'macros' that facilitate the writing of effect rules for actions.

The following are three functions from actions to propositional fluents:

inv, where $\mathsf{H}(s, inv(a))$ says that the action a is invoked at time s. At all other times, $\mathsf{H}(s, inv(a))$ is false.

app, where $\mathsf{H}(s, app(a))$ says that the action a is applicable at time s.

$fail$, where $\mathsf{H}(t, fail(a))$ says that the action a terminated with failure at time t. $\mathsf{H}(t, fail(a))$ is false at all times when the action is not executing, or when it is executing but not terminating, or when it is terminating successfully.

[3] We mean ontology in the classical sense of the word, not merely a taxonomical structure

[4] To be precise, in a well-formed plan, each action must be applicable in any state of the world that may result, according to the effect laws, if the preceding actions are successful. If some of them fail then later actions may be inapplicable in a way that can only be detected at plan execution time.

In addition, we mention one function from propositional fluents to actions:

test, where $test(p)$ or $test(f : v)$ is an action that is always applicable, whose duration is always instantaneous (expressed by $\mathsf{D}([s, s], test(p)))$, and that satisfies

$$\mathsf{H}(s, fail(test(p))) \leftrightarrow \neg\mathsf{H}(s, p)$$

In other words, $test(p)$ succeeds at time s iff p is true at s.

Notice that this function does not represent an *action*, and questions about the executability of the tests, their possible side-effects, and the precision of the results are not relevant for it. It is simply a kind of conditional operator in the logic([5]).

The following abbreviation is introduced:

$\mathsf{D}_v(s, a)$ for $\mathsf{H}(s, inv(a)) \wedge (\neg\mathsf{H}(s, app(a)) \vee \exists s' \exists t[\mathsf{D}([s', t], a) \wedge s' < s < t])$: the action a is invoked at time t but it is either not applicable, or already executing at that time. (This is the case where invocation of the action does not initiate an execution).

The priority of the propositional connectives is defined so that $a \rightarrow b \wedge c$ means $a \rightarrow (b \wedge c)$.

2.4 Axiomatic characterization

The following set of axioms characterizes the obvious properties of these relations.

S1. If an action is being executed, then it must have been invoked and be applicable and non-executing at invocation time:

$$\mathsf{D}([s, t], a) \rightarrow \mathsf{H}(s, inv(a)) \wedge \neg\mathsf{D}_v(s, a)$$

This implies:

$$\mathsf{D}([s, t], a) \rightarrow \mathsf{H}(s, inv(a)) \wedge \mathsf{H}(s, app(a)) \wedge$$
$$\neg\exists s' \exists t[\mathsf{D}([s', t], a) \wedge s' < s < t]$$

S2. If an action is invoked, then it is executed from that time on, unless it is inapplicable or already executing:

$$\mathsf{H}(s, inv(a)) \rightarrow \exists t[s \leq t \wedge \mathsf{D}([s, t], a)] \vee \mathsf{D}_v(s, a)$$

The full version of this axiom is slightly larger in order to also allow for composite actions.

S3. An action can not take place during overlapping intervals:

$$\mathsf{D}([s, t], a) \wedge \mathsf{D}([s', t'], a) \wedge s \leq s' < t \rightarrow s = s' \wedge t = t'$$

[5] Actually, the *test* operator is mostly motivated for its use in composite action expressions where it makes it possible to define conditional actions. The use of composite actions is excluded in the present article, but we retain the definitions for the *test* function anyway since it is integrated with the basic axioms.

