On agent cooperation

van den Broek, J.

Chapter 7

THE MACRO-LEVEL ASPECTS OF THE ALIGNMENT STUDY

A coordinated problem solving perspective

7.1 Introduction

This chapter addresses the macro-level aspects of the alignment study and consists of five sections. Section 7.2 describes the cooperative work condition of the alignment experiment. It describes how two Soar agents cooperate within the task environment and addresses the problems encountered. Section 7.3 presents the results of the cooperative work condition. Section 7.4 discusses the results of the multiagent effort in relation to the single agent effort of the Van den Broek experiment. In fact, this is the second comparison contrasting the single Soar agent performance with that of the multi Soar agent condition. Hence, this entails a comparison of the non-cooperative and the cooperative work condition *within* the Van den Broek experiment. Section 7.5 discusses the results of the multiagent effort in relation to the single agent effort of the Cohen experiment. This in fact is the third comparison concerning the performance of the single Cohen agent and that of the two Cohen agents in the cooperative work condition. Section 7.6 compares the cooperative work conditions of both experiments. This is the fourth comparison between the performances of the multi Cohen agent and the multi Soar agent.

7.2 The cooperative work condition

This section describes the cooperative work condition of the alignment experiment. Within the cooperative work condition, two Soar agents must cooperate in order to accomplish the same model task.

Within the cooperative work condition, the *bottom-up* experimental cycle has been applied (Conte & Castelfranchi, 1994). The starting point of the bottom-up experimental cycle, depicted in figure 7.1, is the task environment in which the two Soar agents are embedded. The agents are endowed with cognition and initial knowledge such as goals, rules following principles, etc. The next step is to set the

agents to the task of collective decision-making. The top frame of figure 7.1 represents the starting point of the bottom-up experimental cycle.

In the process of cooperation (the second frame), it is probable that different kinds of coordination problems of will emerge (the third frame).

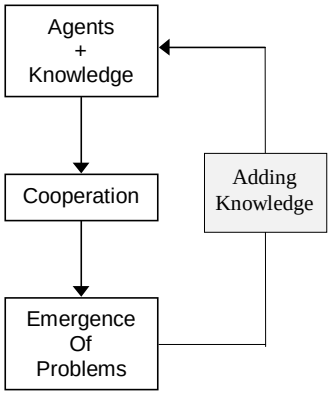


Figure 7.1. The bottom-up experimental cycle.

The way to overcome these emerging difficulties in cooperation is first to analyse the causes of the problem, i.e. to identify what knowledge the agents are lacking in order to cooperate successfully. Based on the analysis, the presumed knowledge required will be added to the agents. The feedback arrow represents this process of adding knowledge. Whether the knowledge added proves to be appropriate in solving the cooperation problem among the agents can be established by rerunning the experiment. In this way, the bottom-up experimental cycle can be explored several times with the intention of achieving a situation in which the agents are capable of cooperating successfully, i.e. to achieve a situation in which a task can be attained through a collaboration process. Knowledge is viewed as being "appropriate" when it enables coherent collective behavior and achieves the collaborative goal set. This means that we are not concerned with optimizing the cooperating process, or with its efficiency, although this experimental cycle could provide the methodological tool to do so. The bottom-up experimental cycle has been applied to provide the experimenter with a means of "discovering" the agent's need for sociality, i.e. coordination ability, without presupposing it.

The agents, being copies of the agent used within the single agent condition, were set to their cooperation task. During the first run, the agents became locked into an action-repetition deadlock. The action-repetition deadlock within the problem-solving process emerges when the agents persist in making different action decisions concerning the same observed state. The emergence of an action-repetition deadlock will be explained with an example run. The sequence of actions as it occurs in the example run and the items learned by both agents are listed in table 7.1.

Table 7.1. The action sequence and items learned within the example run.

Action decision	State	Agent	Applied action	Items learned
1	{4 8 16 24}	Alpha	↑	
	{3 8 15 24}	Alpha		Hyp-action1 ↑
2		Beta	←	
	{3 7 15 23}	Beta		Hyp-action1 →
3		Alpha	↑	No-effect ↑
4			→	
	{3 8 15 24}	Alpha		Hyp-action2 →
5		Beta	→	Repetition 1; No-effect →
6			↑	No-effect ↑
7			↓	
	{4 8 16 24}	Beta		Hyp-action2 ↑
8		Alpha	↑	
9	{3 8 15 24}	Beta	↓	Repetition 2; ineffective state
10	{4 8 16 24}	Alpha	↑	Repetition 1
11	{3 8 15 24}	Beta	?	Repetition 3

Let us assume that the organization is in its initial state {4 8 16 24}, see figure 7.2, and that agent Alpha applies action A ("up")—a random selection—causing a state change to {3 8 15 24}. Alpha deduces that Hyp-action1 is A ("up") because of the positive feedback obtained (see table 7.1). Control now shifts to agent Beta, which applies (second action decision)—again at random—action B ("left"). At this stage it should be emphasized that agent Beta has no knowledge, i.e. no model, of agent Alpha and vice versa. Beta deduces that the best action to pursue (Hyp-action1) must be D ("right") because action B ("left") induced negative feedback. Again, Alpha is in control. Because of its Hyp-action1 preference (RULE 4 see Appendix A), Alpha applies A ("up") and experiences the fact that action A has no effect on this state (see action decision 3). The action potential has been narrowed down to three. However, by applying action C ("down"), the agent would proceed in the opposite direction as indicated by Hyp-action1, which has been implemented as less effective search behavior. This search knowledge has been implemented as RULE 6. Hence, the action potential for this state is reduced to either B ("left") or D ("right"). Alpha has not established a preference for either of these actions and consequently a random choice between the two will be made. Alpha applies D ("right"), moving the organization back to state {3 8 15 24}. The feedback received is positive and Alpha represents that Hyp-action2 is D ("right"). Now in control, Beta finds itself again in state {3 8 15 24}, the state in which it made its last action decision. The fact that agent Beta has moved to a state it just came from is represented as repetition 1 in table 7.1 (see action decision 5). Beta applies action D ("right")—Hyp-action1 preference—and finds that action D has no effect on this state (action decision 5). Beta also will make a random

choice between action A (“up”) and C (“down”), based on SEARCH RULE 6. Beta applies action A and observes that the action has, again, no effect on this state. Consequently, action C (“down”) will be applied, which causes a state transition to {4 8 16 24}. Please note that the last Beta action resulted in a return to the initial state. Negative feedback is obtained, leading Beta to conclude that Hyp-action2 is to be represented as action A (“up”). Alpha, now in control, applies action A (“up”)—Hyp-action1 preference (action decision 8). With Alpha’s last action decision, Beta finds itself back in state {3 8 15 24} for the second consecutive time, which has been represented in table 7.1 as repetition 2. Because Beta now knows that neither of the Hyp-actions can be applied in this state, it will represent it as an ineffective state. Again, Beta applies action C (“down”) (action decision 9). The reaction of Alpha will be to apply action A (“up”) in state {4 8 16 24} because it has not yet established the “ineffectiveness” of state {3 8 15 24} and it still assumes that Hyp-action1 (“up”) must be pursued (action decision 10). Beta returns for the third consecutive time to the same state, i.e. repetition 3. In this way, a cyclic pattern of action decisions emerges from which the agents cannot escape and which therefore yields a deadlock for the cooperation process. The knowledge that both agents hold within their world models is depicted in figure 7.2.

