Understanding Cooperation: an Agent's Perspective

Andreas Lux
DFKI *
PO-Box 2080
67608 Kaiserslautern
Germany
lux@dfki.uni-kl.de

Donald Steiner
Siemens AG, c/o DFKI
PO-Box 2080
67608 Kaiserslautern
Germany
steiner@dfki.uni-kl.de

Abstract

The Multi-agent Environment for Constructing Cooperative Applications (MECCA) is based upon a framework unifying the internal behavior of agents and cooperation among agents. This paper presents a formalized view of agent behavior relying on the basic loop of goal activation, plan execution and scheduling followed by task execution. This allows for a presentation of the semantics of cooperation primitives: interagent messages supporting cooperation, comprised of speech acts operating upon objects occuring in the basic loop. The formal semantics of cooperation primitives gives a meaning to individual messages, independent from the cooperation protocol. Thus, agents can reason about exchanged messages and are able to dynamically create their own methods for cooperation.

1 Introduction

The Multi-agent Environment for Constructing Cooperative Applications (MECCA)[1] establishes a basic loop guiding the behavior of agents incorporating reactivity and deliberation (Steiner et al. 1993). As all agents act according to this loop, cooperation can also be based upon it. Messages are fundamental for supporting cooperation in MECCA; their content is clearly important for successful cooperation. This content is determined not only by the agents' local behavior, but also by the previous messages and the expected direction the conversation will take. Human conversations have been extensively analyzed; speech act theory arose as a result (Austin 1962; Searle 1969), classifying messages into a variety of types. Attention has been directed towards embedding (illocutionary) speech acts in planning theory (Grosz & Sidner 1990), but has, up to now, been limited to the speech acts and not the entire content of the message.

We argue that a sender of a message should have a sufficient model of the recipient to know the recipient's behavior upon receipt of the message. Exactly this model is what determines which message the sender sends and, correspondingly, the entire cooperative process. Thus, sender and receiver must agree upon the meaning (semantics) of the exchanged messages. Furthermore, the semantics of the speech acts vary according to the objects under discussion - the proposal of a goal has a different meaning than the proposal of a plan to achieve the goal. Only until these semantics have been determined, can agents reason about the messages they send and receive to influence the cooperation.

In this paper, we present the formal representations of the basic loop determining both agent behavior as well as cooperation among agents.

2 The Agent Model

An agent is the central computational entity, which serves as an explicit model of all entities participating in a cooperation.[2] It can be decomposed into the following three components: the functional, task-solving component (*agent body*), the cooperative superstrate (*agent head*), and the communication functionality (*agent communicator*) (Steiner, Mahling & Haugeneder 1990).

We assume that functional interfaces to an agent's body and communicator are given and concentrate here on a formal description of the agent's head. The resulting cooperative interaction among agents will be examined in Section 3.1. The agent behaves according to a basic *agent loop* (Steiner et al. 1993) which consists of the four phases of *goal activation*, *planning*, *scheduling* (temporal synchronization) and *execution/evaluation*.

2.1 Formalization of the Agent Model

An agent is formally defined by sets and by functions which operate on these sets. According to the above mentioned phases, the following sets are fundamental to an agent A.

W_A a set of possible world states of A; the current world state of A is denoted $W_{A,c}$.

*German Research Center for Artificial Intelligence

[1]This work has been supported by Siemens as part of the KIK-Teamware and CoMMA-Plat projects

[2]both human and machine agents

$\mathcal{A}_A$ a set of actions A can perform; actions have preconditions, which must hold in $\mathcal{W}_c$ before execution, procedures, which are function calls to either body, head or communicator, and effects, which alter the world state $\mathcal{W}_c$; actions are primitive, non-decomposable plans.

$\mathcal{G}_A$ a set of A's active goals; an active goal $g \in \mathcal{G}_A$ is represented by a particular world state, i.e. $\mathcal{G}_A \subseteq \mathcal{W}_A$.

$\mathcal{P}_A$ a set of hypothetical plan sets for A's goals; plans are similar to actions in that they have preconditions and effects on the world state $\mathcal{W}_c$; their procedures, however, are compositions of actions and other plans; formally, $\mathcal{P}_A = \bigcup_{g \in \mathcal{G}_A} \{\mathcal{P}_g\}$, where $\mathcal{P}_g$ is a set of hypothetical plans for g.

$\mathcal{S}_A$ a set of plans to whose execution A is committed; thus, $\mathcal{S}_A \subseteq \bigcup \mathcal{P}_A$.