S4,S5. Actions of the form $test(p)$ are always applicable, and instantaneous:

$$\mathsf{H}(s, app(test(p)))$$

$$\mathsf{D}([s,t], test(p)) \rightarrow s = t$$

S6. All other actions execute over extended periods of time: never immediately, except for actions of the form $test(p)$:

$$\mathsf{D}([s,t], a) \rightarrow s < t \vee \exists p[a = test(p)]$$

S7. Actions only fail at the end of their execution:

$$\mathsf{H}(t, fail(a)) \rightarrow \exists s[\mathsf{D}([s,t], a)]$$

S8. Definition of success for actions of the form $test(a)$:

$$\mathsf{D}([s,s], test(p)) \rightarrow (\mathsf{H}(s, fail(test(p))) \leftrightarrow \neg\mathsf{H}(s,p))$$

Several of these axioms capture desirable properties directly. For others, all the consequences are not immediately obvious. One useful consequence is the following theorem, previously reported in [3]:

Theorem 1. *In any model for the axiom S3, let $\{[s_i, t_i]\}_i$ be the set of all intervals such that $\mathsf{D}([s_i, t_i], a)$ for a specific action a. Then there is some ordering of these intervals such that $s_i < s_{i+1}$ and $t_i \leq s_{i+1}$ for all i.*

Proof. Suppose the proposition does not hold, and choose an order of the pairs such that $s_i \leq s_{i+1}$, and where each pair only occurs once. Also, choose j so that either $s_j = s_{j+1}$, or $s_j < s_{j+1} < t_j$. If no such j is to be found, then the ordering already satisfies the condition in the proposition.

However, the case $s_j = s_{j+1}, t_j \neq t_{j+1}$ contradicts axiom (S3). The case $s_j < s_{j+1} < t_j$ also contradicts axiom (S3). This concludes the proof. QED.

The value of this observation is that through it, it makes sense to use the fluent $fail(a)$ for characterizing the success or failure of an action with extended duration. If theorem 1 were not to hold, then it would not be clear from $\mathsf{H}(t, fail(a))$ which invocation the failure referred to. This consideration is also the reason for the choice manifested in axiom S1: if an action a is invoked while it is already in the midst of executing, then it is not represented as "failing", since this would confuse matters with respect to the already executing instance. Instead, we use the convention that it is invoked, possibly applicable, but it does not get to execute from that starting time.

We also obtain at once:

Theorem 2. *In any model for the axioms S1 – S8, if $\mathsf{D}([s,t], a)$ and $\mathsf{H}(u, fail(a))$ for some u in $(s,t]$, then $t = u$.*

Informally, we can think of each model in dynamical terms as a possible history in the world being described, and what this theorem says is that if an action is invoked and begins to execute, then if $\mathsf{H}(u, fail(a))$ becomes true at some timepoint u during the execution, the action halts and ends with failure, and if it is able to proceed until its normal ending without $\mathsf{H}(u, fail(a))$ becoming true at any time, then it ends with success.

Any use of this logic will naturally be concerned with the effects of actions. In Cognitive Robotics Logic and its background, the Features and Fluents approach, as well as its successors, this is specified using action laws, which in particular make use of the occlusion predicate, and in combination with assumptions of persistence. In [6] we showed how this logic can be used for specifying an architecture for a logic-based cognitive robotic system where rules specifying failure conditions for an action can be written as implications where the consequent has the form $\mathsf{H}(u, fail(a))$.

2.5 Examples

The following additional abbreviations are introduced. They are generally useful for writing effect rules and applicability restrictions rules for actions.

$\mathsf{G}(s, a)$ for $\mathsf{H}(s, inv(a))$: the action a is invoked ("go") at time s

$\mathsf{A}(s, a)$ for $\mathsf{H}(s, app(a))$: the action a is applicable at time s

$\mathsf{D}_\mathrm{s}([s, t], a)$ for $\mathsf{D}([s, t], a) \wedge \neg\mathsf{H}(t, fail(a))$: the action a is executed successfully over the time interval $[s, t]$, it starts at time s and terminates with success at time t.

$\mathsf{D}_\mathrm{f}([s, t], a)$ for $\mathsf{D}([s, t], a) \wedge \mathsf{H}(t, fail(a))$: the action a is executed but fails over the time interval $[s, t]$, it starts at time s and terminates with failure at time t.

$\mathsf{D}_\mathrm{c}([s, t], a)$ for $\exists u[\mathsf{D}([s, u], a) \wedge t \leq u]$: the action a is being executed, the execution started at time s and has not been terminated before time t. (It may terminate at t or later).