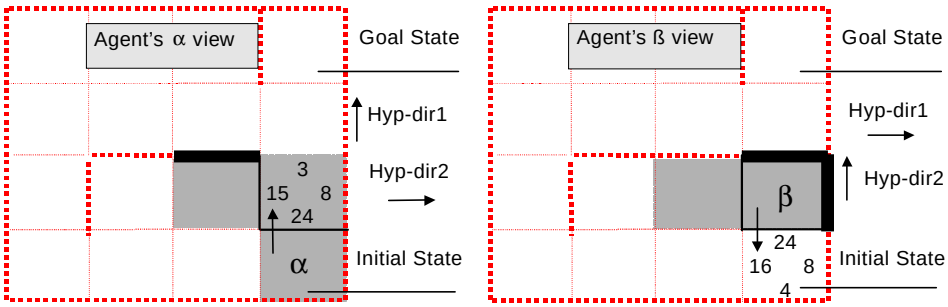


Figure 7.2. The situation in which an action repetition deadlock emerges.

The reason the cooperative deadlock emerges in the form of a cyclic pattern is that agent Beta wants to avoid state {3 8 15 24}, which has been represented as ineffective, while agent Alpha does not possess this knowledge and has therefore no reason not to want to explore that state and to pursue in that direction. This is considered realistic in terms of cooperation processes. Differences in perception of reality are identified as one of the causes of individual conflict (March & Simon, 1993: 141).

The lesson learned in terms of representation is that when models of the environment are learned on the basis of individual action-feedback cycles, the model content may differ from agent to agent simply because they are based on individual learning experience, which enhances the plausibility of the model.

7.2.1 Difference between both mental models

The example of a cooperative deadlock caused by cyclic behavior, depicted in figure 7.2, shows an important consequence of learning agents in a cooperative work condition. Within the cooperative work condition, agents build up a mental model of the task environment based on their own individual experience. Because experience differs at a certain moment in time, the model content on which the agents base their choice of action will also differ accordingly. Incorporating the possibility of different interpretations of reality within the multiple agent decision-making models enhances the plausibility of the model, because it mimics real-world situations in which conflicts emerge.

The cooperative deadlock is perceived as a "social conflict" because it is based upon a cyclic *behavioral pattern* of two agents. It means that both agents perceive different means for reaching the same goal state. The different perceptions of "means" in reaching the goal state relate to the different content of their task environmental models. Agent Beta's model of the task domain holds that both Hyp-actions A ("up") and D ("right") have no effect on state {3 8 15 24} and has represented this state as ineffective. Consequently, Beta tries to avoid that state and proceeds in a different direction. Agent Alpha, not having this knowledge, has no reason to avoid this state and still holds the attainment of that state as preferable. The emergence of conflict is a phenomenon that was not anticipated beforehand. It was something the simulation stumbled upon and therefore it came somewhat as a surprise.

7.2.2 Adding cooperation knowledge

The emerging cooperative deadlock of cyclic behavior has been perceived and dealt with as an indication of the need for additional knowledge in order for the cooperation effort to succeed. The first thing that the agent needs is the ability to recognize that "the other agent" reversed the effect of its own action. The recognition rule has been implemented in such a way that the "incidental" reversals, which are part of the normal trial-and-error learning cycle, will be ignored. In order to do so, the `LEVEL OF REPETITION` was introduced, i.e. the number of consecutive times a state reversal occurs. An action-repetition deadlock will only be detected when a state transition reversal takes place at least two consecutive times, see Appendix A under label `OPERATOR PASS2LEARN`.

The next thing to decide is what action to take when the cooperative deadlock has been recognized. Initially, the rule that was added forces the agent that recognizes the action-repetition deadlock first, to select an action different than the action that was last applied. In the above example sequence of figure 7.2, the action last applied is "A". Alpha must now exclude action "A" as an alternative and choose from the others (B, C, and D). This forced variation adds a certain amount of randomness to behavior in order to overcome cyclic patterns at a local minimum. This random factor is comparable to a *stochastic* activation function used in Boltzmann machines (Russell &

Norvig, 1995). Adding this stochastic activation function to the action selection behavior did not solve the problem since the cyclic behavior remained. The difference, however, was that the cyclic pattern occurred in a span of three different states instead of two. We did not extend the Soar agent's ability for remembering more than one visited state, since the cyclic behavior would persist and spread over even more states, providing, therefore, no "real" solution. A real solution in the case of a knowledge-level system, which the Soar agent evidently is (see chapter 4), would be to add sufficient coordination knowledge to the agent so that it is able to cope with the deadlock situations. That is, we want to look for a knowledge-based solution instead of a trial-and-error approach.

The question then was: What would be a proper knowledge-based solution for the cooperative deadlock. We have stated that particular differences in world representations are the cause of the cyclic pattern. Therefore, the answer must be found in somehow aligning the knowledge content of the models in such a way that the basis for perceiving different means of reaching the goal state disappears. One way of accomplishing this is to give the agents an opportunity of learning additional environmental knowledge. The assumption is that the newly acquired environment knowledge will change the action preference ordering and consequently change the decision-making process of the agents with the potential of lifting the cooperative deadlock situation. The solution proposed is different from the former in that additional *knowledge* about the environment is learned instead of simply implementing a heuristic of randomly trying "something else".

The way an agent learns within the model is to act within the task environment. Therefore, the best action to apply in the case of a cooperative deadlock is to "pass", i.e. to apply action E, and grant the other agent an extra opportunity to act and, by doing so, to learn and change its world model accordingly. Because this second order "pass" decision will be made based on the assumption that the other agent will learn from it, it has been called a "pass2learn" decision. The difference between the default "pass" decision following each state transition and the one induced through a cooperative deadlock is that the latter is based on a percept of the cyclic decision pattern. This means a shift from applying the pass rule based on tacit knowledge to applying the rule based on explicit knowledge. In summary, the pass2learn rule means that a decision will be made by the agent that detects a no-progress situation (percept), in the form of cyclic behavior, to shift control to the other agent.

Next, this pass2learn rule will be applied in the example case as depicted in figure 7.2 and it will be indicated how the problem-solving process changes accordingly. The action sequence and items learned as occurs within the example run demonstrating the pass2learn rule is listed in table 7.2.

7.2.3 Applying the pass2learn rule

Beta will be the first to detect the cooperative deadlock when it returns to state {3 8 15 24} for the second consecutive time and will, consequently, take the initiative for "pass2learn" (see action decision 9 in table 7.2). It means that instead of applying action C ("down") it will shift the control to Alpha.

Table 7.2. The action sequence and items learned as occurs within the example run demonstrating the pass2learn rule.

