

Rational Dialog in Interactive Games

Maria Arinbjarnar

School of Computer Science, Reykjavik University
Kringlan 1, is-103, Reykjavik, Iceland

Abstract

In story driven computer games today there is a noticeable lack of intelligent Non-Player Characters (NPCs). The NPCs currently implemented rarely act in a rational way, although some have an emotional drive and a set of goals to pursue. Their actual interactions are usually scripted and/or very limited.

To tackle the problem of getting NPCs to act in a rational way I propose an engine that uses Multi-Agent Influence Diagrams (MAIDs) and game theory, a mathematical method of decision-making in competitive situations. This engine creates NPCs that are sufficiently autonomous to participate in a rational dialog, that is to decide on a rational sentence to speak based on their own knowledge.

The success of the solution is measured in terms of game theory. The time complexity for an NPC to calculate a rational sentence is linear with respect to the number of sentences that the NPC considers. With some standard optimizations these results could be improved further.

Introduction

There is a continuous call for games that are intellectually challenging, i.e. games that challenge the player to think and solve puzzles without the aid of a walkthrough. One way to tackle this problem in story driven games is to have the narration responsive to player interactions. An important element in an interactive narration is interactive Non-Player Characters (NPCs). NPCs that will respond intelligently to game events and to actions made by other NPCs or human players. I firmly believe that computer games that include rationally interactive NPCs will open up a completely new genre of games.

A *rationally interactive NPC* is an NPC that interacts with both human players and other NPCs in a rational way. Rationality is defined as the choice of actions which best satisfy a person's objectives. These objectives are desires that motivate the individual (Heap *et al.* 1992).

This new genre of games would contain narrations that are developed by both players and NPCs progress in the game world and by their interactions with each other. This would mean that the storyline would not be pre-authored but evolve from players and NPC interactions in the game.

Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The problem domain of creating rationally interactive NPCs is vast and thus I emphasize the creation of a rational dialog in the sense that an NPC decides on what to say in a dialog by finding a rational sentence to speak.

There are three main steps in creating a rational NPC. First it is necessary to create the past life of the NPC so that she has a plausible knowledge base and connections with other characters in the game.

Secondly each NPC needs a causally driven decision mechanism that contains the NPC's knowledge base characteristics and information that she gathers along the way.

The third and final step is to make the NPC capable of finding equilibrium as defined by game theory so that she can be said to maximize her payoff when talking to another NPC or a human player. These final two steps are essentially what makes the NPC autonomous because then she is able to make decisions on how to interact based on her own knowledge, perception and reasoning instead of being controlled by outside constraints.

The chosen game domain is a murder mystery adventure because it offers a well structured game world to build a knowledge base in. In order to create a past life I use the Dynamic Plot Generating Engine (DPGE) (Arinbjarnar 2006). The DPGE creates new murder mystery plots on demand using a Bayesian net. Bayesian nets are causally connected belief networks (Jensen 2001). Each constructed plot is consistent and coherent, meaning that a murderer has a credible motive and had the opportunity and means to commit a murder. The version I employ in this work has been extended with motives and relations between characters of the plot.

The algorithm that creates the plot is not relevant to this article as it has no affect on the NPCs. Only the resulting plot affects them, and not the way the plot was generated. The NPCs are not trying to solve the murder mystery as in a clue style board game. The setting is rather like an Agatha Christie narration where various characters will be talking together discussing various aspects of a recent murder and trying their best not to appear suspicious. The human player could be a detective trying to solve the case or he could play one of the suspects or the murderer. The human player would be able to discuss aspects of the mystery with the NPCs.

The goals of the NPCs is to be "in character", meaning that each NPC tries to act in a way that best represents their

character, an honest NPC will be less inclined to lie than a dishonest NPC.

Each NPC is given a knowledge base with respect to the initial configuration of the world and the NPC's role within the game. As an example, a suspect knows everything about herself but has for the most part only approximate knowledge of the crime and other characters in the plot. The murderer on the other hand has in addition to a complete knowledge of her own person also a fairly complete knowledge of the crime. So each NPC gets a blueprint in the form of a Bayesian net that is identical to the one that plot was generated from. The knowledge that he should have with respect to his role is instantiated. Moreover he is equipped with tools to be able to interact rationally with other NPCs or players. The NPCs are also equipped with a Bayesian net to estimate how other NPCs or players see the world. Finally the NPC has tools to update his knowledge base with information from other NPCs or players.