$\mathcal{F}_A$ a set of functions which are used by A to activate goals (*new_goal*), to find plans (*find_all_plans*), to perform actions (*perform*) etc.; these functions are used to control A's basic loop; the functions might be different among agents.

Thus, an agent A is formally defined by the 6-tuple $A = (\mathcal{W}_{A,c}, \mathcal{G}_A, \mathcal{P}_A, \mathcal{S}_A, \mathcal{A}_A, \mathcal{F}_A)$, where $\mathcal{W}_{A,c} \in \mathcal{W}_A$.[3]

In the following, we formally describe an agent's basic loop based upon the aforementioned sets. The functions in $\mathcal{F}$ are described briefly, with more detail to follow in Section 3.1 where the cooperative aspects are considered.

Goal Activation Assume a set of current goals $\mathcal{G}$ is given. The set is extended with the goal g by the function *new_goal:* $\mathcal{W} \times \{\mathcal{G}\} \rightarrow \mathcal{G}$. According to the current world state $\mathcal{W}_c$, the function *new_goal* selects a world state g which is inserted into the set of active goals $\mathcal{G}$. *new_goal* can be triggered by the agent itself as well as by an external event, e.g. receipt of a message from another agent or effect of the execution of another agent's action. Algorithmically, this phase of the basic loop can be described by:[4]

```
while (g := new_goal(W_c, G)) do
    G := {g} ∪ G
od.
```

Planning The planning approach pursued (Burt 1995) can be characterized as follows: Cooperative plans are built by hypothetical reasoning on the basis of a temporal knowledge representation. Hypothetical reasoning is done by means of abduction (Shanahan 1989); event calculus (Kowalski & Sergot 1986; Eshghi 1988) is used for knowledge representation. Thus, a plan p is a set of events. An event e of a plan p is defined by an instantiated action, a partial ordering, the executing agent and a set of constraints on necessary resources.[5]

$$e = (a_e, ag_e, V_e, N_e, s_e, ((r_1, C_1), \ldots, (r_n, C_n)))$$

with
a_e = action to be executed
ag_e = executing agent
V_e = set of direct predecessor events
N_e = set of direct successor events
s_e = status of the event,
$s_e \in \{hyp, committed, done\}$
r_i = i-th resource
C_i = set of constraints on the i-th resource, possibly empty

The agent function *plan_goal* : $\{\mathcal{G}\} \times \mathcal{W} \rightarrow \mathcal{G}$ selects a goal $g \in \mathcal{G}$ from the set of active goals. The function *find_all_plans* : $\mathcal{G} \times \{\mathcal{A}\} \times \{\mathcal{S}\} \times \mathcal{W} \rightarrow \mathcal{P}$ returns a set of hypothetical plans $\mathcal{P}_g$ which will achieve the goal $g \in \mathcal{G}$, from the world state W_c, ensuring that the plans are compatible with the already scheduled events from $\mathcal{S}$. Knowledge about the post- and preconditions of actions is contained in A. The planning phase can be described by:

```
while (g:= plan_goal(G, W_c)) do
if P_g := find_all_plans(g, A, S, W_c))
then P := P_g ∪ P;
fi;
od.
```

Scheduling The function *schedule_plan* : $\{\mathcal{P}\} \times \mathcal{W} \rightarrow \mathcal{P}$ selects a set of alternative plans $\mathcal{P}_g$ for the goal g from the set of hypothetical plans. Using a cost function *cost* : $\{\mathcal{A}\} \times \mathcal{W} \times \{\mathcal{S}\} \times \mathcal{P} \rightarrow \Re$[6] the function *find_optimal_plan* : $\mathcal{P} \times \{\mathcal{A}\} \times \{\mathcal{S}\} \times \mathcal{W} \rightarrow \mathcal{P}$ calculates the "cheapest" plan p_g^* out of this set. In deciding the overall utility of the plan, not only costs for necessary ressources should be taken into account but also cooperative aspects like e.g. the confidence in partner agents.

After that, the plan p_g^* is scheduled into the global schedule $\mathcal{S}$ of the agent, thereby updating information about predecessor and successor events. A successful scheduling corresponds to a commitment of the agent to the plan; the status of the events of the plan is modified to *committed*. p_g^* must be removed from $\mathcal{P}$, and the other sets in $\mathcal{P}$ must be updated so that they only contain plans which are still correct with the new schedule S. We thus ensure that $\mathcal{P}$ always contains

[3] Where it is unambiguous, an agent's index is omitted, i.e. $\mathcal{W} = \mathcal{W}_A, \mathcal{G} = \mathcal{G}_A$, etc.

