metalogue

METALOGUE: Deliverable D2.1

Model of Repeated Prisoner's Dilemma

The METALOGUE Consortium

September 2014

Version N°: [version 1]

Main Authors: Christopher Stevens, Niels Taatgen, Fokie Cnosson

SEVENTH FRAMEWORK PROGRAMME

Project funded by the European Community under the Seventh Framework Programme for Research and Technological Development

Project ref. no.	ICT – 611073
Project title	METALOGUE – Multiperspective Multimodal Dialogue: dialogue system with metacognitive abilities

Document status	**[Draft, v1.0, v2.0, Final version]**
Contractual date of delivery	_30_ July_ 2014
Actual date of delivery	_September 2014
Document number	D2.1
Deliverable title	Model of Repeated Prisoner's Dilemma
Dissemination level	Public
Type	Report
Number of pages	17
WP contributing to the deliverable	WP 2
WP / Task responsible	University of Groningen
Contributing partners	
Reviewers	Dimitris Spiliotopoulos (UoP)
Author(s)	Christopher Stevens, Niels Taatgen, Fokie Cnosson
EC Project Officer	Pierre-Paul Sondag
Keywords	Prisoner's Dilemma, Theory of Mind, Metacognition

Table of Contents

1. Executive Summary

Metacognition is valuable for negotiation. With it, negotiators can anticipate objections from others and craft agreements more likely to be accepted. In this deliverable, we show ACT-R agents capable of metacognitive reasoning about opponents in the repeated prisoner's dilemma. The agents presented here both use instance-based learning to predict their opponents' behaviour. Two types of metacognitive agent are presented. The opponent-perspective agent is capable of taking an opponent's perspective to anticipate the opponent's future actions and respond accordingly. The modeller agent predicts the opponent's next move based on the previous move of the agent and the opponent. The opponent-perspective agent consistently outscores non-metacognitive agents and strategies. However it does this mainly by exploiting opponents. The modeller agent, by contrast, is much more successful at encouraging cooperation. These simple models provide insights regarding modelling of metacognition in more complex tasks.

2. Introduction

One of the principal aims of work package 2 is to create a cognitive agent capable of metacognition in a negotiation scenario. This includes predicting the behaviour of a partner, reasoning about their goals, and changing strategies when necessary. This will allow the creation of realistic negotiation partners for trainees to practice against. The present deliverable contains the first step toward that goal. In this deliverable, we present cognitive agents that can predict the behaviour of an opponent in the repeated prisoner's dilemma task.

Negotiation scenarios are often mixed-motive. That is, players must weigh their own self-interest against the interests of other parties (Kelley, 1967). One could ignore or minimize the interests of others, but this is potentially costly because it may make them less likely to cooperate in the future. This basic tension is illustrated well in the prisoner's dilemma. The prisoner's dilemma is a 2 x 2 game in which players must choose to cooperate with their opponent (move B) or to defect (move A). This results in one of four possible outcomes. The following is a typical payoff matrix for the prisoner's dilemma game.

	Player 2		
		Cooperate	Defect
Player 1	Cooperate	1,1	-10, 10
	Defect	10, -10	-1, -1

If both players consistently choose cooperate, then they both will enjoy a positive payoff. However, cooperation is risky, because both players have a temptation to defect. If the opponent defects when a player cooperates, the cooperating player will lose a large number of points. Defection also carries risks. When there is more than one round, opponents may retaliate by defecting in later rounds. The optimal strategy is not obvious and depends on the opponent. Therefore, reasoning about an opponent's goals and predicting their future behaviour may provide an advantage.

We chose ACT-R (Anderson et al., 2004) as the modelling architecture because it will allow us to create psychologically plausible agents. Most of the existing prisoner's dilemma agents were developed for optimal performance, not necessarily realistic behaviour (Axelrod, 1980; Nowak & Sigmund 1993). ACT-R is a cognitive architecture designed to closely replicate human behaviour. Furthermore, it has been successfully used in the past to create believable computer opponents (e.g. the game of set; Taatgen et al., 2003).