Action decision	State	Agent	Applied action	Items learned
1	{4 8 16 24}	Alpha	↑	
	{3 8 15 24}	Alpha		Hyp-action1 ↑
2		Beta	←	
	{3 7 15 23}	Beta		Hyp-action1 →
3		Alpha	↑	No-effect ↑
4			→	
	{3 8 15 24}	Alpha		Hyp-action2 →
5		Beta	→	Repetition 1; No-effect →
6			↑	No-effect ↑
7			↓	
	{4 8 16 24}	Beta		Hyp-action2 ↑
8		Alpha	↑	
9	{3 8 15 24}	Beta	Pass2learn	Repetition 2
10		Alpha	↑	No-effect ↑
11		Alpha	→	No-effect →; ineffective state
12		Alpha	←	
	{3 7 15 23}	Alpha		Ineffective state
13		Beta	→	
	{3 8 15 24}	Beta		Ineffective state
14		Alpha	←	Repetition 1
15	{3 7 15 23}	Beta	↑	Repetition 1; No-effect ↑; ineffective state

Control accordingly shifts and Alpha will apply action A ("up") (Hyp-action1) to state {3 8 15 24} and will learn that action A has no effect on that state. Based on RULE 5, agent Alpha will now attempt to apply action D ("right"), which is represented as Hyp-action2, and will learn that this action also cannot be applied to this state. Based on the knowledge that both Hyp-actions cannot be applied here, the state will be represented as ineffective. Alpha will apply action B ("left") (action decision 12) and a transition to {3 7 15 23} occurs, and this state will also be represented as ineffective. Based upon the newly acquired world knowledge, agent Alpha will now also avoid

visiting state {3 8 15 24} and the cyclic pattern that caused the cooperation deadlock has been broken.

What happened was, that based on the ability to recognize a cooperative deadlock, agent Beta took the initiative of applying the pass2learn rule instead of applying action C. In this way, agent Alpha gets an extra opportunity of learning from the environment. In doing so, its model of the task environment is augmented to the extent that it takes a different action decision. Because the action selection behavior of Alpha changes, the cooperative deadlock has been solved and cooperation progresses.

7.2.4 The necessity to model the other agent

A second interesting effect that emerged from the simulation experiments was that the pass2learn rule, as described above, provided only a partial solution. It worked fine when the cooperation deadlock was detected within the state of which the "other" lacks knowledge, as was the case in the example run listed in table 7.2. The pass2learn rule fails in the case where the repetition was detected "outside" the state of which the "other" lacks knowledge. First, we will present an example run demonstrating such a case. Then we will explain how the pass2learn rule has been augmented to overcome the problem.

The reason we describe the problem of cyclic behavior and the solution applied in such a detailed way is because the problem illustrates and underlines an important property that a cognitive agent should have to be able to operate in multiagent situations, i.e. in order to act as a social agent. The property that agents need to have in cooperative work situations is the ability to model the "other" agent or agents (e.g. Van Heusden & Jorna, 1999). The agent model may be viewed in addition to, or as part of their model of the task environment. Within Distributed AI, these so-called "agent models" are an important way of coordinating the activities of a group of agents without imposing centralized control (Bond & Gasser, 1988; Gasser & Hill, 1990). Sociologists, social psychologists, and those within the field of computational organization theory have proposed that the primary mechanism for creating organized societies of individuals is the ability of the individual to generate and use internal models of other agents and for the individual to reflect on actions and their effects (Carley & Newell, 1994). The next example shows that the ability to model the other agent is also needed within this cooperative effort.

7.2.4.1 Illustrating the problem

The example situation depicted in figure 7.3 will be used for illustrating the failure of the pass2learn rule in particular situations. The action sequence and items learned within the example run are listed in table 7.3.

The situation is that Alpha is located in the top-left state {1 5 9 17}, Hyp-action1 is A ("up"); Hyp-action2 is D ("right"). From this it follows that Alpha's movement decision will be A ("up") and in doing so will learn that this action has no effect on

this state. Alternatively Alpha applies D ("right")—Hyp-action2—and a transition to state {1 6 9 18} occurs.

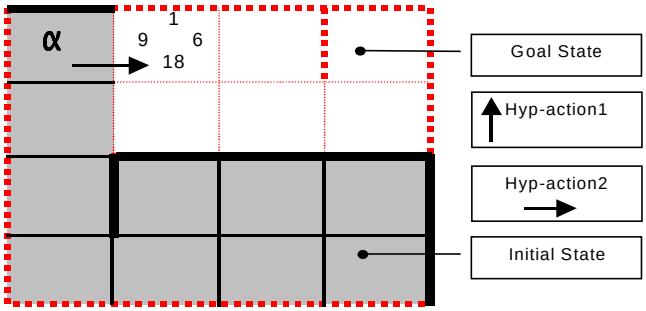


Figure 7.3. α's movement decision.

Table 7.3. The action sequence and items learned as occurs within the example run demonstrating the failure of the pass2learn rule.

Action decision	State	Agent	Applied action	Items learned
1 (figure 7.3)	{1 5 9 17}	Alpha	↑	No-effect ↑
2		Alpha	→	
3 (figure 7.4)	{1 6 9 18}	Beta	→	
4 (figure 7.5)	{1 7 13 19}	Alpha	↑	No-effect ↑
5		Alpha	→	No-effect →; ineffective state
6		Alpha	←	
7	{1 6 9 18}	Beta	→	Repetition 1
8	{1 7 13 19}	Alpha	←	Repetition 1
9	{1 6 9 18}	Beta	Pass2leran	Repetition 2
10		Alpha	↑	No-effect ↑; ineffective state
11		Alpha	←	
	{1 5 9 17}	Alpha		Ineffective state
12		Beta	→	
13	{1 6 9 18}	Alpha	←	Repetition 1
14	{1 5 9 17}	Beta	→	Repetition 1
15	{1 6 9 18}	Alpha	Pass2learn	Repetition 2
16		Beta	→	
Back to action decision 6!	{1 7 13 19}	Alpha	←	

The situation in which Beta makes the third action decision is depicted in figure 7.4. With D ("right") being Hyp-action1 and A ("up") being Hyp-action2, Beta applies D ("right) and a transition to state {1 7 13 19} occurs.

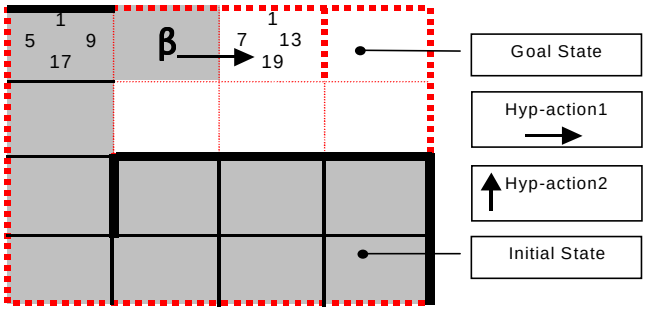


Figure 7.4. β's movement decision.

Alpha, back in control, see figure 7.5, will first try to apply A ("up") (Hyp-action1) and will detect that this action has no effect on this state. Subsequently, D ("right") (Hyp-action2) will be applied and Alpha, again, learns that this action has no effect on this state. State {1 7 13 18} will be represented as ineffective state. Two alternative actions (B, C) are left. Based on rule 6, Alpha will apply action B ("left") thereby enabling a transition to state {1 7 13 19}.

