Modeling intelligent agent beliefs in a card game scenario

Marcel Gołuński[1,a], Roman Tomanek[2,b], Piotr Wąsiewicz[3,a]
[a]Institute of Electronic Systems, Warsaw University of Technology; [b]Faculty of Christian Philosophy, Cardinal Stefan Wyszyński University in Warsaw

ABSTRACT

In this paper we explore the problem of intelligent agent beliefs. We model agent beliefs using multimodal logics of belief, $KD45_{(m)}$ system implemented as a directed graph depicting Kripke semantics, precisely. We present a card game engine application which allows multiple agents to connect to a given game session and play the card game. As an example simplified version of popular Saboteur card game is used. Implementation was done in Java language using following libraries and applications: Apache Mina, LWJGL.

Keywords: agent beliefs, card game engine, Kripke Model, multimodal logic of beliefs, Saboteur

1. INTRODUCTION

People very often participate in so-called games during argumentative dialogues, when somebody may win an argument or be defeated after claiming against an opponent. Conflict of interests is very characteristic in social environments in which several persons interact. Human invention and logic can involve more different games between actors or players e.g. economic competition, social cooperation, chess, soccer, war, or dialogue [1,2,3]. Each of the individuals control some of the variables that determine the eventual outcome and the individual decisions, taken together, determine the behavior of the collective [4]. The mathematical analysis of conflict and other situations of social interaction together with strategic aspects such as move or decision order and the individuals' epistemic characteristics or their attitudes towards risk, belong to so-called game theory, which proper name should be theory of play.

Epistemic game theory has added one more element to actions, strategies and preferences the role of factual and higher-order information, especially in autonomous agent environments [5,6]. In this paper we describe our efforts of making simplified saboteur game [7] server with agents changing their beliefs and actions depending on their internal logic model and game board situations.

2. THEORETICAL BACKGROUND

2.1 Introduction

Modal logic is the logic of necessity and possibility, of 'must be' and 'may be'. This may be interpreter in various ways. If necessity is necessary truth, there is alethic, if it is normative necessity, there is deontic logic. It may refer what is known or believed to be true, in which case, there is epistemic logic.

Logic is concerned with truth and falsity. In modal logic we are concerned with truth or falsity in other possible worlds as well as the real one. In this sense a proposition will be necessary in a world if it is true in all worlds which are possible relative to that world, and possible in a world if it is true in at least one world possible relative to that world. Modal logic is based upon the 'ordinary' (two-valued) Propositional Calculus, and when we use the expression 'Prepositional Calculus' (or the abbreviation 'PC') simpliciter, it is to this non-modal system of logic that we shall be referring [8].

2.2 Systems of modal logic: K, S4, S5, KD45 [9]

The most familiar logics in the modal family are constructed from a weak logic called ***K*** (after Saul Kripke). A variety of different systems may be developed for such logics using ***K*** as a foundation. The symbols of ***K***: '~' for 'not', '→' for 'if…then', '□' for the modal operator 'it is necessary that' and '◊'for the modal operator 'it is possible that. (The

[1] marcel.golunski@gmail.com, mgolunsk@elka.pw.edu.pl
[2] r.tomanek@uksw.edu.pl
[3] pwasiewi@gmail.com, pwasiewi@elka.pw.edu.pl

Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2012, edited by Ryszard S. Romaniuk, Proc. of SPIE Vol. 8454, 84542A · © 2012 SPIE
CCC code: 0277-786/12/$18 · doi: 10.1117/12.2000238

connectives '&', '∨', and '↔' may be defined from '~' and '→' as is done in propositional logic.) ***K*** results from adding the following rules to the principles of propositional logic:

Necessitation Rule: If φ is a theorem of ***K***, then so is $\Box\varphi$.

Distribution Axiom: $\Box(\varphi\rightarrow\psi)\rightarrow(\Box\varphi\rightarrow\Box\psi)$.

(We use 'φ' and 'ψ' as metavariables ranging over formulas of the language.) According to the Necessitation Rule, any theorem of logic is necessary. The Distribution Axiom says that if it is necessary that if φ then ψ, then if necessarily φ then necessarily ψ. The operator ◊ (for 'possibly') can be defined from □ by letting $\Diamond\varphi = \sim\Box\sim\varphi$.

The system ***K*** is too weak to provide an adequate account of necessity. The following axiom is not provable in ***K***, but it is clearly desirable: *(M)* $\Box\varphi\rightarrow\varphi$ (*M*) claims that whatever is necessary is the case.

(4) $\Box\varphi\rightarrow\Box\Box\varphi$

(5) $\Diamond\varphi\rightarrow\Box\Diamond\varphi$

S4 is the system that results from adding (4) to *M*. Similarly ***S5*** is *M* plus (5) and ***system*** *KD45* results from adding (4) and (5) to (*D*) $\Box\varphi\rightarrow\Diamond\varphi$

2.3 Multimodal Logics

Modal logics have been widely studied for reasoning about knowledge and belief, usually based on the monomodal logics *S5* and *KD45* for reasoning about knowledge and belief respectively. Both logics have axioms (4) : $\Box\varphi\rightarrow\Box\Box\varphi$ and (5) : $\sim\Box\varphi\rightarrow\Box\sim\Box\varphi$, which mean that knowledge and belief both satisfy positive and negative introspection.