[4] The formal description of the agent loop is based on a mixture of logic and programming language-like notation. As e.g. in Lisp it is assumed that failure of a function (= boolean value *false*) returns *nil* and that each value not *nil* corresponds to boolean value *true*.

[5] The executing agent can be seen as a "special" kind of resource.

[6] $\Re$ is the set of real numbers

correct plans. Algorithmically, this can be described as follows:

```
while (P_g := schedule_plan(P, W_c)) do
  p*_g := find_optimal_plan(P_g, min_< P_g);[7]
  S := merge_events(p*_g, S) ∪ p*_g;
  P := P\{P_g}
od.
```

Execution The selection of the next event to be executed is made by the function *select_event* : $\{\mathcal{S}\} \rightarrow \mathcal{S}$. From the global schedule $\mathcal{S}$, this function searches the event e for which holds: $\forall e' \in V_e : s_{e'} = done$. Let a_e be the action associated with the event e. To be executable the preconditions of the associated action have to be consistent with the actual world state $\mathcal{W}_c$. This fact is proven by the predicate *applicable* : $\mathcal{A} \times \mathcal{W} \rightarrow \{true, false\}$.

Within the execution phase those events of the agent's global schedule $\mathcal{S}$ can be executed by the function *perform* : $\mathcal{A} \rightarrow \{true, false\}$ for which no temporal restrictions exist because the predecessor events have already been performed. If several such events exist, they are executed in parallel (if possible) or one is selected on a non-deterministic basis. On correct execution of the action - the predicate $perform(a_e)$ yields the result value *true* - the agent's world state $\mathcal{W}_c$ changes. The world state is actualized by the function *update_state* : $\mathcal{A} \times \mathcal{W} \rightarrow \mathcal{W}$ according to the effects of the instantiated action a_e. The status s_e of the executed event e is changed by the function *mark_as_done* : $\mathcal{S} \rightarrow \mathcal{S}$ from *committed* to *done*. This part of the basic agent loop can be described by:

```
while (e := select_event(S)) do
  if (applicable(a_e, W_c) ∧ perform(a_e))
  then W_c := update_state(a_e, W_c);
  S := mark_as_done(e)
  else replan(g)[8]
od.
```

Evaluation In the evaluation phase it is examined whether all events of plan $p_g^\star$ have been executed correctly to achieve goal g, i.e. $\forall e \in p_g^\star$ must hold: $s_e = done$. To check this status, the predicate *all_events_marked_done* : $\mathcal{P} \rightarrow \{true, false\}$ is used. In other words, the agent has to monitor the execution of its scheduled events. Failed actions or actions not being executable due to un-kept commitments from partner agents are reasons why a plan can fail to achieve the goal. For that reason, the agent should consider each new observation, how it effects the actual world state $\mathcal{W}_c$ and whether the scheduled plan $p_g^\star$ is still on track to achieve the goal g.

Thus, observations - when made several times - can even result e.g. in a revision of an agent's knowledge about actions if the actions did not lead to the desired effects. The function *assimilate* : $\mathcal{O} \times \{\mathcal{A}\} \times \mathcal{W} \times \{\mathcal{S}\} \rightarrow \mathcal{W}$[9] handles both of these issues - modification of the world state and revision of the agent's knowledge. The predicate *consistent* : $\mathcal{W} \times \{\mathcal{S}\} \times \mathcal{P} \times \mathcal{G} \rightarrow \{true, false\}$ checks whether the scheduled plan $p_g^\star$ leads to the desired goal g when the agent's world state has changed.

If an agent observes the successful execution of a plan's last event, the desired goal is achieved and is deleted from the active goal list; the predicate *all_events_marked_done* yields the result *true*, and the following holds: $g \subseteq \mathcal{W}_c$. In the other case, a rescheduling process has to be initiated by the function *reschedule* : $\mathcal{W} \times \{\mathcal{S}\} \times \mathcal{P} \times \mathcal{G} \rightarrow \mathcal{S}$ which, in turn, triggers a re-planning process for goal g if it does not succeed. The evaluation sub-process can be described by:

```
while (o := observation()) do
  W_c := assimilate(o, A, W_c, S);
  if ¬ consistent(W_c, S, p*_g, g) then
  reschedule(W_c, S, p*_g, g)
  fi;
  if all_events_marked_done(p*_g) then
  G := G\{g}
  fi;
od.
```

3 The Cooperation Model

The formal agent model serves as the basis for the description of the cooperation model. However, agents do not always plan and act alone in a world, but must often cooperate with each other to commonly achieve their goals. Cooperation arises as several agents plan and execute their actions in a coordinated way. In MECCA, not only single-agent behavior but also cooperation is seen from a goal-based viewpoint. The basic elements of cooperation are the so-called *cooperation primitives* (Lux, Bomarius & Steiner 1992). They are a combination of *cooperation types*[10] and *cooperation objects*, which are either a goal, a plan, a task or unspecific information such as results, parameters, or other knowledge. Cooperation primitives are basic agent head functions, describing communication among agents with a specific intention. They are represented as plans, whose preconditions and effects fix the semantics/intention of the primitives and whose plan procedures consist of a call to the head function handling the communication (head-communicator-interface).

