Modelling the evolution of theory of mind

Lise Pijl
March 2011

Master Thesis
Artificial Intelligence
Dept of Artificial Intelligence
University of Groningen, The Netherlands

Prof. dr. L. C. Verbrugge (Artificial Intelligence, University of Groningen)
Dr. B. Verheij (Artificial Intelligence, University of Groningen)

Abstract

Theory of mind is the ability to attribute mental states to others and understand that these may be different from our own (Premack and Woodruff, 1978). This ability allows us to reason about the beliefs, intentions and goals of others. Some examples of behavior that require theory of mind are cooperation, deception and communication (Baron-Cohen, 1999). We discern different orders of theory of mind. Zero-order theory of mind is not about the mental states of others but about real events. First-order theory of mind allows us to reason about the mental states of others. Second-order theory of mind allows us to reason about what other people think about our mental states. It is assumed that our capability of theory of mind is innate (Brüne and Brüne-Cohrs, 2006), but it takes a few years to fully develop in humans. From the age of two to five, children learn to master first-order theory of mind (Wellman et al., 2001). When children are around six and seven years old, they learn how to apply second-order attribution correctly (Perner and Wimmer, 1985). Whether animals are capable of theory of mind is still under debate (Penn and Povinelli, 2007; Burkart and Heschl, 2007).

There are different theories why humans have evolved theory of mind. It has been suggested that the need for cooperation (Moll and Tomasello, 2007) or the need to deceive and manipulate others or mixed-motive interactions (Verbrugge, 2009) lead to the evolution of theory of mind. A fourth hypothesis is that theory of mind follows from larger group sizes due to changes in habitat (Dunbar, 1992). To these hypotheses we add another one: a competition environment may lead to the evolution of theory of mind.

To test this hypothesis we constructed an agent-based model. In this model, we let agents interact competitively with each other. Agents select their actions based on logical rules. Each agent uses the same mechanism to select their actions and this is common knowledge. The logical rules are evolved over time. The rules in an agent's rule database reflect the agent's capability of theory of mind. Based on the agent's beliefs on the beliefs of its opponent, the agent selects its actions. The advantage of this approach is that the agent's rules and beliefs are insightful.

We ran experiments in which we varied the following parameters: reproduction method (either one-point crossover or linked genes), population size and mutation probability. In 17 of the 18 simulations with one-point crossover the average highest-order rule in the population was second-order or higher. In the simulations where agents reproduced using linked-genes 9

of 18 simulations resulted in an average highest-order rule in the population of second-order or higher. This lower percentage of evolution of second-order rules may be due to the fact that the size of of the rule database was significantly smaller in the simulations where the agents reproduced using linked-genes than with one-point crossover.

We expected that $n+1$-order rules do not persist in the population when agents do not use n-order rules. We found that this was the case in 32 of 36 simulations.

In this project we showed that theory of mind evolves in a competition setting. We conclude that evolving rule databases is an insightful method to investigate the possible driving forces behind the evolution of theory of mind.

Contents

CHAPTER 1

Introduction

To interact with others, we greatly rely on social cognition skills. One of these skills is theory of mind. Theory of mind is the ability to attribute mental states such as beliefs, desires and intentions to others. Since such mental states are not directly observable, we form theories about them. Theory of mind is innate in humans, but it takes several years before the skill is fully developed (Brüne and Brüne-Cohrs, 2006). Whether animals are capable of some form theory of mind is still under discussion.

In the ongoing research on the development of human intelligence many questions are yet unanswered. One of these questions is how and why humans are capable of theory of mind and why we are so good at it compared to other animals. So far, we have only been able to theorize about the driving force behind the evolution of theory of mind. It has been suggested that competition, cooperation or mixed-motive interactions may have played a role (Verbrugge, 2009).

In this thesis a new approach to learn more about the evolution of theory of mind is discussed. We develop an agent-based model where agents interact using rules that may reflect theory of mind. These rules evolve over time. Using this model we test different hypotheses on an agent population.

In this chapter we briefly introduce theory of mind and the research questions and discuss why an agent-based computer simulation is used to investigate these questions. In chapter 2 we discuss research on theory of mind in more detail, elaborate on the research methodology we applied and discuss several design choices of the model based on existing literature and research.

1.1. Theory of mind

Premack and Woodruff (1978) first introduced the term *theory of mind* in 1978. To be capable of theory of mind means that an individual is able to attribute mental states to others and can understand that these mental states may be different from its own.

We discern different orders of theory of mind (image 1.1). An individual without theory of mind has a zero-order theory of mind. First-order theory of mind is the capability to attribute mental states to others. Second-order theory of mind is the capability to attribute mental states to others and also the capability recognize that others may have a theory of mind about your own state.

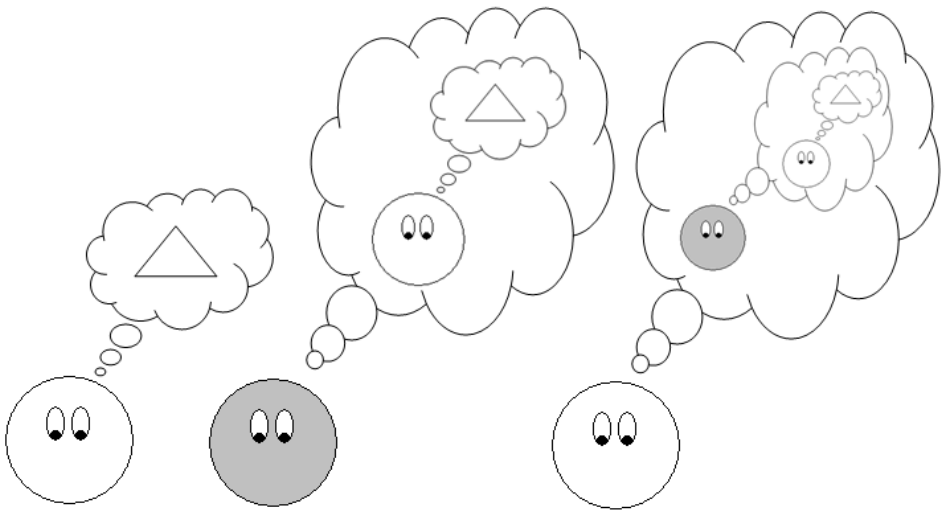


FIGURE 1.1. The leftmost agent has a zero-order theory of mind. It has no theory about the mental states of others. The middle agent has a first-order theory of mind; it has a theory about the mental state of the first agent. The right agent has a second-order theory of mind.

Adults are capable of performing up to fourth-order theory of mind just above chance level (Kinderman et al., 1998). This capability is innate, but it takes several years before the skill is fully mastered. Whether animals are capable of theory of mind is still under discussion. Behavior is found that may indicate that some animals are capable of first-order theory of mind (Hare et al., 2000, 2001; Call and Tomasello, 2008; Moll and Tomasello, 2007; Burkart and Heschl, 2007; Clayton et al., 2007), but there are also other explanations for that behavior that do not involve theory of mind (Penn and Povinelli, 2007; Burkart and Heschl, 2007).

1.2. Research question

At some point in primate development, first-order and higher-order theory of mind arose. There are different theories on the driving force behind the evolution of first-order theory of mind (Dunbar, 1996) and higher-order theory of mind (Verbrugge, 2009): cooperation, mixed-motive interactions and misleading and deception. In this thesis we focus on one hypothesis: competition is a driving force behind the evolution of theory of mind.

To gain insight into how and why higher-order theory of mind evolved, we want to investigate whether competition could be a possible explanation for the evolution of theory of mind. To that end, we built a computer

model where we let heterogeneous and autonomous individuals interact. Individuals will decide their actions using a rule database, which evolves over generations. In this project we address the following questions:

- Can competitive interaction between individuals in a simulated non-spatial environment give rise to the evolution of higher-order theory of mind?
- Will an $n+1$-order attribution evolve only when the larger part of the population has the ability of n-order attribution?

1.3. Why an agent-based computer simulation

To contribute to the research on the question whether higher-order theory of mind arose from competitive interaction, we have built an agent-based computer simulation where the agents choose their actions using a knowledge base. There have been experiments with successfully evolving rule databases before (Van der Vaart and Verbrugge, 2008; Grefenstette, 1992). The gain of this approach is that the evolved rules are insightful. We do not only gain insight into the behavior of the agents, but also in their mental underpinnings.

Obviously we cannot reproduce the evolution of theory of mind in real life. Simulating the evolution allows us to gain insight into the process much more rapidly. Furthermore, we can control the virtual environment and experiment with parameters that may influence the process. This is discussed more thoroughly in chapter 2.

In the next chapter we discuss research on theory of mind in more detail, elaborate on the research methodology we applied and discuss several design choices of the model based on existing literature and research. In Chapter 3 the model is discussed in more detail. In Chapter 4 we present the results of the simulations. These results will be discussed in Chapter 5. In the final chapter we summarize our findings and present our conclusions. The model, the data from the simulations and a PDF version of this thesis can be found on `http://tom.lisepijl.nl`.

CHAPTER 2

Theoretical Background

Theory of mind provides us with the ability to learn from others and teach others, to understand their motivations and goals, and to deceive and mislead others. Humans are quite adept at it; most animals, however, are not. In this chapter we provide an overview of research on theory of mind. We look at the use of theory of mind in humans, and present research on the ability of theory of mind in animals. Next, we discuss several hypotheses on when and why we humans have evolved theory of mind. Then, a short introduction in research methodology on agent-based simulations is given. Lastly, we provide a short introduction on the algorithms that will be used to evolve theory of mind in a computer-simulation.

2.1. Theory of mind in humans

The term *theory of mind* was first coined by Premack and Woodruff (1978). They define theory of mind as follows:

> *An individual has a theory of mind if he imputes mental states to himself and others.*

This means that an individual has a theory of mind if he thinks of other individuals as individuals with their own mental states such as beliefs, intentions and goals. Next, the individual must realize that he himself has beliefs, intentions, goals and so on and that those of others may be different from his own. When interacting with others, reasoning about the mental states of others allows us to predict and understand behavior of others. These mental states cannot be directly observed. This makes reasoning about the mental states of others quite a challenge. We can only observe the behavior of others and derive a theory about the other's mental state. Usually, when we interact with another human, we assume that he too has a theory of mind. Reasoning about what the other believes that we believe requires a higher-order theory of mind, as will be explained next.

Perner and Wimmer (1985) provide an example of the use of first-order and higher-order beliefs. The story is about John and Mary, who are both interested in the location of an ice-cream van. John and Mary are both at the park and it is announced that the van will stay in the park for the afternoon. However, when Mary is on her way home and John is still at the park, it is announced in the park that the van will be at the church for the rest of the day. So, Mary does not know that the ice-cream van moved to

the church. Now, we could ask ourselves the following question: where does Mary go when she wants to get an ice cream? To answer this question we have to access our mental representation of Mary. If we use our theory of mind capabilities, we would answer that Mary will go to the park. After all, she does not know that the ice-cream van moved to the church. She thinks the van is in the park. This is an example of first-order belief attribution.

The story continues. When Mary is on her way home, the ice-cream van passes. Mary asks the driver where he is going and he answers that he will be at the church for the rest of the day. We then ask the following question: where does John think Mary thinks the ice-cream van will be during the rest of the day? To answer this question, we have to access our mental model of John and therein his mental model of Mary. John does not know that Mary knows that the van will be at the church. He believes that Mary believes that the ice-cream van is still at the park. Therefore, we think that John will think that Mary thinks that the ice-cream van is still at the church. This is an example of second-order theory of mind or second-order belief attribution, because we have to access two mental states (John's mental state of Mary's mental state) to answer the question.

In philosophy, there are two approaches to theory of mind: *theory-theory* and *simulation-theory*. In theory-theory it is assumed that when we use theory of mind to predict or explain behavior, we use a theory of human behavior. We are born with cognitive skills, but we need to learn theories to make sense of the world. By observing and experimenting in the social world, we develop and modify theories (Whiten, 2002). We learn that imputing mental states on others help to accurately predict and explain the actions of others. Children learn over time that first-order theory of mind does not always give the expected results and more sophisticated forms of theory of mind are developed.

The alternative is that instead of constructing a theory, we use our own minds to simulate what the other may believe. One way to answer the question 'where will Mary go when she wants to get an ice cream' is to put ourselves in the shoes of Mary and use what we know of Mary's observations to simulate her thought process (Cruz and Gordon, 2002). Thus far, the debate whether theory-theory or simulation-theory describes theory of mind better has not been solved.

As shown, theory of mind allows us to reason about the behavior of others. But it also facilitates other types of behavior. Baron-Cohen (1999) provides us with a list of types of behaviors that require theory of mind (see below). He argues that the evolution of theory of mind like language or bipedalism is a major milestone in primate evolution. After all, language requires a theory of mind. Language allows for changing the knowledge state of the listener. However, this requires one to know that the listener

has indeed knowledge that can be influenced or changed. It requires a theory of mind.

Other types of behavior that require theory of mind according to Baron-Cohen (1999) are:

- **Intentionally communicating with others** Here, communication refers to the acts undertaken to change the knowledge state of the listener. A dog who is barking at a cat may not intend to change the knowledge state of the cat, but simply to make the cat run away. To intentionally inform others requires the belief that others have minds that can be informed.
- **Repairing failed communication** It requires a theory of mind to understand that a message may not be understood and the message needs to be communicated again in a different way.
- **Teaching others** When teaching one wants to change the knowledge state of a less knowledgeable listener.
- **Intentionally persuading others** Persuading is changing someone else's belief about something. Although the goal is often to change the behavior of the other, it is realized by changing the belief and intention state of the other.
- **Intentionally deceiving others** As above, intentionally deceiving others has as goal to change the belief state of the other. In contrast, an animal with camouflage, whose appearance saves it from being eaten by a predator, is not engaging in a deception that requires theory of mind.
- **Building shared plans and goals** When sharing a goal with another person, both must recognize the intention of the other and work out how to coordinate their actions with those of the other to achieve the shared goal. Animals hunting in packs may seem to work together, but often they fail at building shared plans and goals.
- **Intentionally sharing a focus or topic of attention** Looking at the same target at the same time is not shared attention if each is only aware of his own point of view. Shared attention requires a theory of mind only if both individuals are aware of the other being aware of looking at the same target.
- **Pretending** is to temporarily treat an object as if it were another, or as if it had attributes that it clearly does not have. It requires theory of mind in the sense that the pretender has to switch between thinking about his own knowledge of the real identity and the pretend identity.

Despite the usefulness of theory of mind, it takes several years before humans are capable of applying it correctly even though it is assumed that our capability of theory of mind is innate (Brüne and Brüne-Cohrs, 2006).

One-year old infants are already remarkably adept in goal recognition (first-order theory of mind)(Gergely et al., 1995; Woodward, 1998), but they fail at realizing that individuals may have beliefs different from their own. Then, from the age of two to five children acquire full competence on first-order theory of mind tasks (Wellman et al., 2001). When children are around six and seven years old, they learn how to apply second-order attribution correctly (Perner and Wimmer, 1985). Interestingly, it turns out that the children's application of second-order theory of mind depends on the task that has to be carried out (Flobbe et al., 2008). Kinderman et al. (1998) found in an experiment with undergraduates that the participants perform up to fourth-order theory of mind just above chance level. 10 and 11 year-olds perform at third-level theory of mind just above chance (Liddle and Nettle, 2006). This, together with the findings by Flobbe et al. (2008) suggest that children continue developing their theory of mind abilities through their school years.

Liddle and Nettle (2006) also found that, not unexpectedly, the level of theory of mind correlates with a person's social competence. Children in the age of ten and eleven years old were tested on their level of theory of mind. They mention research by others which indicates that the development of theory of mind is strongly influenced by non-heritable factors, such as the quality of parental interaction, quantity of sibling and other family-interaction and social deprivation and maltreatment. In their research they found a correlation between the social competence of the children and their level of theory of mind. Furthermore, a lower theory of mind-level was found in schools with a socio-economic disadvantage.

In this section we discussed what theory of mind is, how it develops in humans and what kind of behavior theory of mind facilitates. We are perhaps not unique in our theory of mind ability. In the next chapter we discuss theory of mind in animals.

2.2. Theory of mind in animals

Humphrey (1976) suggested that the primary driving force behind the evolution of human intelligence is social competition that follows from living in groups. Living in groups has its advantages. The chance of being killed by a predator decreases when living in a large group for several reasons (Dunbar, 1996, p. 17). Humphrey (1976) provides another advantage. He argues that for an individual to stay alive, not much creative intelligence is required. Seemingly advanced techniques such as beating a termite heap with a stick to encourage them to come to the surface, only requires trial-and-error learning or imitation of others and not necessarily advanced reasoning techniques. He guesses that most of the practical problems higher primates face, can be dealt with by learned strategies. When learning those strategies, primates benefit highly from living in a group. Young primates can safely learn by

imitation and trial-and-error whilst being taken care of by other primates. Older animals remain useful as teachers.

However, the presence of dependent animals requires unselfish behavior. Humphrey suggests that although every individual 'is essentially selfish, playing only to win, the selfishness of social animals is typically tempered by what, for want of a better term, I would call sympathy. By sympathy I mean a tendency on the part of one social partner to identify himself with the other and so to make the other's goals to some extent his own' (Humphrey, 1976, p. 313).

If this hypothesis is true, it is not strange that animal behavior that may result from a form of theory of mind is found in animals with complex social lives, such as humans, chimpanzees (Hare et al., 2000, 2001; Call and Tomasello, 2008; Moll and Tomasello, 2007; Burkart and Heschl, 2007) and corvids (Clayton et al., 2007). However, there is critique that behavior that may result from theory of mind can be explained in terms of behavioral rules (Penn and Povinelli, 2007; Burkart and Heschl, 2007).

In this section we present research on chimpanzees and corvids and discuss whether or not these animals are capable of some form of theory of mind.

2.2.1. Primates. Hare, Call and Tomasello (2001) found chimpanzee behavior that could be explained in terms of theory of mind. They devised an experiment where the chimpanzees interacted in a competitive situation. From their experiments they conclude that chimpanzees know what other chimpanzees have and have not seen in the immediate past and that they therefore know what other chimpanzees do and do not know and that they use this information strategically.

The experimental setup is as follows (Hare et al., 2000) (see figure 2.1 on page 10). A dominant and subordinate chimpanzee are housed in four adjacent cages. At the beginning of a trial, each chimpanzee is locked up in one of the extreme cages. The door between the two center cages is fully opened. The apes can access the two center cages through guillotine doors. When these doors are partly raised, the chimpanzees can observe a human placing pieces of food at various locations within the two center cages. They can also see the other chimpanzee looking under its door. Some food is placed somewhere in the open space; other food is hidden behind a barrier so that the dominant chimpanzee cannot see the food. The question is whether the subordinate ape knows that the dominant ape does not know that food was placed behind the barrier and can therefore safely take it. The main finding was that subordinates did indeed go more often to the food that was hidden from the dominant chimpanzee. From this, Hare et al. (2001) conclude that in competitive situations chimpanzees know what other chimpanzees have or have not seen and therefore do and do not know.

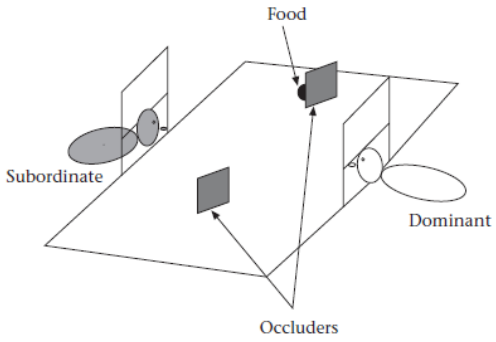


FIGURE 2.1. Experimental setup in the experiments by Hare et al. (2001, pg. 142)

Reaux et al. (1999) performed an experiment on chimpanzees where the chimpanzees begged for food from one of two experimenters. One of the experimenters could not look at the chimpanzee and therefore not observe the begging gesture. When the subject begged for food from the experimenter who could see, the chimpanzee was rewarded with a treat. Six treatment conditions were used. In the first condition, both experimenters were holding opaque screens, but one experimenter used the screen to cover his face. In the second treatment, both experimenters held a bucket, but one had a bucket placed over his head. In the third condition, the eyes of one experimenter were covered by a blindfold whereas the mouth of the second experimenter was covered. In the fourth treatment, one experiment had his eyes closed whereas the other's eyes were open. In the fifth condition, one experimenter was paying attention to the subject whereas the second experimenter was looking at a location above and behind the subject. In the last condition, one experimenter was facing the subject whereas the second experimenter was seated with his back towards the subject.

The chimpanzees did not appear to understand that they should beg from someone who could see them as opposed to someone who could not. The chimpanzees seem to learn stimulus-based rules instead. For example, they seemed to learn that orientation of the experimenter, the face and the eyes related to the receiving of rewards.

The experiments mentioned are cooperation experiments. Moll and Tomasello (2007) characterize a cooperation activity by three features. First, the participants in the cooperative activity share a goal, to which they are jointly committed. Second, and relatedly, the participants take roles in order to achieve this joint goal. And third, the participants are generally motivated and willing to help one another accomplish their role if needed.

From the experiments by Reaux et al. (1999) and other experiments done by others, the Moll and Tomasello (2007) conclude that chimpanzees fail at each of the three features that characterize cooperation. Although

chimpanzees failed in cooperation tasks they are not wholly unaware of the intentions of humans. In other experiments chimpanzees did recognize what the human experimenter tried to accomplish rather than what the human experimenter actually did (see for an overview Call and Tomasello, 2008). Call and Tomasello conclude that chimpanzees understand the actions of others not just in terms of surface behaviors but also in terms of the underlying goals and possibly intentions. But they remark that 'chimpanzees probably do not understand others in terms of a fully human-like belief-desire psychology in which they appreciate that others have mental representations of the world that drive their actions even when those do not correspond to reality.'

Hare et al. (2001) conclude from their experiments and experiments of others that cooperation is a too unnatural situation for chimpanzees. Moll and Tomasello (2007) too were not able to get chimpanzees to cooperate. Hare et al. (2001) suggest that it is likely that primate social-cognitive abilities evolved to a large degree to allow individuals to defeat competitors, so it is in competitive settings that we are most likely to see these abilities expressed.

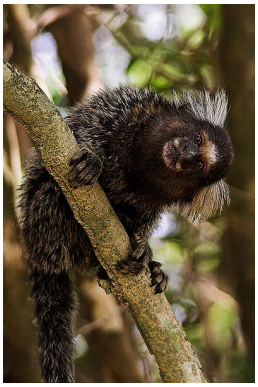


FIGURE 2.2. Marmoset. (Photograph shared by Carmem A. Busko under the Creative Commons Attribution 2.5 Generic license.

The experiment by Hare et al. (2001) was repeated by Burkart and Heschl (2007) with marmosets. Marmosets are small, New World monkeys (see figure 2.2 on page 11). Burkart and Heschl subjected the marmosets to the same test as Hare et al. (2001) did. They also found that subordinate marmosets prefer to take the hidden food over the food visible to the dominant marmoset. To verify whether marmosets indeed know what the other does or does not see, they performed a second experiment on the marmosets. The marmosets had to select one out of six containers where only one container contained food. The gaze of the human experimenter could be used as a cue to select the right container. The containers were attached to a wooden board; three on one side and three on another side. The board

was either vertically or horizontally attached to the wall. The experimenter was positioned in such a way that he could see only one side of the board, so that he could see three containers while the other three were hidden from him (see figure 2.3 on page 12). The marmoset could see the experimenter and all six containers from its initial position.

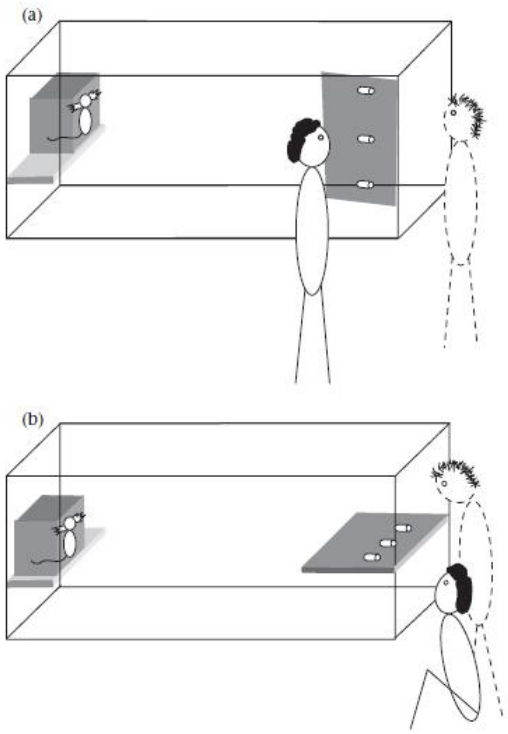


FIGURE 2.3. Experiment by Burkart and Heschl (2007). In a) the experimenter is positioned either left or right from the board. In b) the experimenter is positioned either above or below the board. On each side of the board, three containers are attached but only one is baited. The human experimenter indicated the baited container with a gaze from the corresponding side.

When the marmoset was released from its cage (from which he could not see the experimental setup), the experimenter was already providing the gaze cue (head and body turned towards the container with food in it and looking at the container). The marmoset was allowed to select one container. Overall, the marmosets did not select the container with food in it more than could be expected by random behavior. Neither did the choice of the correct side of the board deviate from chance level. However, the correct position of the baited container (regardless of the side on the board) was selected above chance level. This suggests that marmosets do

not differentiate between what the experimenter could or could not see. They were not able to deal with the visual barrier, but they did follow the gaze of the experimenter to select the container.

Burkart and Heschl (2007) provide two possible explanations for the discrepancy in the findings from these two experiments. The first is that marmosets are able to understand what other individuals do or do not see but they did not show it in the second experiment. This could be due to the unnatural setting of the experiment. A second explanation could be that marmosets do follow the gaze of others, but do not use the knowledge of what the other is looking at to infer what the other may or not know. Instead, in the competition experiment, they may use a two-step mechanism where the marmosets first follow the gaze of the dominant competitor to the visible piece of food and subsequently treat the look-at piece of food as belonging to the dominant competitor. Therefore, the subordinate chooses the remaining piece of food that is hidden from the view of the dominant marmoset.

This hypothesis was tested by presenting two pieces of food to the marmosets. One group had to learn to pick the piece of food that was directly looked at by the experimenter. The other group had to learn to pick the piece of food that was not looked at by the experimenter. If the marmoset would start to grasp the wrong piece of food, the experimenter quickly snatched it away. It was hypothesized that marmosets would learn to use the human gaze as a cue for avoiding a piece a food more quickly than using the gaze as a cue for choosing a piece of food. These results were indeed found. The group that had to choose the piece of food that was looked at by the experimenter performed on chance level. The group that had to choose the piece of food not looked at by the experimenter consistently performed above chance level. This supports the hypothesis that marmosets treat a piece of food that is already looked at by another individual as belonging to that individual and therefore avoid it. This would require no theory of mind at all.

2.2.2. Corvids. Next to chimpanzees and marmosets, corvids (a family of songbirds including crows and ravens) show behavior which could be explained in terms of theory of mind. Corvids live in social societies which share several features in common with chimpanzees. They live in a fission-fusion society (a social group sleep together, but forage in small groups during the day), form long-term alliances and understand third-party relationships. Young corvids experience a long developmental period in which the juveniles interact with individuals who are not necessarily relatives. This allows the juvenile to learn from many different group members. Another commonality between corvids and primates is the relative size of their brains. Corvids have the largest brains relative to their body size of any family of birds, and the same relative size as that of apes (cited in Clayton et al., 2007).

Clayton et al. (2007) did experiments on western scrub-jays. Scrub-jays, like most corvids, hide their food (*caching*), so that they can retrieve it later. However, their caches are susceptible to pilfering by other individuals. Corvids have different strategies to prevent their caches being pilfered. For example, they tend to cache in areas where the density of conspecifics is very low. If other corvids are present, they will wait with caching until potential pilferers are hidden from view by a barrier or until they are distracted. Other protective measures include caching food while being hidden behind a barrier as opposed to caching in full view. This alone does not indicate necessarily that the scrub-jay is aware what the observer does or does not see. The researchers propose a simpler solution. They suggest that the cachers are responding to what they themselves can see. When they cache behind a barrier, the observer is effectively out of sight and therefore perhaps out of mind.

However, scrub-jays also prefer to cache in a shady location as opposed to a well-lit location and far away from an observer as opposed to close to the observer. The 'out of sight, out of mind'-hypothesis would not explain this kind of behavior. This might indicate that a scrub-jay does know what the other does or does not see. Still, approximately 25% of the caches are not at the location which may seem optimal to us, such as behind a barrier or in a shady spot. Earlier research seems to suggest that the cache location also depends the social status of that competitor. For instance, the social relationship between a cacher and an observer affects the choice of the caching location. In an experiment by Clayton et al. (2007) scrub-jays were given food to cache when another scrub-jay was present. The scrub-jay could cache its food close to the other scrub-jay or further away. In presence of a dominant scrub-jay or a subordinate scrub-jay the food was cached further away significantly more often. When a scrub-jay was alone, there was no significant difference between the choice of the two options. Also, when a scrub-jay's partner was present, no significant difference was found. This may indicate that scrub-jays do not perceive their partner as a competitor.