For both D_s and D_f, s is the time when the action was invoked, and t is the exact time when it concludes with success or failure.

As an example of the use of this notation, the following formula states that a condition φ guarantees that an action always succeeds:

$$\mathsf{H}(s, \varphi) \wedge \mathsf{G}(s, a) \rightarrow \exists t[\mathsf{D}_\mathrm{s}([s, t], a)]$$

Ordinary action laws specify the action's effects when it succeeds. They are therefore written as usual and with D_s on the antecedent side: if preconditions apply and the action is performed successfully, then the postconditions result.

As a second example, consider the case of actions that are described in terms of a precondition, a prevail condition, and a postcondition, where the postcondition is at the same time the termination condition for the action [7]. The prevail condition must be satisfied throughout the execution of the action; if it is violated then the action fails. Simple pre/ post/ prevail action definitions can be expressed as follows, if φ_a is the precondition of the action a, ω_a is the postcondition, and ψ_a is the prevail condition:

$$\mathsf{A}(s,a) \leftrightarrow \mathsf{H}(s,\varphi_a)$$
$$\mathsf{D}_\mathsf{s}([s,t],a) \rightarrow \mathsf{H}(t,\psi_a) \wedge \mathsf{H}(t,\omega_a)$$
$$\mathsf{A}(s,a) \wedge \mathsf{D}_\mathsf{c}([s,t],a) \wedge u \in [s,t) \rightarrow \mathsf{H}(u,\psi_a) \wedge \neg\mathsf{H}(u,\omega_a)$$
$$\mathsf{D}_\mathsf{c}([s,t],a) \wedge \neg\mathsf{H}(t,\psi_a) \rightarrow \mathsf{D}_\mathsf{f}([s,t],a)$$

The traditional case of only pre- and postconditions is easily obtained by selecting ψ_a as tautology.

3 The Action Coordinator

We now proceed to using the Cognitive Robotics Logic for specifying the action coordinator as described in section 1. We limit the problem to a relatively simple action coordinator that does not take the success and failure of actions into account.

3.1 CRL Formulation

Starting from the CRL formalism that was introduced in the previous section, we define the action coordinator by extending the logic with an additional object domain for agents, and with two new fluent-valued functions, *inva* and *asta*. The fluent $inva(g,a)$ will express an invocation of the action a by the agent g (i.e., a request for its initiation). The proposition $\mathsf{H}(s, inv(g,a))$ is true iff the agent g invokes a at time s.

Notice the difference between $inv(a)$ that was introduced above, and $inva(g,a)$ that is introduced here. The relation between them will be specified and proved below.

The action coordinator's response to an invocation is represented using the funtion *asta*, for 'action state'. At each point in time, the fluent $asta(g,a)$ has a value representing the current response to an invocation of a that has previously been issued by the agent g. The invocation is represented as a momentary condition, but the response is represented as something that applies over an interval of time. This frees the agents from 'remembering'([6]) what responses they have received for their invocations.

The value of $asta(g,a)$ shall be one of the following discrete values. If the action is executed during the interval $[s,t]$ on behalf of the agent g, then the value is stex (for 'start executing') at initiation time s, ex (for 'executing') in the interior of the interval, and nil at time t. Before any such execution has taken place, the value is also nil. If an invocation $inva(g,a)$ occurs when the value of $asta(g,a)$ is nil, then it switches to pend, for 'pending'. It may retain this value for some time, but it can also switch to either of stex meaning that the invocation was honored by initiating the action, or to ref meaning that the invocation was

[6] I.e., from having to retain that information in its local state.

refused. During and after execution the value is ex and nil as already explained, and after execution the action is available for initiation again. If the action has been refused, on the other hand, a renewed invocation $inva(g, a)$ will change the value of $asta(g, a)$ from ref to pend.