[7] Let p_1, p_2 be plans with $p_1, p_2 \in \mathcal{P}_g$. It holds: $p_1 < p_2$, iff $cost(\mathcal{A}, \mathcal{W}_c, \mathcal{S}, p_1) < cost(\mathcal{A}, \mathcal{W}_c, \mathcal{S}, p_2)$.

[8] where $e \in p_g^\star$

[9] $\mathcal{O}$ is the set of observations made by an agent due to its sensoring or communication facilities.

[10] drawn from speech-act theory (Searle 1969)

Speaker	Hearer
Pre: *Preconditions*	Pre: *Preconditions*
Proc: send (coop_prim)	Proc: recv (coop_prim)
Eff: *Effects*	Eff: *Effects*

Table 1: Generic semantic representation of cooperation primitives

3.1 Formalization of the Cooperation Model

In the extended process view of cooperating agents, local goals of an agent become *shared goals* if they are solved in cooperation with other agents. The common planning phase leads to the development of so-called *multi-agent plans*, plans which are executed by more than one agent. During the planning and scheduling process, a multi-agent plan is subdivided - as far as possible - into several single-agent plans which can be executed by the agents. The execution of single-agent tasks does not only comprise head or body actions but also communicator actions (i.e. sending cooperation primitives) to coordinate the behavior of other agents.

The semantics and pragmatics of cooperation primitives are dependent on the current phase and on the involved cooperation objects. The functions $f \in \mathcal{F}$ have to be extended to react appropriately to the receipt of a cooperation request. The decision whether a cooperation between agents is at all possible, is also made via these functions. The decision process either ends in sending an answer, in executing a body or head action or in ignoring the received message. Cooperation requests for goals are decided by the function *new_goal*, for plans by the function *find_all_plans*, for schedules by the function *schedule_plan* and for queries by the function *answer_query*.

Cooperation primitives are represented as multi-agent actions with specific preconditions and effects. The procedure (`Proc:`) of cooperation primitives calls the send and receive communication functions on the sender (Speaker) and receiver (Hearer) sides, respectively (cf. Table 1).

According to Table 1, the following formal semantic and pragmatic representation describes the preconditions and effects (`P:` and `E:`, resp.) with respect to the sender and receiver of the cooperation primitive. The description is split according to the agent's basic loop phases and the tackled cooperation objects. The preconditions of the cooperation primitives can be seen as applicability conditions. Together with the speech act class and the message type, they yield the semantics of the cooperation primitives. The effects of the cooperation primitives enhance the knowledge of the cooperating agents and trigger functions in the agents' basic loops. Therefore, they can be seen as the pragmatics of the cooperation primitives. In addition to the notation in Section 2.1 we assume the following:

S, H cooperating agents (S = Speaker, H = Hearer); note that agents alternate roles when issuing successive cooperation primitives, e.g. the initiator S of a `PROPOSE` can be the recipient of a following `ACCEPT` message. Furthermore, in this discussion we restrict the hearer H to be one agent, although in principle H can be a set of agents.

RC_e set of resource constraint pairs

$$((r_1, C_1), \ldots, (r_n, C_n))$$

which have to hold for event e; the consistency of an event is defined by the predicate *consistent* : $\mathcal{P} \times \mathcal{S} \times \mathcal{W}_c \times \mathcal{A} \rightarrow \{true, false\}$; the predicate yields *false*, if $RC_e \cup \mathcal{W}_c \cup \mathcal{S} \cup \mathcal{A} \Rightarrow \neg < x > \ \wedge \ < x >$; a plan p is consistent with an agent's global schedule $\mathcal{S}$ iff all events e of the plan are consistent with $\mathcal{S}$, i.e. : $\forall e \in p$: $consistent(\{e\}, \mathcal{S}, \mathcal{W}_c, \mathcal{A})$.