Another protective measure is re-caching food. When scrub-jays are observed by a conspecific caching their food in a certain tray, they re-cache their food. In fact, experiments support the hypothesis that scrub-jays remember which individuals watched them cache and use this information to re-cache their food. Does this mean that scrub-jays are capable of theory of mind or know what others have or have not seen? Clayton et al. (2007, p. 519) say on this: 'In short, these studies show that scrub-jays keep an eye on competition and protect their caches accordingly. Such behaviour would appear to meet the behavioural criteria for one form of theory of mind, namely knowledge attribution, if by the term we mean the ability to attribute different informational states to particular individuals.'

However, it turned out that not all scrub-jays re-cache their food. In a previous experiment it was found that re-caching behavior does not only

depend on whether the scrub-jay was observed earlier by another scrub-jay, but it depended also on previous experience as a thief. Experienced thieves engaged in re-caching much more often than birds who had never been thieves in the past. This means that re-caching is not innate behavior. This type of behavior is called experience projection. According to Clayton et al. (2007) experience projection 'refers to a second form of theory of mind, namely the ability to use one's own experiences, in this case of having been a thief, to predict how another individual might think or behave, in this case what the potential pilferer might do.' So does the behavior of the scrub-jays, chimpanzees and the marmosets as described above result from a theory of mind?

2.2.3. Alternative explanation for theory of mind-like behavior. Penn and Povinelli (2007) provide an alternative explanation for the behavior of the chimpanzees and corvids. They show that the behavior of chimpanzees in the experiments of Hare et al. (2000, 2001) and the behavior of corvids in the experiments of Clayton and Emery (Clayton et al., 2007) do not necessarily require a theory of mind. For example, in the experiment run by Hare et al. (2001), it is possible to explain the behavior of the chimpanzees not in terms of theory of mind, but rather in terms of perception and memory of recent events. For example, the strategy of the subordinate could simply be something like 'Don't go after food if a dominant who is present has oriented towards it'.

For the experiments on corvids (Clayton et al., 2007) the same criticism holds. Although it is possible that corvids are capable of reasoning on the goals or observations of other corvids, it is equally possible that they have simply learned rules as 're-cache food if a competitor has oriented towards it in the past' or 'attempt to pilfer food if the competitor who cached it is not present' or 'try to re-cache food in a site different from the one where it was cached when the competitor was present' and so on.

2.2.4. Summary. Based on the experiments described above, we conclude that there is no conclusive evidence that chimpanzees and corvids are capable of full theory of mind as humans do. Although the behavior of the chimpanzees and corvids can be explained in terms of theory of mind, they can also be explained in terms of behavioral rules.

Behavior that might indicate a theory of mind is so far only observed in a competition setting, as mentioned above. Therefore, we will examine in this Master thesis the evolution of theory of mind in a competition setting.

2.3. Evolution of higher-order attribution in humans

As inconclusive as research about the existence of theory of mind in non-human primates and other animals is, just as inconclusive is research about the origins of first-order theory of mind and higher-order theory of mind in

humans. There are, however, theories on the evolution of first-order theory of mind and higher-theory of mind, which we will discuss in this section.

2.3.1. When did theory of mind evolve. When theory of mind first arose among humans is not clear. We will discuss two possible options (Baron-Cohen, 1999). The first holds that the capability of theory of mind in humans evolved as early as 6 million years ago. The second hypothesis holds that theory of mind evolved 30,000 years ago.

If existing monkey and ape species have a theory of mind, we can assume that theory of mind evolved as early as the common ancestor between us. This would have occurred 6 million years ago. However, there is no convincing evidence that apes are capable of applying theory of mind (see section 2.2). Also, none of the eight behaviors that require theory of mind as mentioned by Baron-Cohen (1999) (see page 7) are displayed by non-human primates or other animals.

The second theory holds that theory of mind arose 30,000 years ago. This theory is supported by palaeo-archaeological evidence. Around that time, statues of impossible entities were made, such as the half-man-half-lion ivory statuette (see figure 2.4) from Holhenstein-Stadel, Germany, and the painting of the half-man-half-reindeer (see figure 2.5), in Trois-Freres, France, both dated around 30,000 years ago. These forms of art are interesting because they are representations of imaginary persons. This shows that the artist was capable of pretend play, since the animal depicted never existed, only in the artist's imagination. Pretend play requires a theory of mind (see page 7).

Second, archaeological evidence shows that our ancestors were concerned with death, because they buried other individuals. Around 28,000 years ago, dead persons were adorned with jewelry. This might show that the decorator cared about how other people either now or in the afterlife perceived the adorned person. It requires theory of mind to care about how other people perceive oneself or are perceived by them.

Interestingly, it is during that same period that the life span of individuals began to increase significantly, indicating increased survivorship of older adults through human evolution (Caspari, 2004). Caspari and Sang-Hee (2006) suggest that this is perhaps not directly a result of some biological attribute, but the result of cultural adaptations. So what does this tell us about the evolution of theory of mind in humans? As far as we can see, not much at this point. But the research mentioned above does show that archeology and anthropology might provide some valuable clues about the circumstances at the time humans might have evolved the ability to apply theory of mind.

2.3.2. Possible drives for the evolution of higher-order theory of mind. In this Master's project we are not so much concerned with the question *when* theory of mind evolved in humans, but rather we want to know *how* and *why* we humans evolved this ability. Different theories are

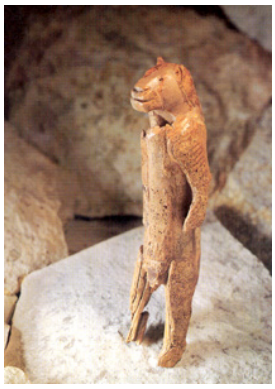


FIGURE 2.4. Half-man-half-lion statuette from Holhenstein-Stadel, southern Germany

FIGURE 2.5. Half-man-half-reindeer painting from Trois-Freres, France

proposed concerning the evolution of higher-order theory of mind in humans. First, we discuss why humans have evolved first-order theory of mind in the first place.

As mentioned in section 2.2 living in social groups provides several advantages. Among others, it allows individuals to learn from each other. First, young animals are allowed a prolonged period in which they can freely experiment and explore while being taken care of by older animals. Second, they are brought into contact with older individuals from which they can learn by imitation. However, since both young and older animals are dependent on others, such learning and caretaking requires unselfish sharing from other animals (Humphrey, 1976). For this to work, it is assumed that theory of mind emerged as an adaptive response to increasingly complex social interaction (Brüne and Brüne-Cohrs, 2006). First-order theory of mind allows individuals to understand and predict the behavior of others to

a certain degree. So why should we benefit from a higher-order theory of mind?

Some authors claim that higher-order social cognition arose because of the need for cooperative planning; others that it provided social glue by enabling gossip and language (Dunbar, 1996). Dunbar (1992) argues that the need for cooperative planning and language follows from larger primate group sizes that were the result of a change of habitat. Others suggest that the main purpose of higher-order social cognition was to manipulate and deceive competitors (cited in Verbrugge, 2009) or recognize deception (Brüne and Brüne-Cohrs, 2006). To these theories, Verbrugge (2009) adds the theory that the need for mixed-motive interactions such as negotiation explains evolution of higher-order theory of mind.

To gain insight into how and why higher-order theory of mind was evolved, we suggest to build a model in which we try to facilitate the evolution of a higher-order theory of mind. In particular, we test the theory that the situation of agents in a competitive environment might result in the evolution of higher-order theory of mind. If we can show that competitive interaction between individuals can result in the evolution of higher-order theory of mind, the competition hypothesis could provide an explanation for the fact that humans evolved higher-order theory of mind.

2.4. Research methodology

To test our hypothesis we will build and examine an agent-based computational model. An agent-based computational model makes it possible to find possible explanations for regularities that results from the local interactions of heterogeneous agents. Often, an explanation cannot be found otherwise, because experiments (for example in real life) would take too long, involve too many test subjects or because it is not possible to vary just one property of the system. Phenomena that have been researched using agent-based computational models are economic classes, spatial unemployment patterns, segregation, epidemics, traffic congestion patterns, alliances and voting behaviors (Epstein, 2006).

In an agent-based computational model, heterogeneous agents interact locally with each other. Each individual determines its own behavior based on local information (*bounded information*). The setup of the individuals and the environment is the *micro-specification*. If a microspecification generates a macrostructure of interest (such as traffic congestion patterns or segregation, as mentioned above), then the microspecification provides a candidate explanation. Generating a macroscopic regularity from a microspecification may provide understanding of the regularity. This way of finding explanations for macro regularities is called *generative social science* (Epstein, 2006). The motto of generative social science according to Epstein could be: 'If you did not grow it, you did not explain its emergence'.

In this project, we will try to grow evolution of higher-order theory of mind. As mentioned before, our microspecification will be the competitive setting. Growing higher-order theory of mind from a competition setting may show that the competition hypothesis is a feasible one.

2.5. Modeling the evolution of higher-order theory of mind

To gain insight into how and why higher-order theory of mind was evolved, we will investigate whether theory of mind is evolved in a population of agents who interact competitively. We prefer to model evolution over learning. In this project, we will evolve theory of mind in terms of rules. Below, we argue why.

2.5.1. Learning or evolution. There are two options for developing higher-order theory of mind in a computer simulation: learning and evolution. Nolfi and Floreano (1999) describe the difference between evolution and learning as follows.

> Evolution is a process of selective reproduction and substitution based on the existence of a geographically-distributed population of individuals displaying some variability. Learning, instead, is a set of modifications taking place within each single individual during its own life time. Evolution and learning operate on different time scales. Evolution is a form of adaptation capable of capturing relatively slow environmental changes that might encompass several generations, such as perceptual characteristics of food sources for a given bird species. Learning, instead, allows an individual to adapt to environmental changes that are unpredictable at the generational level.

This, however, does not mean that every skill is either only evolved or only learned. For example, although it seems that humans evolved the innate capability to learn and express language, much learning is required before a human is able to use language correctly. This might also be true of higher-order theory of mind. The ability is innate, but it requires experience to bring it to fruition. Because the ability is innate, we prefer evolution over learning.

2.5.2. Representing theory of mind in rules. We test the theory that the situation of agents in a competitive environment might result in the evolution of higher-order theory of mind. If we can show that competitive interaction between individuals in a simulation results in the evolution of higher-order social cognition, the competition hypothesis could provide an explanation for the fact that humans evolved higher-order theory of mind. To achieve this, we will supply every agent with a rule database that will be evolved over time. If higher-order theory of mind arises, this will be clearly visible in the evolved rules.

Evolving rules in agent-based simulations has been done before (Van der Vaart and Verbrugge, 2008; Grefenstette, 1992) and the results are promising. The agents in these experiments were quite capable of adapting to their environment. The experiment described by Van der Vaart and Verbrugge (2008) was a pilot project where the possibilities of evolving rules were explored. The rules of the agent determined the actions. Although there was no intention to specifically evolve first-order theory of mind, the specifications did allow for the evolution of rules that represent theory of mind. In this project, we will explore the possibilities of evolving rules that represent first-order and higher-order theory of mind further.

The advantage of evolving rules is that existing knowledge can easily be incorporated in the initial knowledge database (Grefenstette, 1992). A second advantage is that rules are insightful. We not only gain insight into the effective behavior given a certain environment, but also into the mental reasoning processes behind the action selection process (Van der Vaart and Verbrugge, 2008).

2.5.3. Genetic algorithms. The rules will be evolved using a genetic algorithm. Genetic algorithms are based on evolution (for more information on genetic algorithms, see Mitchell, 1996) . A living thing has genes that hold information how to build and maintain an organism. Genes are usually inherited from one's parent(s). Roughly, one can think of a gene as a trait, such as eye color. For a mouse, it may encode the color of his fur. In many cases, it is to the mouse's advantage if the color of its fur matches the colors of its environment, so it is not easily noticed by predators. Genes may change due to *mutation*. This means that genes are not correctly copied, but slightly altered. In many cases, a mutated gene does not result in a directly observable change. If, however, it turns out that the color of the fur of a mouse which lives in a dark environment is a few shades lighter than that of its parents, it may get eaten before it gets a chance to reproduce. Or, if the new tone of fur has a better match to the color of the environment, it may be very successful at reproducing, because it will not get eaten. In terms of evolutionary algorithms, it has a high *fitness* compared to its conspecifics.

Genetic algorithms use these principles from evolution. In our case, the genes are the rules used to select actions (see next chapter for details). The success of the agent's actions determines the agent's fitness. The agents with the highest fitness will reproduce. We will use two reproduction methods in this project: *linked genes* or *crossover*.

Using linked genes reproduction only requires one parent. The offspring inherits the genes of its parent. This method resembles cloning. For crossover the genes of two parents are used to construct the offspring. The genes of each parent are split at a random point. A child receives one part of the genes from every parent. This is called *cut and splice*.

Next, the genes received are subject to some modifications due to mutation. In nature this may be caused by damage due to chemicals, radiation

or viruses or errors that occur during DNA replication. Mutation allows animals or virtual agents to adapt to their environment.

2.5.4. Domination. We already mentioned that individuals will be placed in a competitive setting. Individuals are constantly interacting with each other. The outcome of an interaction depends on the rules an agent uses (see next chapter for details). These rules are evolved using a genetic algorithm. However, to be able to do that, a suitable fitness function is required. A fitness function determines how well the agents perform. Agents with the highest fitness values are allowed to reproduce. Part of the fitness function is the dominance value. The dominance value reflects the social status of an individual. A higher dominance value may lead to better access to food, mates, or safe locations (Hemelrijk, 1999). Losing a fight will decrease the individual's dominance; winning will increase it.

Hemelrijk's DomWorld, an agent-based model used to explain dominance interaction and social structures, contains a formula (2.5.1) that is used to change the dominance of two interacting individuals (Hemelrijk, 1999; Hemelrijk et al., 2003). The parameter StepDom varies from 0 to 1 and represents the intensity of aggression. A high value results in a great change in the domination value when updating it. A low value results in a small change.

The variable w_i describes whether agent i won or lost the dominance interaction, where $w_i = 1$ means that it won and $w_i = 0$ that it lost (2.5.2). Losing or winning an interaction is based on chance; if the relative dominance value of individual i is larger than a random number between 0 and 1, individual i wins the dominance interaction.

$$DOM_i+ = (w_i - \frac{D_i}{D_i + D_j}) * STEPDOM$$

$$DOM_j- = (w_i - \frac{D_i}{D_i + D_j}) * STEPDOM \tag{2.5.1}$$

$$w_i = \begin{cases} 1 & \frac{DOM_i}{DOM_i + DOM_j} > RND(0,1), \\ 0 & else. \end{cases} \tag{2.5.2}$$

What is interesting about this equation is that even when both individuals start with the same dominance value, which means that their chance of winning a dominance interaction is equally large, one individual has a significant larger dominance over the other after a few turns. This is called the winner-loser effect. After winning a dominance interaction, the chance increases that the next dominance interaction is also won, because of the increased dominance value after winning the first interaction. This effect has been observed in real animals too. Research on insects, rodents, molluscs, fish and birds indicate that a previous aggressive interaction can influence the individuals' behavior in subsequent interactions (Chase et al., 1994).

In our model, winning or losing an interaction is not based directly on chance but on the outcome of a fight. When two individuals have won the same number of fights, the winner-loser effect will show. The winner-loser effect is interesting. However, we choose to omit this effect in our project. If any regularities arise in the model, we would have to decide if this is a result of the winner-loser effect or because of other properties of the model. We choose to omit the winner-loser effect to reduce the number of factors that may influence the evolution of theory of mind in this model. Therefore, we use another method of calculating the dominance value. We propose not to use the formula in (2.5.1), but simply use the average number of fights won to represent the dominance value (2.5.3). This will make the dominance values more insightful when interpreting the results of the experiments. The chosen formula and its effects are discussed in section 3.4.

$$DOM_i = \frac{fightsWon_i - fightsLost_i}{fightsWon_i + fightsLost_i} \tag{2.5.3}$$

We will discuss our model more specifically in the next chapter.

2.6. Chapter summary

In this chapter we introduced theory of mind. Theory of mind allows us to attribute mental states to oneself and others. Furthermore, it also allows us to understand that the mental states of others may be different from our own. There are different orders of theory of mind. Zero-order theory of mind is not about the mental states of others. An example of a zero-order sentence is: 'I think the ball is in the basket'. First-order theory of mind is about the mental states of others. An example of a first-order sentence is: 'I believe Sally believes the ball is in the box'. Second-order theory of mind is the ability to understand that others may have a first-order theory of mind about you. An example of a second-order sentence is: 'I think Sally believes that I believe the ball is in the box'.

Our ability in theory of mind is assumed to be innate (Brüne and Brüne-Cohrs, 2006). From the age of two to five children learn to master first-order theory of mind (Wellman et al., 2001). When children are around the age of six or seven they have learned to apply second-order theory of mind (Perner and Wimmer, 1985). Humans can perform up to fourth-order theory of mind at just above chance level (Kinderman et al., 1998). Whether animals are capable of theory of mind and to what extent, is still unclear. However, theory of mind-like behavior in animals was most often observed in a competition setting. Therefore, in this project we will test the hypothesis whether a competition environment could be a driving force behind the evolution of theory of mind.

To investigate whether competition might have been a possible driving force behind the evolution of theory of mind, we use an agent-based model. By programming agents with simple rules and letting them interact

with each other and their environment interesting phenomena may be reproduced. Reproducing an interesting phenomenon in this way allows us to learn something about how it could have emerged, which is, in this project, theory of mind. Agents will interact with each other competitively. The agents select their action using rules. The agents who perform best will be selected for reproduction. Agents reproduce using either the linked genes method or crossover. In this model, the agent's genes are the rules in the agent's rule database. In the next chapter, the model will be discussed in further detail.

CHAPTER 3

An agent-based model of the evolution of theory of mind

To answer the research questions presented in section 1.2 on page 2 and to investigate whether evolving the rule database of rule-based action-selection agents is a successful approach when simulating theory of mind, we built a model where we let heterogeneous, autonomous individuals interact competitively with each other in a non-spatial environment. Agents select their actions using their rule database, their memory of earlier interactions and their beliefs about their opponent's rules and beliefs. The agent's capability of theory of mind is found in the rules that are stored in the agent's rule database. The rule database will be evolved over time.

Firstly, we introduce the task that the agents have to perform and take a look at the tasks that have been omitted for this project but may be used in future research. Secondly, in section 3.3 an explanation of the construction of the rules and rule database is given. We will also explain how agents can learn from experience and reason about the rules used and believed by others. Next, we will provide some more information about the competitive setting (see section 3.4) and look at the evolution of the rule database of the agents (see section 3.5). Lastly, we will explain the experimental setup in section 3.6. In Chapter 4, the results of the experiments will be discussed.

3.1. Task choice

If the model is to provide a plausible cause for the evolution of theory of mind, the task the agents will perform must fulfill two requirements. Firstly, the application of higher-order attribution must provide the individual with a substantial evolutionary advantage over those who apply a lower-order attribution. Secondly, the task should be ecologically plausible. We selected a competition task that fulfills these requirements. Several other tasks were also considered, but have been omitted. We do discuss these tasks, because they may be suitable for future research.

Because of the first requirement, a cooperative task was omitted. This is because if a group of individuals successfully completes a task, every individual receives a small reward. This means that individuals with a first- or lower-order attribution receive the same reward as an agent with a higher-order attribution. Thus, we expect that evolutionary pressure is low.

Next, as mentioned in the previous chapter, chimpanzee and corvid behavior that could perhaps be explained in terms of theory of mind always took place in a competitive situation, never in a cooperative setting. The cooperation tasks that were considered and might be considered for future research were hunting and a coup d'état (Erdal and Whiten, 1996).

Other suitable tasks are tasks of a Machiavellian Intelligence nature and tasks related to commitment (Mant and Perner, 1988). Machiavellian Intelligence may be demonstrated by lying, deceiving, misleading and making and breaking alliances (for examples, see Byrne and Whiten, 1988). These types of tasks seem to require higher-order theory of mind, at least at first sight. These tasks and tasks related to commitment are well worth investigating, but in this project we have chosen another, simpler task.

For now, we propose a competition task where the goal is not to change the knowledge state of the other (as would be the case in a Machiavellian Intelligence task) nor the behavior of the other. Here, theory of mind is used to reason about the action of the agent's opponent. If we succeed in simulating the evolution of higher-order theory of mind in a competition setting, other settings, such as cooperation or commitment, can be explored.

With the above requirements in mind, the following task was constructed.

3.2. Proposed task

There are often conflicts about food and dominance in a primate population (de Waal, 1982). An argument may sometimes result in a physical fight between two individuals. In this model, the individuals fight each other and try to win these fights by correctly reacting to the predicted action of the opponent. In each interaction an agent either defends or attacks. There are two possible actions: an agent defends or attacks with left or with right. If an agent's attack is not blocked, i.e. the attacker selects a different direction than the defender does, it wins the interaction (see figure 3.1). Winning a fight provides the individual with an increased dominance value. A higher dominance value may result in more access to food, mates or safe hiding places (Hemelrijk, 1999). In this experiment, an agent's chance to reproduce is proportional to the agent's dominance value.

Each agent uses rules in order to choose which action it will take. These rules are stored in a rule database. The rule database does not change during the agent's lifetime. Based on its experience with its opponent (which is stored in memory), the agent forms beliefs about the rules used by its opponent. The agent also forms beliefs about the beliefs of the opponent about the agent's own rules. These beliefs are stored into memory. The beliefs are used to select an action using the rules in the agent's rule database.

To execute this task successfully, reasoning about another agent's rules to select a suitable action is not the only method. For example, an agent could select random attack and defend actions or choose its action based

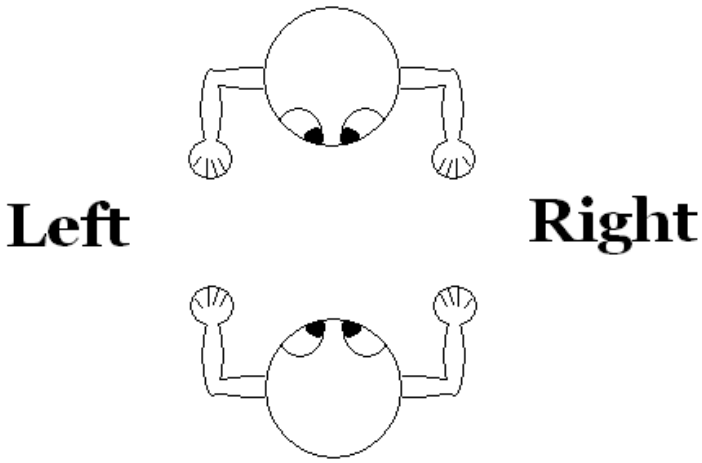


FIGURE 3.1. Two agents interacting. The side an agent selects to attack or defend with (left or right) is not relative to the agent. When one agent attacks left and the other defends right, an attack was successful. When an agent attacks left and the opponent defends left, the attack is blocked.

on the previous action of its opponent. As a second alternative, the agent could change its rules during its lifetime or choose to skip the usage of some rules. A more complicated task such as a Machiavellian Intelligence-task may not have these alternative solutions. However, if the proposed approach is successful, other tasks can be investigated using the same method.

3.3. Inferring actions and beliefs

In this section we show how an agent selects its action and how the agent reasons about the rules used by its opponent and about the beliefs of its opponent. First, we show how the rules of an agent are constructed and how they are stored in the agent's rule database. Next, we discuss the prior knowledge of the agents about each other and their task. Then, we discuss how each agent chooses its action based on the following:

- the rules in the agent's rule database,;
- memory of previous interactions with its opponent;
- beliefs about the opponent's rules,;
- beliefs about the opponent's beliefs about the agent's rules.

3.3.1. Rule construction. The rules that are used by the agent and stored in its rule database are evolved over time and do not change during the agent's lifetime. The rule database contains rules of two types: attack rules and defend rules. A rule consists of a *condition* and an *action*:

$$[condition \rightarrow action]$$

If we look at the rule from an agent's point of view, the rule can be read as 'if *condition* is the case, then I will execute the following *action*'. The rules are evaluated in the order in which they are stored in the rule database. The order of the rules does not change during the agent's lifetime. The first rule in the rule database for which the condition is true, is executed.

First, we provide several examples of valid rules. Then, the syntax of valid rules is discussed.

3.3.1.1. *Zero-order rule.* First, we introduce a zero-order rule. No theory of mind is required to use a zero-order rule. The condition of a zero-order rule is always true. Below is an example:

Example 1.

$$[TRUE \rightarrow RIGHT_A]^S$$

The condition of this attack rule is $TRUE$. This means that the corresponding action is always executed. The subscript A denotes that the rule is an attack rule: $RIGHT_A$ is an attack action where the agent attacks with right. The subscript D denotes that the action is a defend action. The superscript S means that this rule is used by the agent itself (S stands for 'self'). Another valid zero-order rule is the defend rule below:

Example 2.

$$[TRUE \rightarrow LEFT_D]^S$$

A zero-order rule is a behavioral rule. It does not require any kind of reasoning. It is perhaps comparable to left- or right-handedness. A right-handed person prefers to use his right hand over his left one, unless it is beneficial to do otherwise.

3.3.1.2. *First-order rules.* Now, we introduce first-order rules. We call these rules first-order rules, because they can be explained in terms of first-order theory of mind. It requires the agent to form beliefs about the rules its opponent uses. Below is an example of a first-order rule:

Example 3.

$$[B_S[TRUE \rightarrow LEFT_A]^O \rightarrow LEFT_D]^S$$

This first-order defend rule means the following: if the agent believes that its opponent uses the attack rule $[TRUE \rightarrow LEFT_A]^O$ then the agent defends with left. This allows an agent to select an appropriate action when it has established a belief about a rule its opponent uses.

3.3.1.3. *Second-order rules.* However, to outsmart an opponent who uses first-order rules, an agent requires second-order rules. Second-order rules are not only about the agent's beliefs about the opponent's rules, but also about the agent's beliefs about the opponent's beliefs about the agent's rules. Below is an example of a second-order rule:

Example 4.

$$\Big[B_S\big[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\big]^O \rightarrow LEFT_A\Big]^S$$

Again, we read the rule from the viewpoint of an agent that interacts with its opponent. It states that if the agent believes the opponent uses the rule $[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D]^O$, then the agent will attack with left. B_O denotes the belief of the opponent. In this rule, the belief of the opponent concerns the rule of the agent itself: $[TRUE \rightarrow RIGHT_A]^S$. The opponent's rule the agent is considering could be read as follows: 'if the opponent believes that I use the attack rule $[TRUE \rightarrow RIGHT_A]^S$ the opponent will defend with right'.

3.3.1.4. *Third-order rules.* Only a higher-order defend rule, in this case a third-order defend rule, can counter a second-order attack rule. Below is an example of a third-order defend rule which counters the second-order attack rule above.

Example 5.

$$\Big[B_S\Big[B_O\big[B_S[TRUE \rightarrow RIGHT_A]^O \rightarrow RIGHT_D\big]^S \rightarrow LEFT_A\Big]^O \rightarrow LEFT_D\Big]^S$$

It does not end here, however. Fourth- and higher-order rules are also possible.

3.3.1.5. *Rule syntax.* Above we gave an example of zero-order, first-order and higher-order rules. Now, we discuss what a valid rule looks like. To that end, we introduce two rule building blocks which can be combined to create any valid rule. We impose as few constraints as possible on the contents of the rule.

The first valid rule is a zero-order rule:

Rule building block 1. $[TRUE \rightarrow action_\alpha]^m$

For this rule and the following rules the following holds:

- *action* can either represent a $LEFT$ or $RIGHT$ action;
- $\{\alpha..\zeta\}$ each represent the type of the rule: this can be an attack rule (A) or defend rule (D);
- $\{m..z\}$ each represent the agent whose rules are considered: this can be either the agent itself (S) or its opponent (O). For each $\{m..z\}$ it holds that they may be equal or different.

Next, we introduce the second building block used to create first- and higher-order rules:

Rule building block 2. $[B_m rule^n_\alpha \rightarrow action_\beta]^o$

At the position of $rule_{\alpha}^{n}$ we insert a rule of type α that is used or believed to be used by agent n. A rule based on either the first or on the second building block can be inserted here. Note that for a rule to be used when selecting an action, it should be the agent's rule and not the opponent's rule. Rules in the database that are of the form $[condition \rightarrow action_{\alpha}]^{O}$, for example due to mutation, will not be used.

3.3.2. Rule database. The agent's rules are stored in its rule database. When selecting and using rules, the following assumptions are made:

- The rules in the rule database do not change, nor does the order change during an agent's lifetime. The rules only change during the reproduction step (see section 3.5 on page 46).
- The action of the first rule of which the agent believes that the condition is true is executed. This means that when an agent establishes no new beliefs, it keeps executing the same action. It also means that even though the rule database contains higher-order rules that would benefit the agent, they are never reached if a zero-order rule is evaluated first.
- When an agent is initialized, each agent receives a preference (comparable to left- or right-handedness). The first generation of agents receives a random preference; later generations inherit their preference from one of their parents. When no matching attack or defend rule is found, the following rule is used: $[TRUE \rightarrow preference_{\alpha}]^{S}$. This rule is then appended at the end of the rule database.