The intended structure of possible transitions is illustrated in figure 1. Notice that the stex state can only be visited during one single timestep at a time, whereas all the other states can remain for several timesteps.

$$\begin{tikzcd}
\texttt{ref} \arrow[r, shift right] & \texttt{pend} \arrow[l, shift right] \arrow[r] & \texttt{stex} \arrow[dl] \arrow[d] \\
 & \texttt{nil} \arrow[u] & \texttt{ex} \arrow[l]
\end{tikzcd}$$

Figure 1: State transitions for asta(g,a)

The transition from one value to another depends on the following factors. The transitions from pend to stex or ref represent the decisions of the coordinator. The transitions from stex to ex and from ex to nil, or directly from stex to nil reflect the execution of the actions, and normally they are obtained from the process layer where each action is executed. The transitions from nil or ref to pend represent how the action coordinator receives and administrates invocations of actions by the agents.

This transition structure is intended to represent an upper bound on admissible transitions. Specific policies in the action coordinator can be represented by restricting the transitions from pend to stex or ref, but not by relaxing any of the transitions described here.

However, one extension that may be of interest is to have a way for agents to discontinue an ongoing action. In this case the transition from ex to nil is caused by a message to the action coordinator from a deliberative agent, and not from the process layer. This requires an extension to the formalism for expressing how the agent sends that message, and it is not considered in the present article.

3.2 Axioms for the CRL Formulation

The system of transition rules can be expressed using the following axioms. Recall that the function θ represents the predecessor of a given timepoint, and $s < t$ represents that $s = \theta^n t$ for some $n > 0$. We introduce the auxiliary predicate $Blocked(s, a)$ that characterizes those conditions where an action can not initiate even if some agent invokes it. It is formally defined as follows:

NS1. $Blocked(s, a) \leftrightarrow \neg \mathsf{H}(s, app(a)) \lor \exists g[\mathsf{H}(s, acta(g, a) : \mathsf{ex})]$

The following axioms characterize the generic action coordinator.

K0. Fluents of the form $acta(g, a)$ take (at most) one value at each point in time, chosen among the five values mentioned above.

$$\mathsf{H}(s, acta(g, a) : r) \land \mathsf{H}(s, acta(g, a) : r') \rightarrow$$
$$r = r' \land (r = \mathsf{ex} \lor r = \mathsf{pend} \lor r = \mathsf{nil} \lor r = \mathsf{ref} \lor r = \mathsf{stex})$$

We suppose that $acta(g, a)$ also *has* a value for each combination of s, g, and a, but an axiom to this effect does not appear to be needed for the proofs being made below.

K1. If an action is being executed, then it must have been initiated, and from the point of view of each initiating agent it goes through the states stex, ex, and nil:

$$\mathsf{D}([s, t], a) \rightarrow s < t \land \exists g[\mathsf{H}(s, acta(g, a) : \mathsf{stex})] \land$$
$$\forall g[\mathsf{H}(s, acta(g, a) : \mathsf{stex}) \rightarrow \mathsf{H}(t, acta(g, a) : \mathsf{nil}) \land$$
$$\forall u[s < u < t \rightarrow \mathsf{H}(u, acta(g, a) : \mathsf{ex})]]$$

K2. If an action initiates, then it is executed from that time on and in a finite interval of time, so that it has an ending time. It can only initiate if it is applicable at that point in time:

$$\mathsf{H}(s, asta(g, a) : \mathsf{stex}) \rightarrow \mathsf{H}(s, app(a)) \land \exists t[\mathsf{D}([s, t], a)]$$

K3. If an action is executing and not initiating from the point of view of an agent, then it must have been in that state or initiating in the preceding timepoint with respect to the same agent:

$$\mathsf{H}(s, asta(g, a) : \mathsf{ex}) \rightarrow \mathsf{H}(\theta s, asta(g, a) : \mathsf{ex}) \lor \mathsf{H}(\theta s, asta(g, a) : \mathsf{stex})$$

K4. An action can only initiate for an agent if it was pending for that agent at the preceding timepoint:

$$\mathsf{H}(s, acta(g, a) : \mathsf{stex}) \rightarrow \mathsf{H}(\theta s, asta(g, a) : \mathsf{pend})$$