As the game progresses the NPCs will interact with each other or the human player. When an NPC wants to say something then she will:

- Generate possible sentences and possible replies.
- Calculate the optimal reply that she expects for each of her sentences.
- Calculate her set of optimal sentences given the set of optimal replies.
- Pick one sentence in the set of optimal sentences to speak.
- Update her knowledge base with respect to what she decided to say. If the opponent is a NPC then the opponent will also update it's knowledge base.

The method described above is developed using Multi Agent Influence Diagrams (MAIDs) (Koller & Milch 2003). MAIDs are extended Bayesian nets specifically designed to find Nash equilibrium for incomplete data, which is exactly the problem that the NPCs face.

The result is a rationally interactive NPC that is able to participate in a dialog, calculating an optimal sentence to speak based on her knowledge base and what has transpired previously in the dialog. The NPC is able to receive information from other NPCs and players, evaluate it and, if she finds it sound, add it to her knowledge base. Thus the plot in the game develops by means of interactions between the characters very much as it would in real life. The NPCs make no distinction between another NPC and a human player, so if a human player takes the part of a suspect or murderer defined by the plot engine then the NPCs will interact with the human player just as if the human player were another NPC.

The results are measured in terms of game theory and not by whether the dialogs appear natural to humans. Modules such as for natural language processing will need to be added to the engine in order to make the dialog appear reasonable to humans.

In the next section I describe some Related work. In section "Game Theory" I describe the use of game theory. In section "The Engine" the engine is described. In section "Knowledge" I describe the NPCs knowledge base and how

new knowledge gets added to it. In section "Generating Sentences and Replies" I describe how the NPCs generate possible sentences and replies from their knowledge base. In section "Finding Equilibrium" I describe how the NPCs calculate rationality. Followed by a section on "Results" and a section on "Conclusions" where I summarize the results. In section "Future Work" I discuss what needs to be added to the engine so that it could be implemented in NPCs in narrative games.

Related Work

There is a body of ongoing research that deals with autonomous NPCs using many different techniques such as decision trees, planning and neural networks. I review some of them next.

A recent one is Thespian (Si, Marsella, & Pynadath 2005) which emphasizes creating autonomous agents that are both responsive to the user and consistent with their own internal motivations and goals. Moreover the characters and the story line can be easily authored by people lacking programming skills.

The Virtual Story Teller (Theune *et al.* 2004; 2002) deploys a director agent and character agents. The director agent's job is to direct the character agents towards meaningful goals to enforce an emerging storyline, the character agents are very independent and their actions are controlled by their feelings and by the priority of their goals, both of which are continuously changing in light of actions and input from their surroundings. The director agent can forbid or accept an action chosen by a character but he cannot command the character, which essentially keeps the character actions coherent and sensible for that character. The characters are given individual characteristics that have a direct influence on their feelings, reactions and response to input from other characters and the environment. They are also given, primary and secondary goals, that are the driving force behind their actions. Thus a storyline emerges by the NPCs reactions to in-game events as they attempt to fulfill their goals. This has not been implemented in an actual game and it appears that the player would be hard pressed to be at the right place at the right time to interact intelligently with an NPC and participate in the storyline.

An interesting approach that maps existing theories of human behaviorism into computational models uses C Language Integrated Production System (CLIPS) to build a rule based system to manage agent emotions, drives and relationships with other agents (Chaplin & Rhalibi 2004). CLIPS provides a complete environment for the construction of rule and or object based expert systems. That system uses the Iterated Prisoners Dilemma (Heap *et al.* 1992) to give the agents a chance to defect and for other agents to show vengeance if they find out about the defection. The authors implement the Iterated Prisoners Dilemma through a set of norms. They apparently manage to connect a rule based system with emotional drive based agents so that the agents are able to sustain themselves and to coexist to some degree. The results put forth appear to be from limited scenarios where the expected behaviors have been isolated.

Another approach compares adjustable rule sets and Neural Nets to model human behavior (Tolk 2002). The proposers of this approach discover that even with adjustable rule sets, the rules still need to be situationally adequate, that is they need to be closely modeled for the actual scenario at hand and that Neural Networks will give a better performance. On the other hand it is not always clear what the Neural Nets learn, e.g. how they come to determine the expected reaction to stimuli, while the rule based systems are much more transparent.