3.3.3. Memory. As mentioned before, the action that is selected by the agent is determined by the agent's rule database and the agent's memory. Each agent's memory contains the following data for every opponent:

- Action of the agent and the action of its opponent in past interactions (interactions are removed from memory when they are no longer useful, see section 3.3.6);
- the rules the agent believes the opponent uses;
- rules the agent believes the opponent believes the agent uses.

It should be noted that agents are not allowed to believe that the opponent uses two different rules rules that have the same condition and type (attack or defend) but a different outcome. For example, an agent is not allowed to believe that its opponent uses both rules below:

$$[TRUE \rightarrow LEFT_{D}]^{O}$$
$$[TRUE \rightarrow RIGHT_{D}]^{O}$$

In this model, every agent has the same reasoning mechanism and this is known by all agents (see section 3.3.4). We explained earlier that the first rule in an agent's rule database of which the agent believes that the condition is true, is used. So it cannot be the case that an opponent uses two or more rules with the same condition (although they can be in the

rule database, they are just not being used), just as the agent does not use two rules with the same condition. The same holds for higher-order rules. Therefore it is not possible that an agent believes that its opponent uses two or more rules of the same type that have the same condition but a different action. Neither can the agent believe that its opponent does so. Allowing agents to believe that opponents use rules that are the same except for the actual action would violate the assumption that agents are aware that every agent uses the same inference engine.

3.3.4. Initial knowledge about the task and opponents. Each agent uses the inference engine that is described in section 3.3.6. Each agent reasons about its opponent's beliefs and rules perfectly, in the sense that it does not make mistakes when establishing beliefs about its opponent's beliefs and rules. Furthermore, each agent knows that its opponent reasons with the exact same reasoning mechanism and deduces the same beliefs as the agent itself would given the same history of interactions and rule database.

This approach is fruitful if we want to evolve theory of mind because, as discussed in section 3.3.1, the construction of the rules is free from constraints: the agents are allowed to reason about every type of rule or belief by either itself or its opponent. It is difficult to allow an agent to establish beliefs about its opponent's rules and beliefs, if an agent does not have insight into the rules an opponent uses and also knows nothing about the reasoning mechanism of the opponent. Therefore, we make the assumption that every agent reasons perfectly and that every agent knows that its opponents do so too and do not make mistakes or cheat.

3.3.5. Example: inferring actions and beliefs. In this section, we give a brief overview how an agent selects an action using its inference engine and how it reasons about rules used by its opponent and about its opponent's beliefs. In the next section, we explain how the inference engine works and provide a more detailed example.

In this example one agent attacks an opponent several times. The agent's rule database (see table 3.1) contains four attack rules.

$$\Big[B_S\big[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\big]^O \rightarrow LEFT_A\Big]^S$$
$$\big[B_S[TRUE \rightarrow LEFT_D]^O \rightarrow RIGHT_A\big]^S$$
$$\big[B_S[TRUE \rightarrow RIGHT_D]^O \rightarrow LEFT_A\big]^S$$
$$[TRUE \rightarrow RIGHT_A]^S$$

TABLE 3.1. A rule database that contains four attack rules. The first rule is a second-order attack rule. The second and third rule are first-order attack rules. The last rule is a zero-order attack rule.

3.3.5.1. *First contact.* When an agent interacts for the first time with a certain opponent, it has no previous experience with that opponent. Therefore, all rules that involve beliefs about either the opponent's rules or the opponent's beliefs do not match, since there is no information on which to base these beliefs. The only rules that an agent can use in the first turn are zero-order rules. In the first interaction, the agent attacks. The only rule that is applicable during this turn is the zero-order rule $[TRUE \rightarrow RIGHT_A]^S$. Therefore the agent will attack with right. For now, let us assume that the agent's opponent defends with left.

3.3.5.2. *Second interaction.* In the second interaction we let the same agent attack again. In the first attack the agent could not use the first three attack rules, since the agent had no experience with its opponent on which to base the corresponding beliefs. However, when evaluating its rule database in the second interaction, the agent does have experience it can use. When the agent starts to evaluate the rules in its database, then new beliefs are established.

Although the first rule cannot match because more interactions are required to know that the agent's opponent applies the rule $\left[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\right]^O$, the agent does learn that its opponent may know that the agent uses the rule $[TRUE \rightarrow RIGHT_A]^S$. When looking at the second rule, the agent can derive that its opponent uses the rule $[TRUE \rightarrow LEFT_D]^O$ and so the agent can apply the second rule in its rule database.

For now, let's assume that its opponent defended right this interaction, for example, because it applied the following first-order defend rule: $\left[B_S[TRUE \rightarrow RIGHT_A]^O \rightarrow RIGHT_D\right]^S$.

3.3.5.3. *Third interaction.* Again, the same agent attacks. Now the first rule can be used, because the agent now believes that the condition of the first rule is true, since it both believes that its opponent believes that the agent uses the zero-order attack rule $[TRUE \rightarrow RIGHT_A]^S$ and the opponent's action in the previous turn corresponds with the action in the rule. Therefore, the agent attacks with left.

3.3.6. Inference engine. Each agent uses an inference engine to select an action and reason about opponents. The inference engine is the same for every agent. The input of the inference engine is the memory of the agent and the agent's rule database (see figure 3.2).

In order to find an action, rules are evaluated until a rule is found of which the agent believes that the condition is true. During this process an agent might also establish beliefs about the rules used by its opponent and beliefs about the opponent's beliefs. Algorithm 1 on page 33 contains the pseudo-code of this action-selection function.

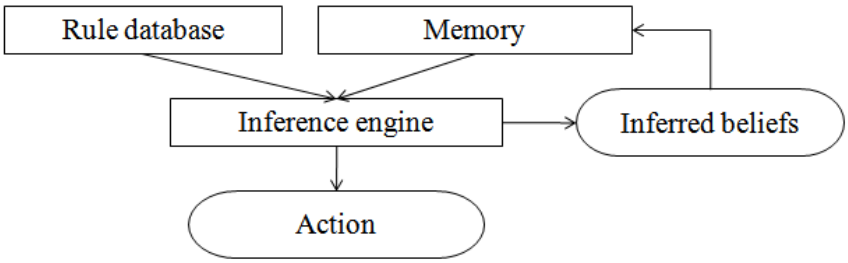


FIGURE 3.2. Schema of the process of inferring beliefs and actions. The inference engine uses the contents of the rule database of the agent and the contents of the agent's memory to select an action. During that process the inference engine may infer new beliefs which are stored into the agent's memory for further use.

Algorithm 1 Pseudo-code of the action-selection mechanism. Its input is the agent's rule database, memory and the required type of the rule (defend or attack). It returns an action (left or right).

```
1: procedure getAction(ruleDatabase, memory, type)
2:     for i = 0 to ruleDatabase.length do
3:         rule ← ruleDatabase[i]
4:         if rule.type ! = type then
5:             break
6:         end if
7:         if rule.user == OPPONENT then
8:             continue
9:         end if
10:        if evaluateRule(rule) then
11:            memory.close()
12:            return rule.action
13:        end if
14:    end for
15:    ruleDatabase.add([TRUE → preference_type]^S)
16:    return preference
17: end procedure
```

For every rule in the rule database, the algorithm first verifies if the rule is of the required type (line 4) and if it has the right user (line 7). Then, the inference engine evaluates the condition of the rule using algorithm 2 (line 10). If the agent believes that the condition is true, the action corresponding to that rule is executed. Lastly, if no rule matches, a zero-order rule containing the preference of the agent is added to the rule database (line

15) and the preference action is executed.

Algorithm 2 Pseudo-code of the function that evaluates the condition of a rule. The input is the rule that is evaluated. It returns true when the condition is believed to be true and false when it is not.

```
1: procedure EVALUATERULE(rule)
2:     beliefOperator ← rule.beliefOperator
3:     rra ← rule.ruleReasonedAbout
4:     user ← rra.user
5:     if condition == TRUE then
6:         return true
7:     end if
8:     if beliefOperator == OPPONENT AND user == SELF then
9:         if retrieveRuleBelievedByOpponent(rra) OR inferRule-
   BelievedByOpponent(rra) then
10:            return true
11:        end if
12:    else if beliefOperator == OPPONENT AND user == OPPONENT
   then
13:        if retrieveRuleUsedByOpponent(rra) OR inferRuleUsedBy-
   Opponent(rra) then
14:            return true
15:        end if
16:    else if beliefOperator == SELF AND user == SELF then
17:        if ruleInRuleDatabase(rra) then
18:            return true
19:        end if
20:    else if beliefOperator == SELF AND user == OPPONENT then
21:        if retrieveRuleUsedByOpponent(rra) OR inferRuleUsedBy-
   Opponent(rra) then
22:            return true
23:        end if
24:    end if
25:    return false
26: end procedure
```

Algorithm 2 evaluates the condition of the agent's rule. In different words, it determines if the agent believes the condition is true based on earlier interactions and the agent's beliefs in memory.

To evaluate the condition of the rule, we use three elements of the condition: the belief operator, the rule reasoned about and the user of the rule that is reasoned about in the condition. As mentioned in section 3.3.1.5, a condition of a rule is either $TRUE$ or it looks like the condition in building block 2: $B_m rule^n_\alpha$. B_m refers to the belief operator, while $rule^n_\alpha$ refers to

the rule reasoned about where n is the user of the rule and α refers to the type of the rule.

In algorithm 2 we discern five different kinds of conditions:

(1) $TRUE$: the condition is true (line 5);
(2) $B_O rule_\alpha^S$: the opponent believes that the agent uses $rule_\alpha$ (line 8);
(3) $B_O rule_\alpha^O$: the opponent believes that it uses $rule_\alpha$ (line 12);
(4) $B_S rule_\alpha^S$: this condition is true if the agent believes itself uses $rule_\alpha$ (line 16);
(5) $B_S rule_\alpha^O$: the agent believes that the opponent uses $rule_\alpha$ (line 20).

If a rule is a zero-order rule, the evaluation of the condition is simple: the condition is true and this is returned to *getAction* (algorithm 1). In the other four cases the rules are about beliefs of either the agent or its opponent.

The second condition is about the belief of an opponent about the agent's rule. The condition is true if it is found that the opponent believes that the agent uses this rule. The functions *retrieveRuleBelievedByOpponent* (algorithms 4) and $inferRuleBelievedByOpponent$ (algorithm 6) check if the agent believes that its opponent believes that the agent uses this rule.

The third condition is true when the agent believes that its opponent uses this rule (functions *retrieveRuleUsedByOpponent* (algorithm 3) and $inferRuleUsedByOpponent$ (algorithm 5)). The same algorithms are also used for the fifth condition. The truth values of these two conditions are the same to the agent. If the agent believes that its opponent uses a certain rule, the agent can be sure that its opponent can also believe that it uses that rule and vice versa.

Lastly, the fourth condition is true when the rule is in the agent's own database. We assume that each agent is capable of introspection.

In lines 9, 13 and 21 two functions are called upon to determine whether the condition is true.

These functions are *retrieveRuleBelievedByOpponent* (algorithm 4) and *inferRuleBelievedByOpponent* (algorithm 6) for the second condition and *retrieveRuleUsedByOpponent* (algorithm 3) and *inferRuleUsedByOpponent* (algorithm 5) for the third and fifth condition.

The function *retrieveRuleUsedByOpponent* and the function *retrieveRuleBelievedByOpponent* search the agent's memory whether the agent has inferred in previous turns if the agent's opponent uses a certain rule or believes that the agent uses a certain rule. However, if this rule cannot be found in memory, the functions *inferRuleUsedByOpponent* (algorithm 5) and *inferRuleBelievedByOpponent* (algorithm 6) are used to infer if it is possible that the rule is respectively used by the opponent or if the opponent believes that the agent uses the rule. Note that an agent only starts to infer beliefs when it has to select an action and not, for example, at the end of

its turn.

We will discuss later why the distinction is made between *retrieveRuleUsedByOpponent* and *inferRuleUsedByOpponent*. First, we will look at the functions *retrieveRuleUsedByOpponent* (algorithm 3) and *retrieveRuleBelievedByOpponent* (algorithm 4). Both functions simply search the agent's memory if the agent has established a belief regarding a rule in an earlier interaction.

Algorithm 3 Pseudo-code of the function that investigates whether the agent already believes that its opponent uses a certain rule. Input: the opponent's rule and the opponent's identity. Output: returns true if the rule is already stored in memory and false if it is not.

```
1: procedure RETRIEVERULEUSEDBYOPPONENT(rule, opponent)
2:     if memory.retrieveRuleUsedByOpponent(rule, opponent) then
3:         return true
4:     end if
5:     return false
6: end procedure
```

Algorithm 4 Pseudo-code of the function that investigates whether the agent believes whether its opponent believes that the agent uses a rule. Input: the agent's rule and the opponent's identity. Output: returns true if the rule is already stored in memory and false if it is not.

```
1: procedure RETRIEVERULEBELIEVEDBYOPPONENT(rule, opponent)
2:     if memory.retrieveRuleBelievedByOpponent(rule, opponent) then
3:         return true
4:     end if
5:     return false
6: end procedure
```

Next, we look at the functions *inferRuleUsedByOpponent* (algorithm 5) and *inferRuleBelievedByOpponent* (algorithm 6). The function *evaluateRule* (algorithm 2) uses these functions to determine if the agent believes the condition of the rule is true while this was not known before.

First, we discuss the function $inferRuleUsedByOpponent$ in algorithm 5. The question that is answered in this function by the agent is the following: does the opponent uses a certain rule? In other words, does the opponent believe that the condition of its rule is true and does the action of the rule match with the action in memory? First, we determine if the opponent believes that the condition of its rule is true. Again, we have five different kinds of conditions.

Algorithm 5 Pseudo-code of the function that investigates whether the opponent possibly uses a certain rule. Input: opponent's rule and the opponent's identity. Output: true when the agent believes its opponent uses the rule and false when it does not.

```
1: procedure INFERRULEUSEDBYOPPONENT(rule, opponent)
2:
Require: The agent does not believe the opponent uses a rule with a similar
    type with the same condition
3:      action ← rule.action
4:      type ← action.type
5:      beliefOperator ← rule.beliefOperator
6:      rra ← rule.ruleReasonedAbout
7:      user ← rra.user
8:      interactions ← memory.getInteractionsWhereOpponent(type, oppo-
    nent)
9:      if interactions.size == 0 then
10:         return false
11:     end if
12:     if beliefOperator == OPPONENT AND user == SELF then
13:         if !retrieveRuleBelievedByOpponent(rra, opponent) then
14:             inferRuleBelievedByOpponent(rra, opponent)
15:             return false
16:         end if
17:     else if beliefOperator == OPPONENT AND user == OPPONENT
    then
18:         if !retrieveRuleUsedByOpponent(rra, opponent) then
19:             inferRuleUsedByOpponent(rra, opponent)
20:             return false
21:         end if
22:     else if beliefOperator == SELF AND user == SELF then
23:         if !retrieveRuleBelievedByByOpponent(rra, opponent) then
24:             inferRuleBelievedByOpponent(rra, opponent)
25:             return false
26:         end if
27:     else if beliefOperator == SELF AND user == OPPONENT then
28:         if !retrieveRuleUsedByOpponent(rra, opponent) then
29:             inferRuleUsedByOpponent(rra, opponent)
30:             return false
31:         end if
32:     end if
33:     for i = 0 to interactions.length do
34:         if action != interactions[i].actionOpponent then
35:             return false
36:         end if
37:     end for
38:     memory.addRuleUsedByOpponentToTemporaryMemory(rule,
    opponent)
39:     return true
40: end procedure
```

- $TRUE$: the condition is true (line 5);
- $B_O rule_\alpha^S$: the opponent believes that this condition is true if the opponent believes that the agent uses $rule_\alpha$ (line 12);
- $B_O rule_\alpha^O$: the opponent believes that this condition is true if it believes that it uses $rule_\alpha$ (line 17);
- $B_S rule_\alpha^S$: the opponent believes that this condition is true if the opponent believes the agent uses $rule_\alpha$ (line 22);
- $B_S rule_\alpha^O$: the opponent believes that this rule is true if the agent believes that the opponent uses $rule_\alpha$ (line 27).

It is easy for the inference engine to assess whether the first condition is true: it always is, also to the opponent.

The second condition is more complicated. The agent believes that its opponent believes the condition is true when the opponent believes that the agent uses $rule_\alpha$. So, function *retrieveRuleBelievedByOpponent* (algorithm 4) is used to evaluate if the agent believes that the opponent believes that the agent uses $rule_\alpha$ by assessing the stored beliefs in the agent's memory.

The third condition is about the opponent's belief about the opponent's rule. For the opponent, this condition is only true if the opponent itself believes it uses $rule_\alpha$. Therefore, the inference engine now uses the function *retrieveRuleUsedByOpponent* (algorithm 3) to determine if the agent established in an earlier interaction whether $rule_\alpha$ is indeed used by the agent's opponent. If that is the case, then this opponent can also believe that it uses the rule since agents are capable of introspection into their own rule database.

The fourth condition is similar, but now the opponent reasons about the agent's beliefs about the agent's rule. The inference engine uses the function *retrieveRuleBelievedByOpponent* (algorithm 4) to determine if the opponent believes that the agent uses $rule_\alpha$. If this is the case, it must also believe that the agent believes that the agent uses $rule_\alpha$, since agents have introspection into their own database and this is common knowledge.

For the fifth condition, the inference engine uses the function *retrieveRuleUsedByOpponent* (algorithm 3) to determine if the opponent uses $rule_\alpha$. If the agent can infer in an interaction that it believes that its opponent uses $rule_\alpha$, then its opponent can infer that the agent can do so.

We have discussed five possible conditions of opponent's rules and how they are handled. For rules that are used by the agent itself, the function *retrieveRuleBelievedByOpponent* (algorithm 4) is used. For rules that are used by the agent's opponent, the function *retrieveRuleUsedByOpponent* (algorithm 3) is used. This is regardless of the believer of the rule. Although it may seem silly to have a rule such as $\Big[B_S\big[B_S[TRUE \rightarrow LEFT_A]^S \rightarrow LEFT_D\big]^O \rightarrow RIGHT_A\Big]^S$, the truth value of the condition is the same

for the agent as a seemingly more sensible rule such as $\Big[B_S\big[B_O[TRUE \rightarrow LEFT_A]^S \rightarrow LEFT_D\big]^O \rightarrow RIGHT_A\Big]^S$. This could be explained as follows.

If the agent is able to infer that its opponent uses a certain rule, so can the opponent infer that the agent is able to infer its opponent's rule, since they use the same reasoning mechanisms. This makes the belief operator quite useless, except in the function *evaluateRule* (algorithm 2).

We have discussed how the inference engine determines if it is possible that the opponent believes that the condition of its rule is true. Now we look at the corresponding action. If the inference engine finds that the opponent believes the condition of its rule is true, the algorithm continues to match the action of that rule to the actions in the agent's memory. If they match, it is possible that the agent's opponent uses the rule. This rule is then stored in the agent's temporary memory. The temporary memory can be seen as a memory buffer of the inference engine. The functions *retrieveRuleUsedByOpponent* (algorithm 3) and *retrieveRuleBelievedByOpponent* (algorithm 4) do not access the temporary memory and therefore cannot make use of its contents. This prevents the inference engine from using beliefs that are derived in the current turn from explaining behavior in previous turns.

For example, let's assume that an agent attacks left using a zero-order attack rule in a first interaction and its opponent defends left, also based on a zero-order attack rule. The agent may derive in the next turn that its opponent believes that it uses the zero-order attack rule $[TRUE \rightarrow LEFT_A]^S$. But, it may, based on this belief, also infer that its opponent used the first-order defend rule $[B_O[TRUE \rightarrow LEFT_A]^S \rightarrow LEFT_D]^O$. But this cannot be the case, since its opponent did not know the agent's zero-order attack rule in the first turn.

In the function $inferRuleUsedByOpponent$ the agent's regular memory is accessed to determine whether the opponent believes that the condition of its rule is true. If the rule the opponent reasons about is not found in the agent's memory (in other words, the rule was not derived in an earlier turn), the function returns to its caller that it could not determine if the rule is used by its opponent (for example line 13 - 16). However, in line 14 the algorithm does check if that rule could be inferred in the current interaction. For example, if we look again at the second-order attack rule $\Big[B_S\big[B_O[TRUE \rightarrow LEFT_A]^S \rightarrow LEFT_D\big]^O \rightarrow RIGHT_A\Big]^S$, the inference engine cannot confirm this condition in the second interaction. However, it can infer that its opponent believes that the agent uses the zero-order attack rule $[TRUE \rightarrow LEFT_A]$. And if its opponent defends with left in the

next interaction, the agent does believe that the condition of the rule is true.

After an action is selected in function *evaluateRule* (algorithm 1), the memory of the agent is closed (line 11). In this procedure, the contents of the temporary memory (the rules that were inferred during this interaction) are transferred to the agent's regular memory, so these beliefs can be used in a future interaction. The interactions on which the inference of the beliefs are based, are removed from memory. There is no use in keeping them, since they do not reflect the current state of beliefs: these actions were the result of a belief set which is now outdated.

Now we look at the function $inferRuleBelievedByOpponent$ in algorithm 6. This algorithm is comparable to algorithm 5 but now the inference engine tries to determine whether the opponent believes that the agent uses a certain rule, where in the previous algorithm the inference tries to determine whether the opponent uses a certain rule. The algorithm is essentially the same as the previous one, but there is one difference. The action-part of the rule is now matched to the action of the agent and not to the action of the opponent.

3.3.7. More detailed example: inferring actions and beliefs in an interaction. In section 3.3.5 we showed an example of two agents interacting. The following example is the same, but this time we explain the reasoning steps in more detail with the workings of the inference engine in mind. Again, we look at the interactions from the viewpoint of the attacking agent. Table 3.2 contains the attack rules of the agent.

$$\Big[B_S\big[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\big]^O \rightarrow LEFT_A\Big]^S$$
$$\big[B_S[TRUE \rightarrow LEFT_D]^O \rightarrow RIGHT_A\big]^S$$
$$\big[B_S[TRUE \rightarrow RIGHT_D]^O \rightarrow LEFT_A\big]^S$$
$$[TRUE \rightarrow RIGHT_A]^S$$

TABLE 3.2. Attack rules of a second-order agent

3.3.7.1. *First contact.* Function *getAction* (algorithm 1) is used to select an action. First, the algorithm checks if the rule is of the right type (we are looking for an attack rule, not a defend rule) and if the user matches (the rule can only be used by the agent itself). Since the first rule is not a zero-order rule, the agent evaluates the condition of the first rule using *evaluateRule* (algorithm 2). Because the rule is about the agent's own belief about a rule used by the opponent, $retrieveRuleUsedByOpponent$ (algorithm 3) is used to see if this rule has been stored in memory during a previous interaction. This is not the case, because there have not been earlier interactions where the agent could have established this belief. Then the agent checks using function $inferRuleUsedByOpponent$ (algorithm 5)

Algorithm 6 Pseudo-code of the function that investigates whether the opponent possibly believes a certain rule is used by the agent. Input: the agent's rule and the opponent's identity. Output: true when the agent believes its opponent believes the agent uses the rule and false when it does not.

```
1: procedure INFERRULEBELIEVEDBYOPPONENT(rule, opponent)
2:     Require The agent does not believe the opponent believes a rule of a similar type with the same condition
3:     action ← rule.action
4:     type ← action.type
5:     beliefOperator ← rule.beliefOperator
6:     rra ← rule.ruleReasonedAbout
7:     user ← rule.user
8:     interactions ← memory.getInteractionsWhereAgent(type, opponent)
9:     if interactions.size == 0 then
10:        return false
11:    end if
12:    if beliefOperator == OPPONENT AND user == SELF then
13:        if !retrieveRuleBelievedByOpponent(rra, opponent) then
14:            inferRuleBelievedByOpponent(rra, opponent)
15:            return false
16:        end if
17:    else if beliefOperator == OPPONENT AND user == OPPONENT then
18:        if !retrieveRuleUsedByOpponent(rra, opponent) then
19:            inferRuleUsedByOpponent(rra, opponent)
20:            return false
21:        end if
22:    else if beliefOperator == SELF AND user == SELF then
23:        if !retrieveRuleBelievedByOpponent(rra, opponent) then
24:            inferRuleBelievedByOpponent(rra, opponent)
25:            return false
26:        end if
27:    else if beliefOperator == SELF AND user == OPPONENT then
28:        if !retrieveRuleUsedByOpponent(rra, opponent) then
29:            inferRuleUsedByOpponent(rra, opponent)
30:            return false
31:        end if
32:    end if
33:    for i = 0 to interactions.length do
34:        if action ! = interactions[i].actionAgent then
35:            return false
36:        end if
37:    end for
38:    memory.addRuleBelievedByOpponentToTemporaryMemory(rule, opponent)
39:    return true
40: end procedure
```

if it can infer whether the opponent uses rule $[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D]^O$, but since there have been no previous interactions this also fails and the agent continues to investigate the second rule.

Since both the second and third rule are about a rule of the opponent, both rules fail and algorithm 1 continues to the fourth and last rule. First, the algorithm checks again if the type and user are correct. Then it continues to evaluate the condition. Since the condition is true because it is a zero-order rule (see section 3.3.1.1), the algorithm returns the corresponding action, which is in this case a right attack, and the algorithm finishes.

For now, we assume that the agent's opponent defends with left. Lastly, the actions of both agents are stored in the agent's memory.

Table 3.3 contains the memory of the agent after the first interaction.

Interactions		Rules used by opponent	Rules believed by opponent
S	O		
$RIGHT_A$	$LEFT_D$		

TABLE 3.3. Memory of the agent after the first interaction

3.3.7.2. *Second interaction.* In the second interaction we let the same agent attack again. The algorithm *getAction* is again used to select an action. For every rule the algorithm first checks if the rule is of the correct type and user. Then, *getAction* continues evaluating the condition of each rule with *evaluateRule*.

Since the first rule is about the agent's own belief about a rule used by its opponent, *retrieveRuleUsedByOpponent* is used to see if the rule $B_S[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D]^O$ has been stored in memory during a previous interaction. This is not the case, since the agent could not have established any beliefs about the opponent's rules in the previous turn, since there was no experience on which to base any beliefs.

Next, *evaluateRule* calls *inferRuleUsedByOpponent* to see if it is possible that its opponent did apply the rule $[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D]^O$ during previous turns. The opponent could only have used this rule in a previous interaction if it believed that the condition was true. Therefore, the inference engine must first determine if the opponent might have believed that condition $B_O[TRUE \rightarrow RIGHT_A]^S$ was true, in other words, if the opponent might believe that the agent uses the rule $[TRUE \rightarrow RIGHT_A]^S$. To investigate this, *inferRuleUsedByOpponent* calls the function *retrieveRuleBelievedByOpponent* (algorithm 4) to search the agent's memory if it derived in an earlier interaction that its opponent may believe that the agent uses the rule $[TRUE \rightarrow RIGHT_A]^S$.

However, this search will return nothing, since the opponent could not have established any beliefs in the previous turn, since it was the first time

the agent interacted with its opponent. Because *retrieveRuleBelievedByOpponent* returned false, *inferRuleUsedByOpponent* calls *inferRuleBelievedByOpponent* to investigate if the agent can establish the belief that its opponent might believe that the agent uses the rule $[TRUE \rightarrow RIGHT_A]^S$. This is the case, since the agent executed a right attack action during the first interaction. This belief is stored into temporary memory. The function *inferRuleUsedByOpponent* returns false to *evaluateRule* since there is no belief that the agent's opponent used the rule $[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D]^O$.

Since *retrieveRuleBelievedByOpponent* returned that it was not believed in a previous interaction that the agent's opponent might believe that the agent used the rule $[TRUE \rightarrow RIGHT_A]^S$, the function *inferRuleBelievedByOpponent* is called to investigate if that belief can be established using the information that became available after the previous interaction. This is the case, so this rule is stored into the agent's temporary memory. However, this new belief cannot be used for inferring other new beliefs, so *inferRuleUsedByOpponent* returns that the condition $\big[B_S\big[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\big]^O$ could not be confirmed.

Now, the agent looks at the second rule: $\big[B_S[TRUE \rightarrow LEFT_D]^O$. The function *getAction* first checks if the type and user of the rule is correct. This is the case. The function *evaluateRule* is used to evaluate the condition of the rule. Since the rule concerns the agent's own belief about an opponent's rule, *evaluateCondition* uses *retrieveRuleUsedByOpponent* to search the agent's memory if the agent has established the belief in an earlier interaction that the agent's opponent uses the defend rule $[TRUE \rightarrow LEFT_D]^O$. Such a rule is not found, since the agent could not have known its opponent's defend rule in the previous turn, since it did not have any information to base this belief on in the previous interaction.

However, the agent does have experience with its opponent that it can use. The function *inferRuleUsedByOpponent* is then used by *evaluateCondition* to evaluate if the opponent could have used the rule $[TRUE \rightarrow LEFT_D]^O$ in the previous turn. This rule is a zero-order rule and therefore the inference engine only has to match defend actions of the opponent in previous interactions with the action in the rule. The agent's opponent did defend with left in the previous turn, so the action of the rule and the action in the agent's memory correspond. Therefore, the inference engine draws the conclusion that the rule $[TRUE \rightarrow LEFT_D]^O$ is indeed used by the opponent. This rule is also stored into temporary memory and *inferRuleUsedByOpponent* returns to *getAction* that the rule the agent reasons about is indeed used by the agent's opponent. Therefore, the condition of the second rule does match and the agent attacks with right.