K5. If an action is executing for one agent, or if it is inapplicable, then it can not be initiated for any agent:

$$Blocked(s, a) \rightarrow \forall g[\neg \mathsf{H}(s, asta(g, a) : \mathsf{stex})]$$

K6. Consider an action a that is inert (nil) or refused for a particular agent at a particular timepoint. If it is invoked by the agent then it must be pending at the next timepoint, otherwise it must retain the same value.

$$\mathsf{H}(\theta s, asta(g, a) : r) \land (r = \mathsf{nil} \lor r = \mathsf{ref}) \rightarrow$$
$$(\mathsf{H}(s, inva(g, a)) \rightarrow \mathsf{H}(s, asta(g, a) : \mathsf{pend})) \land$$
$$(\neg \mathsf{H}(s, inva(g, a)) \rightarrow \mathsf{H}(s, asta(g, a) : r))$$

K7. An action can only switch from another state to being pending as the result of an invocation from the agent in question:

$$(\mathsf{H}(s, asta(g, a) : \mathsf{pend}) \land \mathsf{H}(\theta s, asta(g, a) : r) \land r \neq \mathsf{pend}) \rightarrow \mathsf{H}(s, inva(g, a))$$

K8. If an action is pending from the point of view of an agent, then in the next timestep it must be initiated, refused, or still pending:

$\mathsf{H}(\theta s, asta(g,a) : \mathsf{pend}) \rightarrow$
$\quad \mathsf{H}(s, asta(g,a) : \mathsf{stex}) \vee \mathsf{H}(s, asta(g,a) : \mathsf{ref}) \vee \mathsf{H}(s, asta(g,a) : \mathsf{pend})$

K9. There exists a timepoint s_0 such that $asta(g,a)$ has the value nil for all times $\leq s_0$:

$\exists s_0 \forall s, g, a[s \leq s_0 \rightarrow \mathsf{H}(s, asta(g,a) : \mathsf{nil})]$

We also introduce the following policy rule.

P1. If an action is pending and applicable, then initiate it:

$\mathsf{H}(\theta s, asta(g,a) : \mathsf{pend}) \wedge \neg Blocked(s,a) \rightarrow \mathsf{H}(s, acta(g,a) : \mathsf{stex})$

This policy rule is the first example of a rule that restricts the transitions for an action from being pending, to being initiated or refused. This particular rule *forces* a pending action to initiate as soon as it is not blocked by not being applicable, or by another instance of the same action being executed. One can of course think of alternative rules that instead require the initiation to be delayed or refused in specific circumstances. It is intended that rules K0 through K9 shall remain fixed, whereas policy rules can be exchanged.

3.3 Properties of the CRL Formulation

The logical structure that was defined in the previous subsection allows for a number of interesting cases. In particular, consider a situation where two separate actions are pending and become unblocked at the same time, but where it is not possible to execute them concurrently. With the axioms shown above, including the policy rule P1, both will be initiated. We take the view that in this case, one should set things up so that both actions do execute, but at least one of them will fail, possibly after only one timestep. Although this convention may seem peculiar at first, please notice that the conflict between two concurrent actions may also arise later on during their execution, and it may be due to external events that could hardly have been predicted when the actions started. Since we anyway have to accomodate actions that fail for such reasons in the course of their execution, we can as well represent the starting-time conflict in the same way.

The representation shown above allows one to express that each occurrence of an action is done on the request of one or more agents. There must be at least one agent for which it is being performed, according to axiom K1. If an action a is pending for more than one agent g and then becomes unblocked at a particular timepoint s, then the policy axiom P1 requires that the action initiates for all those agents at the same time. However, if P1 is not used then it is possible to initiate the action for some of the invoking agents but not for all of them, or to not initiate it at all.

The rules K0 through K9 characterize the action coordinator in a number of ways. Rules K1, K2, and K3 specify how the execution of an action, as expressed using the $\mathsf{D}()$ predicate, is controlled by the action states as represented by $acta(g, a)$. Rules K6 and K7 specify how those action states interact with the messages from the cognitive agents, as expressed using the $inva(g, a)$ fluents. Axioms K0, K4, K5, K8 and K9 specify the permitted values and permitted transitions for the action states, although the other axioms also imply some restrictions on those transitions.