The logic *S5* also has axiom (*T*) : $\Box\varphi\rightarrow\varphi$, which means that knowledge is veridical, while *KD45* also has axiom

(D) : $\Box\varphi\rightarrow\sim\Box\sim\varphi$ which means that belief is consistent.

We consider multimodal logics with m pairs of modal operators $\Box_i$ and $\Diamond_i$ where $1 \le i \le m$. We use p and q to denote primitive propositions. Formulae of our language are:

$\varphi ::= \bot \mid \top \mid p \mid \sim\varphi \| \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \varphi\rightarrow\varphi \mid \Box_i\varphi \mid \Diamond_i\varphi \mid$

Definitions for Multimodal Logics

The modal depth of a formula φ is the maximal nesting depth of modal operators occurring in φ. For example, the modal depth of $\Box_1(\Box_2 p \vee \Diamond_1 q)$ is 2.

A Kripke frame is a tuple $(W;\tau;(R_i)_{1\le i\le m})$ where *W* is a non-empty set of possible worlds, $\tau\in W$ is the actual world, and each R_i is a binary relation on *W*, called the accessibility relation for $\Box_i$ and $\Diamond_i$. If $R_i(w;u)$ holds, then we say that the world u is accessible from the world w via R_i.

Kripke Model

A Kripke model is a tuple $(W;\tau;(R_i)_{1\le i\le m},h)$, where $(W;\tau;(R_i)_{1\le i\le m})$ is a Kripke frame and *h* is a function mapping worlds to sets of primitive propositions. For $w\in W$, the set of primitive propositions "true" at *w* is *h(w)*. Given a Kripke model $M=(W;\tau;(R_i)_{1\le i\le m},h)$ and a world $w\in W$, the *satisfaction relation* ⊨ is defined as usual for the classical connectives with two extra clauses for the modalities as below:

$M;w \vDash \Box_i\varphi$ iff $\forall v\in W\,[\,R_i(w;v)$ implies $M,v\vDash\varphi\,]$ M

$M;w \vDash \Diamond\varphi$ iff $\exists v\in W\,[\,R_i(w;v)$ and $M;v\vDash\varphi\,]$

We say that φ is satisfied at *w* in *M* if $M,w\vDash\varphi$'. We say that φ is satisfied in *M*, and write $M\vDash\varphi$', and call *M* a *model* of φ, if $M;\tau\vDash\varphi$. *M* is a model of a set Γ of formulae if $M\vDash\varphi$' for every $\varphi\in\Gamma$.

If we allow all Kripke models (with no restrictions on the accessibility relations) then we obtain a multimodal logic which has a standard Hilbert-style axiomatisation denoted by $K_{(m)}$. Other normal multimodal logics are obtained by adding certain axioms to $K(m)$.

2.4 Multimodal Logics of Belief

To reflect proprieties of belief, one can extend K(m) with some of the axioms below:

(D) $\Box_i\varphi \rightarrow \sim\Box_i\sim\varphi$	belief is consistent
(I) $\Box_i\varphi \rightarrow \Box_i\varphi$ if $i > j$	subscript indicates degree of belief
(4) $\Box_i\varphi \rightarrow \Box_i\Box_i\varphi$	belief satisfies positive introspection
(4_s) $\Box_i\varphi \rightarrow \Box_j\Box_i\varphi$	belief satisfies strong positive introspection
(5) $\sim\Box_i\varphi \rightarrow \Box i\sim\Box_i\varphi$	belief satisfies negative introspection
(5_s) $\sim\Box_i\varphi \rightarrow \Box_j\sim\Box_i\varphi$	belief satisfies strong negative introspection

The following systems are intended for reasoning about multi-degree belief:

$KDI4_s = K(_m) + (D) + (I) + (4s)$

$KDI4 = K(_m) + (D) + (I) + (4)$

$KDI4_s5 = K_{(m)} + (D) + (I) + (4_s) + (5)$

$KDI45 = K(_m) + (D + (I) + (4) + (5)$

For multi-agent systems, we use subscripts on $\Box$ and $\Diamond$ to denote agents and assume that $\Box_i\varphi$ stands for „agent *i* believes that φ ' is true" and $\Diamond_i\varphi$ stands for : „φ is considered possible by agent *i*".

For distributed systems of belief we can use the logic system $KD4_s5_s$:

$KD4_s5_s = K(m) + (D) + (4_s) + (5_s)$

In this system, all agents have full access to the belief bases of other agents: they are united as „friends".