An approach that vaguely resembles the decision mechanism of my prototype engine is from (Silverman *et al.* 2006). They use three sets of trees called short term Goals, Standards for behavior of self and others, and long term Preferences, or GSP trees. These trees are specific to each character motive modeled, e.g. they are specific for sex, culture and other character specific attributes. The trees have probabilities to determine the next action based on the previous action. For example, a Somalian civilian female that is in a state of “wanting to belong” is 70% likely to *obey orders* and 30% likely to try to *protect her friends*. These trees are clearly very specific.

The way that my engine differs from related work is that I combine game theory and Bayesian networks to create an autonomous NPC that uses his own knowledge base and his own deductive reasoning to participate in a rational dialog. The NPC evaluates input from outside in light of his own knowledge and with respect to his personal characteristics. The NPC is not guided in any way by outside rules, constraints or directions.

Game Theory

When the players are evaluating what to say they use game theory to model the problem of finding an optimal sentence to speak. The traditional representation of game theory is strictly mathematical, exact and explicit (Fudenberg & Tirole 1991). The resulting high abstraction does not have a very transparent connection with real world scenarios. It is a non-trivial endeavor to model even the simplest of human interactions using game theory. J. C. Harsanyi discusses this point in great detail and in his own words (Harsanyi 1967) p:163:

In our own view it has been a major analytical deficiency of existing game theory that it has been almost completely restricted to complete information games, in spite of the fact that in many real-life economic, political, military, and other social situations the participants often lack full information about some important aspects of the “game” they are playing.

Let us first consider the dynamics of an incomplete information game, which is a game where some players lack full information of the game. We will assume that each player knows his own payoff function and state to keep the complication within reasonable bounds and we will only look at two player games. We will further assume that the players don’t know the opponents’ current payoff function but know all payoff functions that the opponent could be using. The latter is referred to as *exogenous data*, which means external

data that is not derived from the players status at any given time but is a factor in the equilibrium-calculations. Examples of exogenous data are strategy spaces, pay off functions, types, and prior distributions. This is considered common knowledge among the players (Fudenberg & Tirole 1991). The payoff function is affected by anything that could matter to the player such as goals, feelings, characteristics etc.

Harsanyi argues that the *type* of a player can represent the character of a player and his beliefs of other player’s strategies, including their payoff functions (Harsanyi 1995). Each player is considered to have a finite set of possible types $\Theta_i \in \Theta$ with an objective prior probability distribution p . The complete set of all possible types is exogenous data, common knowledge, but each players type θ_i is not exogenous data. This means that a player knows all the possible types that a player could be of but he does not know which type his opponent is currently of.

The *exogenous* data of the game is then the set of all possible types Θ and all possible payoff functions contingent to the types. Bayesian equilibrium then uses prior calculations commonly referred to as *ex ante* calculations. These *ex ante* calculations use Bayes rule to calculate the probability of each possible *type* $\theta \in \Theta$ that the opponent could be of (Fudenberg & Tirole 1991), hence the name *Bayesian* equilibrium.

It is important to realize that since Bayesian equilibrium, like Nash equilibrium (Nash 1951), is essentially a consistency check, players’ beliefs about others’ beliefs do not enter the definition – all that matters is each player’s own beliefs about the distribution of types and his opponents’ type-contingent strategies (Fudenberg & Tirole 1991). If this “type-centric” interpretation is not applied then the game will enter an infinite recursion. The two interpretations, type-centric and player-centric, are always equivalent from a game theoretic point of view (Harsanyi 1995).