Lastly, the contents of the temporary memory are stored into the agent's memory. Interactions where the agent attacked and the opponent defended

are removed from memory, because they are no longer relevant. Let's assume the agent's opponent defended right in this interaction, for example, because it applied the following first-order defend rule: $\left[B_S[TRUE \rightarrow RIGHT_A]^O \rightarrow RIGHT_D\right]^S$. The actions of both agent are also stored in memory.

Table 3.4 contains the memory of the agent after the second interaction.

Interactions		Rules used by	Rules believed
S	O	opponent	by opponent
$RIGHT_A$	$RIGHT_D$	$[TRUE \rightarrow LEFT_D]^O$	$[TRUE \rightarrow RIGHT_A]^S$

TABLE 3.4. Memory of the agent after the second interaction

3.3.7.3. *Third interaction.* Again, the agent attacks. When requested for an action, the inference engine calls on *getAction*. First, the algorithm looks at the first rule in the database. The type and user is correct, so the algorithm uses *evaluateRule* to evaluate the condition of the rule. First, *retrieveRuleUsedByOpponent* is used to find out if the rule in the condition has been stored in memory during a previous interaction. This is not the case, so *evaluateRule* calls $infer RuleUsedByOpponent$ to find out if the agent's opponent used the rule $\left[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\right]^O$ in previous interactions. The agent's opponent could only have applied this rule if it believed that the condition of this rule was true. So, *retrieveRuleBelievedByOpponent* is used to find out if the agent believes that its opponent believes that the agent uses rule $[TRUE \rightarrow RIGHT_A]^S$. This rule is already stored in memory.

Next, $infer RuleUsedByOpponent$ matches the action of the opponent's rule with actions in memory. The action matches with those in memory, so the agent establishes the belief that its opponent uses the rule $\left[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\right]^O$. This rule is stored in temporary memory. Since the condition of the first rule of the agent's database is true, the corresponding action is returned.

Lastly, the agent stores the contents from its temporary memory into its permanent memory and earlier interactions where the agent's opponent executed a defend action are removed. The agent's opponent defends with right (perhaps because it only has zero- and first-order rules) and the agent stores the actions of both agents in memory.

Table 3.5 contains the memory of the agent after the third interaction.

3.3.8. Initial rule database. The first generation of agents is given a simple rule database to select actions from. The database contains only zero-order rules (see table 3.3.8).

Interactions		Rules used by opponent	Rules believed by opponent
S	O		
$LEFT_A$	$RIGHT_D$	$[TRUE \rightarrow LEFT_D]^O$ $\big[B_O[TRUE \rightarrow RIGHT_A]^S$ $\rightarrow RIGHT_D\big]^O$	$[TRUE \rightarrow RIGHT_A]^S$

TABLE 3.5. Memory of the agent after the third interaction

Defense rules
$[TRUE \rightarrow preference_D]^S$

Attack rules
$[TRUE \rightarrow preference_A]^S$

TABLE 3.6. Zero-order rule database

3.4. Competitive Environment

The environment in which the agents interact is competitive in nature in two ways. Firstly, interactions between agents are competitive. Secondly, only the most successful individuals reproduce. In this section we take a look at the environment of the agents.

3.4.1. Environment. The environment the agents live in is non-spatial, since we are not looking for spatial effects. During a generation agents interact with each other. Two agents are randomly selected for each interaction: one agent is the attacker and the other is the defender. After one generation the most successful agents are selected for reproduction.

3.4.2. Interaction. A new round of interactions starts when a new population is created. Agents are selected randomly for interaction. During an interaction, the following steps occur:

(1) Each agent selects an action; new beliefs that are established in that process are stored;
(2) the actions are executed;
(3) the agents update their memory;
(4) the winner is selected and the dominance ranking is updated accordingly.

More information on action-selection can be found in section 3.3.

3.4.3. Dominance. How well the individuals do in comparison with other agents is reflected in the dominance value. A higher dominance value may lead to better access to food, mates or safe locations (Hemelrijk, 1999). Individuals with the highest dominance are allowed to reproduce. Losing a

fight will decrease the individual's dominance; winning will increase it. In section 2.5.4 on page 21 we introduced the formula below which we use to calculate the domination value of agent i. The dominance value lies in the range $[-1, 1]$.

$$DOM_i = \frac{fightsWon_i - fightsLost_i}{fightsWon_i + fightsLost_i} \tag{3.4.1}$$

In this formula, the interactions where an attack was blocked are omitted. In retrospect, this is an odd choice since the attacks where the defender are blocked are not taken into account. This has an unfortunate effect. Assume an agent is a good defender and did not lose any interactions, but is a poor attacker and won only a few interactions. According to formula 3.4.1, the agent has a high dominance value. Now, assume an agent which is a good attacker and wins many interactions, but is a poor defender and loses almost as many interactions as it won. The agent's dominance value is low. This is summarized in table 3.7. This has the effect that defensive agents are favored over those that have better attack capabilities. However, we expect that this will not influence the simulation results much. If most agents are good defenders but poor attackers, agents who are capable of successfully attacking their opponents have a higher dominance value than agents who are not good attackers. Since agents are not punished for having large databases, there is no disadvantage to having rules that either allow agents to be good defenders or good attackers.

Attack capabilities	Defend capabilities	Numerator	Denominator value	Dominance
Bad	Good	Small, positive	Small, positive	Large
Good	Bad	Small, positive	Large, positive	Small

TABLE 3.7. Expected effect of dominance function.

The dominance value is a measure for the fitness value which determines how successful the agents are. Only the most successful agents are allowed to reproduce. In the next section we explain how agents reproduce and how they evolve over generations.

3.5. Evolution of the rule database

Since theory of mind follows from the rule database, we evolve the rule database over time (see section 2.3 on page 15 for a short overview). We use roulette-wheel selection to select agents for reproduction. The children of the selected agents will inherit a mutated version of the rule database. We test two methods of inheritance: fully linked genes and crossover.

3.5.1. Fitness function. Regardless of the reproduction method, we require a fitness function. The fitness function determines how well the individuals are doing. The chance of reproduction is proportional to the agent's fitness. The fitness function is shown in equation (3.5.1). The fitness function for agent i (f_i) depends only on the dominance value of agent i (DOM_i) which is described in equation (3.4.1).

$$f_i = DOM_i \tag{3.5.1}$$

3.5.2. Roulette-wheel selection. In this model, we use roulette-wheel selection (also called fitness proportionate selection) to select individuals who will reproduce (de Jong, 2008, p.139). Suppose we want to select an agent for reproduction from a set of n agents. Instead of placing n agents on the roulette-wheel evenly (like a normal roulettewheel), we attribute each agent space on the roulette wheel relative to its fitness. The result is that more successful agents have more space on the roulette wheel and thus a higher chance to get selected (see figure 3.3). Agents with a higher fitness have a higher probability to get selected, but at the same time less successful agents can also get selected. The result is that diversity is maintained in the population since not only the best agents are selected for reproduction.

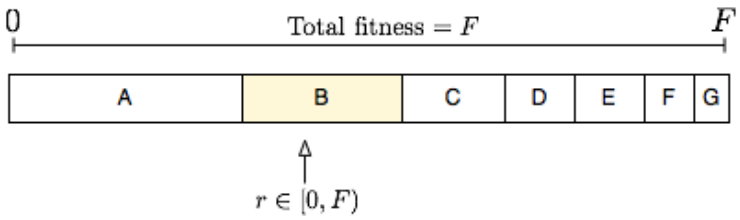


FIGURE 3.3. An example of roulette wheel selection. r is a random number within the range of the total fitness. Here, individual B is selected.

For the pseudo-code of this selection mechanism, see algorithm 7. The fitness values of each agent are first transformed to a suitable range. Then, the total fitness value is calculated. First, the fitness of each agent is squared and then added to the total fitness value (line 6). Squaring the fitness value allows us to favor agents with a higher fitness more strongly over those with a lower fitness. Then, a random number between zero and the total fitness value is selected (line 8). This can be compared to the phase in the roulette game where the ball stops spinning and ends in one of the pockets. Next, it is determined which agent corresponds with the pocket (line 10 - 16). We use the squared dominance value to favor agents with a high fitness over those with a lower fitness (lines 6 and 12).

Algorithm 7 Pseudocode of roulette wheel selection algorithm.

```
 1: procedure getParentUsingRWS(agentSet, dominance)
Require: Fitness values are transformed from range [−1..1] to [0..2].
 2:     totalFitness = 0
 3:     for i = 0 to agentSet.size do
 4:         agent ← agentSet[i]
 5:         fitnessValue ← dominance(agent)
 6:         totalFitness += fitnessValue * fitnessValue
 7:     end for
 8:     r ← random number between 0 and totalFitness
 9:     cumulativeFitness = 0
10:     for i = 0 to agentSet.size do
11:         fitnessValue ← dominance(agent)
12:         cumulativeFitness = fitnessValue * fitnessValue
13:         if cumulativeFitness > r then
14:             return agentSet[i]
15:         end if
16:     end for
17: end procedure
```

3.5.3. Reproduction. After the parents have been selected, the parents will reproduce. The children inherit the rule database of one or two parents, depending on the reproduction method. We test two methods, linked genes and crossover. Linked genes inheritance is quite simple. The offspring inherits genes from a single parent (or in this case the rule database) with a few changes due to mutation.

Crossover requires two parents; the offspring will inherit genes from both of them. In one-point crossover the gene set is cut at some point. Each child gets one part from each parent. In n-point crossover the gene set is divided into more sections (de Jong, 2008, p.41). Crossover is expected to give results faster than linked genes, because it allows for more diversity in the agent set. We ran experiments with 1-point crossover.

Besides changes due to crossover, the rule database is subject to mutation (see section 2.5.3 on page 20). Mutation occurs on database level and on rule level. We vary the mutation probability between the following values: 0.0005, 0.001, 0.005.

On the database level, the possible mutations are:

- A randomly selected rule is duplicated and inserted randomly in the rule database;
- a randomly selected rule is removed from the rule database;
- a randomly selected rule is moved to a random location in the rule database;
- a zero-order rule is inserted in the rule database;
- two rules swap locations in the rule database;

- a randomly selected rule changes type: an attack rule becomes a defense rule and vice versa.

On the rule level, the possible mutations are:

- Belief operators are changed, inserted or removed;
- the corresponding action changes from $LEFT$ to $RIGHT$ or vice versa;
- the user of the changes from $SELF$ to $OPPONENT$ or vice versa.

If the rule reasons about another rule, this nested rule is also subject to mutation.

After selection and reproduction, a new generation of agents is born. The dominance ranking is reset and the agents start interacting.

3.6. Experiments and expected results

The basic setup will be as follows. First we will run experiments varying the parameter values mentioned in table 3.8. Each experiment will be run twice. This results in 36 simulations. The results of the experiment will be discussed in Chapter 4. Appendix A.1 and table 3.9 contain an overview of the simulations that were run.

Parameter	Values
Number of agents	200, 350, 500
Mutation probability	0.005, 0.001, 0.0005
Reproduction method	linked genes, crossover
Average number of attack actions to each opponent	10

TABLE 3.8. Varied parameters in experiments

	linked-genes			crossover		
MC / AC	200	350	500	200	350	500
0.0005	13	17	21	15	19	23
0.001	1	5	9	3	7	11
0.005	25	29	33	27	31	35

TABLE 3.9. Numbers of the experiments in which the agent attacked every opponent on average 10 times.

3.6.1. Evaluation criteria. The model was created with two criteria in mind. Firstly, we want to see higher-order theory of mind (at least second-order theory of mind) evolve. Secondly, if theory of mind evolves we want to gain an understanding why higher-order theory of mind evolves at it does.

If theory of mind does not evolve at first, we can change the setup of the model.

As discussed in section 3.3.1 on page 27 there is hardly any limit on the contents of a rule. The rules can be about any agent, any action and any type of rule. It could be possible that no rules that represent theory of mind are evolved within the simulation time, because the search space is too large. In that case can limit the search space by imposing constraints on the construction of the rules. For example, we expect that an efficient attack rule is about a defend rule, as we have seen in the examples of higher-order rules in section 3.3. If no theory of mind evolves, we could limit the rule syntax in such a way that attack rules are only about defend rules and vice versa. Preferably we want to see theory of mind evolve with as few constraints as possible.

3.6.2. Expected results basic setup. We expect that competitive interaction between individuals in a simulated non-spatial environment with the constraints mentioned earlier gives rise to the evolution of higher-order theory of mind and that the average order of attribution will increase over generations.

We predict that several parameters influence the evolution of higher-order cognition: the reproduction method, the mutation probability, the number of interactions per opponent and the number of agents. Varying these parameters may give us insight into what parameters may influence the evolution of higher-order cognition. The following parameters were omitted due to lack of time: different types of initial rule database and the number of agents who survive a generation and continue to live in the next generation. Next, we discuss the predicted influence of varying the four parameters.

3.6.3. Reproduction. There are two methods of inheritance considered: linked genes and one-point crossover. We expect that crossover may lead to the evolution of higher-order theory of mind faster. Because rule databases are combined using one-point crossover, more diverse agents are created.

3.6.4. Number of interactions. We expect that the number of interactions will limit the order of theory of mind that can arise. To be able to use a third-order attack rule, an agent has to attack its opponent at least four times. This can be easily explained. Let us assume that one agent attacks another agent four times. The first time both the attacking and the defending agent use a zero-order rule because neither agent has interacted with the other, so there are no memories and earlier established beliefs that would allow both agents to select higher-order rules.

During the second turn, both agents can use a first-order rule since they both have now experience to base beliefs about the used rules by their opponents on. During the third turn both agents can use second-order rules and during the fourth turn both agents can use third-order rules.

We ran experiments where each agent on average attacked each other agent 10 times and experiments where each agent attacked the other agent on average 20 times. Theoretically, this could result in respectively ninth-order rules and nineteenth-order rules.

Since we could not find any difference in results in both sets of simulations, we have set the number of interactions to 10. This halves the simulation duration. The total number of interactions during one generation is calculated using equation 3.6.1.

(3.6.1)
$$numberOfInteractions = numberOfAgents * numberOfAgents * 10$$

3.6.5. Mutation probability. We expect that when a low mutation probability is used, it will take longer to evolve higher-order rules than using a higher mutation probability, because less diversity is introduced in the population. However, we expect that a very high mutation probability does not result in higher-order rules quickly if at all. Because rule databases change fast when using a very high mutation probability, we expect that rule databases cannot adapt quickly enough to the rule databases of others, because the databases of others are also likely to change. The lowest mutation value we choose stills allows for changes in rule databases when the agents reproduce, while the highest value does not result in very large databases and in rule databases that are very different from those in a previous generation.

3.6.6. Number of agents. We expect that a larger number of agents will provide a larger diversity in the agent set, therefore allowing higher-order rules to evolve faster. Experiments with a population size larger than 500 resulted in simulations that took more than a week to run, so we did not experiment with larger population sizes. In preliminary experiments with a population size smaller than 200 we found that the evolution of higher-order rules takes a long time, probably because there is little diversity in the population because of the limited number of agents.

3.7. Chapter summary

In this chapter we showed how the model is constructed. Agents interact competitively, using logical rules to select their actions. Whether a rule is used by an agent depends on the agent's previous experience with its opponent, the beliefs an agent has about its opponent's rules and the beliefs an agent has about its opponent's beliefs.

After one generation, the agents reproduce using either the linked-genes reproduction method or one-point crossover. The agents with the highest dominance values are most likely to be selected for reproduction. The offspring inherit a mutated version of their parents' rule database. The

following parameters are varied in simulation: population size, mutation probability and the reproduction method.

With the experiments in Chapter 4 we hope to investigate whether competitive interactions between individuals may result in the evolution of higher-order theory of mind.

CHAPTER 4

Results

In this chapter we present the results of the experiments (see section 3.6 for an overview). We discuss how the simulation results are analyzed and show how the experiment parameters influenced the evolution of theory of mind in the model.

4.1. Analysis

To analyze the results, we developed a method to determine the order of a rule. We look at the contents of a rule to determine the order of a rule. We explain how to determine the order of a rule. Then we give an example. Finally, the pseudo-code of the algorithm that determines the order of a rule is discussed.

4.1.1. Determining the order of a rule. A rule template is shown below. This is building block 2 (see also section 3.3.1.5 on page 29). The variables m, n and o can have the values *SELF* (S) or *OPPONENT* (O). Actionsα and β can have the values *ATTACK* (A) or *DEFEND* (D).

$$[B_m rule^n_\alpha \rightarrow action_\beta]^o$$

To determine the order of a rule, we do not just count the number of belief operators but we also look at the contents of a rule. The order of a rule is increased when the following two conditions are met:

(1) $n \neq o$, where n and o have the values *SELF* or *OPPONENT*;
(2) $\alpha \neq \beta$ where α and β have the values *ATTACK* or *DEFEND*.

In other words, two conditions must be fulfilled for an order of a rule to increase. Firstly, an attack rule must reason about a defend rule or the other way around. Secondly, a rule used by the agent itself must reason about the rule of an opponent or the other way around.

4.1.2. Example.

Example 6.

$$\Big[B_S\Big[B_O\big[B_S[TRUE \rightarrow RIGHT_A]^O \rightarrow RIGHT_D\big]^S \rightarrow LEFT_A\Big]^O \rightarrow LEFT_D\Big]^S$$

Example 6 is a third-order rule. We will show how its order is determined. This defend rule, used by the agent itself, reasons about an opponent's attack rule. So, the two conditions above are met. This means that this rule is at least a first-order rule. Then we look at the nested rule: $\Big[B_O\big[B_S[TRUE \rightarrow RIGHT_A]^O \rightarrow RIGHT_D\big]^S \rightarrow LEFT_A\Big]^O$. The attack rule of the opponent reasons about a defend rule of the agent itself. So the order is increased to two. The next nested rule is $\big[B_S[TRUE \rightarrow RIGHT_A]^O \rightarrow RIGHT_D\big]^S$. This defend rule reasons about the attack rule of the opponent, so the order of the rule is increased to three. The nested rule $[TRUE \rightarrow RIGHT_A]^O$ does not reason about a rule, so we conclude that the order of the rule is three.

When we analyze the rule in example 7, we notice that the first condition is violated, since the attack rule used by the agent itself reasons about a defend rule used by the agent. So, we say that this rule has no order. This is denoted by a question mark. This allows us to distinguish these no-order rules from true zero-order rules.

Example 7.

$$\Big[B_O\big[B_S[TRUE \rightarrow RIGHT_D]^S \rightarrow LEFT_D\big]^S \rightarrow RIGHT_A\Big]^S$$

4.1.3. Algorithm. In algorithm 8 we describe the algorithm used to determine the order of a rule in more detail. In lines 8 - 16 of algorithm 8 we describe four situations in which the order of a rule is not increased.

- When a rule that is used by the agent itself reasons about a rule used by the agent itself;
- when a rule that is used by the agent's opponent reasons about a rule that is used by the agent's opponent;
- when an attack rule reasons about an attack rule;
- or when a defend rule reasons about a defend rule.

Example 8.

$$\Big[B_O\big[B_S[TRUE \rightarrow RIGHT_D]^S \rightarrow LEFT_D\big]^O \rightarrow RIGHT_A\Big]^S$$

If a rule does not belong to one of the four categories mentioned above, an order is returned. The order depends on the order of the rule that the rule reasons about. If the order of the rule that is reasoned about is indeterminable, the order of the rule is set to one (line 19). According to this method, the rule in example 8 is a first-order rule. The first part of this rule, $\big[B_S[TRUE \rightarrow RIGHT_D]^S \rightarrow LEFT_D\big]^O$, has no determinable order, but since the agent takes into account the action of its opponent in this rule, we can safely say that this rule is at least a first-order rule. We could argue that this rule is a second-order rule, because the agent considers its opponent's beliefs about its own defend rule, but we choose to err on the safe side.

In all other cases, the order of a rule is the order of the rule that the rule reasons about incremented by one (line 21).

Algorithm 8 Pseudo-code of algorithm used to determine the order of a rule.

```
 1: procedure GETORDER(rule)
 2:     user ← getUser(rule)
 3:     type ← getType(rule)
 4:     rra ← getRuleReasonedAbout(rule)
 5:     if rule.isBehaviorRule() then
 6:         return 0
 7:     end if
 8:     if user == SELF AND getUser(rra) == SELF then
 9:         return ?
10:     else if user == OPP AND getUser(rra) == OPP then
11:         return ?
12:     else if type == ATTACK AND getType(rra) == ATTACK then
13:         return ?
14:     else if type == DEFEND and getType(rra) == DEFEND then
15:         return ?
16:     end if
17:     orderRRA ← getOrder(rra)
18:     if orderRRA == ? then
19:         return 1
20:     else
21:         return orderRRA + 1
22:     end if
23: end procedure
```

4.2. Example of a rule database

In this section we take a closer look at one particular database. Although different experiments yielded different results, this particular database and its rules are representative for the rules evolved in most simulations. Table 4.2 shows the contents of a rule database. This is the rule database of the best performing agent in experiment 31 (see Appendix A) at the end of the simulation. In this experiment 350 agents interacted. They reproduced using crossover. The simulation ran for 2000 generations. The agent won more than half of its attack actions (see table 4.1) and it managed to lose only a fifth of the interactions in which it was attacked.

The rules shown in table 4.2 are rules that are actually used by the agent. Most rules in a rule database are never actually used. This is due to ordering of the rules (the first rule that matches is used), duplicated rules, rules with the same condition, rules of which the user is the opponent and

	Won	Not won
Attack action:	1800	1700
	Not lost	Lost
Defend action:	2800	700

TABLE 4.1. Estimated interactions won and lost by the agent during one generation. The agent is particularly strong in defending itself. The numbers are an estimation. It is only known how often the agent won or lost, but the not the total number of interactions the agent participated in. On average this agent attacks 3500 times and is attacked 3500 times during one generation.

not the agent itself (see algorithm 1) and rules of which the condition is never true. Only 10% to 30% of the rules are used by agents in simulations. We expect that these junkrules do benefit the agent. A rule that is currently not used may become useful when the rule mutates. Currently, there is no penalty for having a large rule database.

So why was the particular agent so successful? First we look at the agent's defend rules, which are rule 2, 4, 8 and 11. Of these rules rules 2 and 8 are particularly successful (see table 4.3). The agent lost all interactions where it used rule 4, but since this rule is hardly used it won't effect the agent's performance very much.

The agent's zero-order defend rule is rule 11. Although this rule does not look like a zero-order rule, its condition is always true if the agent has the rule $[TRUE \rightarrow action_D]^S$ in its rule database (see algorithm 2 on page 2). The rule $[TRUE \rightarrow LEFT_D]^S$ is indeed somewhere in the agent's rule database, but is not depicted here because it is never actually used.

In the second rule (the second-order rule) something interesting happens. The condition of the opponent's rule is about a rule of the agent itself: $B_O[TRUE \rightarrow LEFT_D]^S$. However, the agent never actually applied this rule, but applying the first-order defend rule 8 did result in an action where the agent defended on left. In section 4.5 this phenomenon is discussed in more detail.

The agent's attack rules are rule 1, 3, 5, 6, 7, 9 and 10. The agent lost more interactions than it won using rule 1, 3 and 5. However, these rules (together with the sixth and ninth rule) are hardly used. The seventh and tenth rule are respectively zero-order and second-order rules.

Although this agent was the best performing agent of the population, there are several other agents with a similar performance. These agents outperformed other agents mostly because they had more rules that were actually used (there was not much difference in total database size) and because the rules were smarter. In table 4.4 the rule database of the worst performing agent is shown.

1.	$\Big[B_O\big[B_S\big[B_O[TRUE \rightarrow RIGHT_A]^S \rightarrow LEFT_A\big]^S \rightarrow LEFT_D\big]^O \rightarrow RIGHT_A\Big]^S$(108)(2)
2.	$\Big[B_S\big[B_O[TRUE \rightarrow LEFT_D]^S \rightarrow RIGHT_A\big]^O \rightarrow RIGHT_D\Big]^S$(1223)(2)
3.	$\Big[B_S\Big[B_O\Big[B_S\Big[B_O\big[B_S\big[B_O[TRUE \rightarrow RIGHT_D]^O \rightarrow RIGHT_D\big]^S \rightarrow RIGHT_D\Big]^S \rightarrow RIGHT_D\Big]^O \rightarrow LEFT_D\Big]^O \rightarrow LEFT_D\Big]^O \rightarrow LEFT_A\Big]^S$(18)(1)
4.	$\Big[B_O\Big[B_S\Big[B_O\big[B_O\big[B_O[TRUE \rightarrow LEFT_D]^O \rightarrow RIGHT_A\big]^O \rightarrow RIGHT_D\Big]^S \rightarrow LEFT_D\Big]^O \rightarrow LEFT_A\Big]^S \rightarrow LEFT_D\Big]^S$(11)(?)
5.	$\Big[B_O\Big[B_S\Big[B_O\big[B_O\big[B_S[TRUE \rightarrow LEFT_D]^O \rightarrow RIGHT_A\big]^S \rightarrow RIGHT_D\Big]^S \rightarrow LEFT_D\Big]^O \rightarrow LEFT_D\Big]^S \rightarrow LEFT_A\Big]^S$(2)(?)
6.	$\Big[B_S\Big[B_O\big[B_O\big[B_O[TRUE \rightarrow LEFT_D]^S \rightarrow LEFT_D\big]^S \rightarrow RIGHT_D\Big]^S \rightarrow LEFT_A\Big]^O \rightarrow RIGHT_A\Big]^S$(10)(?)
7.	$\big[B_S\big[B_S[TRUE \rightarrow RIGHT_A]^S \rightarrow RIGHT_D\big]^O \rightarrow LEFT_A\big]^S$(2274)(2)
8.	$\big[B_S[TRUE \rightarrow LEFT_A]^O \rightarrow LEFT_D\big]^S$(1279)(1)
9.	$\big[B_O\big[B_S[TRUE \rightarrow LEFT_D]^O \rightarrow LEFT_D\big]^O \rightarrow LEFT_A\big]^S$(31)(1)
10.	$[TRUE \rightarrow RIGHT_A]^S$(1086)(0)
11.	$\big[B_S[TRUE \rightarrow LEFT_D]^S \rightarrow RIGHT_D\big]^S$(1003)(?)

TABLE 4.2. Rule database of best performing agent in the last generation of experiment 31. The first number to the right of the rules represents the number of times the rule was used during the last generation. The second number represents the order of the rule. Rules that were never used by the agent are not printed.

If we let the best agent (and its best performing colleagues) interact against agents from another simulation with the same experiment parameters, this agent is not the best performing agent. This means that, not unexpectedly, this agent has adapted to the behavior of the agents in its

No. rule	Utility	No. rule	Utility
1.	-.20	7.	0.23
2.	0.94	8.	0.62
3.	-1.0	9.	0.09
4.	-1.0	10.	-0.34
5.	-1.0	11.	0.15
6.	0.6		

TABLE 4.3. Utility of rules of the agent. The value ranges from -1 to 1. For attack rules a positive number means that more interactions were won than not won using a particular rule. For defend rules a positive number means that more interactions were not lost than lost using a particular rule.

$$\Big[B_O\Big[B_O\big[B_O[TRUE \rightarrow RIGHT_A]^O \rightarrow LEFT_D\big]^O \rightarrow RIGHT_D\Big]^S \rightarrow RIGHT_A\Big]^S(205)(?)$$
$$[TRUE \rightarrow RIGHT_A]^S(3211)(0)$$
$$\big[B_S[TRUE \rightarrow LEFT_D]^S \rightarrow RIGHT_D\big]^S(3580)(0)$$

TABLE 4.4. Rule database of worst performing agent.

own simulation but cannot cope with agents from other simulations. This can be explained as follows.

In figure 4.1 a graph is shown which describes the average composition of the rule databases of the agents of one simulation over time. We calculate this as follows. At the end of each generation n, we retrieved from each agent what proportion of the rules in its rule database were zero-order rules, first-order rules, second-order rules and so on. We use this data to calculate what the average rule database composition for all agents in generation n is. Then, we calculated the average rule database composition of the population for each generation.

The average rule database composition (see figure 4.1) is from the same simulation as the agent we have discussed. 24% of the rules in the population are zero-order rules, 19% are first-order rules and 24% are second-order rules.

Ideally, a rule database contains rules to handle all possible rules an opponent may use. In that case, the composition of a rule database would look more like the plot in figure 4.2. It would consist of two zero-order rules (a defend rule and an attack rule) and four first-order rules (one for every zero-order rule possibly used by the opponent). For every possible first-order rule used by an opponent second-order rules are constructed. This results in a lot more second-order rules than first-order rules, and more third-order rules than second-order rules and so on. An ideal database would be able to produce an appropriate response to every possible rule an opponent may

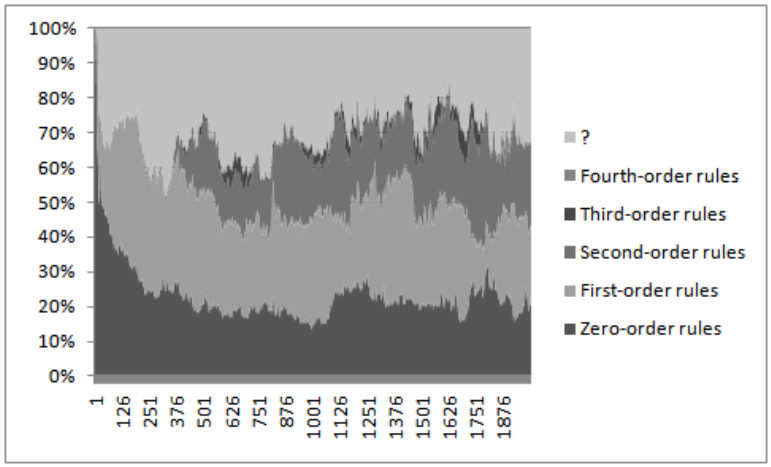


FIGURE 4.1. Composition of an average rule database.