RC'_e modified constraints of an event e, where the following cases can occur: $RC'_e \Rightarrow RC_e$, i.e. RC'_e is a refinement of RC_e; $RC'_e \not\Rightarrow RC_e$, i.e. RC'_e is a modification of RC_e

$p' \Rightarrow p$ the plan p' is a refinement of the plan p, i.e. : $\forall e' \in p' : RC'_e \Rightarrow RC_e$

$p' \not\Rightarrow p$ the plan p' is a modification of the plan p, i.e. : $\exists e' \in p' : RC'_e \not\Rightarrow RC_e \vee (\exists e' \in p' \wedge e' \notin p) \vee (\exists e \in p \wedge e \notin p')$

C_g constraint set associated with the active goal g; for C_g the same cases can occur as for RC_e; a variation of the above mentioned predicate *consistent* applied to the pair (g, C_g) is used to check for consistency of goal constraints

$g' \Rightarrow g$ goal g' is a refinement of goal g, i.e. : $C'_g \Rightarrow C_g$

$g' \not\Rightarrow g$ goal g' is a modification of goal g, i.e. : $C'_g \not\Rightarrow C_g$

$\oplus\ (x)$ abbreviation for $\mathcal{W}_c = \mathcal{W}_c \cup \{(x)\}$

$\ominus\ (x)$ abbreviation for $\mathcal{W}_c = \mathcal{W}_c \backslash \{(x)\}$

$y +$ `TYPE`(y) abbreviation for: if y is the direct result of a function, it should also be sent in a message of cooperation type `TYPE` to the cooperating agent

When working in cooperation with other agents, the basic loop presented in Section 2.1 becomes distributed (Steiner et al. 1993). Thus the following holds:

Goal Activation In the goal activation phase, a cooperation is instantiated if an agent can not find a local plan for the goal or finds a plan which would involve actions to be carried out by other agents. The agent can also propose a goal to a cooperating agent if it

thinks that a cooperative plan is cheaper or more effective than a local plan. The proposal of a goal g is answered by the partner after examining the function *new_goal*. The situation is as in Figure 1.

Derived from this agent behavior, benevolence of an agent can be defined as follows: an agent always accept a partner's proposal (also in the phases of planning, scheduling, execution), if the proposal does not contradict its current world state.

Thus, in the cooperative context, the function *new_goal* with parameters $(\mathcal{W}_c, \mathcal{G})$,[11] returns the following values:[12]

$$\begin{cases} g + \texttt{ACCEPT}(g), if\ consistent((g, C_g), \mathcal{S}, \mathcal{W}_c, \mathcal{G}) \\ \{\} + \texttt{REJECT}(g), if\ \neg consistent((g, C_g), \mathcal{S}, \mathcal{W}_c, \mathcal{G}) \\ g' + \texttt{REFINE}(g, g'), where\ g' \Rightarrow g \\ g' + \texttt{MODIFY}(g, g'), where\ g' \not\Rightarrow g \end{cases}$$

Planning During the planning phase an agent creates all hypothetical plans $\mathcal{P}_g$ for a selected goal g thereby keeping them consistent with already existing hypothetical plans $\mathcal{P}$. The agent can find plans which are incomplete. These partial plans have gaps which are conceptually represented in the event-based representation by "abstract events". Finding appropriate sub-plans, i.e. refinement of an "abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events". "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.

When cooperating about hypothetical plans, the partner agent can also find alternative plans which are not yet in $\mathcal{P}_g$ and propose them as hypothetical plans. To realize this cooperative aspect, *find_all_plans* $(g, \mathcal{A}, \mathcal{S}, has\_hyp\_plan(< Agent >, \mathcal{P}_g) \cup \mathcal{W}^*)$ yields the following values:[13]

$$\begin{cases} \mathcal{P}_g + \texttt{ACCEPT}(P_g), if\ \forall p \in \mathcal{P}_g : consistent(p, \mathcal{S}, \mathcal{W}_c, \mathcal{A}) \\ \{\} + \texttt{REJECT}(P_g), if\ \forall p \in \mathcal{P}_g : \neg consistent(p, \mathcal{S}, \mathcal{W}_c, \mathcal{A}) \\ \mathcal{P}'_g + \texttt{REFINE}(P_g, P'_g), where\ \forall p' \in \mathcal{P}'_g : p' \Rightarrow p \in \mathcal{P}_g \\ \mathcal{P}'_g + \texttt{MODIFY}(P_g, P'_g), where\ \exists p' \in \mathcal{P}'_g : p' \not\Rightarrow p \in \mathcal{P}_g \end{cases}$$

Scheduling In the scheduling phase, the optimal plan p^* for the goal g has to be inserted in the agents' global schedules. To select the optimal plan p^* out of the set of hypothetical plans an agent must know the costs for the plan's actions, especially for the actions which have to be executed by the cooperating agents. If the agent does not know these costs, it has to ask the cooperating agents. Within the function *find_optimal_plan* it has to send an ASK message to the cooperation partners which themselves provide the agent with the requested information by a TELL message.[14] After that, the agent has sufficient knowledge to evaluate the list of hypothetical plans. It can decide on the optimal plan $p^* \in \mathcal{P}_g$ which in turn should be accepted by the cooperating partners. Only for weighty reasons - unforeseen change of a world state - should the cooperating agent be able to reject the optimal plan.