In another kind of multi-agent system, agents are „opponents" who play against each other, and each agent may want to simulate the epistemic states of the others. To write a program for one agent, we may need to use modal operators of the other agents. A suitable logic for this problem is: $KD45(_m)$:

$KD45(_m) = K(_m) + (D) + (4) + (5)$

The logic $KD45(_m)$ has been intensively studied. We use a subscript in $KD45(_m)$ to distinguish the logic from monomodal logic *KD45,* while there is not such a need for the other considered multimodal logics [10].

The given axioms correspond to the following frame restrictions:

Axiom	Corresponding Condition
(D)	$\forall u \exists v\ R_i(u; v)$
(I)	$R_j \subseteq R_i$ if $i > j$
(4)	$\forall u;v;w\ [Ri(u; v) \wedge Ri(v;w) \rightarrow Ri(u;w)]$
(4_s)	$\forall u;v;w\ [Ri(u; v) \wedge Ri(v;w) \rightarrow Ri(u;w)]$
(5)	$\forall u;v;w\ [Ri(u; v) \wedge Ri(u;w) \rightarrow Ri(w; v)]$
(5_s)	$\forall u;v;w\ [Ri(u; v) \wedge Ri(u;w) \rightarrow Ri(w; v)]$

3. AGENT BELIEFS

3.1 Overview

Agent beliefs can be represented using modal logic. In case of our sample card game Saboteur[4] we want to model what agents believe about roles of other players in the game. We also want our agent to imagine what other players might presume about roles played by each game participant. That's why *KD45($_m$)* modal logic system fits to our needs.

Given the *KD45($_m$)* logic system axioms and assumptions we can implement Kripke Model as a graph with the following properties.

3.2 Kripke Model as a graph

Kripke Model can be presented as a directed complete graph $D = (V, A)$ consisting of:

- **Vertices** *(V)* (or nodes) representing **possible worlds** in a Kripke Model.
- **Directed Edges** *(A)* (or arcs), which are ordered pairs of vertices, symbolize **transitions**, relations between possible worlds.

From this point onward we will use word graph in the meaning of Kripke Model structure representing *KD45$_{(m)}$* system. For a sample graph observe figure 1. Every pair of vertices in the graph is connected with two directed edges. Each edge points in a different direction. Edges represent accessibility of possible worlds.

In our model a possible world (see figure 2) is one set of roles agents might play in the game. In one possible world each agent plays exactly one role. Apart from the meaning of symbols S and M visible in figures 1-4, let's assume that we have 3 agents and we know that each one of them can be S or M (have role S or role M). One agent might be only S or M not both. Agent can be only S or M there is no other option. Now if we know that only one of three agents is S and the remaining two are M, we can guess who is who. Possible world represents one potential variant of roles for each agent. A set of symbols SMM visible in the top possible world (figure 1) means that the first agent is S, second and third agents are M. To consider all possibilities we create a set of all possible permutations of roles – all possible worlds. For three agents in question it would be SMM, MSM, MMS there is no other option. One of those worlds is the actual world, and we want our agent to discover which one.

If a possible world is accessible it might be an actual world (at that moment of time, because accessibility of worlds changes over time). To represent accessibility we use directed edges called **transitions (T)** (see figure 3). If both edges between nodes SMM and MMS exist it means that one of these worlds might be actual. If a possible world is not accessible (there is no edge pointing to this world) it means that assumption of agent roles in that node is invalid. Existence of edges represents agent beliefs. If only one world is pointed by existing edges it is the actual world.

The problem to be solved is how to determine if edge between world A and world B exists. How to represent change of belief over time and how to represent edges that point to “more” possible worlds.

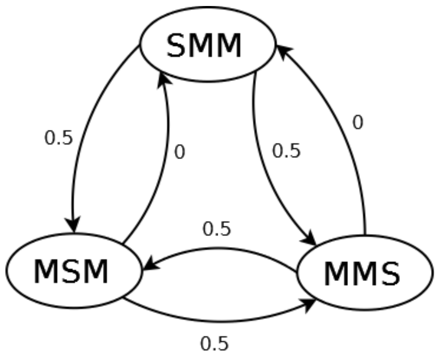

Figure 1. Simple Kripke Model for a card game „Saboteur” for three players.

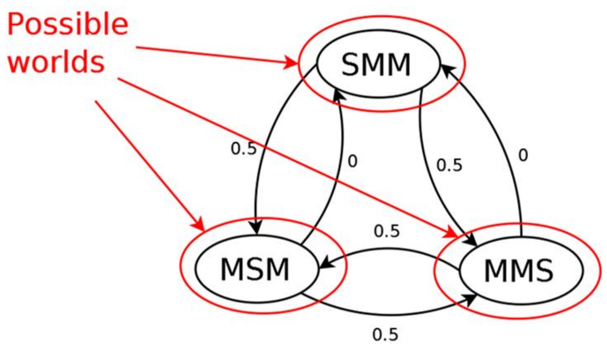

Figure 2. Simple Kripke Model for a card game „Saboteur” for three players, possible worlds.