In general terms then Bayesian equilibrium (Fudenberg & Tirole 1991) is found by first calculating the probability of each of the opponents types θ_{-i} . Then by calculating what strategy, contingent on his type, the opponent would play $s_{-i}(\theta_{-i})$ with respect to each of the strategies that the player would play, $s'_i(\theta_i)$.

$$(s'_i(\theta_i), s_{-i}(\theta_{-i})) =$$

$$(s_1(\theta_1), \dots, s_{i-1}(\theta_{i-1}), s'_i(\theta_i), s_{i+1}(\theta_{i+1}), \dots, s_i(\theta_i))$$

We then find an equilibrium by maximizing the sum of player’s payoff u_i , weighted with the probability that her opponent is of a given type $p(\theta_{-i}|\theta_i)$ for each of the player’s strategies $s'_i \in S_i$ as is shown below.

$$s_i(\theta_i) \in$$

$$\arg \max_{s'_i \in S_i} \sum_{\theta_{-i}} p(\theta_{-i}|\theta_i) u_i(s'_i, s_{-i}(\theta_{-i}), (\theta_i, \theta_{-i}))$$

The Engine

The engine is the *decision mechanism* of the NPC. It is created from the plot that the DPGE generated. This means that the murderer and each of the suspects are created as

independent NPCs', fully equipped with a knowledge base describing their world with respect to the plot. The NPC uses the engine to calculate a rational sentence to speak in terms of game theory.

In game theoretic terms the NPC that is about to speak is the *player i* and the NPC or human that the player is speaking to is the *opponent -i*. When mapping a dialog into a "Bayesian game" then one game is one NPC finding one sentence from a set of sentences to speak against one opponent. When the opponent replies then we have another Bayesian game. This means in game theoretic terms that we have a *two player, non-cooperative, sequential game*. There is one player against one opponent where the player has some n sentences (decisions) to choose from. The player knows his type θ_i but not her opponents type θ_{-i} . According to the definition of exogenous data then the player knows all the possible types that the opponent could be of and for each of those types the player will know all payoff functions contingent to each type. This means that the player knows all the possible set of replies that the opponent could have and that for each of those sets the player knows what the opponent would gain by each reply. The player also has some estimate of how probable the opponent is to be of any given type. The game is sequential which means that the player will speak and then the opponent will reply.

The type does not refer to character types such as suspect or murderer, instead it refers to the status of the NPC's knowledge base. This means that an NPC that has not been told of an affair is of one type but when the same NPC is told about an affair then he becomes another type. This also means that in the worst case in a brute force algorithm then all possible types of a player would mean a crossproduct of all the states of the variables in the Bayesian net of the NPC.

This is where the Multi Agent Influence Diagrams (MAIDs) (Koller & Milch 2003) come to the rescue. A MAID is an extended Bayesian net for finding equilibrium in agents decision problems. It is important to note that the game theoretic game just described is what Koller and Milch refer to as a "single agent decision problem" (Koller & Milch 2003). This means that there is only one agent that needs to make a decision in each game. The agent is the player in game theoretic terms. The opponent does not make his decision until after the current player has acted.

For each run of the engine it needs to set up the Bayesian game. First it generates sentences and all possible replies to each of the sentences from the knowledge base. The knowledge base is described in the next section followed by a section that describes how the sentences and replies are generated.

When a Bayesian game has been set up then the engine finds an equilibrium, the set of optimal sentences, using MAIDs and then the NPC can pick one of these optimal sentences and speak. This is described in section "Finding Equilibrium". The engine also adds any knowledge gained to the knowledge base.

Knowledge

The NPC has a set of knowledge subnets where she maps everything she knows. She also has a set of knowledge subnets

to map what she believes her opponents know. Moreover she has a set of knowledge subnets to map what she assumes her opponents assume that their opponents know.

Each of the subnets that have just been described above is divided into many small subnets. Each of these small subnets contains two or more variables that describe the state of some piece of knowledge. One such subnet is shown in Figure 1. The subnets name is *mother* because it handles the piece of knowledge that describes who if any of the suspects was the victim's mother. Each of the variables has three states, two possible female names and null. The null represents the possibility that none of the suspects is the victim's mother. The two female names are suspects. The top right variable *mother prem* is connected to other *...prem* variables in the net. Each of these knowledge piece subnets have exactly one such variable. This is a *premises* variable that corresponds to the plot generated at the start. Some of these premises variables are instantiated at the start of the game to represent what each NPC knows with respect to the generated plot.