The agent proposing the optimal plan p^* "tentatively' 'commits to it. The acceptance of the optimal plan by the cooperating agent is a commitment to execute the corresponding actions, and triggers a definitive commitment by the receiving agent.

The receiving agent looks at the optimal[15] plan calling the function *find_optimal_plan* : $\{\mathcal{P}\} \times \mathcal{A} \times \{\mathcal{S}\} \times \mathcal{W} \rightarrow \mathcal{P}$, which results in

$$\begin{cases} p^* + \texttt{ACCEPT}(p^*), if\ consistent(p^*, \mathcal{S}, \mathcal{W}_c, \mathcal{A}) \\ \{\} + \texttt{REJECT}(p^*), if\ \neg consistent(p^*, \mathcal{S}, \mathcal{W}_c, \mathcal{A}). \end{cases}$$

For reasons of space, we skip the formal descriptions of PROPOSE(p^*), ACCEPT(p^*) and REJECT(p^*), which are very similar to the ones in the planning phase.

Execution Actions of a plan are executed by the agents by calling the function *perform*. Depending on whether the result *res* of the execution should be transferred to another agent A, the function *perform* creates as effect a new goal $goal(knows(A, res))$ which, in turn, triggers sending a TELL message. Also, failure of an action has to be transmitted to the partner. Cooperation in the execution phase thus comprises message exchange between the cooperation partners. See Figure 3.

For reasons of space, we conclude with a short informal look at the remaining cooperation primitives REQUEST, ORDER, and ASK.

REQUEST und ORDER can be treated as special cases of PROPOSE. The difference from the cooperation primitive PROPOSE is that the recipient of a REQUEST message can only answer with an ACCEPT message or a REJECT message; the recipient of an ORDER message can only answer with an ACCEPT message. The functions *new_goal*, *find_all_plans*, *find_optimal_plan* are thus restricted in their decision about the cooperation object, e.g. *find_all_plans* $(g, \mathcal{A}, \mathcal{S}, has\_hyp\_plan(< Agent >, p) \cup request(< Agent >, p) \cup \mathcal{W}^*)$ yields

$$\begin{cases} p \in \mathcal{P}_g + \texttt{ACCEPT}, if\ consistent(p, \mathcal{S}, \mathcal{W}_c, \mathcal{A}) \\ \{\} + \texttt{REJECT}, if\ \neg consistent(p, \mathcal{S}, \mathcal{W}_c, \mathcal{A}). \end{cases}$$

[11] where $(has\_goal(< Agent >, g) \in \mathcal{W}_c)$

[12] Clearly, if $g' \Rightarrow g$ and $g' \in \mathcal{G}$, then g is consistent with $\mathcal{G}$. We do not go into the details in this paper, but the result of *new_goal* also depends upon the expected reply, e.g. {ACCEPT/REJECT} vs. {MODIFY/REFINE}.

[13] As with *new_goal*, exactly which value is returned depends among others upon the expected reply.It has to be noted that a given ACCEPT is not yet a commitment for execution rather that the plan proposed is consistent with its current plans. The agent commits to a plan in the scheduling phase.

[14] See next paragraph for the formal semantics of ASK and TELL.

[15] from the partner's perspective

[16] in general: $knows(self, x) \Leftrightarrow x \in W_c$.

		Speaker	Hearer
PROPOSE(g)	Pre:	$g \in \mathcal{G}$	-
	Eff:	$\oplus$ *proposed*(g, H)	$\oplus$ *has_goal*(S, g)
ACCEPT(g)	Pre:	*has_goal*$(H, g) \wedge g \in \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	-	$\oplus$ *has_goal*(S, g)
REJECT(g)	Pre:	*has_goal*$(H, g) \wedge g \notin \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	$\ominus$ *has_goal*(H, g)	-
MODIFY/REFINE(g, g')	Pre:	*has_goal*$(H, g) \wedge g \notin \mathcal{G} \wedge g' \in \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	*proposed*(g', H)	$\oplus$ *has_goal*(S, g')