Now in control, Beta detects that Alpha restored the effect of its last applied action decision, i.e., Beta finds the organization back in the same state that it was when it last applied an action, i.e. repetition is 1. Because of Hyp-action1, Beta will apply action D ("right") (action decision 7). Now Alpha also detects the first repetition and again applies action B ("left"). Detecting the second repetition, Beta takes the initiative of "pass2learn" (action decision 9).

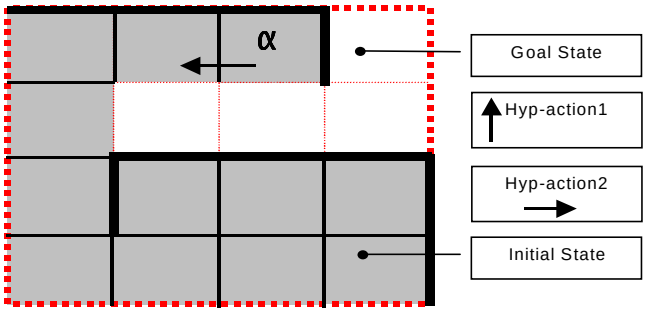


Figure 7.5. β detects repetition.

Alpha, getting the opportunity to make an extra movement decision, will apply action A ("up") (Hyp-action1) and will detect that this action has no effect on this state. Based on this, and on the fact that Hyp-action2 leads to an ineffective neighbor state, the current state will be represented as ineffective. When Alpha applies action B ("left"), we are back in the situation depicted in figure 7.3. In addition, this state will

be represented as ineffective by Alpha. Beta again selects D ("right") in this state because its world representation did not change in the intervening period. Now Alpha detects the cyclic pattern first and takes the initiative of "pass2learn" when the repetition level is 2 (see action decision 15) granting Beta an extra movement decision. Again, Beta will apply D ("right") restoring the situation depicted in figure 7.4. When agent Alpha again selects action B ("left"), this sequence of actions results again in a cyclic pattern of agent decision behavior, because the agents are back in the same state as when action decision 6 was taken. Of course, agent Alpha could have selected action C ("down") resulting in a different action selection pattern. The point is, however, to show that the cyclic pattern remains because the pass2learn mechanism fails in this situation in that the knowledge gained from the additional pass2learn decision is either the wrong piece of information or none at all. As a result, state {1 7 13 19} has been represented within Alpha as ineffective, while Beta has no such knowledge and has therefore no intention of avoiding that state. This will be explained as follows.

In the example run we describe, agent Beta detects the repetition first (the transformation from figure 7.5 back into figure 7.4, see also action decision 9 in table 7.3) and takes the initiative for applying the pass2learn rule. Consequently, agent Alpha makes an extra movement decision and the deadlock situation is solved. However, when returning to state {1 6 9 18} Beta will keep trying to reach state {1 7 13 19} with action D ("right"), being Hyp-action 1, and Alpha tries to avoid it because of the inefficiency representation of that state. In other words, agent Alpha did gain additional world knowledge from the pass2learn decision but the problem is that in this situation Beta should learn that state {1 7 13 19} is indeed inefficient and should avoid it. The vital differences in world representations remain therefore and the pass2learn decision of Beta does not result in the required change of behavior. From this it follows that the pass2learn rule must be applied more intelligently, that is, we must add more knowledge to the rule.

7.2.5 Adding more knowledge to the pass2learn rule

If, in the same situation, Alpha were to take the initiative for the pass2learn decision instead of Beta (see action decision 10 in table 7.4), then Beta would gain additional environmental knowledge. In the situation depicted in figure 7.4, Beta wants to apply action D ("right") and in doing so move to a state {1 7 13 19} represented within Alpha as ineffective. In the case where Beta would know that D ("right") cannot be applied in state {1 7 13 19}, another movement decision would have been made in state {1 6 9 18}. Therefore, the most rational thing to do in this situation is for Alpha to take the initiative for a pass2learn decision since Beta would learn the appropriate, i.e. missing, knowledge.

Table 7.4. The action sequence and items learned as occurs within the example run demonstrating the knowledge-based pass2learn rule.

Action decision	State	Agent	Applied action	Items learned
1	{1 5 9 17}	Alpha	↑	No-effect ↑
2		Alpha	→	
3	{1 6 9 18}	Beta	→	
4	{1 7 13 19}	Alpha	↑	No-effect ↑
5		Alpha	→	No-effect →
6		Alpha	←	
7	{1 6 9 18}	Beta	→	Repetition 1
8	{1 7 13 19}	Alpha	←	Repetition 1
9	{1 6 9 18}	Beta	→ (!)	Repetition 2
10	{1 7 13 19}	Alpha	Pass2learn	Repetition 2
11		Beta	→	No-effect →
12		Beta	↑	No-effect ↑; Ineffective state
13		Beta	↓	
14	{2 7 14 19}	Alpha	↑	
15	{1 7 13 19}	Beta	↓	Repetition 1
16	{2 7 14 19}	Alpha	→	(Neighbor relation!); Repetition 1
17	{2 8 14 20}	Beta	→	No-effect →
18		Beta	↑	
	{1 8 13 20}	Beta		Goal attainment

The question now is: How does an agent determine what knowledge the "other" lacks and whether or not it can be learned by applying the pass2learn rule? To be able to do so, an agent must have some sort of model of the other agent. In this case, it means that the agents must be able to detect which piece of information the other agent lacks. An agent model in the general sense is a representation that an agent holds about the abilities, goals, and preferences of the other agents within the multiagent task environment with which it interacts.

As we have stated, the knowledge advantage in this situation is for agent Alpha because agent Alpha holds knowledge of the fact that in state {1 7 13 19} action D ("right") has no effect, see figure 7.5. Furthermore, agent Alpha knows that agent Beta wants to apply action D ("right"). This is established by observing that the situation has been reversed, which can only be brought about by an action antagonistic to the action last applied by Alpha. In this case, D ("right") is the reverse of Alpha's last-applied action B ("left"). Thus, Alpha's "Beta model" holds that Beta wants to apply action D. Alpha's world representation holds that applying D in state {1 6 9 18} would result in a transition to state {1 7 13 19} in which action D can not be applied. Combining the knowledge of what Beta wants to achieve and knowing that this makes

no sense provides Alpha with the knowledge to take the initiative of making the pass2learn decision.

The pass2learn rule has been extended and enriched by adding a simple agent model. The effect of the knowledge-based rule is that it provides an agent with the opportunity of *deciding* to apply the pass2learn action instead of the simpler "first to detect" criterion. Knowledge-based means in this situation that the pass2learn decision is based on the knowledge that an agent has of both the simulated task environment and the knowledge that the other agent has or apparently lacks. This conclusion following the bottom-up experimental cycle must be that taking the initiative of breaking the detected action-repetition deadlock must be based on the knowledge that one has of the other.

The implemented agent model is indeed a very simple model even to the extent that one could argue if it is in fact a model. The information on the basis of which an agent takes the pass2learn initiative is in fact a memory trace, i.e. a piece of information held within the short-term memory. The decision of whether to apply a pass2learn action will be taken based on comparing this piece of information and what is known of the task environment contained within the task-environment model. Because this information is contained within the short-term memory only, it means that in the next situation this information is not available. It also means that an agent cannot trace back and tell which action the other applied three decision cycles before, for instance, and why this was wrong based on its own knowledge. When one defines a mental model as a model that:

1. Contains information about an external item,
2. Resides in long-term memory, and
3. Can be used at any time within the problem-solving process,

then the implemented model does not match this definition.