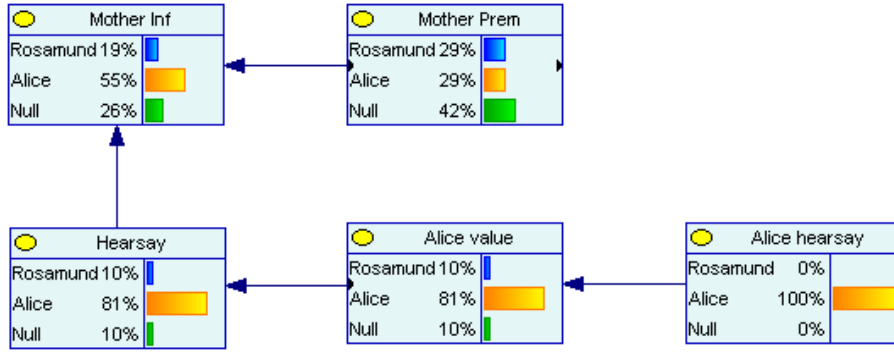
The top left variable *mother inf* is an inference variable. This variable represents the inferred knowledge of both actual data from the premise variable and information that the NPC has been told by an opponent. The *Alice hearsay* variable shows what the opponent said. In this example the NPC was talking to Alice and Alice claims to be the victim's mother. The *Alice value* variable shows how much value the NPC places on Alice's claims. The *value* variables are influenced by the NPC's gullibility and by how reliable she believes her opponent is. If the NPC then talks to other opponents and these opponents also give information on this piece of knowledge then that is added in a similar way. Thus the NPC can receive conflicting evidence from its opponents and needs to evaluate each hearsay with respect to the characters reliability. The hearsay variable aggregates all hearsay and the inference variable aggregates the premises and hearsay variable. If the NPC would receive concrete evidence then only the premises variable would be instantiated.

The premise and inference variables are used to generate sentences which is described below. There is ample opportunity to play around with possible characteristics and emotions when handling the knowledge variables. Intelligence would for instance influence how much precedence, if any, the hearsay variable would have over the premises variable. This could also be influenced by gullibility and some emotions. Moreover the hearsay value variable could take more factors into account such as emotions or connections to the opponent or the type of knowledge being handled. The NPC might put a higher value on the opponent's claim of being the parent than on the opponent's claim of not having a dark secret.

Generating Sentences and Replies

The sentences are generated from the knowledge variables. The process is started by an arbitrary seed. This seed could come from another NPC or from a player or from some event or interaction in the game. First a knowledge variable matching the seed is found. For instance the *Mother Prem* variable in Figure 1 would match the seed *mother*.

Figure 1: Knowledge subnet



Then all the parents and children of the *mother prem* variable are retrieved. These are premises that are connected in the Bayesian net to the *mother prem* variable. All of these variables can be considered “relevant” because they either match the seed or have causal relations to a variable that matches the seed.

Each state in a relevant knowledge variable is used to create two sentences, one for the affirmative and one in denial. For instance the *Mother Prem* variable in Figure 1 would generate 6 possible sentences, namely: “Rosamund is the victims mother”, “Rosamund is not the victims mother”, “Alice is the victims mother”, “Alice is not the victims mother”, “none of the suspects is the victims mother” and finally “one of the suspects is the victims mother”. Knowledge variables are also combined to create sentences that are based on more than one knowledge variable. For example the following sentence contains three knowledge variables: “Rosamund is a tall, blond female”.

The expected replies are generated in the same way as the sentences except that now each of the knowledge variables used to generate sentences is used as a seed for generating replies.

Finding Equilibrium

The NPCs want to find an equilibrium to determine a set of optimal sentences from those generated from the seed. It is important to realize that the game theoretic “game” that the NPCs face is sequential, e.g. the NPC makes a move and then his opponent makes a move. They do not make their moves in parallel which simplifies the computation of equilibrium significantly.

First the engine sets up a game theoretic game using Multi Agent Influence Diagrams (MAIDs) (Koller & Milch 2003). The MAIDs offer an algorithm to find equilibrium in incomplete information games for single agents decision problems in linear time. The MAIDs map the decision variables of the Bayesian net with a “relevance graph” to discover in what order the decisions should be evaluated. There are two decisions that need to be evaluated by the NPC: The assumed reply of the opponent and the sentence that the NPC should speak with respect to the assumed reply.

Since the set of optimal sentences is dependent on their replies the first step is to calculate the set of optimal replies

to each of the sentences. The second step is to determine the set of optimal sentences with respect to the replies that each of them is liable to receive.