[4] Abridged Saboteur rules are presented further on

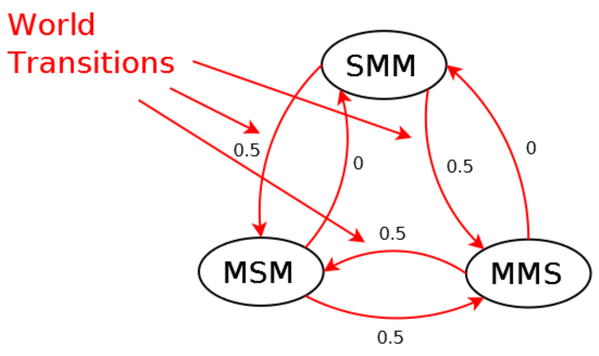

Figure 3. Simple Kripke Model for a card game „Saboteur" for three players, world transitions.

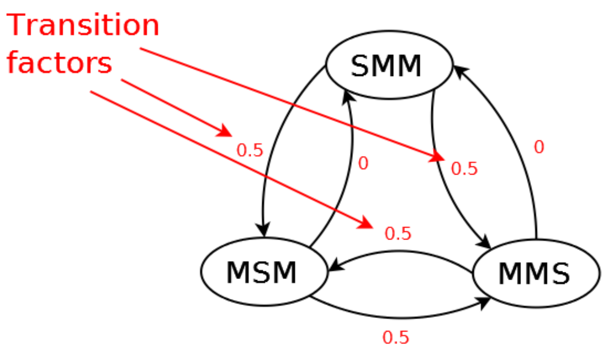

Figure 4. Simple Kripke Model for a card game „Saboteur" for three players, transition factors.

In our model each edge has a **transition factor (TF)** assigned (numerical value between 0 and 1, see figure 4). Transition factor defines if transition (edge) is "present" and world pointed by that edge is accessible. To determine if transition is existent we compare transition factor with a parameter called **transition threshold (TT)**. To determine all existent and nonexistent transitions consider the following formulae:

$$\forall v\,\forall u\,(v!=u \wedge TF(v,u) > TT \rightarrow T(v,u)) \tag{1}$$

$$\forall v\,\forall u\,(v!=u \wedge TF(v,u) \leq TT \rightarrow \sim T(v,u)) \tag{2}$$

where:

$v, u \in V$

$TF(v,u)$ is a transition factor between possible worlds v and u

TT is a transition threshold

$T(v,u)$ is an "existent" transition between possible worlds v and u

Note that in figures 1-4 transition factors for two transitions are set to 0. The reason for that arises from the fact that agent who "created" that graph also plays a role in the game, and in that case it's agent 1 who surely isn't S but is unsure whether agent 2 or 3 is S.

In the beginning of the game, when nothing can be inferred about agent roles, each transition factor (except the zero ones described above) are set to 0.5 which means that we have no idea which world is more plausible.

Transition factor of value 1 means agent is positive that the world pointed by that transition is accessible. Transition factor of value 0 implies otherwise, complete disbelief. But the very existence of transition is dependent not only on the value of transition factor but on the relation between transition factor and transition threshold as (1) and (2) implies. For example if $TF(v,u) = 0.7$ but $TT = 0.9$ then $\sim T(v,u)$.

3.3 Transition Factor change function

After every player turn in the game world agents update beliefs "stored" in their Kripke Models by changing transition factors. During his turn hypothetical player P performs an action that will be evaluated by other players (agents) and player P will be considered as more likely to be M or S (if no action was taken no transition factor change will happen).

When an action is taken by player P and that action is considered to be of S-role kind each possible world where that player is S should be more accessible, and each world where player P is M should be less accessible. That change is reflected by transition factor change function. Transition factor can be increased (more accessible) or decreased (less accessible). Please consider following formulae:

Transition factor increase:

$$TF_n = TF_{n-1} + \frac{1 - TF_{n-1}}{\alpha} \tag{3}$$

Transition factor decrease: $$TF_n = TF_{n-1} - \frac{TF_{n-1}}{\beta} \tag{4}$$

where:

TF_n is a transition factor for current turn n that will be calculated

TF_{n-1} is a transition factor for turn n-1 before player P took an action

α is a belief increase ratio

β is a belief decrease ratio (disbelief)

$\alpha, \beta \geq 1$. For value of 1 belief change is drastic, greater the value of α,and β the change of belief is lesser.

3.4 Sample Game: Simplified Saboteur

Saboteur is a card game designed by Frederic Moyersoen. The players take on the part of dwarves. Either they are miners (M) whose role is to get to hidden treasure, or they are saboteurs (S) trying to prevent miners from getting to treasure. If the miners manage to create a path to the treasure they win. However, if the miners fail, the saboteurs win. Initially players do not know the role of other players in the game, they must deduce other players role over the course of the game.

The game is based on cards. Cards are divided into two groups: action and path cards (there are also gold nugget and role cards, which are unimportant in our simplified version). Among the 44 path cards, there is one start card and three goal cards. One of the latter holds the treasure, the other two only a stone. Goal cards are concealed (put face down on the table) until reached by miners path. Start card is put face up on the table; it's the starting point of a miners tunnel. Over the course of the game, a maze of pathways from the start card to the goal cards is created (see figure 5). Path is created by path cards.