3. Description of the Agents

To evaluate the metacognitive agents presented here, we used a previous model developed by Lebiere, Wallach, & West (2000) as a baseline. This model is not metacognitive because it bases its decisions only on the payoffs of its previous moves. It does not attempt to learn or reason about its opponent. We present two types of metacognitive model: opponent-perspective and modeller. The opponent-perspective agent assumes that the opponent will select the move with the highest expected value. The modeller agent tracks the frequencies of the opponent's previous moves and attempts to predict the opponent's next move probabilistically.

3.1 Declarative Memory in ACT-R

The basic unit of declarative memory in ACT-R is the chunk. A chunk is a schema with one or more slots that holds information or references to other chunks. For example, here is how ACT-R might represent knowledge about the concept "cat."

Cat
  Isa animal
  Legs 4
  Has-tail yes

The *isa* slot is a special slot that indicates the chunk-type. A chunk-type is simply a way of classifying different types of declarative knowledge. So in this model, there may be chunk-types for insects, cars, buildings and so on.

In the following simulations, we use an instance-based learning approach to allow the agents to adapt to their opponents. The outcome of each trial is stored as a chunk in ACT-R's declarative memory. The basic unit of declarative memory in ACT-R is the chunk. If no chunk currently exists that matches the current trial's outcome, then a new chunk will be created. If the outcome matches an existing chunk, the existing chunk will become more active. The agents attempt to predict outcomes or opponent behaviours by retrieving a chunk from memory that matches the current situation.

The likelihood of retrieval of a chunk is determined by its activation level. Chunks with higher activation levels are more likely to be retrieved. The activation level of a chunk (i) is derived from the following equation.

$$A_i = \left(\ln\left(\frac{n}{1-d}\right) - d * \ln(L)\right) + N\left(0, \frac{\pi * s}{\sqrt{3}}\right)$$

In this equation, n is the number of presentations of chunk i. L represents the amount of time that has passed since the creation of the chunk and d is the rate of activation decay. The rightmost term of the equation represents noise added to the activation level. This equation dictates that the activation level of a chunk is determined by how frequently it is retrieved. In the standard ACT-R activation equation, activation also depends on how recently the chunk has been retrieved. However, here we use a simplified equation that assumes all references to the chunk are uniformly distributed throughout the life of the chunk.

3.2 The self-payoff agent

This agent is a replication of the model reported in Lebiere, Wallach, & West (2000). It does not attempt to learn or predict the opponent's behaviour. Instead, it simply looks at the most likely payoff of each of its possible moves. Then it selects the move associated with the highest payoff.

The self-payoff agent remembers the previous rounds of the game using four declarative memory chunks. Each chunk represents one of the four possible outcomes of the game.

```
A1-A2 isa outcome move1 A move2 A payoff1 -1 payoff2 -1
A1-B2 isa outcome move1 A move2 B payoff1 10 payoff2 -10
B1-A2 isa outcome move1 B move2 A payoff1 -10 payoff2 10
B1-B2 isa outcome move1 B move2 B payoff1 1 payoff2 1
```

The isa slot indicates that these chunks are of type "outcome" The move1 and move2 slots represent the moves chosen by players 1 and 2 respectively. The payoff slots contain the number of points both players received. In every round, the model creates a new outcome chunk in the goal buffer. When the model selects its move, it records it in the move1 slot. At the end of the trial, the opponent's move and the resulting payoffs are also recorded in this chunk. This chunk is then removed from the goal buffer and merged with an existing chunk in declarative memory, reinforcing it and increasing its activation level.