FIGURE 4.2. Composition of the rule database in an ideal situation.

use. The best performing agent does not have such a database, as we can observe by looking at the database in table 4.2. The agent'fs database is specifically suited to handle the rules of its opponents in its simulation but the rule database is not adapted for other possible rules it may encounter in different simulations.

This is not unexpected. An ideal database requires many rules that have to be in the correct order. More importantly, those rules are not useful if the rules they reasoned about are not used by the agent's opponents. There is no incentive to evolve rules that are never required. Therefore, if those rules are not required, they do not necessarily persist in the population.

4.3. Short-cut rules

When we started the experiments, we expected that the most effective attack rules would be rules that reasoned about the defend action of the opponent and effective defend rules would be rules that reasoned about the attack action of the opponent. However, we found in simulations rule databases that contain attack rules that reason about the opponent's belief about the agent's attack rule and not about the opponent's defend action. An example of this phenomenon is presented below:

Example 9.

$$\left[B_O[TRUE \rightarrow LEFT_A]^S \rightarrow RIGHT_A\right]^S$$

The condition of this attack rule is about the opponent's belief about the agent's attack rule. If the agent believes that its opponent believes the agent will attack with left, the agent attacks with right. The defend action of the opponent is not taken into account.

The fact that these rules are persistent in the population show these rules do benefit the agent. If most opponents in a population use the first-order rule $\left[B_S[TRUE \rightarrow LEFT_A]^O \rightarrow LEFT_D\right]^S$ applying the rule above is useful. The agent is one step ahead of its opponent; it uses a short-cut. Short-cut rules are not effective on their own. They are only useful if they are co-evolved with 'normal' rules such as $\left[B_S[TRUE \rightarrow LEFT_A]^O \rightarrow LEFT_D\right]^S$.

We found that in 34 of the 36 simulations short-cut rules with a functionality of a second-order rule were used as a response to first-order rules that were already used in previous generations. For example, the short-cut order rule in example 9 was first applied when the first-order rule $\left[B_S[TRUE \rightarrow LEFT_A]^O \rightarrow LEFT_D\right]^S$ is already present in the rule databases of agents in previous generations.

4.4. Evolution of higher-order rules

In chapter 1 we hypothesized that $n+1$-order attribution evolves only when the larger part of the population has the ability of n-order attribution, since it only makes sense to use $n+1$-order rules if other agents apply n-order rules. We found that in 32 of the 36 experiments $n+1$-order rules arose after n-order rules are found in the population. We found this by analyzing graphs of the average rule database composition.

In the four experiments where $n+1$-order rules arose before the corresponding n-order rules, the $n+1$-order rules were either short-cut rules or the rules did not persist in the population. In experiment 15 (see figure 4.3) an example of the evolution of $n+1$-order rules can be seen without the prior evolution of n-order rules.

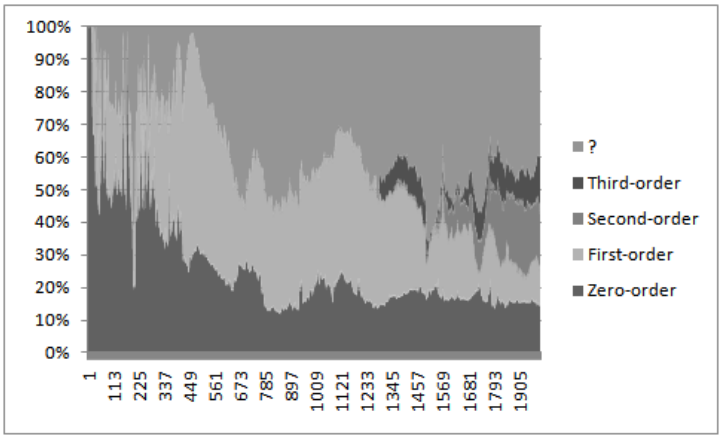


FIGURE 4.3. Average rule database composition for experiment 15. From generation 1296 to 1515, third-order rules persist in the population, but no second-order rules are used. The rule persists because of the presence of a short-cut rule.

An example of when $n+1$-order rules were found when no n-order rules are present can be found in experiment 15 (see figure 4.3). In this particular example the following third-order rule persisted in the population:

$$\Big[B_S\Big[B_S\big[B_O[TRUE \rightarrow LEFT_D]^O \rightarrow RIGHT_A\big]S \rightarrow RIGHT_D\Big]^O \rightarrow LEFT_A\Big]^S$$

It turned out that is was quite useful for the agents to use this rule, since the following short-cut rule with a second-order rule functionality was used in the population:

$$\big[B_O[TRUE \rightarrow LEFT_D]^S \rightarrow RIGHT_D\big]^S$$

This rule states that if the agent believes that its opponent believes that the agent's zero-order defend rule is to defend with left, then the agent will defend with right. This short-cut order rule was only useful because the following first-order rule was used in the population:

$$\big[B_S[TRUE \rightarrow LEFT_D]^O \rightarrow RIGHT_A\big]^S$$

In this rule the agent will attack with right when it believes that its opponent uses a zero-order defend rule in which it defends with left. We have shown that in the experiments where $n+1$-order rules evolve when there are no n-order rules present, the $n+1$-order rules are still useful.

4.5. Errors in reasoning

Although every agent has a perfectly functioning reasoning mechanism (see section 3.3), agents can make wrong inferences about the rules or beliefs

of their opponent. This may occur when an agent fails to establish certain beliefs about the rules or the beliefs of its opponent. We will provide a short example to explain this phenomenon.

$$\begin{bmatrix} B_S\big[B_O[TRUE \rightarrow RIGHT_D]^S \rightarrow LEFT_A\big]^O \rightarrow LEFT_D\big]^S \\ \big[B_S[TRUE \rightarrow RIGHT_A]^O \rightarrow RIGHT_D\big]^S \\ [TRUE \rightarrow LEFT_D]^S \end{bmatrix}$$

TABLE 4.5. Rule database that can lead to wrong inferences

We introduce an agent with the rule database shown in table 4.5. Assume that in the first interaction the agent has to select a defend action. It will use the zero-order defend rule $[TRUE \rightarrow LEFT_D]^S]$. The agent's opponent attacks with right.

In the second turn, the agent defends again. The agent first evaluates the second-order rule (for a detailed explanation on evaluating rules, see section 3.3). The agent will not use the second-order rule. When evaluating this rule the agent investigates if it is possible that its opponent believes the agent's zero-order defend rule is $[TRUE \rightarrow RIGHT_D]^S$. This is not the case.

So, the agent will next evaluate the first-order rule. It establishes the belief that its opponent's zero-order attack rule is $[TRUE \rightarrow RIGHT_A]^O$. Therefore, the agent defends with right. Up to this point, the agent did not establish incorrect beliefs.

However, when the agent is attacked again it will err. When evaluating the second-order rule, the agent will verify whether it is possible that its opponent believes that the agent uses the zero-order defend rule $[TRUE \rightarrow RIGHT_D]^S$. The agent will conclude that this is indeed the case, since it is in the agent's memory that the agent defended with right in the previous interaction.

What happened here was that the agent failed to derive that its opponent may believe that the agent's zero-order defend rule is $[TRUE \rightarrow LEFT_D]^S]$, because never did the agent evaluate a rule in which this belief may have been established.

For this agent, this is not necessarily a bad thing. Even though it did not correctly infer its opponent's beliefs, it may still give an appropriate response to the action of its opponent.

4.6. Order of rules

In most experiments the order of the rules in the agents' databases ranged from zero to three. The highest-order rule found was a fifth order rule. Appendix A contains graphs of the average maximum order of rules in the rule databases for each agent in the population of the simulations we

ran. We calculated these values as follows. First, we determined for the rule database of every agent in the population the order of the highest-order rule in the rule database. We then calculated the average maximum order for the population.

Each line in the graphs in Appendix A represents a run of a particular experiment. On the x-axis the number of generations is depicted. On the y-axis the average maximum order of the rules in the agents' databases is shown.

In table 4.6 we summarized the average maximum order at the end of the simulations for all the parameter values. Note that these values are not the highest values for the average maximum order. In some simulations the average maximum order increased and decreased later on.

	linked-genes			crossover		
MC / AC	200	350	500	200	350	500
0.0005	1.0, 2.8	1.0, 2.0	1, 2.4	3.0, 1.0	2.4, 1.9	2.0, 2.5
0.001	1.1, 2.0	1.0, 1.7	1.1, 1.8	2.0, 2.9	3.0, 2.0	2.8, 3.3
0.005	2.5, 2.0	2.0, 2.0	2.0, 2.1	2.0, 1.9	2.0, 2.1	2.1, 2.5

TABLE 4.6. Average maximum order of rules at the end of an experiment. Each value represents one simulation. In chapter A.2 the average maximum order over time is shown.

In all experiments first- and higher-order rules evolve, but the stability and the order of the rules varies. In the following sections we discuss the influence of the parameters on the evolved rules. Sometimes the effects are difficult to determine, because each experiment was only run twice. The reason is that simulations may take more than a week to run.

4.7. Number of interactions

In preliminary experiments we found that in order for theory of mind to evolve, a minimum number of interactions is required. If agents do not interact with each other often enough, the agents have no experience with their opponents that allows them to use higher-order rules. In that case, no theory of mind will arise, because the agents are not capable of using higher-order rules.

This can be explained as follows. In order to make use of an attack rule with order n, an agent has to attack at least $n+1$ times to benefit from its rules. Let us assume that one agent attacks another agent four times. The first time both the attacking and the defending agent use a zero-order rule because neither agent has interacted with each other, so there are no memories and earlier established beliefs that would allow both agents to select higher-order rules. In the second turn, both agents can use a first-order rule since they both have now experience which they can use to

determine of the condition of their first-order rule is true. During the third turn both agents can use second-order rules and during the fourth turn both agents can use third-order rules.

We then ran experiments in which each agent attacked on average each opponent either 10 times or 20 times. There is no notable effect of changing this parameter, apart from the fact that it takes at least twice as long to run an experiment in which each agent attacks its opponent 20 times.

4.8. The effect of reproduction

In experiments we used two reproduction methods: linked-genes and one-point crossover (see section 3.6.3). We expected that crossover would lead to the evolution of higher-order rules faster than linked genes, because rule databases of the offspring of the agents would be more diverse in the crossover condition. From the graphs in Chapter A.2 we can see that this is indeed the case in 7 of the 9 cases where we compared the results of experiments in which only the reproduction method was varied. Furthermore, in 17 of the 18 experiments where the agents reproduced using crossover the average maximum order was two or higher. In the experiments where the agents reproduced using linked genes this happened in half of the experiments.

An unexpected effect was that linked-genes reproduction led to much more variation of the average maximum order over generations. The variation was less when the mutation probability increased. We discuss the effect of the mutation probability in section 4.10.

Another effect we found was that in experiments where the agents reproduced using crossover the rule databases were larger than in experiments where the agents reproduced using linked genes. Table 4.7 shows the average rule database size at the end of an experiment (after 2000 generations) for all experiments. We did not find an explanation for this phenomenon.

	linked-genes			crossover		
MC / AC	200	350	500	200	350	500
0.0005	23	23	25	35	38	38
0.001	33	36	27	49	51	52
0.005	93	53	87	101	108	95

TABLE 4.7. Average rule database size at the end of a simulation

4.9. The effect of the number of agents

In experiments we varied the number of agents to 200, 350 and 500 agents. From the graphs in section A.4 we could not observe an effect in the experiments where the agents reproduced using cross-over. From the graphs it can seen that in the experiments where agents reproduced using

the linked genes method, higher-order rules are evolved faster when the population size is increased. However, this effect was only observed when the mutation probability was 0.0005 and 0.001 and not in the experiments where the mutation probability was 0.005.

There is a possible explanation that accounts for both the lack of effect in the crossover group and the lack of effect in the linked-genes group with a mutation probability of 0.005. A larger population causes more diversity in the population, because when agents reproduce, mutation occurs. Having more agents may thus lead to the evolution of higher-order rules more quickly. In the experiments where agents reproduced using crossover more diversity is already introduced because the rule databases of two agents are recombined when they reproduce. Similarly, the lack of effect of the number of agents in the experiments where agents reproduced using linked-genes may be attributed to the fact that the mutation probability in those experiments was high. Therefore, there was more diversity in the population.

Since we did not include a method to measure the genetic diversity in the population for the difference experiments, we cannot verify this hypothesis.

4.10. The effect of the mutation probability

In experiments we used the following values for the mutation probability: 0.0005, 0.001 and 0.005 (see 3.6.5 for the choice on the mutation probability values). We expected that higher-order rules would evolve more quickly when the mutation probability was high. However, when we look at the graphs in section A.3 we could not find such a relation.

We discussed in section 4.8 that the maximum average order of the rule databases of the agents varied more over time when the agents reproduced using linked genes as opposed to when the agents reproduced using crossover. A possible explanation for this phenomenon is that there is a difference in diversity in the two conditions. When agents reproduce using linked-genes diversity in the rule databases is introduced by mutation. In the populations where agents reproduce using crossover, diversity is not only introduced by mutation but also by recombining the rule databases of the agents' parents (see section 3.5.3).

Assume we have a population of agents who reproduce using linked genes. One agent in the population evolves a first-order rule while the others have zero-order rules. This agent is highly successful. Its dominance value is very high compared to the values of others. This agent reproduces many children and the average maximum order of the next generation increases. Because one agent is responsible for so many children, the average maximum order may increase from 0 to 1, or from 1 to 2 in one generation. Similarly, assume we have a population which reproduces using linked genes where almost every agent has a first-order rule that gives an appropriate response to a particular zero-order rule. An agent who does not use such a zero-order rule does not suffer from the second-order rule. If the agent has

a high dominance value and produces many offspring, the second-order rule may become useless since it can never be applied and the rule disappears from the population altogether.

This is less likely to happen in a population in which the agents reproduce using crossover. Because the diversity is larger in the population not all agents necessarily lose most interactions against an agent with a higher-order rule. Suppose that an agent has evolved a second-order attack rule that is capable of dealing with a specific first-order defend rule. Not all agents may use such a defend rule and the agent may perform well in the population but not against all agents.

This finding can also be observed from the dominance values in the experiment data. There is less variation in the dominance values in experiments where the agents reproduced using crossover. We did not include a measurement for the rule database diversity in the population. This may have aided in verifying this explanation.

4.11. Summary

In this chapter we have given an example of a rule database (section 4.2) and we showed that higher-order rules are evolved when agents interact with each other in a competition setting (section 4.6). In most experiments the order of the rules varied between 0 and 3. The highest-order rule found was a fifth order rule. We found that $n + 1$-order rules only persist in the population when the corresponding n-order rule is present or when there are short-cut rules in the population (section 4.3). Also, we have shown that the evolved rule databases are adapted to the population (section 4.2). Agents do not always establish correct beliefs about their opponents' beliefs or rules (section 4.5), but this does not necessarily result in poor action selection.

We found that the average maximum order of rules varies more greatly over time in populations where agents reproduced using linked genes (section 4.8). The reproduction method also influenced the average rule database size at the end of a population: crossover resulted in larger databases. As expected, letting agents reproduce using crossover led to the evolution of higher-order rules faster than when the agents reproduced using the linked genes method.

CHAPTER 5

Discussion

In this chapter we discuss what the results presented in Chapter 4 implicate for research on theory of mind. We look at the role of competition in the evolution of theory of mind and discuss the orders of theory of mind that evolve.

5.1. The role of competition in the evolution theory of mind

The first hypothesis presented in Chapter 1 was that competition is a possible driving factor behind the evolution of theory of mind. We showed that a competitive task can lead to the evolution of agents with a higher-order theory of mind. In our model, the highest order of rules that persisted in the population were fourth-order rules. Coincidentally, this corresponds with the highest order of theory of mind humans are capable of (Kinderman et al., 1998).

The fact that we were capable of evolving theory of mind in a competitive situation corresponds with the fact that theory of mind-like behavior in animals is mostly found in a competition setting (see section 2.2).

5.2. Evolving higher-order rules

The second hypothesis was that $n+1$-order attribution evolves only when the larger part of the population has the ability of n-order attribution, because it only makes sense to use $n+1$-order rules if other agents apply n-order rules. We found this phenomenon in 32 of the 36 experiments (see section 4.4). We also found that $n+1$-order rules do not persist in a rule database when there is no corresponding n-order rule unless short-cut rules with a similar function are found (see section 4.3).

In short-cut rules the agent selects its action based on the agent's opponent's beliefs on the agent's rule. The agent does not take the opponent's action into account. An example of a short-cut rule is the rule in example 10. This rule is not assigned an order because the rule does not take the action of the opponent into account (section 3.3).

Example 10.

$$\left[B_O[TRUE \rightarrow LEFT_A]^S \rightarrow RIGHT_A\right]^S$$

If a short-cut rule with the functionality of an n-order rule is found in the population, rules with order $n+1$ persist. The short-cut rule in example

10 only benefits the agent when most agents in the population use the rule $\left[B_S[TRUE \rightarrow LEFT_A]^O \rightarrow LEFT_D\right]^S$. These short-cut rules were found in 34 of the 36 experiments (see section 4.3). This shows that when most individuals in the same population use a similar strategy, individuals can exploit this. In this model there was no punishment for evaluating longer rules. However, in real life it seems plausible that higher-order theory of mind requires more effort. Using a short-cut may reduce this effort.

We discussed in section 4.3 that short-cut rules do not necessarily work when agents interact against agents from a different population. When we placed agents who performed well in their own population in a population that was the result of an experiment with the same parameters, the agent did not perform so well. This indicates that an agent's behavior is adapted to its own environment. The same phenomenon can be found in nature. If we place the mouse with light fur (as discussed in section 2.5.3) in a dark environment, it may not perform as well as it would in its own light colored environment since the mouse is not adapted to its new environment.

The highest rules found in simulations were rules with an order up to five but those rules did not persist. However, in most simulations the most common higher-order rules are up to order three. In theory, given the fact that each agent attacked each opponent on average ten times, ninth order rules could have been evolved. Rules of order ten and higher would not benefit the agent, since those would never be applied (see section 3.6.4 for more on this). One possible explanation for the fact that the order of rules was usually not larger than three or four is the fact that the resulting databases were too small for higher-order rules to be effective. A rule database that contains third-order rules that are suitable to respond to every possible second-order rule an opponent may use, contains almost 300 rules (see page 58 for an explanation). The largest database that was found only had 108 rules (see table 4.7 on page 64).

Letting the simulations run for a longer period could lead to higher orders of theory of mind. We choose not to run longer simulations, since running a simulation already lasted up to a week.

5.3. Interpretation of theory of mind in the model

When we started to develop this model, we chose to evolve rules that represent theory of mind. The contents of the rule could be freely varied but the rules had a fixed format (see section 3.3). We made two strong assumptions about the agent's reasoning capability. Firstly, an agent is consistent in its rule evaluation and given the information it has it reasons perfectly. In other words, given a specific instance of a rule database and memory, the inferred actions and beliefs are always the same for every agent. Secondly, all agents behave this way and this is known by all agents. This allows agents to reason about other agents' rules and beliefs. If an agent cannot be sure about its opponent's rules nor about the opponent's reasoning

mechanism, it is almost impossible for the agent to establish beliefs about its opponent's rules and beliefs.

This approach has similarities with theory-theory which is discussed in section 2.1. The agents have a reasoning mechanism, just as humans have cognitive capabilities. The agents in this model evolve their theories whereas humans develop them during their lifetime (Whiten, 2002). We also could have approached the problem from the simulation-theory point of view. In that case we would have given the agents sensible rules in which the agents respond appropriately to the expected action of their opponent. The quality of each agent's reasoning mechanism could differ. For example, the agent may be able to keep up only with a limited order of theory of mind, because of computational limitations. This would require the agent to assess, or rather simulate, the 'cleverness' of its opponent. After finishing this project we still feel that evolving the agent's rules rather than the reasoning mechanism is a practical choice, because it allows easy insight into the workings of the agent.

5.4. Relevance for research on theory of mind

In this project we showed that it is possible to develop an agent-based model in which the evolution of theory of mind can be simulated. Unfortunately, since there is little known about the evolution of theory of mind in humans, we cannot compare the results of the model to empirical data. The next step is to devise tasks that are hypothesized to be a driving force behind the evolution of theory of mind, such as mixed-motive interactions and cooperation.

Lastly, with this project we showed that we do no longer only have to theorize about the evolution of theory of mind, but that we can actually simulate it in an insightful way. This model has to be extended with costs and perhaps other environmental factors, but it is worthwhile to continue research on the evolution of theory of mind with the proposed setup.

5.5. Simulations

The simulations were run on the Millipede Cluster[1] of the University of Groningen. Running simulations on the cluster allowed us to run all experiments simultaneously. The cluster contains 255 nodes with multiple Opteron 2.6 GHz cores. These nodes are connected to 110 TB of storage. For each experiment, one core was used. In each experiment, two simulations were run (each experiment was run twice) successively on one core. It would have been possible to reprogram the simulation in such a way that two simulations would have been run on two cores, thus reducing the time it would take to run experiment by half.

The duration of each experiment varied between one day (for a simulation with 200 agents) and a week (for a simulation with 500 agents). A

[1] http://www.rug.nl/cit/hpcv/faciliteiten/HPCCluster

queuing mechanism is used on the Millipede cluster to allow everyone a fair share of computing time. However, it happened rarely that a job was in queue before it was run. [2]

[2] I strongly recommend everyone who is running simulations that take more than a few hours to request on account on the Millipede cluster and use it to run simulations. It is easy (if you have some skill with Linux) and it allows you to use your computer for other tasks (such as writing a thesis, for example).

CHAPTER 6

Conclusion and future work

To contribute to the research on higher-order social cognition we constructed a model that allows us to experiment with the evolution of higher-order theory of mind (see Chapter 3). In this model the agents were given a competition task. The agents competed with individuals within the population. The agents' behavior followed from rules in the agents' rule databases. Rules allow agents to reason about their opponent's beliefs and rules. The rule databases were evolved over time. Two reproduction methods were used: linked genes and one-point crossover. We have chosen a model in which the form of the rules is fixed, but the contents of the rules can be varied. Two assumptions were made. Firstly, all agents have the same reasoning mechanism which works perfectly given the agent's rules and the agent's observations. Secondly, this information is common knowledge. We found that competition is a driving force behind the evolution of theory of mind in this model (see Chapter 4 and Chapter 5).

In this chapter we summarize the main findings of this project and propose how the model can be expanded and used in future projects.

6.1. Conclusion

In the agent-based model higher-order theory of mind has evolved in a competition setting. In every simulation higher-order rules evolved. The order of the rules usually varied between zero and three. The highest-order rule evolved was a fifth-order rule (section 4.6). The size of the rule databases varies between 23 and 108, depending on the experiment parameters. Only 10% to 30% of the rules in the rule database are actually used. This does not mean those rules are useless. Unused rules may be useful to maintain genetic diversity, since unused rules may become useful when they change due to mutation (section 4.2).

We found that the agents' rules are adapted to the environment. When the best performing agent from one population interacts with agents from another population that was simulated using the same experiment parameters, the agent was not as successful as it was in its original population. One reason is that the evolved rule database of the agents are too small to give an appropriate response to every possible rule of the opponent (section 4.2). A second reason is that some of the rules in the agents' rule databases are specifically suited to deal with rules in the population. These are the

short-cut rules (section 4.3). A short-cut rule takes only the opponent's belief about the agent's rule into account, but not the action of the opponent. This rule can backfire against an agent, unless most agents in the population use the corresponding rule. We expected that $n + 1$-order rules evolve when the corresponding n-order rules are present in the population. We found that $n + 1$-order rules persist when the corresponding n-order rules are found or when the corresponding short-cut rules are also found in the population (section 4.3).

6.2. Future work

In the current model we made the assumption that each agent has the same perfect reasoning mechanism and this is common knowledge; it would be very interesting to investigate if higher-order theory of mind will arise when less constraints are applied to the model. One option is to expand the action set with a random action. We suspect this will not lead to the evolution of higher-order theory of mind at all, because there is no better response to an opponent who selects random actions than to select a random action yourself. Another option is to introduce errors in action execution (the agents selects an action but another is executed) or errors in memory. This adds a probabilistic factor to the model.

As discussed in section 5.3 we can deviate from the theory-theory approach used in this model to the simulation-theory. We give the agents sensible rules but let the reasoning mechanism evolve over time. This requires the agents to make predictions about their opponents' reasoning capabilities.

In the current model the agents are not punished for having large rule databases. 70% to 90% of the rules in the agents' rule databases are not used. We expect that these junk rules help the agent in evolving useful rules by maintaining genetic diversity (section 4.2). To test this hypothesis we can punish agents with larger rule databases. Also, we can punish the agents for using rules that require extensive reasoning about the mental states of others, since, in real life, this may require mental effort. We expect that this results in more short-cut rules and that the rule databases of the agents are even more strongly adapted to the population.

The order of the rules evolved in simulations were usually not higher than three. We expect that this is caused by the limited database size at the end of a simulation (see section 5.2). Running a simulation for a longer period than 2000 generations may result in agents with larger rule databases and perhaps rules with an order higher than three. However, simulations with 500 agents already take a week to run. Running the experiment on the Millipede cluster of the university allowed us to run all the experiments simultaneously.

The current program can be rewritten in such a way that multiple computer nodes are used to calculate the outcome of interactions within a generation. This allows us to calculate the outcome of more interactions at

the same time and thus reduces simulation time. This will also allow us to experiment with larger populations.

In this project we have shown that it is possible to simulate the evolution of theory of mind in a competition setting using an agent-based model in which the agents select their actions using rules. No longer do we have to theorize about the evolution of theory of mind; we can simulate it in an insightful way. This allows us to test other hypotheses on the evolution of theory of mind (see section 2.3.2), which are cooperation, mixed-motive interactions and deception.

APPENDIX A

Results of the experiments

In this Appendix we show some of the results in graphs. The complete data set may be found on `http://tom.lisepijl.nl`. Chapter C contains further information on the files in the data set.

In this chapter present the average maximum order of the rule databases of the agents in graphs. For every agent in the population we determined what the highest order of the rules in the agent's database was. We then averaged these orders for every agent in a generation.

The x-axis represents the generations. The y-axis represents the average maximum order of rules for every agent in the simulation. Each line represents one run of the simulation. These results of the simulations are not averaged, so we can clearly see if an order increases from n to $n + 1$ over time.

In section A.1 the experiments and their corresponding parameters are listed. Section A.2 contains the average maximum order over time for every experiment that was run. The graphs are placed in such a way that the influences of the reproduction method are easily observed. In section A.3 the effects of different mutation probabilities are compared and in section A.4 the effects of different population sizes are compared. Lastly, in section A.5 the graphs of the average maximum order of the rules are ordered by the average size of the rule database at the end of the different simulations.

A.1. Overview experiments

We devised 36 experiments, of which 18 experiments (the experiments with an even number) were only run in preliminary experiments. Table A.1 contains an overview.

A.2. Comparing reproduction method

On each page, two graphs are printed. The parameters of the experiments that resulted in these graphs are equal, except for the reproduction method.