Action cards are played by putting them face-up in front of oneself or a fellow player. By means of action cards, players can hinder or help each other, take a card out of the pathway maze, or gain information about the goal cards.

Each player starts with a number of cards in his hand. Each turn player must play or discard a card, or pass if he can do neither. At the end of player's turn he draws one card from a deck (until the deck is used up).

The game ends when the miners get to treasure, or when the deck is used up and each player in turn has passed because he had no playable cards left in hand.

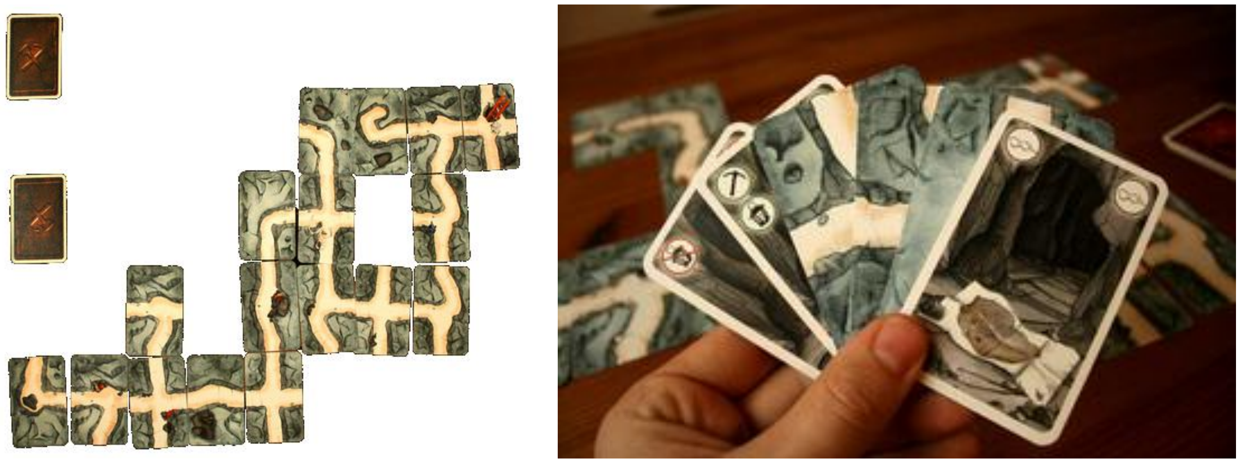

Figure 5. Saboteur Card Game (Source: http://www.annarbor.com/entertainment/saboteur-card-game-review/).

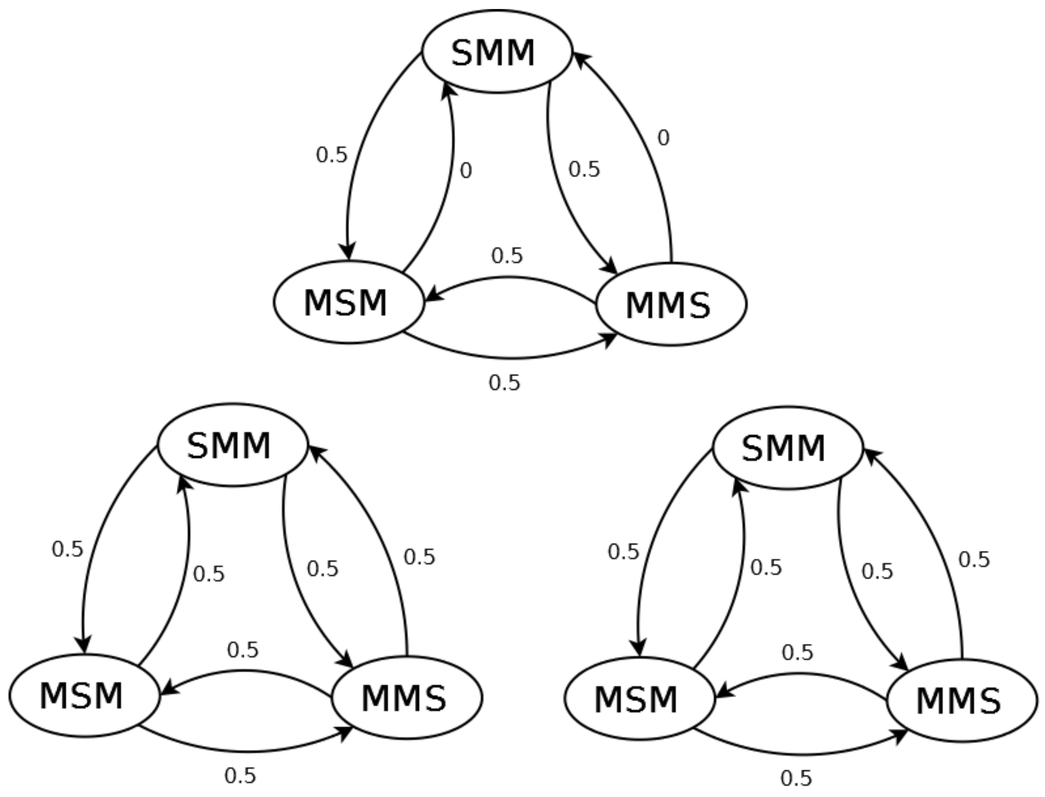

Figure 6. Krike models representing agent beliefs and agent assumptions of other players beliefs.