This definition furthermore implies that an agent model will be available from the very start, at which stage it contains only initial or general information about the social agents that the multiagent environment contains and of which our subject agent is part. Based on experience in dealing with other agents, the knowledge contained within the agent model will increase, as was the case with the task environment model.

As we have said, the implemented agent model is not very sophisticated. However, this is simply to do with the implementation level. A more elaborate and durable agent model could well be implemented and applied if wanted. The current implementation level has proven to be essential in demonstrating that information about the other should be incorporated in one's own decisions in order to cooperate successfully. Furthermore, the agent model proved to be indispensable because otherwise cooperative deadlock could not be solved, which would hamper the cooperation work condition in a severe way.

Despite the fact that the current implementation of the knowledge-based pass2learn rule should be considered an ad-hoc solution—because it is tied to a specific task environment—a general principle can be distilled from this cooperation experiment. The general principle is that in order to function in a multiagent social context, it is necessary for the agents to be able to model each other. In other words, the agent must not only deal with the other agents as physical (possibly dynamic) objects, as it would if the object was in some nonsocial task situation. Other agents also must be treated as having goals and taking actions to attain those goals (Carley & Newell, 1994). In this sense, the knowledge-based application of the pass2learn rule could be viewed as "social" knowledge in dealing with the "social" goal of overcoming the cooperative deadlock. The label "social" is used because the solutions evolve—as opposed to emerge—during interaction and so are the result of cognitive activity. They are not social in the sense that they are the product of a particular cultural-historical environment (Carley & Newell, 1994).

7.2.6 Summary and conclusion

This section described the cooperative work condition of the alignment experiment. Within the cooperative work condition, two Soar agents must cooperate in order to accomplish the same model task. The simulation showed the emergence of "social conflict" between the agents. The conflict as a social phenomenon emerged because the agents hold different views of the task environment. Differences in perception of reality are identified of one of the causes of individual conflict by March & Simon (1993). The cooperation deadlocks are solved by the agent through the coordination mechanism of changing the "pass control" sequence. Specific knowledge has been added to the agents in order for them to be able to determine which agent at which point in the coordinated problem-solving effort should hand over control instead of applying an action. It proved to be vital that the agents model each other in this coordination effort. The knowledge-based coordination mechanism that has been established proved successful in that no conflicts emerged that could not be solved.

7.3 The experimental results

Table 7.5 presents an overview of the results of the alignment experiment. The first three rows contain data that has been presented earlier. The first two rows contain the data of both experimental conditions of the original experiment; presented in chapter 5. The last two rows contain the data of both experimental conditions of the replicated experiment. The data of the single Soar agent (third row) has been presented in chapter 6. Furthermore, section 6.4 presented a discussion about the difference of learning performance between a single Soar agent and a single Cohen agent (comparison 1). The data concerning the cooperative work condition of the replicated experiment (last row) is new and has not been discussed yet.

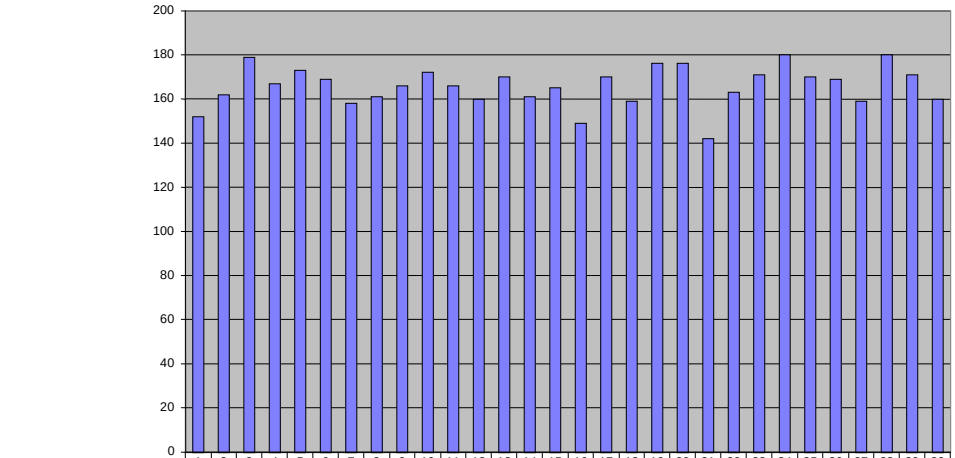
Table 7.5. The results of the alignment study.

Condition	Mean	Standard Dev.	N	Average path length
1 Cohen agent	66.03	19.73	30	15.14
2 Cohen agents	88.03	14.22	30	11.36
1 Soar agent	167.87	9.07	30	5.96
2 Soar agents	165.87	8.97	30	6.03

Therefore, we will present and elaborate on the cooperative work data of the replicated experiment first in section 7.3.1. After that, three discussions will take place. First, we will compare and discuss the performance of the multi Soar agent condition in relation to the single Soar agent condition (comparison 2) in section 7.4. The second discussion will focus on comparing the multi Cohen agent performance with the single Cohen agent performance (comparison 3) 7.5. The third discussion will focus on comparing the multi Cohen agent performance with the multi Soar agent performance (comparison 4) in section 7.6.

7.3.1 The results of the cooperative work condition of the replicated experiment

Table 7.5 presents the multiagent performance (last row) as the average number of goal-state attainments (165,87) and the average solution path expressed as state transitions (6.03). These numbers are based on the goal-state attainment count, i.e. the number of times the goal state has been reached within an experimental run, of the 30 runs, and are depicted in figure 7.6. The number on the x-axis represents the run number. The number on the y-axis represents the goal-state attainment count within that run. For the sake of convenience, the maximum number of goal-state attainments for each run has been printed below the run number. Each single experimental run must be viewed as a separate simulation experiment in which the agents were reloaded and set to the task based on a priori knowledge only. This means that the model of the task environment created within a particular run did not transfer to the next run

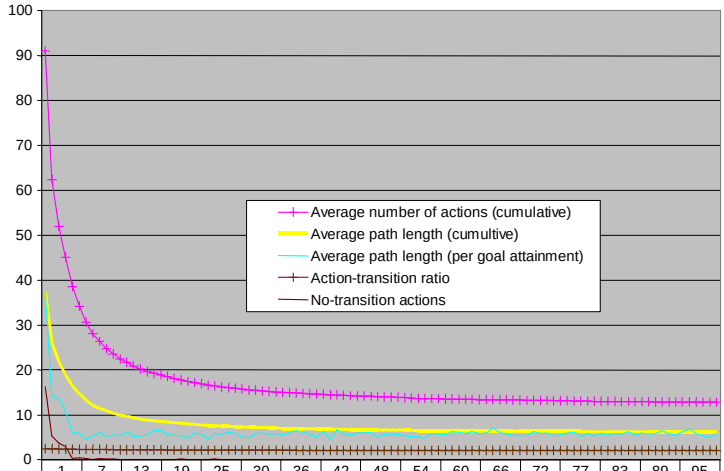


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
max.# of goal-state attainments	152	162	179	167	173	169	158	161	166	172	166	160	170	161	165	149	170	159	176	176	142	163	171	180	170	169	159	180	171	160

Figure 7.6. The distribution of maximum goal-state attainments for the multiagent condition over 30 runs.