The self-payoff model uses the relative activation levels of these chunks to determine the most likely outcome of a given move. It does this using a set of four production rules. The first two productions retrieve two outcome chunks, one in which move1 is A and one in which move1 is B. The remaining productions then select move A if payoff A is higher or move B is payoff B is higher. Pseudocode descriptions of the productions are included below.

```
IF the goal is to play the prisoner's dilemma
AND I have not selected a move
AND I have not retrieved any previous outcomes
THEN retrieve an outcome chunk where I played move A

IF the goal is to play the prisoner's dilemma
AND I have not selected a move
AND I have retrieved a previous outcome where I played move A
```

```
THEN retrieve an outcome where I played B

IF the goal is to play the prisoner's dilemma
AND I have not selected a move
AND I have retrieved previous outcomes where I played the moves A and B
AND The payoff for move A is higher than move B
THEN select move A
```

IF the goal is to play the prisoner's dilemma
AND I have not selected a move
AND I have retrieved previous outcomes where I played the moves A and B
AND The payoff for move B is higher than move A
THEN select move B

3.3 The opponent-perspective agents

The opponent-perspective agents attempt to predict the opponent's move by taking the opponent's perspective. The opponent-perspective agents use the same declarative memory chunks as the self-payoff agent. But instead of determining their own most likely outcomes, they predict the most likely outcomes for their opponent. They then predict that their opponent will select the move with the highest payoff. Based on this prediction, they select an appropriate response.

Due to the nature of the prisoner's dilemma, there is not an optimal countermove for each possible opponent move. Regardless of an opponent's move, defecting will always yield the highest immediate score. Thus there are several possible strategies. One strategy (cooperative) is to always cooperate when the opponent is going to cooperate. This would reinforce the opponent's B1B2 chunk and make it more likely that the opponent will cooperate in the future. An alternative strategy (aggressive) is to cooperate half the time and to defect the other half. This allows the player to take advantage of the opponent's cooperation, but will likely make the opponent less likely to cooperate in the future. Regardless, it makes sense to always defect when one's opponent is going to defect. Otherwise, the player will lose a large number of points. Two different opponent-perspective agents were developed: one using the cooperative strategy and one using the aggressive strategy.

The perspective models use the same declarative chunks and production rules described above. However, their production rules instead compare the opponent's payoffs rather than their own payoffs. Based on this comparison, the models will predict the opponent's next move. Then, one of the following three productions will be selected (depending on the predicted move and the agent's strategy):

IF the goal is to play the prisoner's dilemma
AND I haven't selected my move
AND I predict that my opponent is going to defect
THEN select defect

IF the goal is to play the prisoner's dilemma
AND I haven't selected my move
AND I predict that my opponent is going to cooperate
THEN select defect

IF the goal is to play the prisoner's dilemma
AND I haven't selected my move
AND I predict that my opponent is going to cooperate
THEN select cooperate

3.4 The modeller agent

The modeller agent represents a different form of metacognition than the opponent-perspective agents. The modeller attempts to build a mental model to predict the opponent's most likely next move based upon their previous moves. Unlike the opponent-perspective, the modeller does not make any assumptions about the specific strategy used by the opponent. Instead, it tracks how the opponent responds to each of the four possible outcomes in the game (double-defect, defect-cooperate, cooperate-defect, and double-cooperate). It then predicts that the opponent will make the same move when the outcome appears again.

The memory structure of the modeller agent is different from that of the self-payoff and opponent-perspective agents. The model does not start with any predefined chunks, but after every trial, it will create a new chunk like the following example:

SEQUENCE0 isa sequence move1 A move2 B next-move A

Move1 represents the player's move and move2 represents the opponent's move. Next-move represents the opponent's move on the following round. This chunk represents an instance in which the player defected and the opponent cooperated; in the next round, the opponent responded with a defection.

Before deciding on a move, the modeller agent will retrieve a previous instance that matches the current situation. For example, after a double-cooperation round, the modeller will attempt to retrieve a chunk in which both move1 and move2 are B. Based on this retrieved chunk, it will predict the opponent's next move. Like the cooperative opponent-perspective agent, the modeller will select the same move that it thinks its opponent is going to select. If the modeller fails to retrieve a similar instance, it will select a move randomly.