3.5 Agent Logic

In our agent logic we create separate Kripke Model (graph) for each player in the game. One graph represents agent’s beliefs. Every other graph represents agent’s assumptions of other players beliefs (see figure 6 for an 3-agent example – initial state of graphs). In figure 6 the top graph represents agent’s beliefs. Two lower graphs represent agent assumptions of other players beliefs. Note that graphs representing other players beliefs do not take into consideration the fact that agent 1 is not a saboteur (because other players do not have that knowledge). After each action taken in the game by player P agent updates each graph except for that player’s graph.

Actions taken by other players in the game must be evaluated by agents. It seems that the most straightforward evaluation method should look like this:

If player P:

- plays a good path card he is most probably a miner.
- plays bad path card he is most probably a saboteur.
- plays rock-fall[5] card for the miners sake he is most probably a miner, otherwise he’s a saboteur
- blocks[6] a player he has opposite role to this player*
- unblocks a player he has the same role as this player**

[5] If a player plays the rock-fall action card he may take a path card of his choice (except start and goal cards) out of the maze of pathways.
[6] There are player blocking and unblocking action cards. If a player is blocked he cannot play path cards.

* Having opposite roles implies that the transition factors of all the transitions to possible worlds in which those players have opposite roles should be increased, while transition factors of all the transitions to possible worlds in which those players have the same role should be decreased.

** Having the same role implies opposite to *.

In case of depth-one logic agent updates each graph only based on the above presented evaluation of player P action.

In case of depth-two logic agent includes belief graph transitions of the most probable ally in agent’s graph.

Algorithm pseudocode for depth-one logic:

```
//After every player action
Evaluate player P action;
if ( player P is a miner or saboteur)
   update every belief graph transition factors [except for player P graph]
   using transition factor change function [formulae 3,4].
else (if player P blocked or unblocked someone)
   update transition factors according to * and **
if (only one world is accessible in agent’s graph)
   agent knows what are other agents roles
```

In depth-one logic agent uses his own graph to reason about other agents roles. He may use other players graphs to change strategy. For example to stop pretending that he is a miner if he assumes that most of other agents believe that he is a saboteur.

Algorithm pseudocode for depth-two logic:

```
//After every player action
Execute depth-one algorithm
if (this is agent’s turn)
   assess the most probable ally using ***
   use the most probable ally’s graph to update agent’s graph, get all the
   transitions (existent and inexistent) from ally’s graph and apply transition
   factor change function [formulae 3,4] to agent’s graph for each ally’s
   transition
```

*** to get the most probable ally, get all the accessible possible worlds from all graphs (agent graph and each player graph). If agent is a saboteur (miner) it should select player who is considered to be saboteur (miner) in the biggest number of accessible possible worlds.

4. SYSTEM IMPLEMENTATION

4.1 Card Game Engine

Simplified saboteur was implemented as a sample game in our card game engine.

Card game engine was implemented in Java programming language in a Client-Server architecture. Server implementation allows for simple addition of new protocols and transports. Currently the only available server side acceptor is Apache Mina TCP/IP acceptor.

Server

Server is implemented in Java. Server architecture allows for simple addition of new communication channels, Main communication channel already implemented is Apache MINA framework [11] (New IO,TCP/IP). The work is being done to introduce http protocol access to card game server.

Agent Client API

Currently agents connect to the server using main communication channel, which is Apache Mina framework with java serialization of messages. Each agent creates TCP/IP connector and connects to the card game server. Work is being done to introduce XML API which will be platform and language independent.

User Client

User Client is a work in progress. Implementation is being done using Java programming language and LWJGL [12] java game framework. Screenshot from client GUI is shown in figure 7.