The average goal-state attainment count over 30 experimental runs is 165.87 with a standard deviation of 8.97. This is based on the total number of goal-state attainments (4976) divided by the number of runs (30). The number of state transitions (1000) divided by the average goal-state attainment count (165.87), produces an average path length of 6.03 over 30 runs.

As was the case within the non-cooperative work condition (chapter 6), the performance of the cooperative work condition will be expressed as the average number of state transitions needed to attain the goal state. The learning curves of the multiagent condition are depicted in figure 7.7.

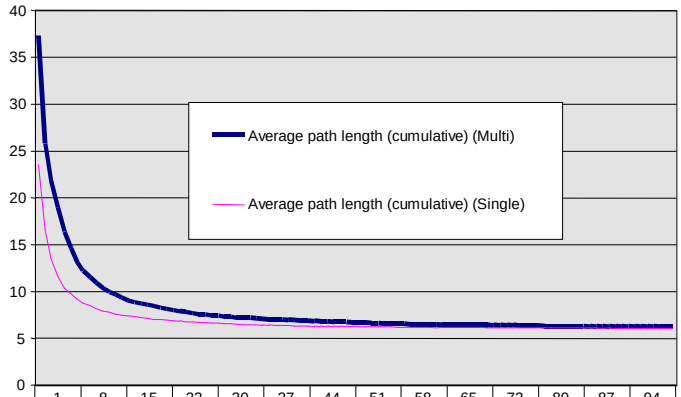


	1	7	13	19	25	30	36	42	48	54	60	66	72	77	83	89	95
Average number of actions (cumulative)	90,97	30,66	21,71	18,49	16,61	15,75	15,00	14,57	14,19	13,84	13,53	13,40	13,25	13,14	13,02	12,92	12,85
Average path length (cumultive)	37,33	13,24	9,71	8,45	7,69	7,36	7,07	6,91	6,77	6,63	6,51	6,46	6,41	6,37	6,32	6,28	6,26
Average path length (per goal attainment)	37,33	4,60	6,27	5,57	4,53	5,07	5,43	6,30	6,00	5,27	5,60	6,03	5,50	5,60	5,83	5,97	6,17
Action-transition ratio	2,4	2,3	2,2	2,2	2,2	2,1	2,1	2,1	2,1	2,1	2,1	2,1	2,1	2,1	2,1	2,1	2,1
No-transition actions	16,3	0,3	0,0	0,0	0,1	0,1	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0

Figure 7.7. Multiagent learning curves.

The curves shown in figure 7.7 are of the same type as those used in the non-cooperative work condition—discussed in chapter 6—and, consequently, are computed in the same way. Full details of the multiagent performance data can be found in Appendix E. What figure 7.7 makes clear is that the multiagent learning characteristics are not different in the qualitative sense, i.e. their shape, from those of the single agent condition. The two conditions differ however in the quantitative sense concerning the specific performance levels. For instance, the average number of state transitions for the first goal-state attainment is 37.33 for the cooperative condition and 23.57 for the non-cooperative condition. The difference of 13.76 is considered as the coordination costs of the cooperation effort. Coordination costs are involved because cooperative deadlocks, in the form of action-repetition deadlocks, emerged within the multiagent condition.

To make the coordination costs visible, we need to compare the average path lengths of both experimental conditions; this is done in figure 7.8.



	1	8	15	22	30	37	44	51	58	65	72	80	87	94
Average path length (cumulative) (Multi)	37,33	12,23	9,11	8,01	7,36	7,04	6,85	6,70	6,53	6,47	6,41	6,34	6,30	6,26
Average path length (cumulative) (Single)	23,57	8,738	7,4	6,861	6,584	6,402	6,283	6,257	6,194	6,183	6,156	6,112	6,093	6,057

Figure 7.8. The learning curves of the non-cooperative work and the cooperative work condition based on the average cumulative path length.

What figure 7.8 shows, is that the difference between the two conditions in terms of average path length diminishes as the problem-solving process evolves. This means that the coordination costs decrease because the learning of the agents progresses. The finding that the coordination costs are especially high in the first stage of the cooperation process is in line with what one would expect in real-world terms. When people start cooperating, they need to work at coordinating the effort. As the cooperation process evolves, the coordination costs will diminish as the agents have learned to "tune in". Whether the coordination costs will delimit to zero or remain, either as a constant or periodically, depends on the task domain involved.

The trend of diminishing coordination costs becomes even more clear when the performance of both conditions are expressed as the average path lengths measured per goal attainment, instead of a cumulative measure. The average solution paths per goal attainment are represented in figure 7.9.

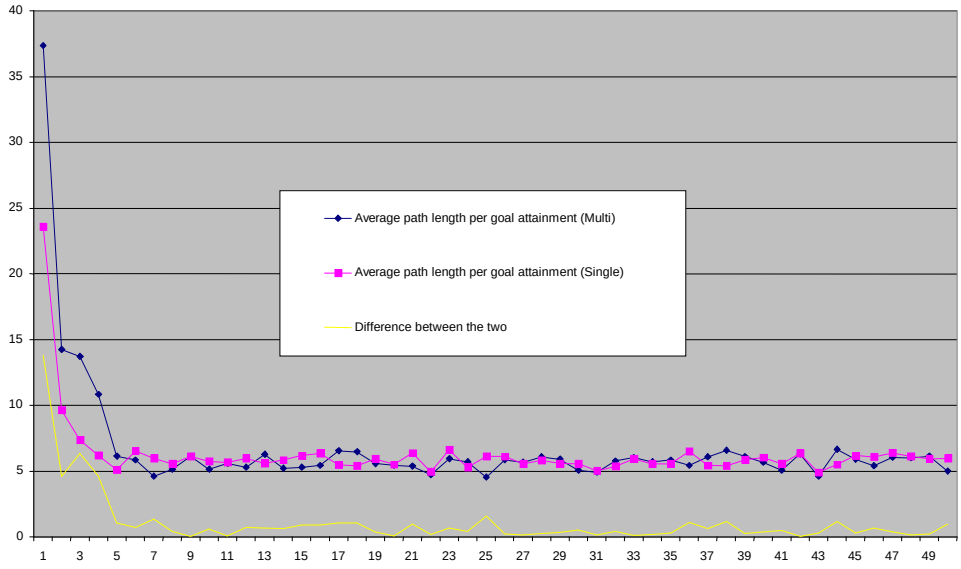


Figure 7.9. Average solution paths measured for each goal attainment of both conditions.

The third curve expresses the difference between the average solution paths of the non-cooperative work condition (second curve) and the cooperative one (first curve). figure 7.9 demonstrates that the initial difference in terms of path length is, again, 13.76 state transitions. The second time that a goal state is achieved, the difference is reduced to 4.6 state transitions[1]. Around the sixth goal attainment occasion, the performance difference between the two conditions is reduced to zero. The variation beyond that point is due to a combination of variations of random placement over 30 runs, and of not taking always the most efficient route. Therefore, we conclude that coordination costs are not involved beyond that point.