This means that the engine generates all the possible strategies (sentences) that the NPC can play, $s'_i(\theta_i)$, and all the strategies (replies) that the opponent can play contingent on each of their possible types $s_{-i}(\theta_{-i})$. This is done by generating the sentences and replies as previously described.

These strategies are then used to find the weighted payoff as detailed for the Bayesian equilibrium (Fudenberg & Tirole 1991),

$$p(\theta_{-i}|\theta_i)u_i(s'_i, s_{-i}(\theta_{-i}), (\theta_i, \theta_{-i}))$$

Next the engine sums up the payoff for each of the players strategies and from the sum collects the set of strategies that carries the maximum value.

Finally the NPC can simply pick some random sentence from the set of optimal sentences to speak. The engine adds any knowledge gained to the NPC’s knowledge base as described in section “Knowledge”.

In order to better clarify how the engine evaluates sentences I will take as an example an NPC named Horace that is talking to an opponent named Linda. Horace has just been asked whether he owed the victim money. First the engine generates a set of possible sentences for Horace to speak. Horace could both say “I’m in debt” or “I’m not in debt” and he could also say “I’m rich” or “I’m poor” since that has a logical connection to whether someone is indebted or not. There could of course be other options but these will do as an example. The engine also creates all possible replies to each of the possible sentences.

If Horace was indebted to the victim then he may want to lie as admitting to having a motive will make him suspicious. Now lying has risks; if Linda catches Horace in a lie then Horace will appear to be even more suspicious than if he tells the truth. Horace may also have high moral standards and not care to lie.

Lets say that Horace was indebted to the victim, then admitting to it will carry a penalty due to increased suspicion. Denying it will carry a penalty due to risk of being caught lying and possibly because of moral standards. This would mean that admitting to being indebted is likely to get a higher payoff, especially if the engine calculates that the op-

timal response to denying the debt would indicate that Linda knows that Horace is lying. Horace can also state that he is poor. That may have the highest payoff. Being poor is not a motive for murder, it is not contradicting what Horace knows is true and unless he believes that Linda may claim that he is lying then claiming to be poor is likely to carry the highest payoff in this example.

If Horace is not indebted to the victim then either claiming to be rich or denying the debt is likely to carry the highest payoff.

When Horace has chosen a sentence to speak then his opponents knowledge base is updated with what Horace just said as described in section "Knowledge". Horace's knowledge base is also updated so that Horace will remember what he told Linda.

If Linda is an NPC then Linda's engine would start by adding the knowledge acquired from Horace's sentence to Linda's knowledge base. Then from the newly updated knowledge base the engine will generate all possible sentences for Linda to speak using Horace's sentence as a seed. Then Linda's engine draws up a new game theoretic game and the engine finds an equilibrium for Linda just as Horace's engine found an equilibrium for him. It is important to understand from this example that:

- Finding one sentence to speak is one game theoretic game.
- The goals of the NPC's are subjective such as not to lie or become suspicious.
- There are no explicit goals, such as "solve the mystery".
- Characteristics, feelings, knowledge, beliefs about the opponents reactions etc. influence the NPC payoff.
- The NPC's add to their knowledge base and the next game theoretic game reads from an updated knowledge base.
- The NPC does not care whether he is talking to a human or another NPC, it is not a factor in the engines calculations.

Results

The NPCs are able to calculate an optimal sentence using game theory and they are able to participate in a dialog, (see example 1).

Example 1 (Dialog) *Rosamund and Alice are talking together and Rosamund got the seed wedlock which is a possible motive for murder. The conversation drifts into stating their actual relations with the victim and ends when Rosamund has nothing to add to the conversation.*

Rosamund says: *Horace did not hold me in wedlock.*

Alice says: *Horace did not hold me in wedlock.*

Rosamund says: *I'm not Horace's wife.*

Alice says: *I'm not Horace's wife.*

Rosamund says: *I'm Horace's parent.*

Alice says: *Horace did not have an affair with a male.*

Rosamund says: *I'm Horaces mother.*

Alice says: *Horace was having an affair with me.*

Rosamund can't think of anything to say!

As can be seen from the example the sentences generated are far from what humans would expect in a dialog. The engine does not attempt to generate sentences that humans would find interesting, natural and fully coherent. What needs to be added to the engine to create human likable sentences is discussed in "Future Work" subsection "Creating Human Like Dialogs".