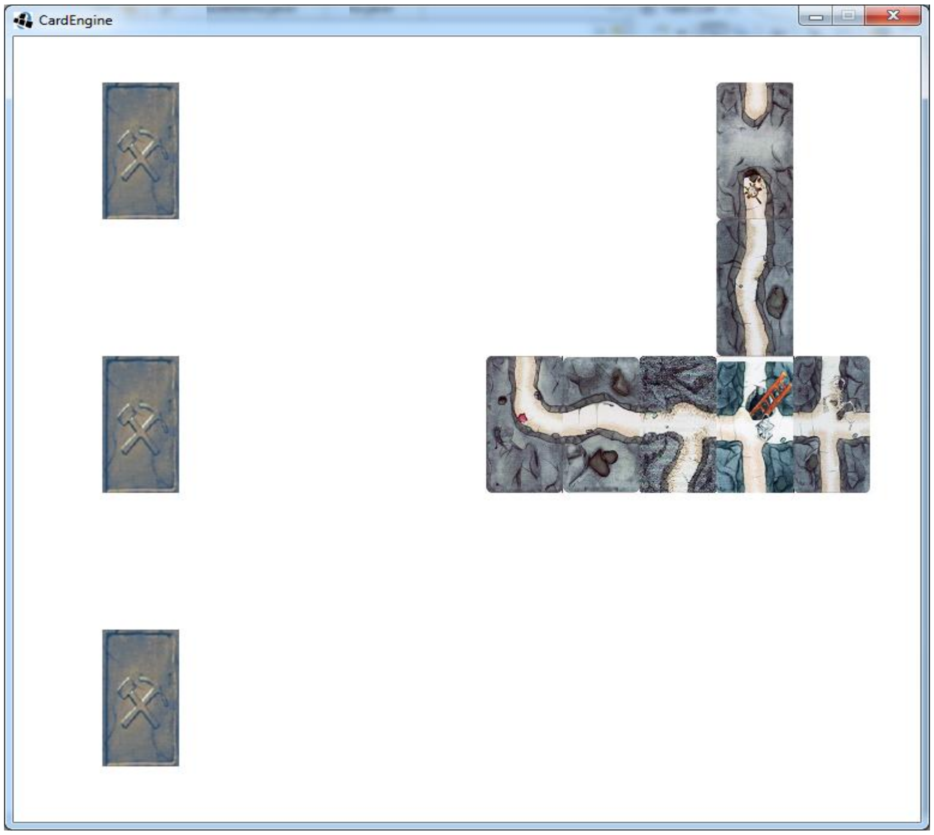

Figure 7. Screenshot from card game engine LWJGL user client.

4.2 Agent logic implementation

Currently whole agent logic is implemented in Java. We have created our own graph implementation to represent Kripke Model. Agents play their roles in a straightforward way. For testing purposes only standard simple strategy of saboteurs and miners was implemented.

We plan on integrating agent logic with OrientDB graph database to speed up graph based operations. OrientDB is an open-source deeply scalable Document-Graph database management system DBMS written (100%) in multiplatform java language. It supports ACID Transactions. On crash it recovers pending documents. It includes SQL language with extensions to handle relationships without JOINs, manage trees and graphs of connected documents. It has small footprint in the Local mode. Perfect for scenarios where the database is embedded. It is extremely light - about 1MB for the full server and super fast - it stores up to 150.000 documents per second, 10 billions of documents per day. It provides native management of trees and graphs, which is 100% compliant with TinkerPop Blueprints standard for GraphDBs. Blueprints became for many graph databases analogous to the Java Database Connectivity (JDBC) in relational databases. It serves as the foundational technology for: Pipes - a lazy, data flow framework, Gremlin - a graph traversal language, Frames - an object-to-graph mapper, Furnace - a graph algorithms package, Rexster - a graph server [13].

5. RESULTS

In a simple test scenario where agents play their roles in a straightforward way Kripke Model KD45(m) implemented as a graph works well. Due to lack of user client which is not yet finished, we couldn’t test agent logic (correctness of beliefs) in a complicated scenario, where player actions might be sophisticated and player’s true goals hidden.

The main problem of presented algorithm is to find correct values for parameters *TT*, α and β. Experiments show that larger α and smaller β give better results. With smaller β incorrect transitions to false possible worlds are removed faster (as shown in figure 8). With smaller α the level of "trust" grows slower, avoiding over-trusting after some insignificant decoy actions taken by saboteurs.

Another thing to consider is the adjustment of α and β parameters for different actions performed by players. For example player who played a blocking action card, or rock-fall card which removed an important pathway should be treated much more severe (smaller β used for transition factor change function) than player who might have mistakenly played wrong path card.

Lack of "hidden agendas" in player strategies didn't give us an opportunity to test the ability of agents to reason about what other players might believe.

In simple test scenario which was employed, depth-two logic didn't improve agent's ability to indicate other player roles. Depth-one logic was suitable enough.

6. SUMMARY

We have implemented *KD45(m)* modal logic system using graph representation of Kripke Model in a card game scenario. Saboteur card game gives interesting opportunities when it comes to the problem of agent beliefs about other players. That's why we have chosen Saboteur as a sample game for our game engine.

Presented algorithm proves useful in a simple scenario. Proposed transition factor change function gives flexibility in defining how much transition factor should be changed and based on what conditions. Unfortunately, due to lack of user client we couldn't have tested algorithm performance in real-life scenario.