7.3.1.1 Action-repetition deadlocks

The largest part of the coordination costs concerns the fact that a number of additional actions have to be taken by the agents in order to resolve emerging action-repetition deadlocks. First, it takes a number of state transitions to detect the repetition deadlock, and then it takes a number of actions to solve it. Because the "repetition level" has been set to two, this means that it is only in cases in which an agent has been put back two consecutive times to the same state, that the repetition will be detected. Since the pass2learn rule has been knowledge-based, it does not entail that the agent that detects the repetition deadlock takes action automatically. It is very well possible that the

[1] The data can be obtained by subtracting curve 3 of Appendix E from curve 3 of Appendix D. For the first seven goal attainments, the differences are: 13.76; 4.6; 6.33; 4.63; 1.03; 0.07; 1.37

agent that detects the action-repetition will not apply the pass2learn action, but that the other agent, subsequently detecting the repetition deadlock, will. Thus, a repetition level of two does not mean that it always takes two state transitions to fire the pass2learn rule. It does mean, however, that it takes at least two state transitions and a maximum of three, depending on the contents of the task environment models of the agents. In addition to detecting the repetition deadlock, it takes one state transition for the other to learn that two states are directly related, and are neighboring states. Alternatively, it takes only one additional action when the task-related knowledge to be learned involves establishing that an action is not allowed. In summing up, each occurring repetition deadlock takes two or three transitions to detect and one action to fire the pass2learn rule. Furthermore, it takes one action for the other agent to learn the additional information. Whether the application of the action results in a state transition depends on the situation. When the learning involves establishing a neighbor relation, it will take an additional state transition. When the knowledge involves an action that is not allowed, it will not. This means that to resolve an action-repetition deadlock, it may take a maximum of four state transitions (involving five actions) and a minimum of two (involving four actions).

Table 7.6 shows the number of times an action-repetition deadlock emerged over the 30 experimental runs. The data covers only measurements over the first six goal attainments since beyond that point no action-repetition deadlocks emerged.

Table 7.6. The distribution of emerging action-repetition deadlocks.

Experiment Run #	Goal Attainment 1	Goal Attainment 2	Goal Attainment 3	Goal Attainment 4	Goal Attainment 5	Goal Attainment 6	Total per experimental run
Total	121	33	27	21	0	2	202
Average	4,03	1,10	0,90	0,70	0,00	0,07	6,73
Stand. Dev.	3,00	2,14	2,09	1,73	0,00	0,25	1,05

By comparing the action-repetition deadlock totals per goal attainment, it becomes clear that the coordination costs are especially high in the first attempt at reaching the goal state. The next three goal-state attainment attempts show a gradual drop in the total number of occurrences. After that point, the occurrence of action-repetition deadlocks becomes incidental and beyond the sixth goal attainment non-existent. As an average over 30 experimental runs, an action-repetition deadlock emerged 6.73 times. When we compare figure 7.9 and table 7.6 it becomes clear that both sets of data confirm, each in its on way, that the coordination costs occur within the first six goal-state attainments.

7.3.1.2 Coordination costs expressed as no-transition actions

Besides coordination costs in the form of extra state transitions to solve action-repetition deadlocks, another kind of coordination cost was involved within the cooperation process. It concerns the fact that each agent must discover which actions are not allowed, i.e. the actions that have no effect on particular states. The coordination costs emerge because each agent has to find this out and learn it. It takes one action for agent Alpha to detect that action A ("up") is not possible in state {3, 8, 15, 24}, but agent Beta also has to detect that this is not possible, at the cost of an action. This inefficiency of the cooperation process can only be observed by comparing data concerning the applied actions, instead of looking at the state transitions as has been done above.

Figure 7.10 shows the average number of actions applied per goal-state attainment for both conditions, minus the number of transitions, times two[2]. The actions that remain and which are plotted concern the number of actions applied for detecting actions that are not allowed. More generally stated, the curves express the actions not involved in bringing about state transitions.

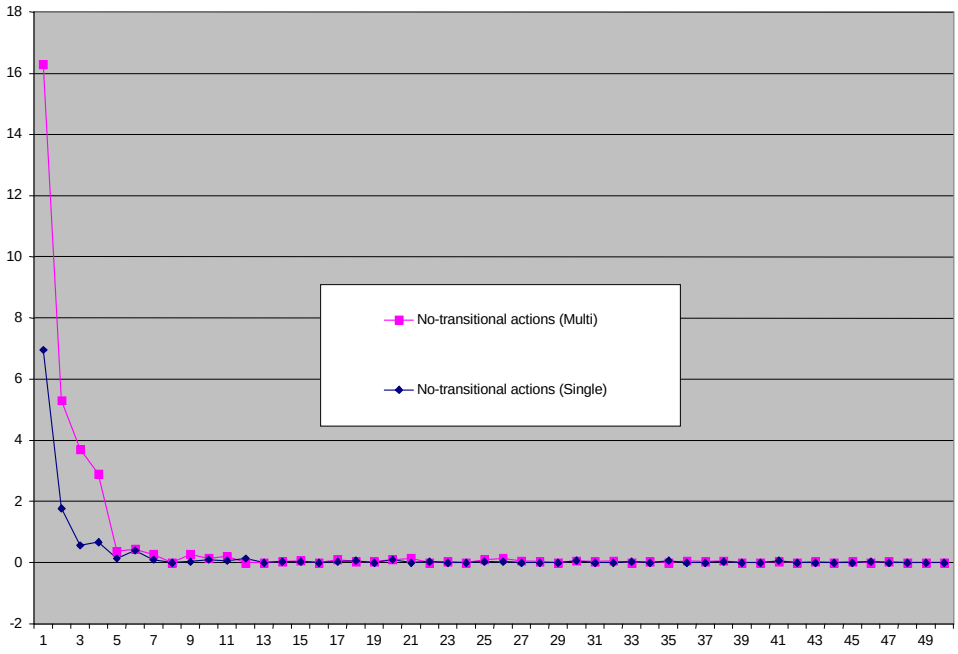


Figure 7.10. Comparing the average number of actions needed per goal attainment for both experimental conditions.

[2] Because every state transition requires a "pass-control" action.

The difference between the “no-transition action” curves indicates the coordination costs involved in terms of applied actions. Figure 7.10 also shows that after the sixth goal attainment, coordination costs in the form of extra action application are no longer involved. In other words, beyond that point the coordination costs are reduced to zero, because there is no need for exploring the task environment in terms of no-transition situation, i.e., every relevant situation is known to both agents.