We shall show below that the 'K' series axiomatization in axioms K0 through K9 restricts the fluent $acta(g, a)$ to the finite-state automaton that was informally described in a previous subsection. However, we first prove some other results since along the way they provide a needed lemma. Notice that the axioms do not *only* represent the automaton; they also characterize the use of multiple invoking agents and the relationship between their invocations.

3.4 Relation to previous formulation

We shall now demonstrate that the axioms S1, S2, S3, and S6 that were defined above follow from the proposed axioms for the action coordinator, including the policy axiom P1. In doing so we achieve two objectives. First, the action-based behavior that was described by the 'S' series axioms is replaced by a more finegrained machinery that is arguably a more precise description of how the deliberative layer works in an intelligent autonomous agent. Secondly, we have verified that the proposed specification for the new, four-layer architecture with distributed cognitive capabilities is consistent with the logical architecture that had been introduced before.

In order to properly relate the old and the new axiomatization, we shall need an axiom that relates fluents of the form $inv(a)$ to the constructs used in the new axioms. Since in the 'S' series axioms, an action initiates if and only if $inv(a)$ holds at a timepoint where the action is applicable and not already executing, we adopt the following axiom:

NS2. $\mathsf{H}(s, inv(a)) \leftrightarrow \exists g[\mathsf{H}(\theta s, acta(g, a) : \mathsf{pend})]$

Using this definition as the bridge, we shall show that the axioms S1, S2, S3 and S6 in the old set of axioms can be obtained as consequences of the new set of axioms, and in particular axioms K0 through K5 plus K9 and the policy axiom P1. Notice that P1 is necessary here, since the 'S' series axiomatization prescribed that an invoked action shall start executing as soon as it is not blocked. Axioms K6 through K8 will not be needed for these proofs, which is not surprising since they represent the decision machinery for the agent coordinator.

The functions $test$ and $fail$ are outside this consideration, so that axioms S4, S5, S7, and S8 are not to be treated. Also, axiom S6 is modified by removing the reference to the $test$ function, becoming

S6'. All actions execute over extended periods of time:

$\mathsf{D}([s,t],a) \rightarrow s < t$

We notice at once that S6' is subsumed by the new axiom K1, and proceed with the others.

The following is the definition for the abbreviation $\mathsf{D}_v(s,a)$ that was introduced above, for reference:

NS3. $\mathsf{D}_v(s,a) \leftrightarrow$
$\mathsf{H}(s, inv(a)) \wedge (\neg\mathsf{H}(s, app(a)) \vee \exists s'\exists t'[\mathsf{D}([s',t'],a) \wedge s' < s < t'])$

We begin with a few lemmas.

Lemma 1. $\mathsf{D}([s,t],a) \rightarrow \mathsf{H}(s, inv(a))$

Proof. Assume $\mathsf{D}([s,t],a)$. By K1, $\exists g[\mathsf{H}(s, acta(g,a) : \mathsf{stex})]$. By K4, $\exists g[\mathsf{H}(\theta s, acta(g,a) : \mathsf{pend})]$. By NS2, $\mathsf{H}(s, inv(a))$. QED.

Lemma 2. $\mathsf{D}([s,t],a) \rightarrow \neg Blocked(s,a)$

Proof. Assume $\mathsf{D}([s,t],a) \wedge Blocked(s,a)$. By K1 from $\mathsf{D}([s,t],a)$, there is some g such that $\mathsf{H}(s, acta(g,a) : \mathsf{stex})$. According to K5 this contradicts $Blocked(s,a)$. QED.

Lemma 3. $\exists g[\mathsf{H}(s, acta(g,a) : \mathsf{ex})] \leftrightarrow \exists s', t'[\mathsf{D}([s',t'],a) \wedge s' < s < t']$

Proof. The right to left direction of the implication follows directly from K1. For the left to right direction, assume $\mathsf{H}(s, acta(g,a) : \mathsf{ex})$. It follows from K3 that there are preceding timepoints from s and back where the value of $acta(g,a)$ is ex, until it arrives to one s' where the value is stex, so that $\mathsf{D}([s',t'],a)$ for some t', according to K2. Such a timepoint s' must exist according to axiom K9. According to K1 it must be the case that $s' < t'$ and $s < t'$. QED.