No.	Agent count	Mutation probability	Interactions	Reproduction	
1	200	0.001	10	linked genes	
2	200	0.001	20	linked genes	preliminary
3	200	0.001	10	crossover	
4	200	0.001	20	crossover	preliminary
5	350	0.001	10	linked genes	
6	350	0.001	20	linked genes	preliminary
7	350	0.001	10	crossover	
8	350	0.001	20	crossover	preliminary
9	500	0.001	10	linked genes	
10	500	0.001	20	linked genes	preliminary
11	500	0.001	10	crossover	
12	500	0.001	20	crossover	preliminary
13	200	0.0005	10	linked genes	
14	200	0.0005	20	linked genes	preliminary
15	200	0.0005	10	crossover	
16	200	0.0005	20	crossover	preliminary
17	350	0.0005	10	linked genes	
18	350	0.0005	20	linked genes	preliminary
19	350	0.0005	10	crossover	
20	350	0.0005	20	crossover	preliminary
21	500	0.0005	10	linked genes	
22	500	0.0005	20	linked genes	preliminary
23	500	0.0005	10	crossover	
24	500	0.0005	20	crossover	preliminary
25	200	0.005	10	linked genes	
26	200	0.005	20	linked genes	preliminary
27	200	0.005	10	crossover	
28	200	0.005	20	crossover	preliminary
29	350	0.005	10	linked genes	
30	350	0.005	20	linked genes	preliminary
31	350	0.005	10	crossover	
32	350	0.005	20	crossover	preliminary
33	500	0.005	10	linked genes	
34	500	0.005	20	linked genes	preliminary
35	500	0.005	10	crossover	
36	500	0.005	20	crossover	preliminary

TABLE A.1. Overview of experiments

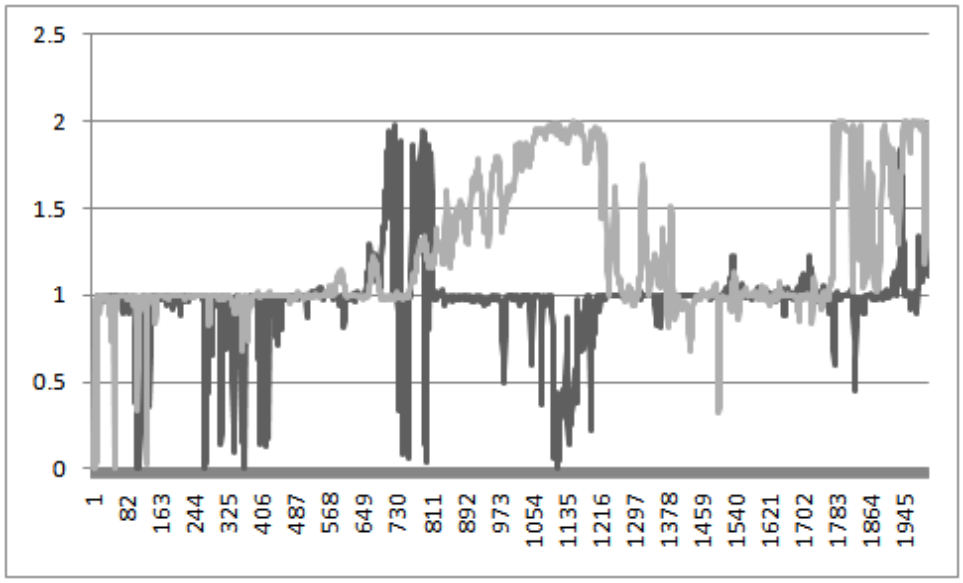


FIGURE A.1. Experiment: 1. Number of agents: 200; mutation probability: 0.001; reproduction method: linked genes; interactions: 10.

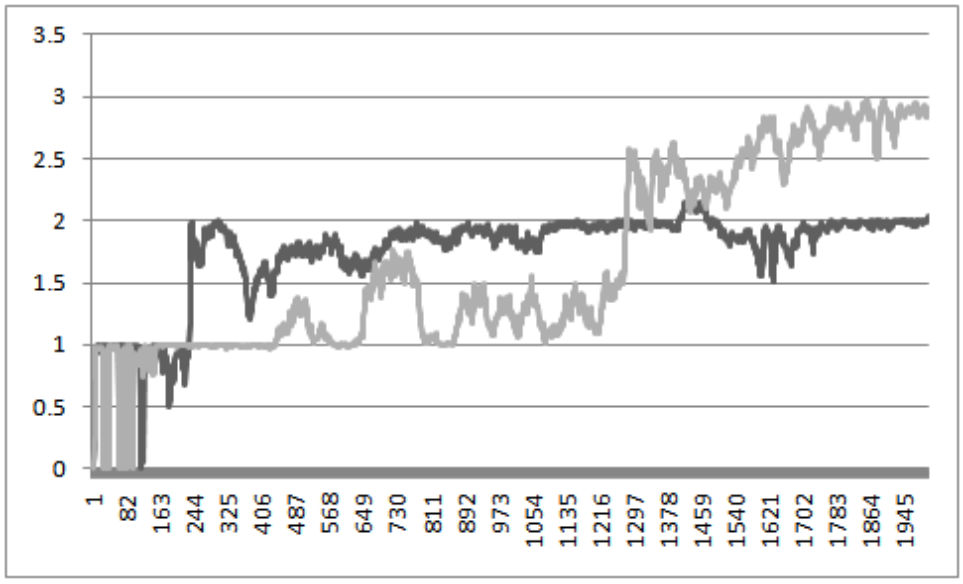


FIGURE A.2. Experiment: 3. Number of agents: 200; mutation probability: 0.001; reproduction method: crossover; interactions: 10.

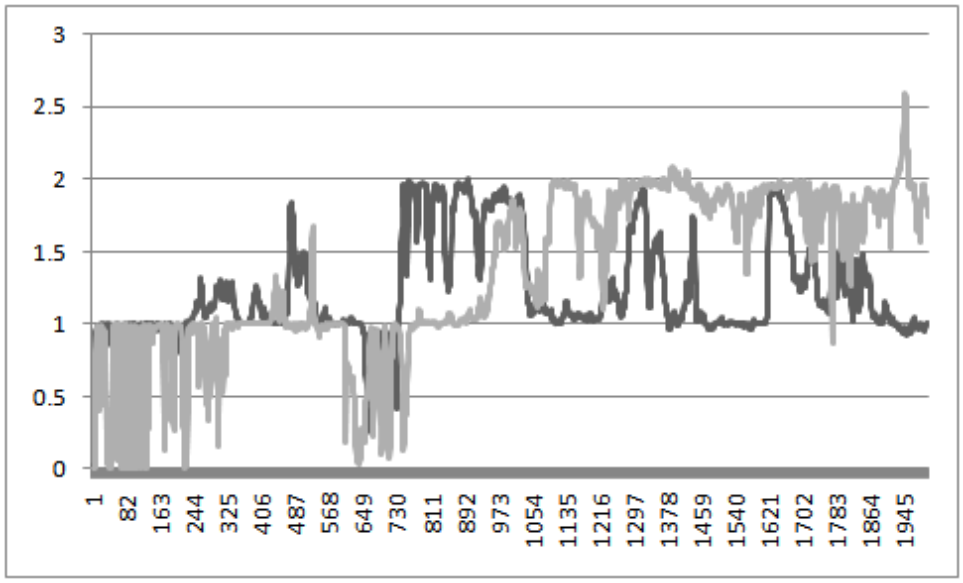


FIGURE A.3. Experiment: 5. Number of agents: 350; mutation probability: 0.001; reproduction method: linked genes; interactions: 10.

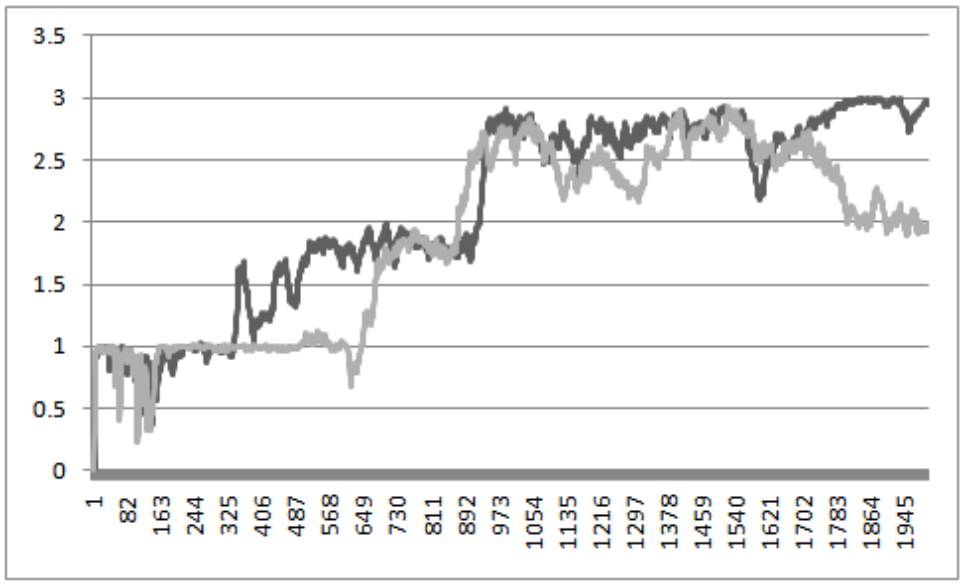


FIGURE A.4. Experiment: 7. Number of agents: 350; mutation probability: 0.001; reproduction method: crossover; interactions: 10.

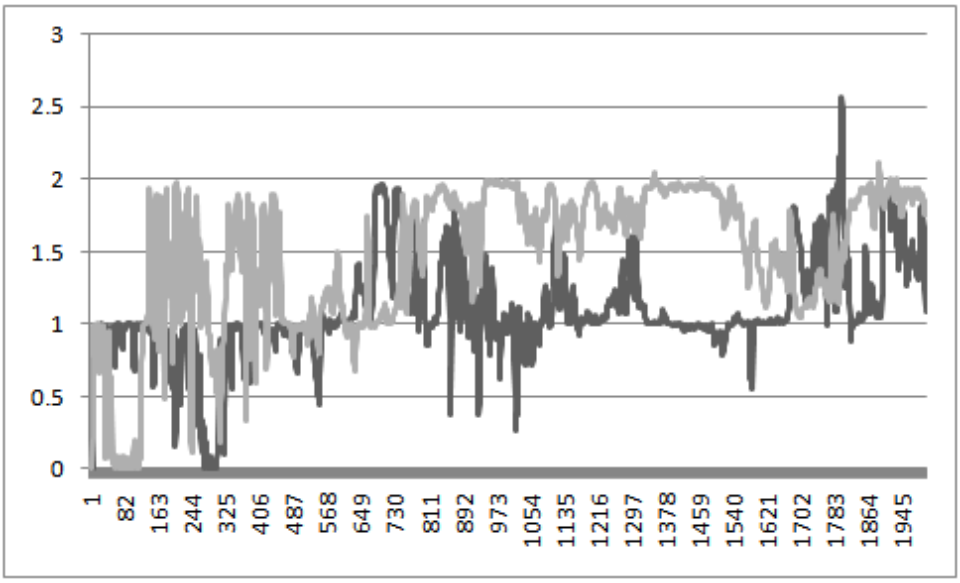


FIGURE A.5. Experiment 9. Number of agents: 500; mutation probability: 0.001; reproduction method: linked genes; interactions: 10.

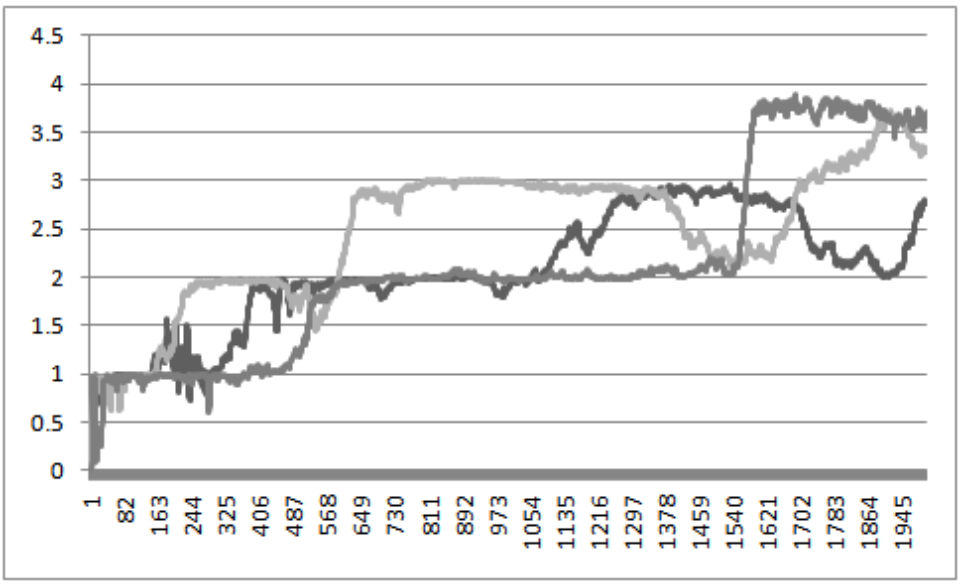


FIGURE A.6. Experiment: 11. Number of agents: 500; mutation probability: 0.001; reproduction method: crossover; interactions: 10.

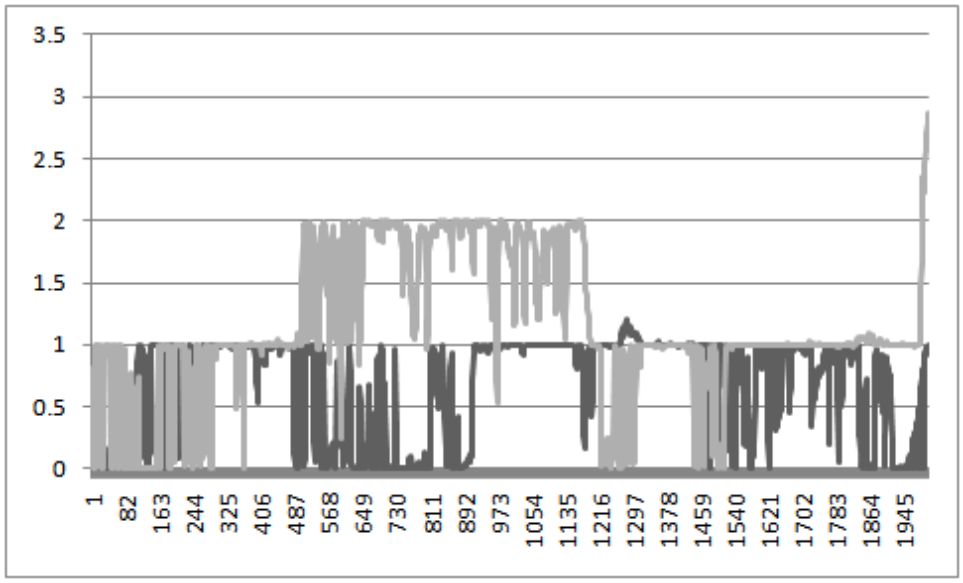


FIGURE A.7. Experiment: 13. Number of agents: 200; mutation probability: 0.0005; reproduction method: linked genes; interactions: 10.

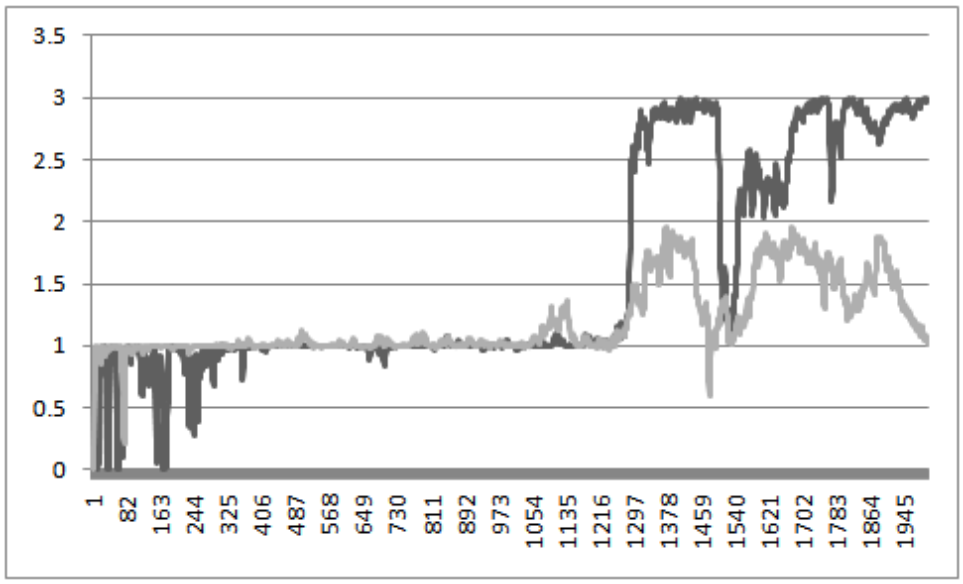


FIGURE A.8. Experiment: 15. Number of agents: 200; mutation probability: 0.0005; reproduction method: crossover; interactions: 10.

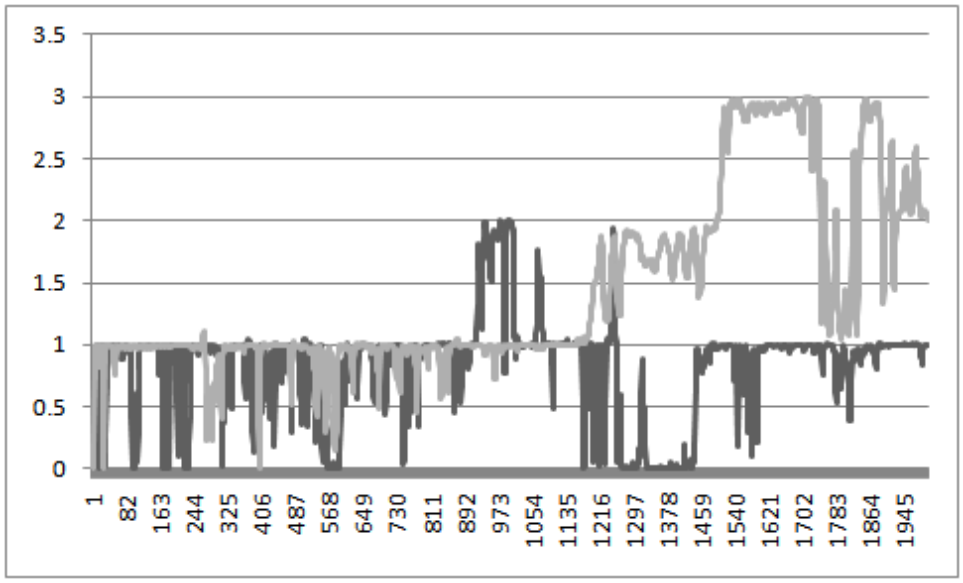


FIGURE A.9. Experiment: 17. Number of agents: 350; mutation probability: 0.0005; reproduction method: linked genes; interactions: 10.

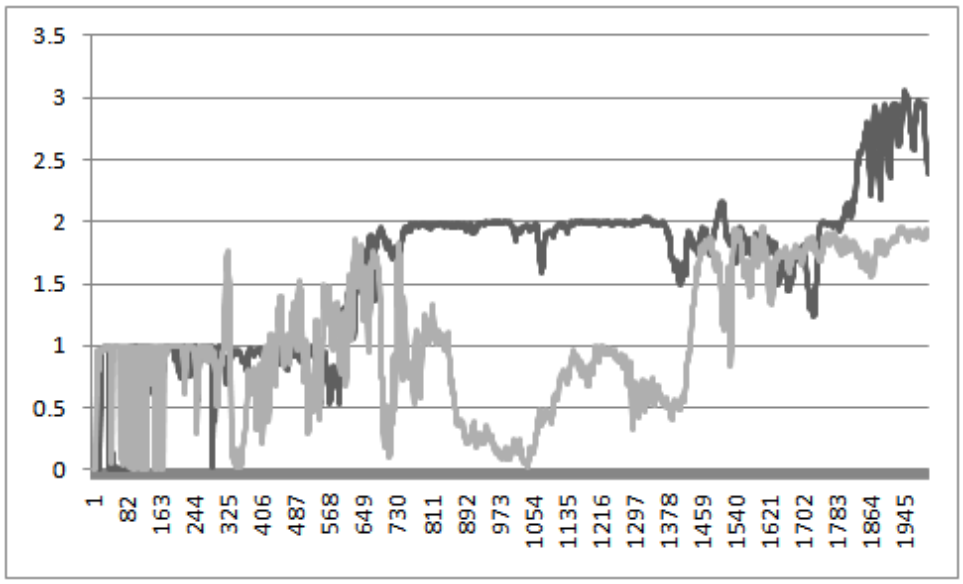


FIGURE A.10. Experiment: 19. Number of agents: 350; mutation probability: 0.0005; reproduction method: crossover; interactions: 10.

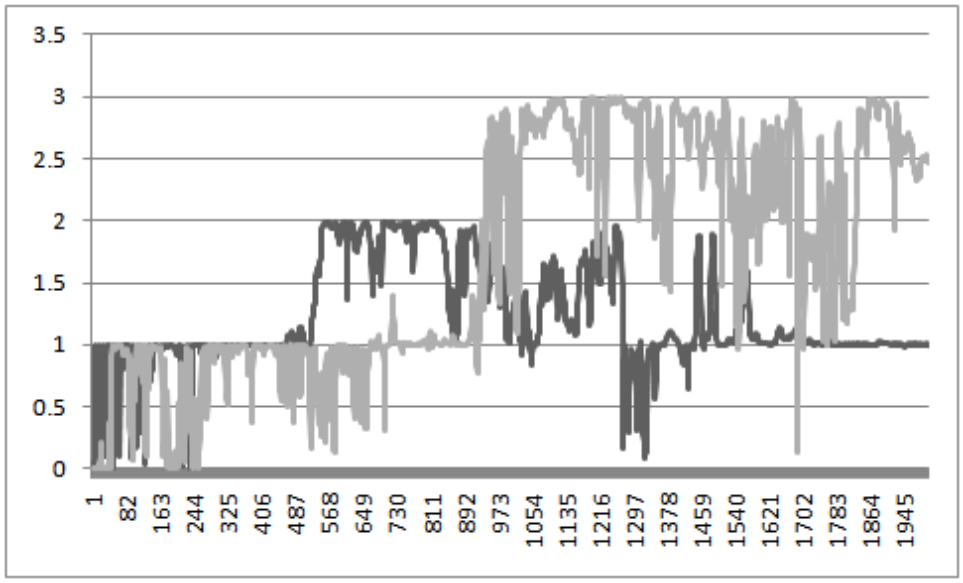


FIGURE A.11. Experiment: 21. Number of agents: 500; mutation probability: 0.0005; reproduction method: linked genes; interactions: 10.

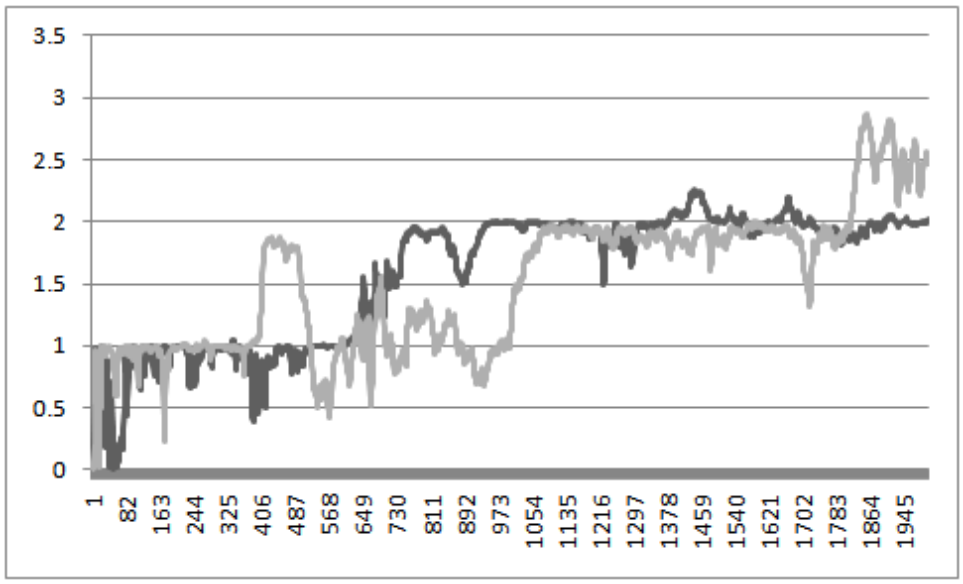


FIGURE A.12. Experiment: 23. Number of agents: 500; mutation probability: 0.0005; reproduction method: crossover; interactions: 10.

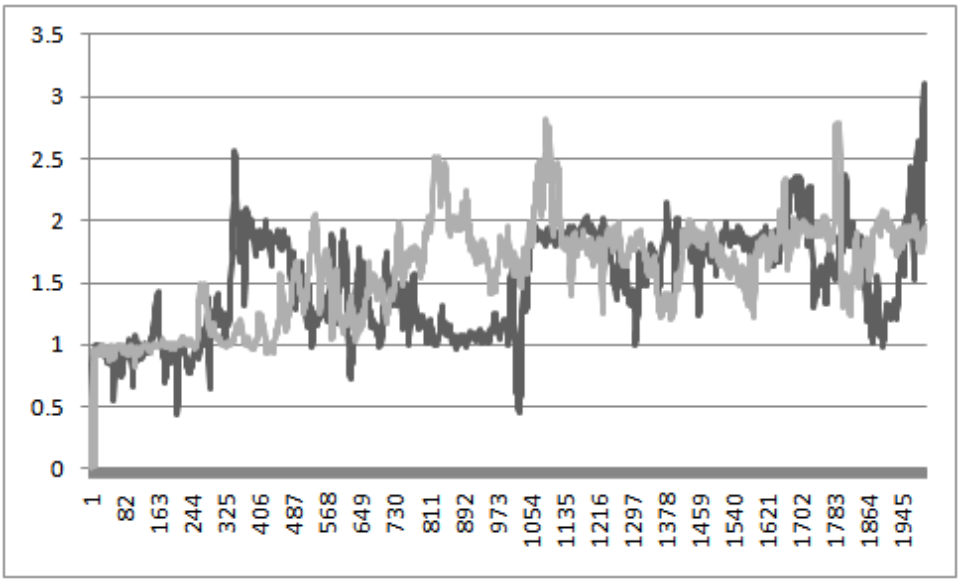


FIGURE A.13. Experiment: 25. Number of agents: 200; mutation probability: 0.005; reproduction method: linked genes; interactions: 10.

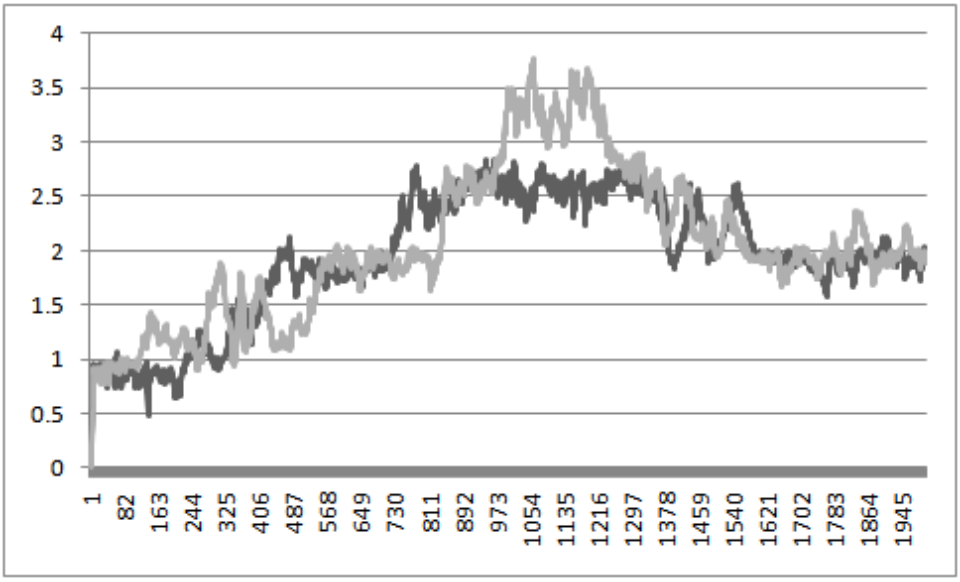


FIGURE A.14. Experiment: 27. Number of agents: 200; mutation probability: 0.005; reproduction method: crossover; interactions: 10.

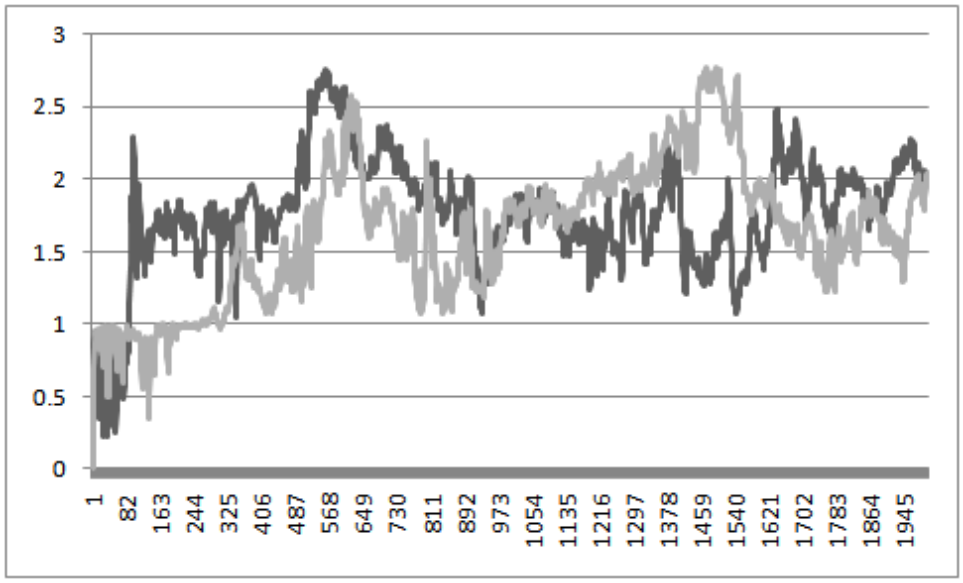


FIGURE A.15. Experiment: 29. Number of agents: 350; mutation probability: 0.005; reproduction method: linked genes; interactions: 10.