The NPCs Rosamund and Alice are clearly able to hold a dialog where they state anything logically relevant to the topic *wedlock*. When Rosamund ceases to find anything optimal to say then the conversation halts. They have managed to state that they were not married to the victim and thus did not murder him to be released from *wedlock*.

Over 75% of the sentences are computed within 3 minutes and more than 50% in less than 1 minute. It helps that most of the time there were not that many sentences generated. Moreover the time complexity is linear with respect to the number of sentences (decisions) each time, because the algorithm uses MAIDs, see (Koller & Milch 2003).

Conclusion

This engine serves as a proof of concept and the current implementation is small and limited. The NPCs have a knowledge base of just over 3000 nodes in their Bayesian nets.

The engine is successful in generating a rational dialog, it successfully calculates a rational sentence for an NPC to speak. It uses the knowledge base of the NPC to generate sentences and replies and calculates an equilibrium according to game theory. This means that they are able to evaluate the set of possible sentences not only by what they themselves think but also by how they expect their opponent may think and reply. Moreover the NPCs are able to adapt to what they hear from their opponents. They do this by adding the hearsay nodes with respect to their opponent's last spoken sentence. It is clear that there is a basis for creating characters that can participate in a dialog and calculate on the fly a rational sentence to speak each time.

There is still a lot of work to be done in order to have the NPCs fully functional in a narrative game. The speech needs to be made more acceptable to humans and the calculations need to be optimized. This is better discussed below.

Without any optimizations, over 75% of the time equilibrium is found within 3 minutes and more than half the time equilibrium is found in less than 1 minute. If this method would be compared to a typical heuristic search or a solution using a lookahead then it can be seen that these times are promising. An optimized heuristic search is unlikely to be able to reach 5 levels deep in its search tree within 3 minutes in an equivalent search space.

Another pleasing property of the proposed solution is that the MAIDs have a linear growth (Koller & Milch 2003) and thus the complexity is linear with respect to number of sentences considered each time.

Finally this is a solution that uses inference and causality along with game theory to find an optimal sentence instead of the common approach of using a lookahead or popular search techniques. This approach is more abstract and it is not dependent on as detailed prior descriptions of search

spaces as many search techniques are. Moreover it does not have the computational problems of the search techniques or techniques that apply a lookahead.

Future Work

As has been described in the article the engine focuses solely on finding the rational sentence to speak in a dialog using game theory. There is considerable room for improving and for extending the engine. Here I will first point out how the engine could be optimized then I will describe what needs to be added so that the NPCs would be able to generate “human like dialogs”; Sentences in natural language and context that humans would find to be natural and rational in a dialog.

Optimizing Calculations

There are a few fairly obvious optimizations that can be done to speed up the calculations.

First, dynamic programming can be applied by storing the value of each sentence in a hash-table by its premises so that it could be easily retrieved instead of recalculated when it is again relevant. It is highly likely that the stored value would still be perfectly valid which would make this a very efficient optimization. When running the engine it becomes fairly clear that a large number of sentences are unnecessarily recalculated.

Secondly, it is not always appropriate to find an equilibrium. It is in fact only appropriate if the decision is dependent on the reply. For instance introducing oneself is not dependent on the reply. It is somewhat dependent on the circumstances, whether one needs to be formal or not, but the opponent’s reply will not influence the player’s payoff. So some method should be developed for the player to decide whether finding an equilibrium is at all necessary. When it is not necessary to find an equilibrium then some other method that better fits the circumstances should be used.

Thirdly, the sentences are generated in a brute force way and this can of course be improved. The complexity of the algorithm for finding equilibrium is linear with respect to the number of sentences evaluated. This means that refining the sentence generation process should significantly reduce computing time. This calls for an algorithm that can choose which sentences are contextually logical in the current dialog. Such an algorithm could use techniques from the field of “Dialog Games”. For a general overview of dialog games see (McBurney & Parsons 2002).

Finally, if the NPC believes that his opponent is indecisive in some area then he can maximize his payoff from his own net assuming that the opponents reply is very predictable. For instance if the NPC believes that his opponents knowledge of the murder weapon is very general then he can simply use his own general knowledge of a weapon to decide what to say instead of finding an equilibrium for that group of sentences. This is because the NPC has then no reason to expect that what he says about the murder weapon would in any way challenge his opponent.