Using Lemma 3 the definition of $\mathsf{D}_v()$ can be rewritten as
$\mathsf{D}_v(s,a) \leftrightarrow \mathsf{H}(s, inv(a)) \wedge Blocked(s,a)$.

We proceed now to the proofs of propositions S1, S2, and S3 which had the status of axioms in the earlier articles.

Proposition S1. $\mathsf{D}([s,t],a) \rightarrow \mathsf{H}(s, inv(a)) \wedge \neg\mathsf{D}_v(s,a)$

Proof. Lemmas 1 and 2 give $\mathsf{D}([s,t],a) \rightarrow \mathsf{H}(s, inv(a)) \wedge \neg Blocked(s,a)$. By tautology, $\mathsf{D}([s,t],a) \rightarrow \mathsf{H}(s, inv(a)) \wedge (\neg\mathsf{H}(s, inv(a)) \vee \neg Blocked(s,a))$ which is equivalent to proposition S1 using the definition of $\mathsf{D}_v()$ as rewritten above. QED.

Proposition S2. $\mathsf{H}(s, inv(a)) \rightarrow \exists t[s \leq t \wedge \mathsf{D}([s,t],a)] \vee \mathsf{D}_v(s,a)$

Proof. Assume $\mathsf{H}(s, inv(a))$. By the definition of inv, $\exists g[\mathsf{H}(\theta s, acta(g,a) : \mathsf{pend})]$. Policy axiom P1 gives $\mathsf{H}(s, acta(g,a) : \mathsf{stex}) \vee Blocked(s,a)$, and K2 gives $\exists t[\mathsf{D}([s,t],a)] \vee Blocked(s,a)$. Axiom K1 then gives $\exists t[\mathsf{D}([s,t],a) \wedge s \leq t] \vee Blocked(s,a)$. The assumption gives $\exists t[\mathsf{D}([s,t],a) \wedge s \leq t] \vee (\mathsf{H}(s, inv(a)) \wedge Blocked(s,a))$ and the rewritten definition of $\mathsf{D}_v()$ concludes the proof. QED.

Proposition S3. $\mathsf{D}([s,t],a) \wedge \mathsf{D}([s',t'],a) \wedge s \leq s' < t \rightarrow s = s' \wedge t = t'$.

Proof. Assume $\mathsf{D}([s,t],a) \wedge \mathsf{D}([s',t'],a) \wedge s \leq s' < t$. Furthermore, for the purpose of proof by contradiction, assume $s < s'$.
K1 obtains $\exists g[\mathsf{H}(s', acta(g,a) : \mathsf{ex})]$. From NS1 it follows $Blocked(s',a)$.
However K1 also implies $\exists g'[\mathsf{H}(s', acta(g',a) : \mathsf{stex})]$, which according to K5 is a contradiction. Therefore $s = s'$.

Next, assume $t < t'$. From $\mathsf{D}([s,t],a)$ and using K1, $\mathsf{H}(t, acta(g,a) : \mathsf{nil})$ follows. From $\mathsf{D}([s,t'],a)$ and using K1, it follows $\mathsf{H}(t, acta(g,a) : \mathsf{ex})$. According to K0 this is a contradiction. If $t' < t$ then the same contradiction is obtained due to symmetry. It follows that $t = t'$. QED.

3.5 Finite-state characterization of action state

We return now to the finite-state characterization of action state.

Theorem 3. In any model for the axioms specified above, the sequence of values that are assigned by $\mathsf{H}()$ to $asta(g,a)$ for given g and a and for successive s, must be restricted to the state transitions that are shown in figure 1. It can stay in the same state for several steps in time, except for state stex where it can only stay for one step in time.