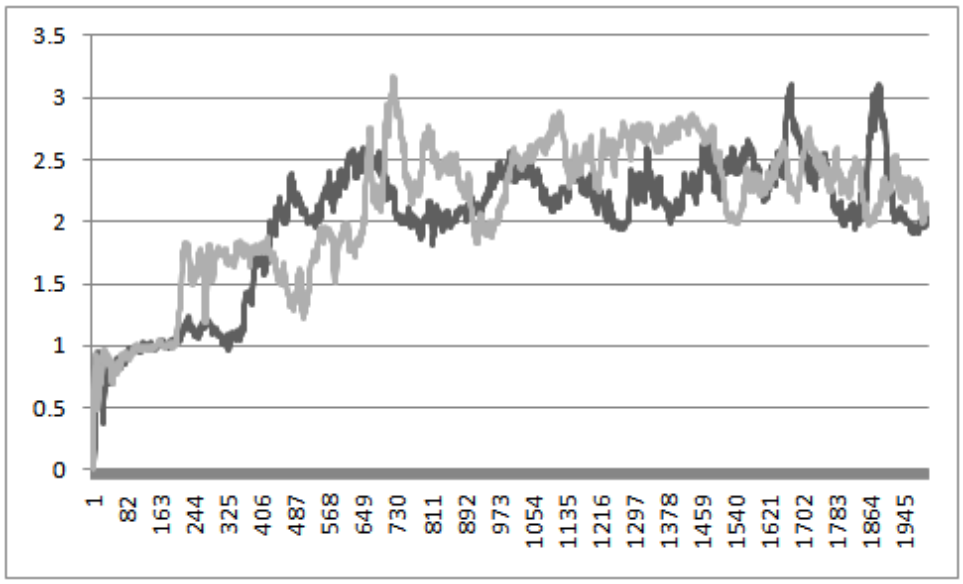


FIGURE A.16. Experiment: 31. Number of agents: 350; mutation probability: 0.005; reproduction method: crossover; interactions: 10.

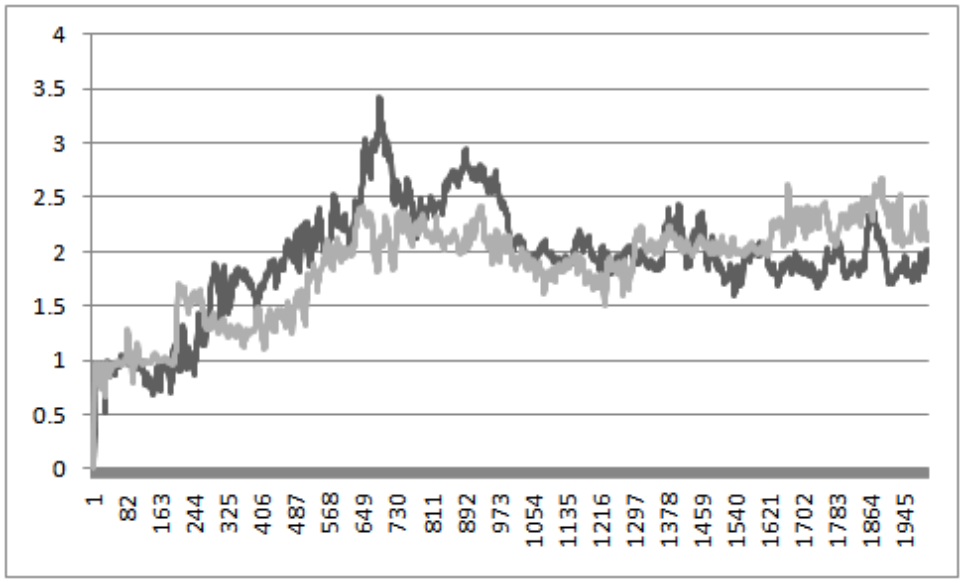


FIGURE A.17. Experiment: 33. Number of agents: 500; mutation probability: 0.005; reproduction method: linked genes; interactions: 10.

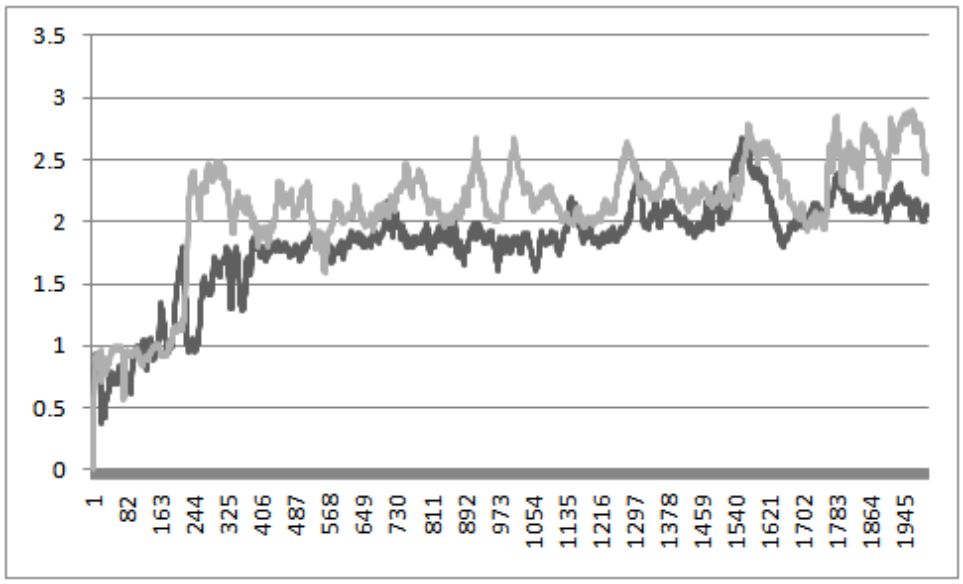
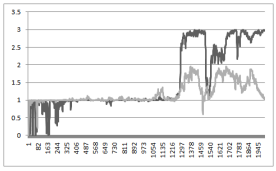
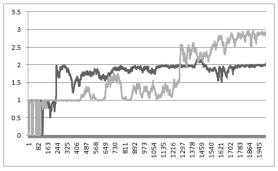


FIGURE A.18. Experiment: 35. Number of agents: 500; mutation probability: 0.005; reproduction method: crossover; interactions: 10.

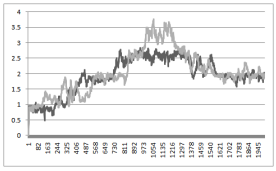
A.3. Comparing mutation probabilities



(a) Mutation probability: 0.0005

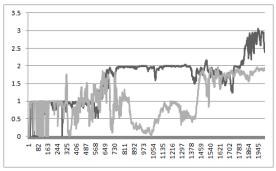


(b) Mutation probability: 0.001

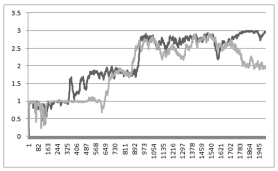


(c) Mutation probability: 0.005

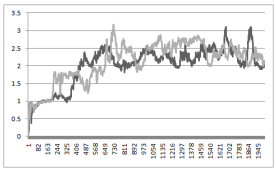
FIGURE A.19. Agents: 200; reproduction: crossover



(a) Mutation probability: 0.0005

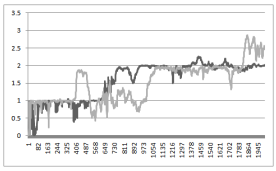


(b) Mutation probability: 0.001

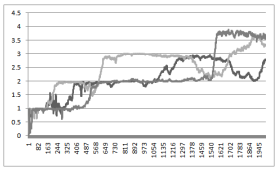


(c) Mutation probability: 0.005

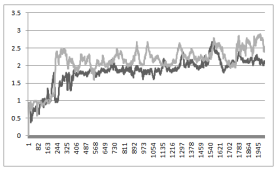
FIGURE A.20. Agents: 350; reproduction: crossover



(a) Mutation probability: 0.0005

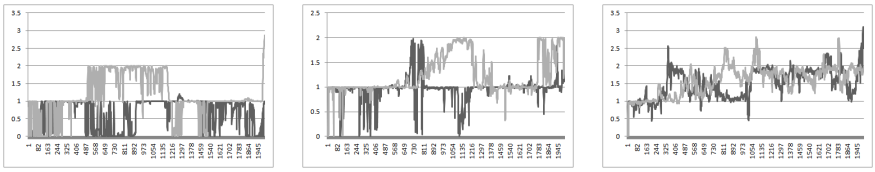


(b) Mutation probability: 0.001



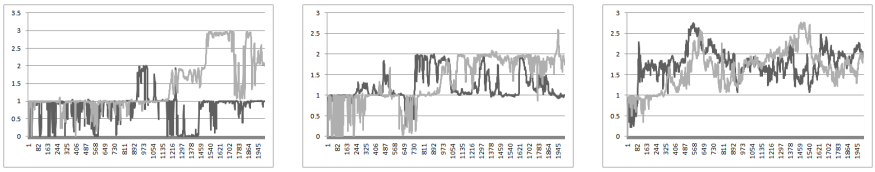
(c) Mutation probability: 0.005

FIGURE A.21. Agents: 500; reproduction: crossover



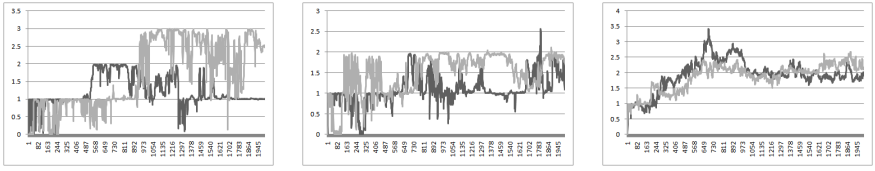
(a) Mutation probability: 0.0005 (b) Mutation probability: 0.001 (c) Mutation probability: 0.005

FIGURE A.22. Agents: 200; reproduction: linked-genes



(a) Mutation probability: 0.0005 (b) Mutation probability: 0.001 (c) Mutation probability: 0.005

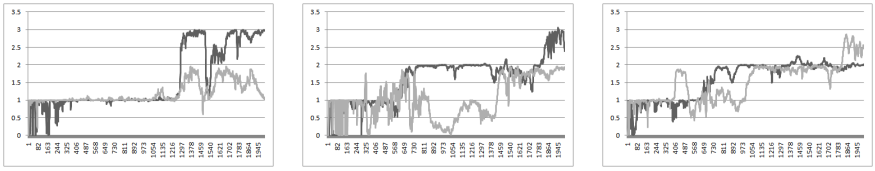
FIGURE A.23. Agents: 350; reproduction: linked-genes



(a) Mutation probability: 0.0005 (b) Mutation probability: 0.001 (c) Mutation probability: 0.005

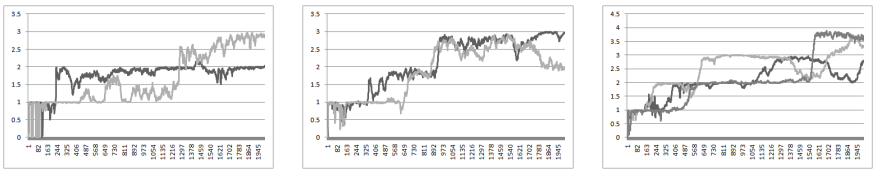
FIGURE A.24. Agents: 500; reproduction: linked-genes

A.4. Comparing population size



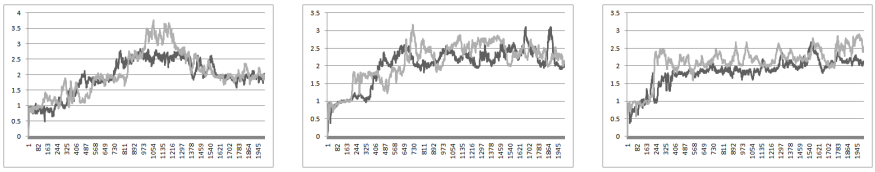
(a) Population size: 200 (b) Population size: 350 (c) Population size: 500

FIGURE A.25. Mutation probability: 0.0005; reproduction: crossover



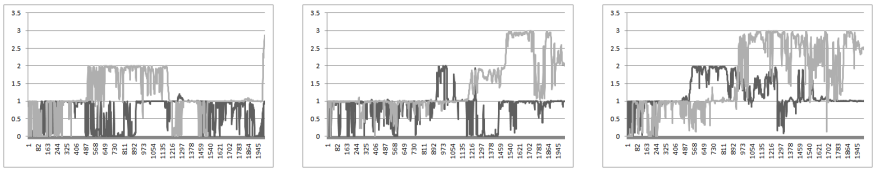
(a) Population size: 200 (b) Population size: 350 (c) Population size: 500

FIGURE A.26. Mutation probability: 0.001; reproduction: crossover



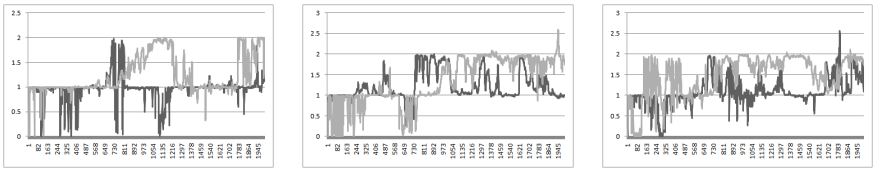
(a) Population size: 200 (b) Population size: 350 (c) Population size: 500

FIGURE A.27. Mutation probability: 0.005; reproduction: crossover



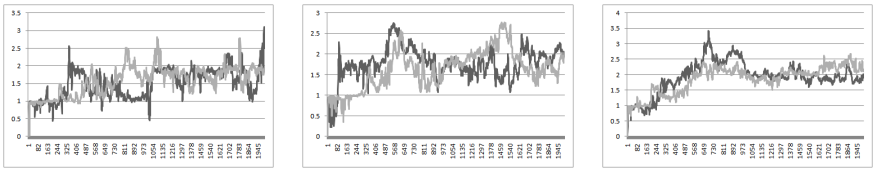
(a) Population size: 200 (b) Population size: 350 (c) Population size: 500