4. Simulation Results

The performance of the self-payoff and metacognitive agents were tested against the self-payoff agent to determine if metacognition provides an advantage. In addition, the models were tested against two normative prisoner's dilemma strategies: tit-for-tat (TFT; Axelrod, 1980) and win-stay-lose-shift (WSLS) (Wedekind & Milinski, 1996). There are several variations on the tit-for-tat strategy, but all of the variations will tend to copy the previous move of their opponent. We used a strict TFT strategy that always copied the previous move of the opponent. The WSLS strategy, by contrast, continues to make the same move until it loses points, then it will change moves. These two fixed strategies were included to test the model's flexibility in dealing with different types of opponents.

Twelve simulations were run. The self-payoff, cooperative opponent-perspective, aggressive opponent-perspective, and modeller agents were all played against the self-payoff, TFT, and WSLS agents. Each simulation consisted of 100 runs of 100 trials. Results were averaged over the runs. A summary of their performance can be found in Table 1.

Table 1. Average scores and win percentages

Agent 1	Agent 2	Agent 1 mean score	Agent 2 mean score	Joint Score
Self-payoff	Self-payoff	-52	-54.8	-106.8
Self-payoff	TFT	-44.71	-59.71	-104.42
Self-payoff	WSLS	223.44	-268.16	-44.72
Opponent-perspective (cooperative)	Self-payoff	-55.02	-45.82	-100.84
Opponent-perspective (cooperative)	TFT	-42.17	-59.57	-101.74
Opponent-perspective (cooperative)	WSLS	337.37	-410.03	-72.66
Opponent-perspective (aggressive)	Self-payoff	22.29	-166.31	-144.02
Opponent-perspective (aggressive)	TFT	-66.09	-85.49	-151.58
Opponent-perspective (aggressive)	WSLS	419.7	-511.9	-92.2
Modeler	Self-payoff	-4.58	-128.18	-132.76
Modeler	TFT	23.44	15.44	38.88
Modeler	WSLS	104.44	84.44	188.88

The self-payoff model performed somewhat poorly overall. Against itself and against TFT, it earned a low negative score. It earned a very high score against WSLS, but it only did so by exploiting WSLS's cooperation. Thus the average joint score was still negative.

The opponent-perspective agents have higher individual scores than the self-payoff model, but their joint outcomes are still poor. The cooperative agent still earned low negative scores against the self-payoff and TFT agents while exploiting the WSLS agent. The aggressive agent performed similarly, except that it also learned to exploit the self-payoff agent. However, both of these agents failed to discover opportunities for cooperation with the TFT and WSLS agents. As a result, the joint outcomes that these agents produce are still quite low (far from the ideal 200).

The modeller agent, by contrast, does succeed in both achieving favourable outcomes for itself and learning to cooperate with other agents when possible. In terms of individual outcomes, the modeller achieves a higher score against the self-payoff agent than both the cooperative opponent-perspective agent and the self-payoff agent itself. Further, it earns a higher score than all of the other agents when playing against TFT. In terms of cooperation, the modeller achieves higher joint outcomes when playing against the TFT and WSLS agents than any of the other agents, demonstrating that the agent can quickly learn when cooperation with an opponent is possible. In fact, the modeller is the only agent that achieves a positive joint outcome against these agents. Against the self-payoff agent, the modeller agent does not achieve cooperation, but it still manages to obtain a near-zero score for itself against an aggressive opponent. Thus the modeller proves that it can perform well against both cooperative and aggressive strategies.

Overall, the modeller agent is the best of the metacognitive agents because of its ability to flexibly adapt to different opponents. Against cooperating agents, it will quickly learn to cooperate and achieve high positive scores. Against aggressive agents, it will learn to play defensively and defect most of the time. The opponent-perspective agents can sometimes achieve higher individual scores, but they do so by playing relatively aggressively and exploiting opponents. This aggressive strategy does not work well against agents like TFT that swiftly punish defection. The opponent-perspective agents are probably aggressive because they use the same decision process as the self-payoff model, which looks only at the possible payoffs for the next trial. Because the short-term payoffs are skewed in favour of defection, the opponent-perspective agents predict that their opponents will have a strong tendency to defect. This does not necessarily mean that taking the opponent's perspective is not helpful in the prisoner's dilemma. But it may be necessary for the agent to have access to a larger set of strategies so that it can find one that resembles the opponent's behaviour.