Proof. Axiom K9 specifies that for initial timepoints the value must be nil. In a given model satisfying the axioms, and for given g and a there, consider the set of all intervals $[s_i, t_i]$ such that $\mathsf{D}([s_i,t_i],a)$ holds and $\mathsf{H}(s_i, asta(g,a) : \mathsf{stex})$. We have already proved that these intervals must be disjoint (proposition S3). Within each of these intervals the transitions in figure 1 are satisfied according to axiom K1. Furthermore, at the endpoint t_i in each of those intervals the value is nil.

Consider now the intervals from the ending time t_i of one interval in this set, to the starting time s_{i+1} of the next interval. The state must be nil at t_i and stex at s_{i+1}. According to axiom K4 it must be pend at θs_{i+1}. Consider now the possible state sequences from nil to pend. Within that interval it can not take the value stex because in that case axiom K2 contradicts the construction. It can also not be ex for the same reason, using lemma 3. It is therefore restricted to nil, ref, and pend, according to axiom K0. A transition from pend to nil would violate axiom K8. Also, transitions between ref and nil in either direction would violate axiom K6. The remaining transitions between these three values are allowed by the diagram.

The timepoints before the first action interval and after the last one satisfy the same restrictions. This concludes the proof. QED.

4 Additional facilities in the action coordinator

The specification of the action coordinator in the previous section provides a simple-minded one that just invokes actions as they are requested while respecting a minimal set of restrictions. It can be programmed by the proper choice of policy rules and by axioms specifying the applicability fluent $app(a)$. The definitions for $app(a)$ are of course domain specific. They were not an issue in the proofs in the previous section since applicability is used in the same way in the 'S' series and the 'K' series of axioms.

The general notion of an action coordinator strongly suggests that a number of other facilities can also be included in it, in particular:

– The use of composite actions, where the action coordinator is in charge of invoking successive sub-actions in a composite action. Plans can be seen as composite actions.
– The definition of goal-directed behavior, where the action coordinator is able to represent the relation between a goal and a sequence of actions that is supposed to lead to that goal. If one of the actions in the sequence fails then the action coordinator should identify a revised plan and start executing it instead.
– A more detailed description of the execution of individual actions in the robot's physical world, for example by relating the logical description of actions to their quantitative description using difierential equations.

We have addressed each of these topics in some earlier articles [3, 5, 6]. It now seems possible that the concept of an action coordinator can provide a unifying framework within which these and other aspects of deliberative behavior can be addressed in a coherent way.

Acknowledgements

The initial ideas for this work occurred during a two-week visit to the IIIA in Barcelona at the invitation of Ramón Lopez de Mantaras. I take this opportunity to thank him and his colleagues there for their hospitality and for many stimulating discussions.

References

1. Pable Noriega and Carles Sierra. Towards layered dialogical agents. In *Proc. ECAI Workshop on Agents, Theories, Architectures and Languages (ATAL'96)*, pages 69–81. Springer Verlag, 1996.
2. Erik Sandewall. *Features and Fluents.* Oxford University Press, 1994.

3. Erik Sandewall. Relating high-level and low-level action descriptions in a logic of actions and change. In Oded Maler, editor, *Proc. of International Workshop on Hybrid and Real-Time Systems*, pages 3–17. Springer Verlag, 1997.
4. Erik Sandewall. Cognitive robotics logic and its metatheory: Features and fluents revisited. *Electronic Transactions on Artificial Intelligence*, 2:307–329, 1998.
5. Erik Sandewall. Logic-based modelling of goal-directed behavior. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *Proc. of Conference on Principles of Knowledge Representation and Reasoning*, pages 304–315, 1998.
6. Erik Sandewall. Use of cognitive robotics logic in a double helix architecture for autonomous systems. In Michael Beetz, Joachim Hertzberg, Malik Ghallab, and Martha Pollack, editors, *Advances in Plan-Based Control of Robotic Agents*, pages 226–248. Springer Verlag, 2001.
7. Erik Sandewall and Ralph Rönnquist. A representation of action structures. In *Proc. of [U.S.] National Conference on Artificial Intelligence*, pages 89–97, 1986.