FIGURE A.28. Mutation probability: 0.0005; reproduction: linked-genes



(a) Population size: 200 (b) Population size: 350 (c) Population size: 500

FIGURE A.29. Mutation probability: 0.001; reproduction: linked-genes



(a) Population size: 200 (b) Population size: 350 (c) Population size: 500

FIGURE A.30. Mutation probability: 0.005; reproduction: linked-genes

A.5. Plots ordered on rule database size

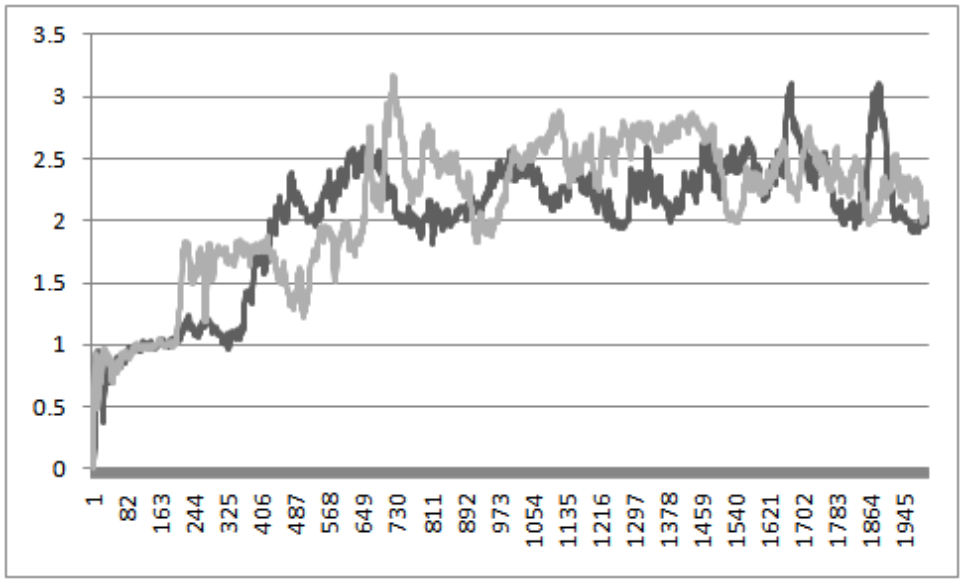


FIGURE A.31. Experiment: 31

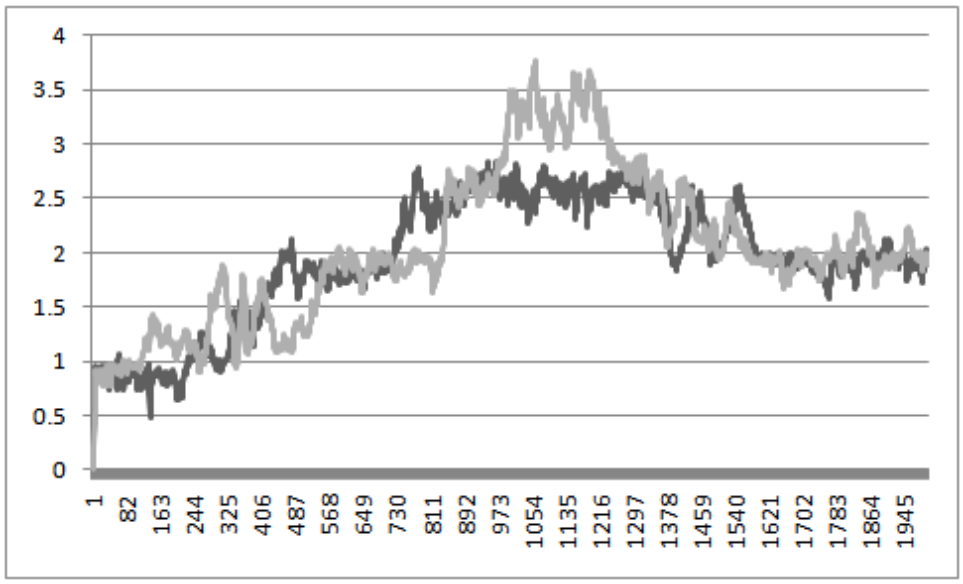


FIGURE A.32. Experiment: 27

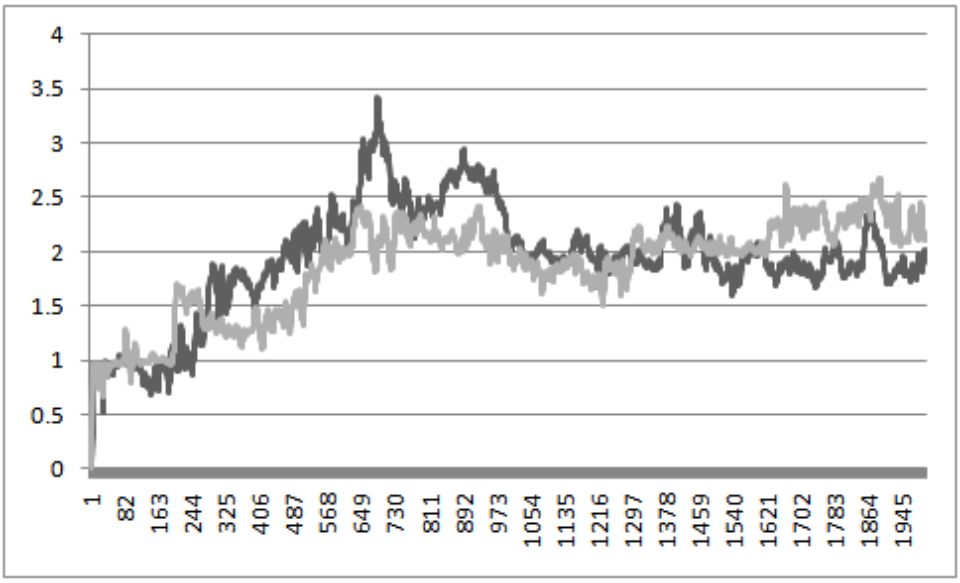


FIGURE A.33. Experiment: 33

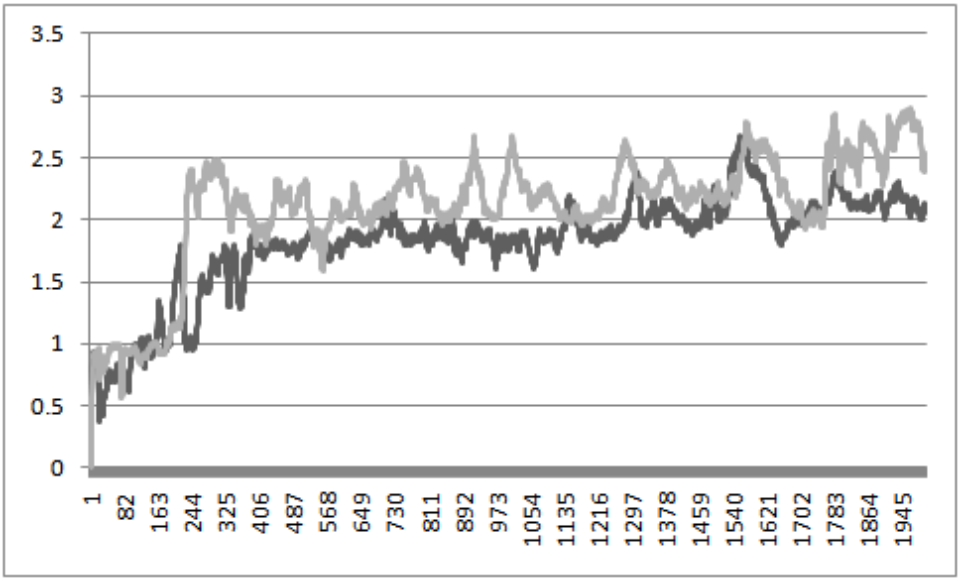


FIGURE A.34. Experiment: 35

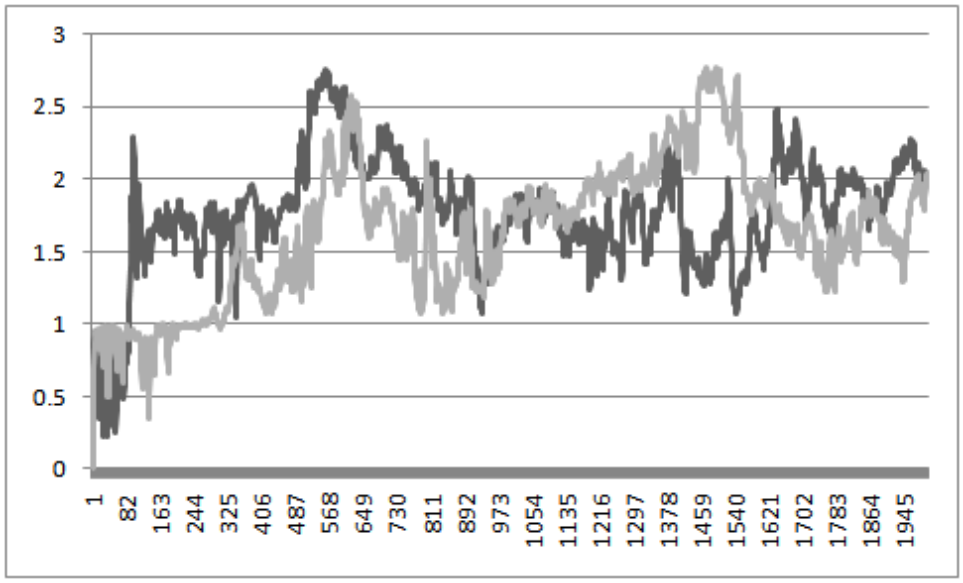


FIGURE A.35. Experiment: 29

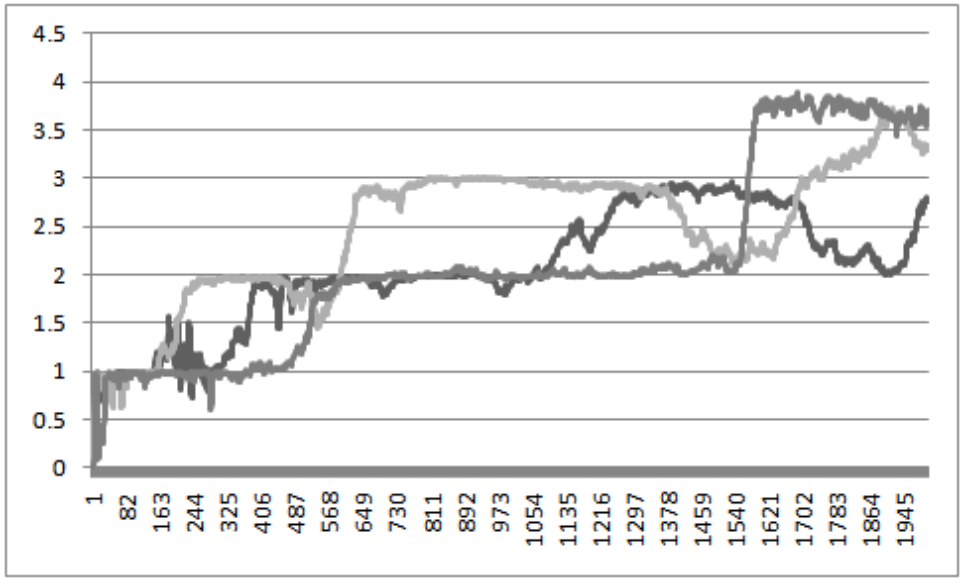


FIGURE A.36. Experiment: 11

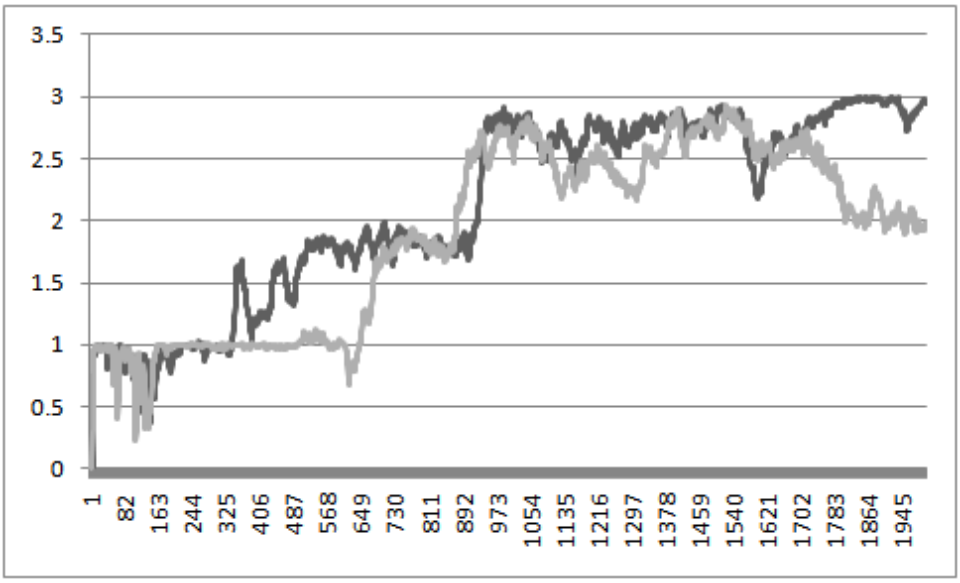


FIGURE A.37. Experiment: 7

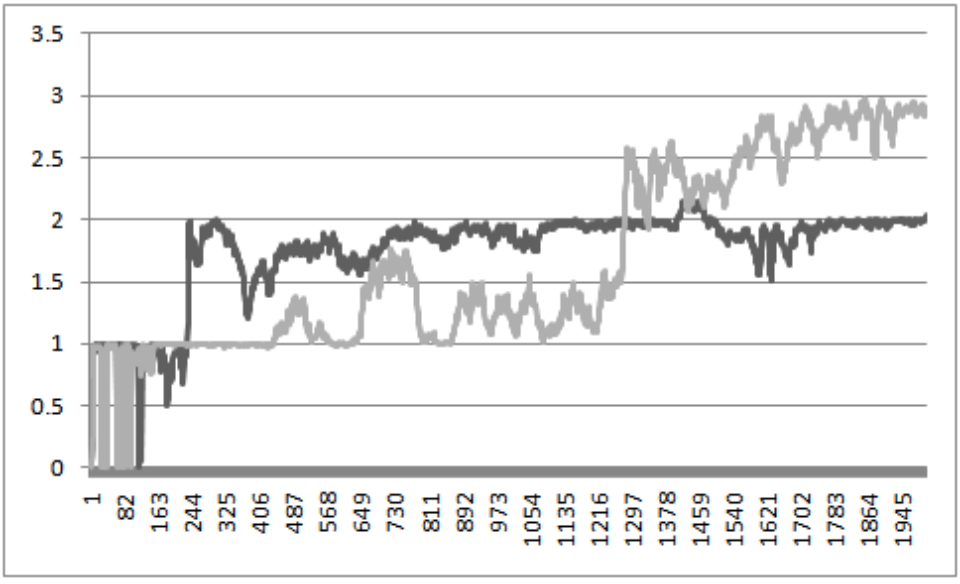


FIGURE A.38. Experiment: 3

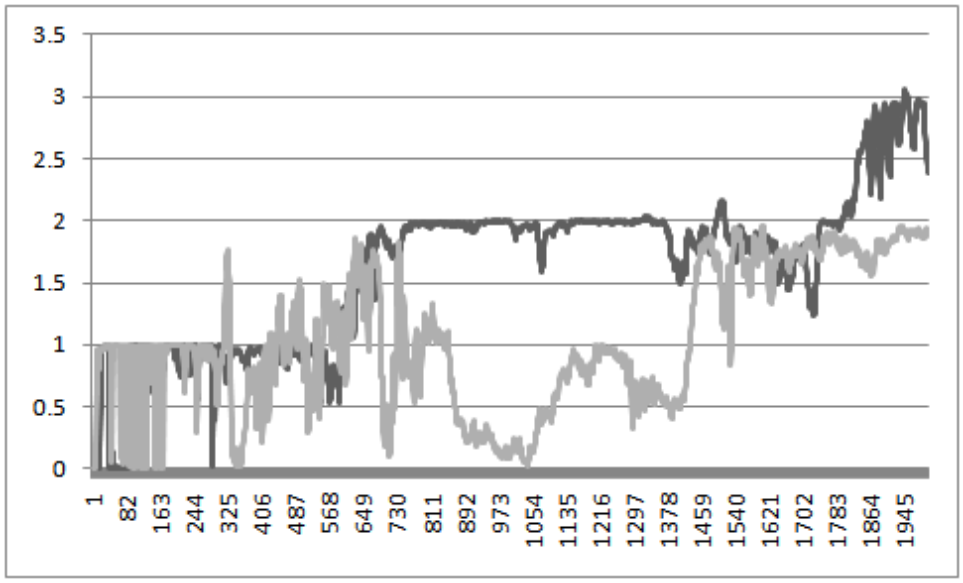


FIGURE A.39. Experiment: 19

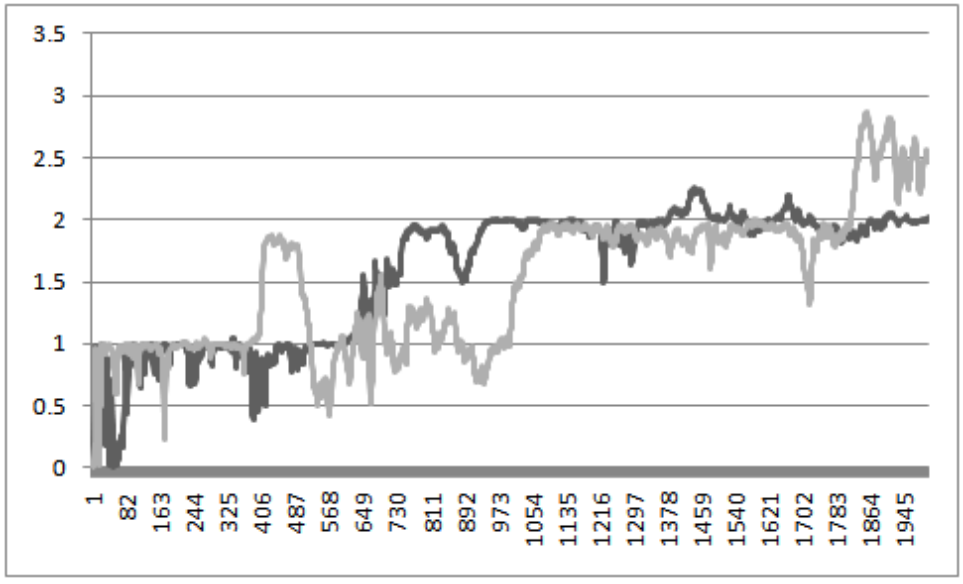


FIGURE A.40. Experiment: 23

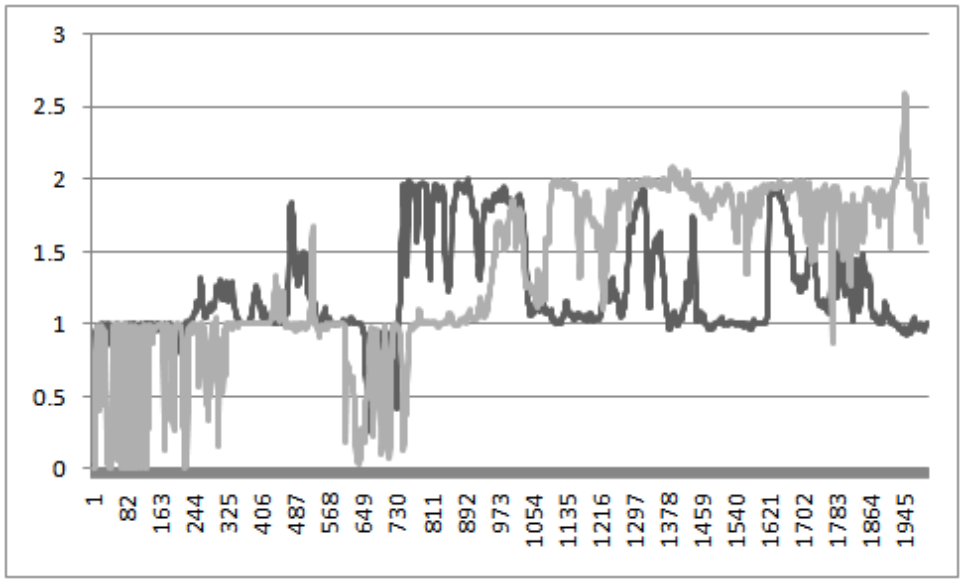


FIGURE A.41. Experiment: 5

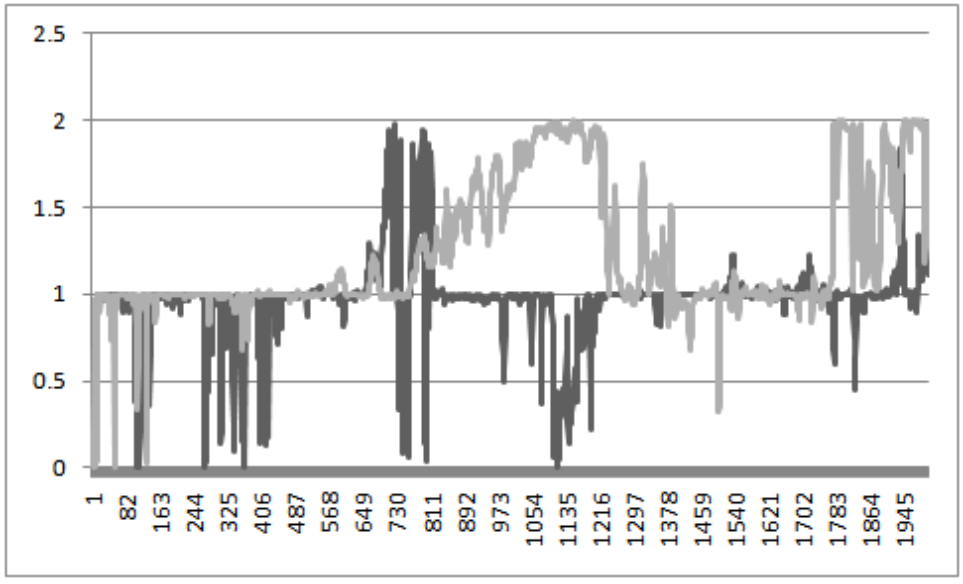


FIGURE A.42. Experiment: 1

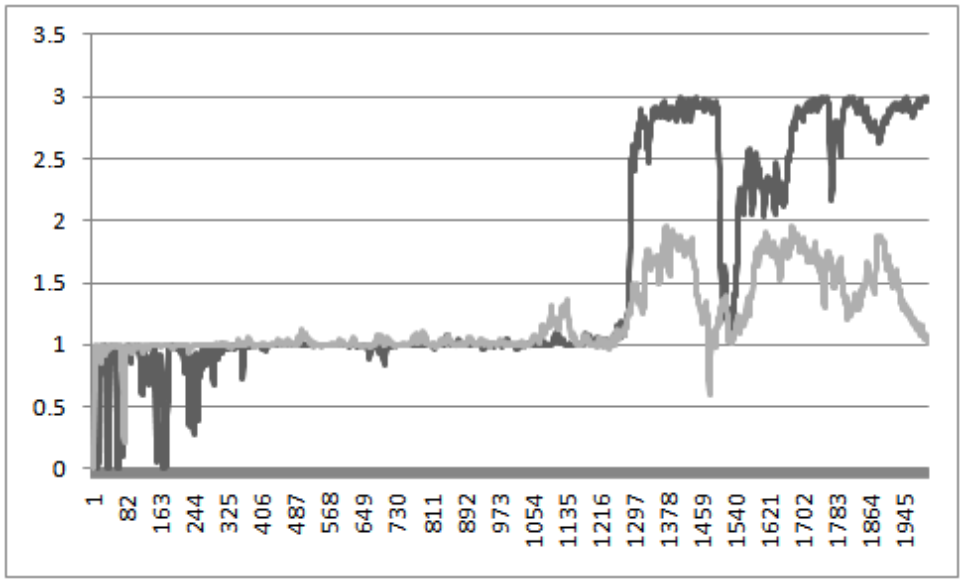


FIGURE A.43. Experiment: 15

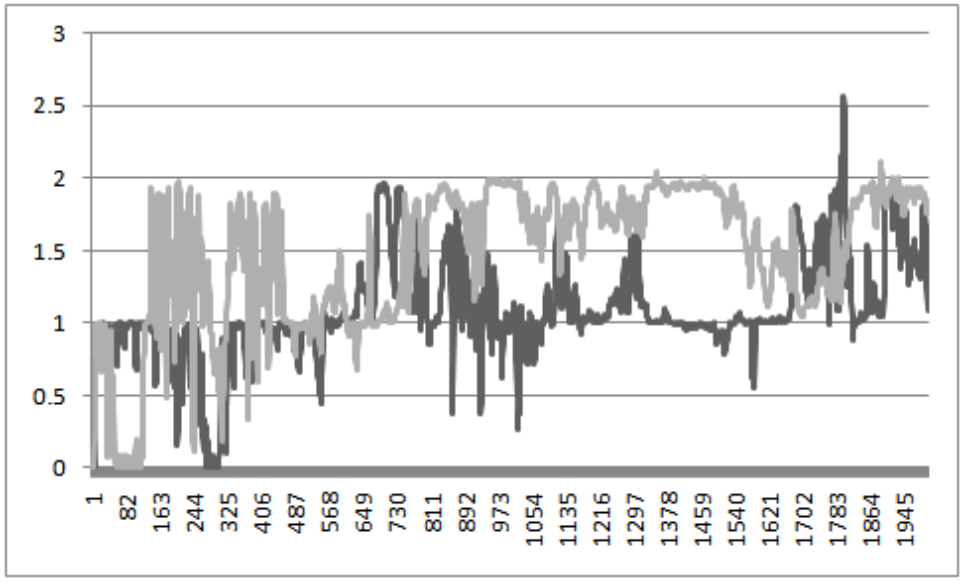


FIGURE A.44. Experiment: 9

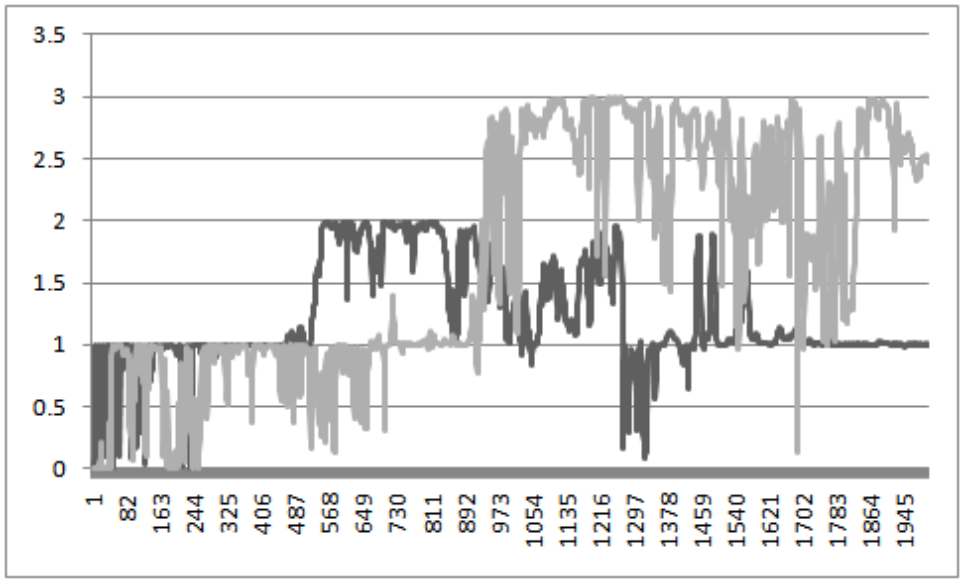


FIGURE A.45. Experiment: 21

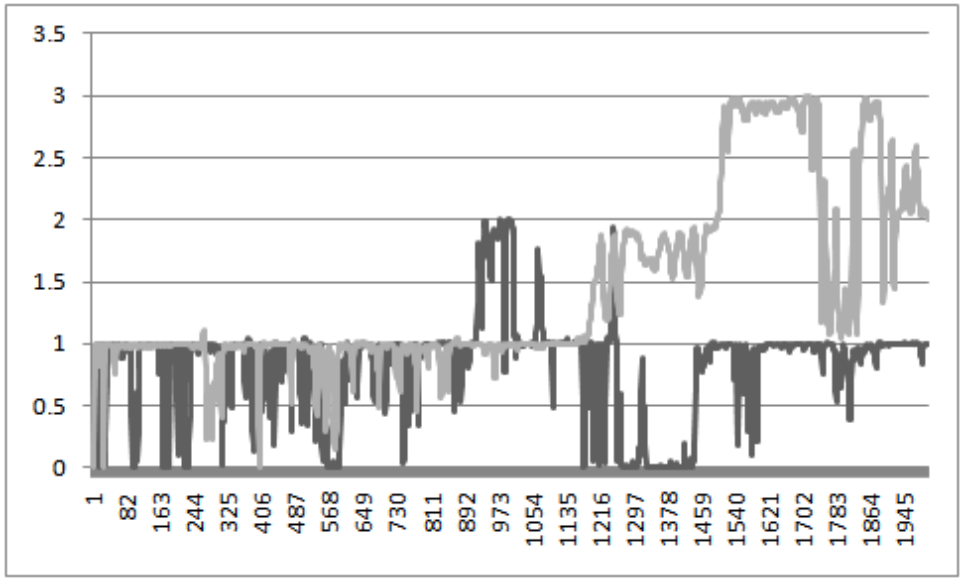


FIGURE A.46. Experiment: 17

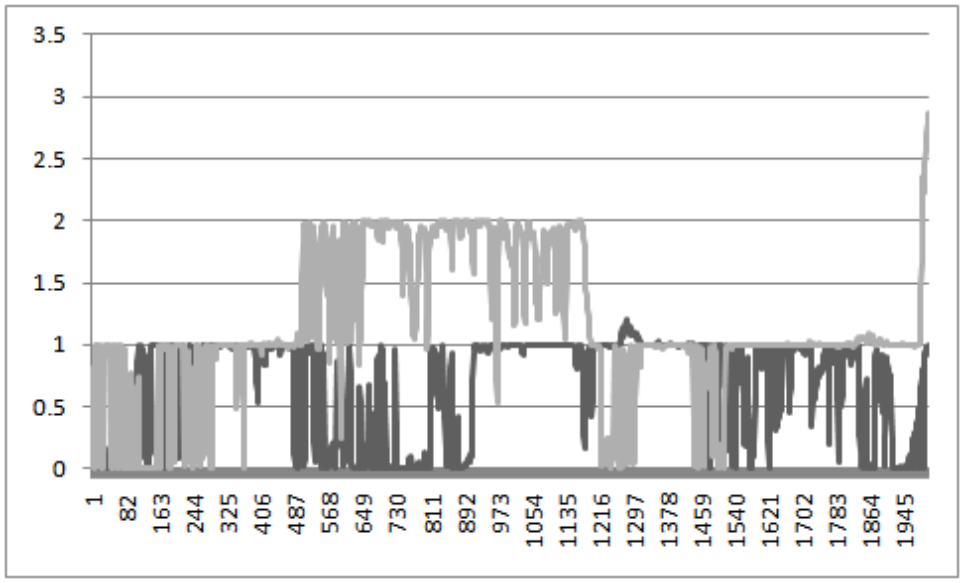


FIGURE A.47. Experiment: 13

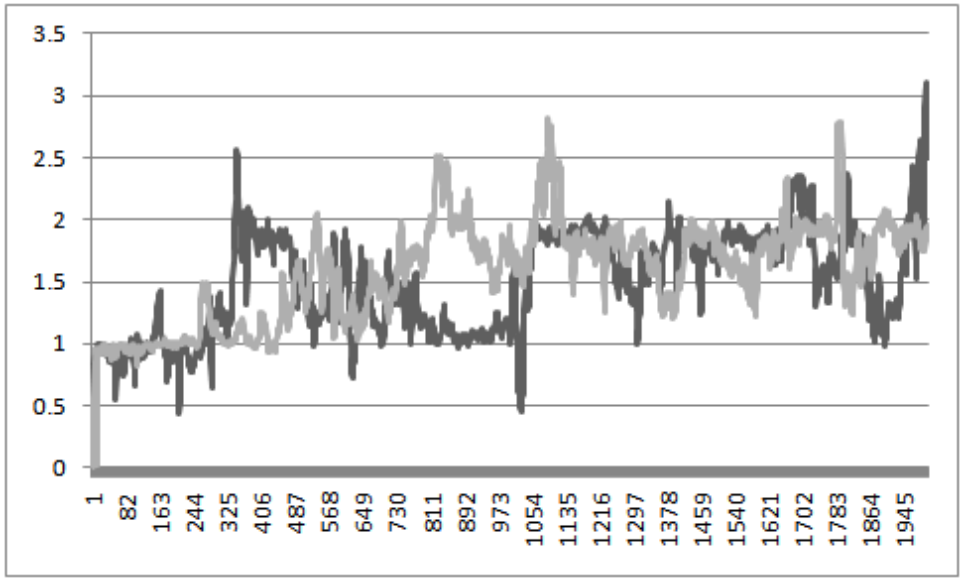


FIGURE A.48. Experiment: 25

APPENDIX B

Model manual

B.1. Requirements

To run the program java1.6.0 or newer is required, as well as the file ToM.zip which can be requested by sending an e-mail to the author of this thesis or downloaded on `http://tom.lisepijl.nl`.

B.2. Running the simulation

To prepare the simulation, extract the file ToM.zip first. The folder ToM\Experiments contains an experiment setup called 'experiment.props'. This file contains the experiment parameters. Without a parameter file, the program will not run. Now, open a command line or terminal and change directory to the folder 'ToM'.

Then, enter the following command 'java -jar ToM.jar -Xms800m' and press enter. The simulation then starts. It prints the parameter values and reports that the agent set is initialized. If debug information is requested the program starts to print information on the evolved agents. If not, nothing will show on the console. When the simulation has ended, the command prompt is returned.

The output from the simulation is printed to the data folder in the 'ToM'-folder.

B.3. Parameter files

Every file in the 'Experiments'-folder will be treated as a parameter file regardless of name and contents. Errors in the parameter file will cause the program to abort. It is possible to place multiple parameter files in the experiment folder. Adding more parameter files when the simulation is already running is pointless, since those parameter files will not be read.

In Java, it is good practice to use the extension .props for parameter files. However, you can name these files any way you like.

An example of the contents of a parameter file is presented in table B.1.

B.3.1. debug. Setting this value to TRUE results in more detailed output. This is only useful for debugging purposes.

B.3.2. name. The name of the experiment can be set. It should contain at least one character.

debug=false
name=Exp
agentcnt=350
generations=2000
interactioncnt=10
pctCSAgents=1
pctFOAgents=0
pctSOAgents=0
pctTOAgents=0
randomAgents=false
repetitions=1
RWS=true
LinkedGenes=false
mutationchance=0.001
survivors=0
reporter=true

TABLE B.1. Contents of a parameter file

B.3.3. agentcnt. This parameter determines the number of agents in each generation. Values higher than 500 are not recommended, since it requires a large working memory and may result in memory errors.

B.3.4. generations. This parameter determines for how many generations the experiment will run.

B.3.5. interactioncnt. This parameter determines how often an agent on average will attack each opponent.

B.3.6. pctZOAgents, pctFOAgents, pctSOAgents, pctTOAgents. Sets the percentage of respectively zero-order agents, first-order agents, second-order agents and third-order agents at the start of the experiment. This numbers must add up to one; otherwise the program will exit with an error.

B.3.7. randomAgents. If this value is set to TRUE, agents will start with a random database.

B.3.8. repetitions. Sets the number of times the experiment will be run with the settings in the parameter file.

B.3.9. RWS. If this is set to TRUE, agents are selected using the roulette-wheel selection method. If this value is FALSE, agents are randomly selected.

B.3.10. linkedGenes. If this value is set to TRUE, agents reproduce using the linked genes method. If this value is set to FALSE, agents reproduce using recombination.

B.3.11. mutationChance. This parameter determines the mutation probability.

B.3.12. survivors. This parameter sets the percentage of agents selected using the reproduction method set in the parameter 'linkedGenes' that survives from one generation to the other.

B.4. Possible problems

B.4.1. Java reports a memory issue. The command 'java -jar ToM.jar -Xms800m' tells Java to reserve 800 MB of memory for the program. However, if a simulation has many agents or runs for a very long time, this may not be enough. Change '800m' to a larger size.

B.4.2. Output directory already exists. To prevent overwriting results of earlier experiments, the program requires that each experiment has a different name. An experiment name consists of two parts: a name given by the user in the parameter file and a short summary of parameters. An example of a name is mentioned below.

`testAC350IN10GE1000MC0.01LGfalseRWStrueRNDfalseSU0.0`

In this experiment name, 'test' is the name of the experiment given by the user. The last part is a summary of the experiment setup. Changing the name of the experiment in the parameter file will solve this problem.

APPENDIX C

Experiment data

Running the model results in several data files. The contents of the data files will be discussed in this chapter.

C.1. The folders RdbRun0, RdbRun1, and so on

In these folders the rule databases of agents are stored. This allows us to let agents from different experiments, repetitions interact. During the simulation the rule databases of the five best performing agents of every hundredth generation.

C.2. The files avgOrder1.csv, avgOrder2.csv, and so on

These files contains the average maximum order of the agents' rule databases for each generation and the average rule database size for each repetition. Each row represents a generation. The first column contains the average maximum order of all the agents' rule databases. The second column contains the average maximum order of the 10% best performing agents. The third column contains the average rule database size.

C.3. The files dominanceValues1.csv, dominanceValues2.csv, and so on

These files contain the dominance values of all agents for each generation. Each row represents a generation. The dominance values are sorted from large to small.

C.4. The files evolvedRules1.txt, evolvedRules2.txt, and so on

These files contain the rule databases of all agents at the end of a simulation. The rules that were never used are not included. It also contains information on the size of the agents' rule databases. The first number after the rule represents the number of times the rule is used. The second number represents the proportion of the interactions won/not won or not lost/lost using that particular rule. The last number refers to the order of the rule.

C.5. The file experimentData.csv

This file summarizes the parameters of the simulation.

C.6. The files orderSpread1.csv, orderSpread2.csv, and so on

These files contain the average rule database composition. Every row represents a generation. The first column represents the proportion of zero-order rules, the second column the proportion of first-order rules, the third column the proportion of second-order rules, and so on. The tenth column represent the proportion of rules that was not assigned an order.

C.7. The files preference0.csv, preference1.csv, and so on

These files contain the agents preference. This data was not used in the analysis of the results. It only reflects an innate property that may never be used. The preference of an agent is only used when an agent does not have a zero-order rule to select its action. This does not happen often.

C.8. The files results0.txt, results1.txt, and so on

These files contain the rule databases of the ten best performing agents for each generation.

Bibliography

Baron-Cohen, S. (1999). Evolution of a theory of mind? In *The Descent of Mind: Psychological Perspectives on Hominid Evolution*. Oxford University Press, Oxford.

Brüne, M. and Brüne-Cohrs, U. (2006). Theory of mind - evolution, ontogeny, brain mechanisms and psychopathology. *Neuroscience and Biobehavioral Reviews*, 30:437–455.

Burkart, J. M. and Heschl, A. (2007). Understanding visual access in common marmosets, *Callithrix jacchus*: perspective taking or behaviour reading? *Animal Behaviour*, 73:457–469.

Byrne, R. and Whiten, A. (1988). *Machiavellian Intelligence*. Oxford University Press, Oxford.

Call, J. and Tomasello, M. (2008). Does the chimpanzee have a theory of mind? 30 years later. *Trends in Cognitive Sciences*, 12(5):187–192.

Caspari, R. and Sang-Hee, L. (2006). Is human longevity a consequence of cultural change or modern biology. *American Journal of Physical Anthropology*, 129:512–517.

Caspari, R. & Lee, S. H. (2004). Older age becomes common late in human evolution. *Proc. Natl Acad. Sci. USA*, 101:1089510900.

Chase, I. D., Bartalomeo, C., and Dugatkin, L. A. (1994). Aggressive interactions and inter-contest interval: how long do winners keep winning? *Animal Behaviour*, 48:393–400.

Clayton, N. S., Dally, J. M., and Emery, N. J. (2007). Social cognition by food-caching corvids. the western scrub-jay as a natural psychologist. *Philosophical Transactions of Royal Society B*, 362:507–522.

Cruz, J. and Gordon, R. M. (2002). Simulation theory. In Nadel, L., editor, *Encyclopedia of Cognitive Science*. John Wiley and Sons Ltd, US.

de Jong, K. A. (2008). *Evolutionary Computation: A Unified Approach*. The MIT Press, Cambridge (MA).

de Waal, F. (1982). *Chimpanzee Politics: Power and Sex among Apes*. The John Hopkins University Press, London.

Dunbar, R. (1992). Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 20:469–493.

Dunbar, R. (1996). *Grooming, Gossip and the Evolution of Language*. Faber and Faber, London.

Epstein, J. M. (2006). *Generative Social Science*, chapter 1, pages 4–45. Princeton University Press, New Jersey.

Erdal, D. and Whiten, A. (1996). Egalitarianism and Machiavellian intelligence in human evolution. In Mellars, P. and Gibson, K., editors, *Modelling the Early Human Mind*, chapter 12. McDonald Institute Monographs, Cambridge.

Flobbe, L., Verbrugge, R., Hendriks, P., and Kramer, I. (2008). Children's application of theory of mind in reasoning and language. *Journal of Logic Language and Information*, 17:417–442.

Gergely, G., Násady, Z., Csibra, G., and Bíró, S. (1995). Taking the intentional stance at 12 months of age. *Cognition*, 56:165–193.

Grefenstette, J. J. (1992). The evolution of strategies for multiagent environments. *Adaptive Behavior*, 1(1):65–90.

Hare, B., Call, J., Agnetta, B., and Tomasello, M. (2000). Chimpanzees know what conspecifics do and do not see. *Animal Behaviour*, 59:771 – 785.

Hare, B., Call, J., and Tomasello, M. (2001). Do chimpanzees know what conspecifics know? *Animal Behaviour*, 61:139 – 151.

Hemelrijk, C. K. (1999). An individual-orientated model of the emergence of despotic and egalitarian societies. *Proc. Roy. Soc. Lond. B*, 266:361–369.

Hemelrijk, C. K., Wantia, J., and Dätwyler, M. (2003). Female co-dominance in a virtual world: Ecological, cognitive, social and sexual causes. *Behaviour*, 140(10):1247–1273.

Humphrey, N. (1976). *Growing points in ethology*, chapter 9, pages 303–317. Cambridge University Press, Cambridge, UK.

Kinderman, P., Dunbar, R., and Bentall, R. P. (1998). Theory-of-mind deficits and causal attributions. *British Journal of Psychology*, 89:191–204.

Liddle, B. and Nettle, D. (2006). Higher-order theory of mind and social competence in school-age children. *Journal of Cultural and Evolutionary Psychology*, 4(3-4):231–246.

Mant, C. M. and Perner, J. (1988). The child's understanding of commitment. *Developmental Psychology*, 24(3):343–351.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge (MA).

Moll, H. and Tomasello, M. (2007). Cooperation and human cognition: the Vygotskian intelligence hypothesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1480):639–648.

Nolfi, S. and Floreano, D. (1999). Learning and evolution. *Autonomous Robots*, 7(1):89–113.

Penn, D. C. and Povinelli, D. J. (2007). On the lack of evidence that non-human animals possess anything remotely resembling a 'theory of mind'. *Philosophical Transactions of the Royal Society B*, 362:731–744.

Perner, J. and Wimmer, H. (1985). "John thinks that Mary thinks that..." attribution of second-order beliefs by 5- to 10-year-old children. *Journal of Experimental Child Psychology*, 39:437–471.

Premack, D. G. and Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 1:515–526.

Reaux, J. E., Theall, L. A., and Povinelli, D. J. (1999). A longitudinal investigation of chimpanzees' understanding of visual perception. *Child Development*, 70(2):275–290.

Van der Vaart, E. and Verbrugge, R. (2008). Agent-based models for animal cognition: a proposal and prototype. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1145–1152, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Verbrugge, R. (2009). Logic and social cognition. *Journal of Philosophical Logic*, 38(6):649–680.

Wellman, H. M., Cross, D., and Watson, J. (2001). Meta-analysis of theory-of-mind development: The truth about false beliefs. *Child Development*, 72:655–684.

Whiten, A. (2002). Theory of mind. In Nadel, L., editor, *Encyclopedia of Cognitive Science*. John Wiley and Sons Ltd, US.

Woodward, A. L. (1998). Infants selectively encode the goal object of an actor's reach. *Cognition*, 69:1–34.