The work shown here demonstrates that metacognitive reasoning about an opponent can improve outcomes both for oneself and for one's opponent in the prisoner's dilemma. By learning an opponent's strategy, an agent can determine if it is safe to cooperate, or if it is better to defect. This increases the probability that the agent and its partner will discover a stable, mutually beneficial outcome. Metacognition also helps an agent detect and defend itself against cheaters. We expect that these benefits extend not only to other simple games but also to more complex negotiation scenarios. It remains for further work to discover how metacognitive reasoning can be best employed to achieve success in these other tasks.

These agents provide a useful starting point for modelling efforts in simple, two-person negotiation games such as the game of nines. The logic used here for comparing non-metacognitive and metacognitive agents could be extended to agents in these other tasks. There are several ways in which metacognition may improve performance of agents in the game of nines. For example, it may be useful to assume the opponent's perspective and consider what deals the agent would accept in their place. This may narrow down the search space of possible offers. Similarly, the agent could make inferences about the opponent's beliefs about the agent's goals. This could allow the agent to determine whether it can convince an opponent that it needs a larger share than it actually does. In addition to serving as realistic training partners, these cognitive agents may give insights about the best metacognitive strategies to use in negotiation scenarios.

5. Conclusions

The work presented here indicates that metacognitive reasoning about an opponent's behaviour can provide an advantage in the repeated prisoner's dilemma. The ability to predict an opponent's next move helps to determine when it is safe to cooperate and when one should defect. This suggests that performance even in simple games may benefit from developing a theory of mind about one's opponent. This is an aspect of simple games that is relatively under-explored, especially in ACT-R.

The prisoner's dilemma captures a fundamental quality of both the parliament and call-center scenarios. In both cases, people must deal with deal with other actors whom they are not sure they can trust. The model outlines a simple strategy that people might use to deal with this issue. The approach described here also serves as a proof-of-concept for the model of the Game of Nines developed in Deliverable 2.2. Both models attempt to address a similar question: Am I dealing with an opponent that will cooperate with me? Both models also attempt to predict when the opponent is going to cooperate and when he or she is going to defect. The models then respond in kind. The success of the prisoner's dilemma model shows that this general strategy is an effective way to deal with uncertainty about opponents in mixed-motive games. The prisoner's dilemma model itself will not be a component of the final METALOGUE system, but it is a necessary step in the development of the Game of Nines and Tragedy of the Commons models.

6. References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, *111*(4), 1036.

Axelrod, R. (1980). More Effective Choice in the Prisoner's Dilemma. *Journal of Conflict Resolution*, *24*(3), 379–403.

de Weerd, H., Verbrugge, R., & Verheij, B. (2013). Higher-order theory of mind in negotiations under incomplete information. In *PRIMA 2013: Principles and Practice of Multi-Agent Systems* (pp 101-16) Springer: Berlin Heidelberg.

Gallese, V., & Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, *2*(12), 493–501.

Fisher, R., Ury, W. L., & Patton, B. (2011). *Getting to yes: Negotiating agreement without giving in*. Penguin.

Kelley, H., Beckman, L., & Fischer, C. (1967). Negotiating the division of a reward under incomplete information. *Journal of Experimental Social Psychology*, *361-98*.

Lebiere, C., Wallach, D., & West, R. (2000). A memory-based account of the prisoner's dilemma and other 2x2 games. *Proceedings of International Conference on Cognitive Modeling*. 185-93.

Nowak, M., & Sigmund, K. (1993). A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, *364*, 56–58.

Rapoport, A., Guyer, M.J., & Gordon, D. G., (1976). *The 2x2 Game*. Ann Arbor, MI: University of Michigan Press

Wedekind, C., & Milinski, M. (1996). Human cooperation in the simultaneous and the alternating Prisoner's Dilemma: Pavlov versus Generous Tit-for-Tat. *Proceedings of the National Academy of Sciences of the United States of America*, *93*(7), 2686–9.

7. Annexes

7.1 Annex 1. [Title]

7.2 Annex 2. [Title]