Creating Human Like Dialogs

First, the sentences need to be converted into natural language using natural language processing modules. This

should not be difficult as most such modules expect some first order logic semantics and the sentences are generated by the engine using first order logic derived from the Bayesian net.

Secondly, it is necessary to remove inappropriate sentences such as “I am a male”. Sometimes it is appropriate to state ones sex, for instance when filling out an application or a form. In order to effectively generate appropriate sentences it is good to use known methods from the field of dialog games (McBurney & Parsons 2002) so that sentences can be generated with respect to the type of dialog that the NPC is currently participating in. This could also help in optimizing calculation. This should be done by a specialized algorithm when generating sentences and replies prior to calculating the equilibrium.

Finally, in addition to adding natural language processing it would also be a good idea to add a module or modules that handle other aspects of speech, such as intonation and how to phrase the sentence. Does the NPC want to be demanding, polite or questioning? Example: “Close the window”, “Please close the window” or “Isn’t it cold in here?” For graphical display it would also be necessary to add some modules for facial expressions and body language.

Acknowledgments

I would like to thank my advisor Luca Aceto who gave me his backing to tackle this unprecedented work at the school of Computer Science at Reykjavik University. Many thanks to Finn Verner Jensen for valuable input and the department of Machine intelligence at Aalborg University. Thanks to Daniel Kudenko and Hannes Högni Vilhjálmsson for reviewing and helpful comments. Thanks to anonymous reviewers for helpful comments.

References

- Arinbjarnar, M. 2006. Murder she Programmed: A Dynamic Plot Generating Engine for Murder Mystery Games. Bachelor of Science project at Reykjavik University <http://nemendur.ru.is/maria01/greinar/BSc.pdf>.
- Chaplin, D. J., and Rhalibi, A. E. 2004. IPD for Emotional NPC Societies in Games. *Proceedings of the International Conference on Advances in computer entertainment technology*.
- Fudenberg, D., and Tirole, J. 1991. *Game Theory*. The MIT Press.
- Harsanyi, J. C. 1967. Games with Incomplete Information Played by Bayesian Players, I-III. Part I. The Basic Model. *Management Science* 14(3):159–182.
- Harsanyi, J. C. 1995. Games with Incomplete Information. *The American Economic Review* 85(3):291–303.
- Heap, S. H.; Hollis, M.; Lyons, B.; Sugden, R.; and Weale, A. 1992. *The Theory of Choice*. Blackwell.
- Jensen, F. V. 2001. *Bayesian Networks and Decision Graphs*. Springer-Verlag.
- Koller, D., and Milch, B. 2003. Multi-Agent Influence Diagrams for Representing and Solving Games. *Games*

and Economic Behavior 45(1):181–221. Full version of paper in IJCAI '03.

McBurney, P., and Parsons, S. 2002. Dialog Games in Multi-Agent Systems. *Informal Logic* 22(3):257–274.

Nash, J. 1951. Non-Cooperative Games. *Annals of Mathematics* 54(2):286–295.

Si, M.; Marsella, S. C.; and Pynadath, D. V. 2005. Thespian: Using Multi-Agent Fitting to Craft Interactive Drama. In *Proceedings of the International Conference on Autonomous Agents and Multi Agent Systems* 21–28.

Silverman, B. G.; Bharathy, G.; O'Brien, K.; and Cornwell, J. 2006. Human behavior models for agents in simulators and games: part II: gamebot engineering with PMFserv. *Presence: Teleoperators and Virtual Environments* 163 – 185.

Theune, M.; Faas, S.; Nijholt, A.; and Heylen, D. 2002. The Virtual Storyteller: Story creation by intelligent agents. *Proceedings of the Workshop on Storytelling in Collaborative Virtual environments at ACM Collaborative Virtual Environments* 95–100.

Theune, M.; Rensen, S.; Akker, R.; Heylen, D.; and Nijholt, A. 2004. Emotional Characters for Automatic Plot Creation. *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment* 95–100.

Tolk, A. 2002. Human Behaviour Representation - Recent Developments. *presented at the Lecture Series on Simulation of and for Military Decision Making*.