Figure 1: Cooperation Primitives for Goal Activation

		Speaker	Hearer
PROPOSE($\mathcal{P}_g$)	Pre:	$\mathcal{P}_g \in \mathcal{P}$	-
	Eff:	$\oplus$ *prop_hyp_plan*$(\mathcal{P}_g, H)$	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}_g)$
ACCEPT($\mathcal{P}_g$)	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \in \mathcal{P}$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	-	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}_g)$
REJECT($\mathcal{P}_g$)	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \notin \mathcal{P}$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	$\ominus$ *has_hyp_plan*$(H, \mathcal{P}_g)$	-
MODIFY/REFINE$(\mathcal{P}_g, \mathcal{P}'_g)$	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \notin \mathcal{P} \wedge \mathcal{P}'_g \in \mathcal{P}_g$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	*prop_hyp_plan*$(\mathcal{P}'_g, H)$	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}'_g)$

Figure 2: Cooperation Primitives for Planning

The cooperation primitive ASK is needed to ask another agent for some information. Information can be either hypothetical knowledge under cooperation as e.g. the current goal, the current plan etc. or world knowledge like e.g. the costs of an action.

This section has introduced the operational semantics and pragmatics of the cooperation primitives. The cooperation primitives are the basis of our cooperation model. By composing cooperation primitives, so-called cooperation methods can be build as multi-agent plans (Lux, Bomarius & Steiner 1992).

4 Related Work and Outlook

The work presented in this paper is based on embedding the philosophical notion of speech acts (Austin 1962; Searle 1969) in a planning framework (cf. (Cohen & Perrault 1979) as a seminal work in this area).

Recently, general theories of communication and coordination in human work environments have been proposed (Winograd & Flores 1986; Malone & Crowston 1991), but the main idea looking at cooperation in multi-agent environments in terms of communication structures goes back to Hewitt's pioneering work on actors and open systems (Hewitt 1977; Hewitt 1986). However, he gives no formal syntax or semantics for such communication structures. Halpern and others (Halpern & Moses 1985; Fischer & Immerman 1986) analyzed the process of communication to prove certain properties such as correctness of distributed protocols. Their work is based on a logic of common knowledge and belief, however, they missed formalizing concepts like intention, commitment, capability etc. which are important for cooperation and coordinated action. Cohen and Levesque were the first to set up a formal theory of belief, goal (choice), intention and commitment (Cohen & Levesque 1987; Cohen & Levesque 1990). This was recently criticized by Singh (Singh 1992a) as being conceptually problematic and in parts unduly complicated. Furthermore, Singh shows that their theory is false or counterintuitive in most natural scenarios. Singh himself proposed a clear formalization of intentions and know-how in a general model of logic and time (Singh 1992b). He uses this formalization to provide a new semantics for communication in multi-agent systems. This formal semantics yields a rigorous framework for establishing various design rules regarding communication among agents. Whereas Cohen/Levesque and Singh were interested in speech acts and mental states as building blocks for coordination and analysis of behavior, Shoham was the first who attempted to base a programming language in part on speech acts (Shoham 1993). This most closely resembles our work as the main emphasis is not only on a formal underpinning but also on an algorithmic design and on the implementation of cooperating agents. However, the first version of AGENTO lacks notions such as desires, intentions and plans and is thus less suitable for constructing high-

		Speaker	Hearer
TELL(res)	Pre:	$res \in \mathcal{W}_c \wedge goal(knows(H, res))$	-
	Eff:	$\oplus\ knows(H, res)$	$\oplus\ knows(S, res) \wedge res \in \mathcal{W}_c$[16]

Figure 3: Cooperation Primitives for Execution

level cooperative commands and agent behavior.

Our work on cooperation primitives emphasizes the semantics of the exchanged messages, thus following Singh's work, but extends it by also taking into account the cooperation context. This is achieved by a planning formalism where fundamental work on reasoning about actions has been done e.g. in (Allen 1984) and (Georgeff & Lansky 1987). Because our model deals with human-computer cooperation, it was greatly influenced by work on discourse modeling, which addresses the relationship between mental state and acting by speaking (cf. e.g. the work of Allen, Grosz, Kautz, Litman, Pollack, Sidner and others in (Cohen, Morgan & Pollack 1990)).

In summary, based upon the conceptual view of the MECCA agent and cooperation models, this paper has demonstrated a first viable approach towards a formalism tightly integrating cooperation with an agent's basic loop.