7.3.2 Summary and conclusion

This section presented the performance data of the cooperative work condition of the Van den Broek experiment. The last two rows of table 7.5 show that the cooperative work condition does not outperform the non-cooperative work condition. Instead, the multiagent effort actually performs less well. Figure 7.8 shows the same outcome in a different way in that the overall performance curve of the cooperative work effort does not cross the curve of the non-cooperative work performance curve. The reason that the non-cooperative work condition outperforms the cooperative work condition is due to the coordination costs. The emergence of cooperation deadlocks in the form of action-repetition deadlocks is the main reason why coordination costs are involved. The coordination costs have been expressed as longer solution paths. This means that more state transitions are needed in order to solve the differences of perceptions. Additionally, coordination costs have also been expressed as the application of additional actions because both agents have to augment and change their task environment model. The involvement of coordination costs is the reason why the cooperative work condition performs less well. The coordination costs decrease very rapidly. Beyond the sixth goal attainment attempt, the coordination costs are reduced to zero. In that problem-solving phase, the cooperative-work condition performs at the same level as the non-cooperative condition.

7.4 Conclusions and discussion concerning the replicated experiment

Within the replicated experiment, the learning performance of the multiagent condition cannot outperform the learning performance of the single agent condition, in principle. The simple reason is that the cooperative condition cannot gain since there are no augmentative grounds for cooperating. Section 3.2.2 discussed a number of reasons why people may cooperate. One of the reasons was augmentative cooperation, which is based on the idea that people are limited in their information-processing capacities. By combining their capacities and aggregating their efforts, an ensemble of individuals can perform a task that would have been impossible, or would have been performed less well by each one of them individually. However, this is not the case. Even more strongly, the single agent performs at almost the theoretical best level. Therefore, the multiple agent condition cannot perform better than best!

The conclusion is therefore that the Soar agent performing the task does not need cooperation, nor does it gain from it; consequently, the cooperation will only hamper the task performance because of the cooperation costs involved.

The second conclusion is that cognitive agents need agent models in order to be able to cooperate and coordinate their actions. The agents must view the other as having goals and taking action to attain those goals. It is only by modeling the agents in this way, that the agents can determine the need for information and knowledge within the other, and are able to coordinate as appropriate. It means that the agents need a knowledge-based coordination mechanism to solve social conflict.

The general principle that emerged from the experiment is that it takes considerable effort to coordinate the cooperation process even under the conditions of:

1. A common goal,
2. Complete benevolence to cooperate, and
3. A task assignment applies.

This means that even when cooperation takes place based on the benevolence and veracity assumption, cooperation and coherent coordination are far from guaranteed. Difficulties of timing and local perspectives can lead to uncooperative and uncoordinated activity. The cooperation condition of the replicated experiment showed that by incorporating the possibility of different interpretations of reality within the multiple agent decision-making models, the plausibility of the model increases, because it mimics real-world situations in which conflicts emerge.

7.5 Conclusions and discussion concerning the original experiment

The first two rows of table 7.5 clearly show that the multiagent condition of the original experiment outperforms that of the single agent. In this version of the experiment, the cooperation effort does pay off. This makes sense because the performance of the single Cohen agent (66.03) is 2.7 times below that of the theoretical best level performance (178.6). Hence, there is considerable room for improvement, and cooperation on augmentative grounds makes sense. The question however is how this augmentative gain has been achieved.

According to Cohen, "this question sheds light on the *organizational* aspects of organizational learning, the way in which learning of the whole may be greater than the sum of the learning of its parts" (p. 175). The relevance of the simulation experiment for organization theory, according to Cohen, is the following. "If, for a given task, the results of learning by multiple agents can be shown to differ from those of learning by a single agent, we can begin to broaden the discussion of the organizational design to include its impact on the ability *of the organization*[3] to learn, as well as on more traditional questions of efficiency" (p. 175). In my opinion, the

[3] Italics added

Cohen experiment clearly demonstrates that the cooperative gain of the multiagent problem-solving effort was based on individual learning characteristics and was not based on the ability of the organization to learn.

Chapter 6 presented a functional account for the reason why the Cohen agent does not perform as well as the Soar agent. The problem of learning capacity as explained is also the source of the performance improvement of the cooperation condition. The overall performance within the multiagent condition was positively influenced due to the cooperation effort. The representational problems reported proved to have less of a delusive or hampering influence within this cooperative work condition. This effect occurs because the multiagent condition has been based on two matrices in which feature-action associations accumulate instead of one matrix. The effect of two learning engines makes the representational situation quite different. In the two-agent condition, much less "transfer of knowledge" between the states occurs. We will make this clear by returning to the same example used in chapter 6 (see section 6.4) but now from the multiagent perspective. In state {3 6 11 22}, agent X will go "down". This will be an improvement and features 6 and 22 will now be more strongly associated with "down" in matrix $M_{(x)}$. In the two-agent condition, however, this implication will not be carried over to the new state {4 6 12 22} because agent Y selects the action on the basis of the contents of matrix $M_{(y)}$. When the system eventually goes "left" from this state, on the basis of matrix $M_{(y)}$ the resultant strengthening of the "left" association with features 6 and 22, again, will not weaken what was learned earlier. In this way, there is much less interference from previous experience, which results in a better overall performance within the multiagent condition. In this way, the two memories work in an augmentative fashion.

Based on the above discussion, three conclusions can be drawn. The first is that for a given task, the results of learning by multiple agents can be shown to differ from learning by a single agent in the case of augmentative gain. The required condition is that the agent must perform significantly below that of the theoretical optimum performance due to a cognitive or learning overload of the task. This is the case in the Cohen model. The capacity of the learning engine to hold information was significantly inadequate. Within the cooperative effort, part of this limitation was lifted. The second is that if it is demonstrated that the results of learning by multiple agents can be shown to differ from those of learning by a single agent, this does not automatically mean that we can begin to broaden the discussion of the *organizational design* to include its impact on the ability of the organization to learn. The explanation presented above is entirely based on learning characteristics of the individual agent and does not postulate any organizational learning effect. The prospect that learning performance improvements within groups can be explained and understood based on cognitive characteristics is seen as support for the methodological individualistic position defended and explained in chapter 2. The third is that the omission of coordination costs within the cooperation process of the Cohen experiment makes the outcome less plausible. From the point of view of organization theory, it is not

plausible that agents do not need to invest in the cooperation process in terms of effort or time, and that they only seem to gain. In contrast, within the replicated experiment, coordination cost was involved in the form of resolving cooperative deadlocks.

7.6 Conclusions and discussion concerning the multiagent level of both experiments

When comparing the multilevel performance of both versions of the experiment, noticeable differences occurred between the two versions. The multiagent performance can be discussed in a quantitatively and a qualitatively way.

The quantitative aspect is that the multiagent condition of the Van den Broek experiment performed better than the multiagent condition of the Cohen experiment, see table 7.5. The performance of Soar agents was double that of the Cohen agents.

The qualitative aspect shows a paramount difference in that in the Van den Broek experiment, “social conflict” emerged, which resulted in coordination costs because the agents put in effort to solve them. The cooperation condition of the replicated experiment showed that by incorporating the possibility of different interpretations of reality within the multiple-agent decision-making model, the plausibility of the model is enhanced, because it mimics real-world situations in which conflicts emerge. Differences in perception of reality are identified as one of the causes of individual conflict. Because conflict does not emerge within the Cohen experiment, no coordination costs are involved. Therefore, the cooperative work condition of the Cohen experiment only gains from the augmentative effects and performs better than the non-cooperative condition, which is considered implausible from the point of view of organization theory.