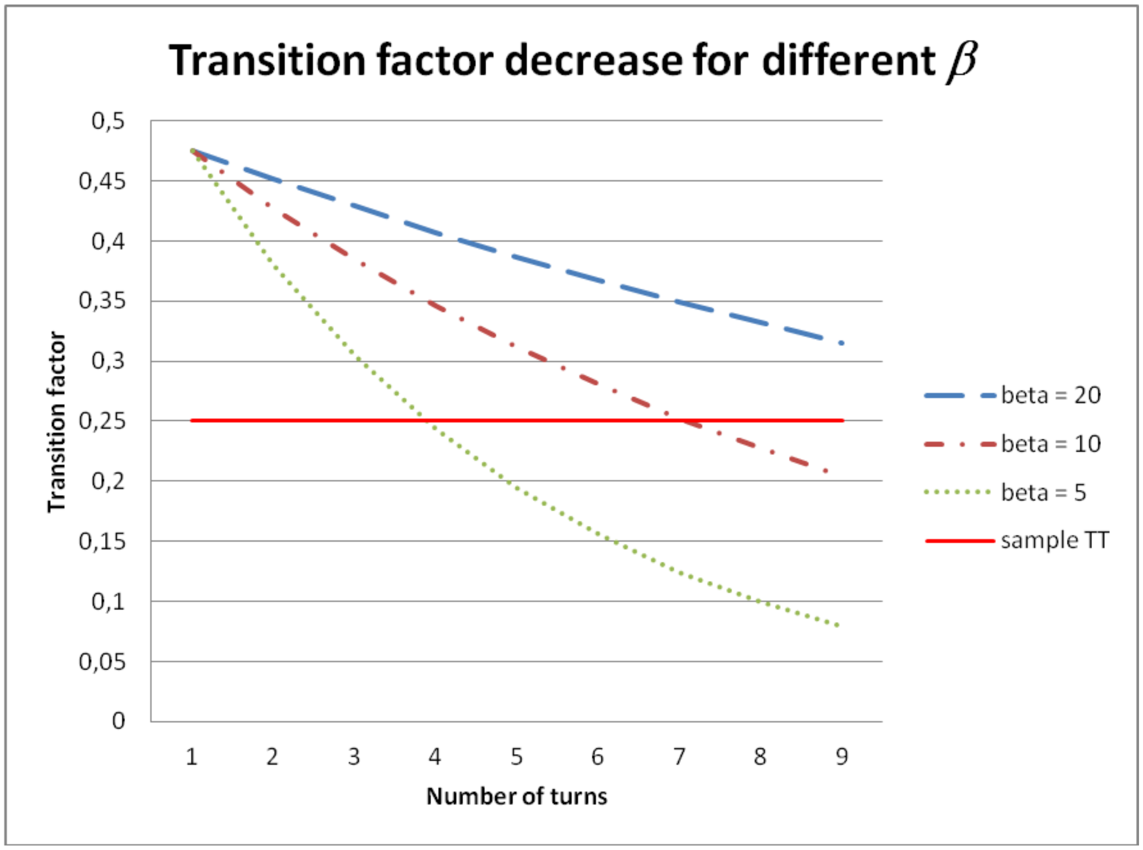

Figure 8. Transition factor decrease in function of game turns for three different β values.

Currently work is being done to finalize user client. We also plan on introducing some declarative language in our card game engine that will allow for creation of different card games. Now saboteur rules are programmed imperatively in Java code.

REFERENCES

[1] Huth M., Ryan M., [Logic in Computer Science: Modelling and Reasoning about Systems], Cambridge University Press (2004).

[2] van Benthem J., “Logic in Games”, ILLC, University of Amsterdam <http://staff.science.uva.nl/~johan/lig/> (September 2011).

[3] van Benthem J., van Ditmarsch H., van Eijck J., Jaspars J., “Logic in Action”, Free Internet Lectures <http://www.logicinaction.org/> (April 2011).

[4] Harrenstein B. P., [Logic in Conflict], PhD thesis, Universiteit Utrecht, Printed by Ponsen & Looijen B.V (2004).

[5] Shoham Y., Leyton-Brown K., [Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations], Cambridge University Press (2009).

[6] Harper L. W., Delugach H. S., “Using Conceptual Graphs to Represent Agent Semantic Constituents”, Proc. ICCS, 333-345 (2004)

[7] Moyersoen F., “Saboteur Rules”, AMIGO Spiel and Freizeit GmbH, Distribution in the US: Z-Man Games, <http://www.zmangames.com/cardgames/files/saboteur/Saboteur_US_Rules.pdf> (2004).

[8] Hughes G. E., Cresswell M.J., [A New Introduction to Modal Logic] , Routledge (1996).

[9] Cresswell, M. J., “Modal Logic”, in L. Goble (ed.), The Blackwell Guide to Philosophical Logic, Oxford: Blackwell, 136-158 (2001).

[10] Rajeev G., Linh Anh Nguyen, “Clausal Tableaux for Multimodal Logics of Belief”, Fundamenta Informaticae 94, IOS Press, 21–40 (2009).

[11] Apache mina framework, <http://mina.apache.org/>

[12] Lightweight Java game library < http://www.lwjgl.org/>

[13] OrientDB Documentation <http://www.orientdb.org/>