As described in (Steiner et al. 1993) cooperation primitives can be combined to form special plans called cooperation methods (such as contract net). Future work will investigate how the described semantics of cooperation primitives can be used to dynamically alter a cooperation method during the cooperative process. For example, after receiving a request for bids with a deadline in the contract net method, a potential bidder could request an extension of this deadline. The manager would then need to determine how long of an extension, if any, would be compatible with its current plan.

Further, as common within human cooperation, "merging" cooperation primitives may reduce communication overhead. For example, the classic "Shall we have lunch tomorrow" expresses not only the goal of having lunch but also a partial plan for executing this goal tomorrow.

Acknowledgment

We thank our colleagues Alastair Burt, Michael Kolb (both DFKI) and Hans Haugeneder (Siemens) for stimulating discussions and fruitful comments on the reported work. Michael Kolb did a great job by implementing most parts of the MECCA system.

References

J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

J. L. Austin: How to Do Things with Words. Oxford University Press, 1962.

A. Burt. Abduction Based Cooperative Planning in Multi-Agent Systems. PhD Thesis. Imperial College, London, forthcoming.

P. Cohen, J. Morgan, M. Pollack (eds.). Intentions in Communication. Bradford Books at MIT Press, 1990.

P. R. Cohen, H. J. Levesque. Persistence, Intention, and Commitment, *in: M. P. Georgeff, A. L. Lansky (eds.): Reasoning about Actions and Plans*, Morgan Kaufman, 1987.

P. R. Cohen, H. J. Levesque. Intention is choice with commitment, *in: Artificial Intelligence, 42(3)*, 1990.

P. R. Cohen, C. R. Perrault. Elements of a Plan-Based Theory of Speech Acts. *in: Cognitive Science 3: 177-122*, 1979.

K. Eshghi. Abductive planning with event calculus. *in: R. A. Kowalski, K. A. Bowen (eds.): Proceedings of the 5th International Conference on Logic Programming.* 1988.

M. J. Fischer, N. Immerman. Foundations of Knowledge for Distributed Systems, *in: Proc. of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge, 171-185*, Monterey, CA., 1986.

M. P. Georgeff and A. L. Lansky (eds.). *Reasoning about Actions and Plans.* Morgan Kaufman, 1987.

B.J. Grosz, C. Sidner. Plans for discourse. *in: P. Cohen, J. Morgan, M. Pollack (eds.) Intentions in Communication.* Bradford Books at MIT Press. 1990.

C. Hewitt. Offices are Open Systems. *in: ACM Transactions on Office Information Systems, 4(3):271-287*, 1986.

C. Hewitt. Viewing Control Structures as Patterns of Passing Messages. *in: Artificial Intelligence 8, 323-364*, 1977.

J. Y. Halpern, Y. Moses. A guide to the modal logics of knowledge and belief: preliminary draft, *in: Proceedings IJCAI-85, 480-490*, Los Angeles, CA., 1985.

R. A. Kowalski, M. Sergot. A logic based calculus of events. New Generation Computing 4, 67-95, 1986.

A. Lux, F. Bomarius, D. Steiner. A Model for Supporting Human Computer Cooperation. *in: Proceedings of the AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems.* San Jose, CA., July 1992.

T. W. Malone, K. Crowston. Toward an Interdisciplinary Theory of Coordination. Technical Report

CCS TR #120. Center for Coordination Science, MIT, Sloan School of Management, Cambridge, MA., 1991.

J. R. Searle. *Speech Acts.* Cambridge University Press, Cambridge, 1969.

M. Shanahan. Prediction is deduction but explanation is abduction. *in: Proceedings IJCAI-89, 1055-1060*, 1989.

Y. Shoham. Agent-oriented programming. *in: Artificial Intelligence, 60: 51-92*, 1993.

M. P. Singh. A Theory of Actions, Intentions, and Communications for Multiagent Systems. PhD thesis, University of Texas, Austin, 1992.

M. P. Singh. A Critical Examination of the Cohen-Levesque Theory of Intentions. *in: Proceedings of the 10th ECAI*, Vienna, Austria, 1992.

D. Steiner, D. Mahling, and H. Haugeneder. Human Computer Cooperative Work. *in: M. Huhns (ed.): Proc. of the 10th International Workshop on Distributed Artificial Intelligence.* MCC Technical Report Nr. ACT-AI-355-90, 1990.

D. Steiner, A. Burt, M. Kolb, C. Lerin. The Conceptual Framework of MAI^{2}L. *in: Proceedings of MAAMAW-93*, Neuchâtel, Switzerland, August 1993.

T. Winograd, F. Flores. Understanding Computers and Cognition: A New Foundation for Design. Addison Wesley, Reading, MA., 1986.