Towards encoding of the transition relation in dialogue games model checking

Anna Sawicka[1], Magdalena Kacprzak[2], and Andrzej Zbrzezny[3]

[1] Polish-Japanese Academy of Information Technology, Warsaw, Poland
[2] Bialystok University of Technology, Bialystok, Poland
[3] Jan Długosz University, Czestochowa, Poland
asawicka@pja.edu.pl, m.kacprzak@pb.edu.pl, a.zbrzezny@ajd.czest.pl

Abstract. We can understand a protocol as a set of rules used by the communicating entities i.e. people or computers. These rules specify allowed interactions between them. Every day people use protocols unconsciously during their conversations since they help to achieve the goal of the conversation (e.g. a compromise, a persuasion). In the paper, we focus on argumentative dialogues in which players can perform actions representing speech acts like *claim*, *question*, *scold* etc. Since we consider dialogues which have an emotional undertow we want to design a system for semantic verification of properties of dialogue games with emotional reasoning. This framework is based on interpreted systems designed for a dialogue protocol in which participants have emotional skills. The verified properties of the protocol are represented using the extension of CTL logic with commitment and emotion modalities (formulated in [7]) and GERDL language for a dialogue game specification ([18]).

Keywords: dialogue game, description language, dialogue protocol, emotions

1 Introduction

Dialogues are certainly the most popular form of communication, but also - the most complex one. To capture some aspects of the dialogue, we can formalize it by defining e.g. a set of participants, a set of possible actions, a set of rules, and so on.

Argumentation systems are formal frameworks for describing games where participants try to reach an agreement, solve a conflict or convince somebody and one of such systems is called a dialogue game [2, 19]. A dialogue can be treated as a two-player game, rules of which are intended to formalize and ensure correctness of the communication [11, 8, 21]. Formal dialogue systems can be also used as a schema for dialogues reasoning about emotions [10].

Currently, with increasing size and complexity of many systems, there is also a growing demand for the verification of properties of these systems. It is important, that the design meet some requirements. One of the solutions is to use model checking [1, 3, 13, 23, 15, 22], which is an automatic verifying technique for concurrent systems such as distributed systems, real-time systems, multi-agent systems, protocols, concurrent programs, and so on. To use this method, we can describe properties of dialogue protocols using propositional temporal logics e.g. linear temporal logic (LTL), computation tree logic (CTL), an extended computation tree logic (CTL*), the universal and existential fragments of these logics, and other extended logics.

Our goal is to adapt existing methods of verification to a new field - dialogue games and to show what mechanisms occur in human argumentative dialogues. We want to combine two approaches, dialogue games and model checking techniques used in multi-agent systems, and adapt them to our system designed for verification of dialogue games with emotional reasoning. We consulted the psychological aspect of such a reasoning with a group of psychologists, and our current model of emotions is an effect of this collaboration. We introduced a description language that meets our needs of describing rules of the dialogue game with emotional reasoning - Game with Emotional Reasoning Description Language (GERDL). Our aim is to create a universal tool, which allows verification and simulation of different dialogue systems.

The paper is structured as follows. In Sec. 2 we present our background and inspirations. Section 3 describes interpreted system for dialogue games with emotional reasoning, Kripke model as a basis for the application of model checking. In Sec. 4 we present a short description of the Game with Emotional Reasoning Description Language (GERDL), a few information about the language used for defining properties to be verified and some details about future work. Conclusions are given in Sec. 5.

2 Inspirations

Our dialogue systems verification system is inspired by existing concepts, such as dialogue systems, multi-agent systems, and model checking.

In dialogue games, a dialogue is treated as some kind of a game played between two parties. A dialogue game can take place both between artificial agents or between the man and the machine. In argumentative dialogues, players can perform actions affecting commitments. In our system they can also affect their emotions. We consider model designed for human-computer communication, which is based on strict rules (same as [2, 6]) and is inspired by Prakken's argumentative dialogue games [9, 16].

The dialogue game is usually specified by three basic categories of rules. *Locution rules* define a set of locutions (actions, speech acts) the player is allowed to utter during the game. Locutions express communication intentions of interlocutors. Such rules specify for example that player can claim, argue, justify, question, concede something etc.

The *structural rules* specify available responses for each specific locution. For example, after one interlocutor claims something, the other one can concede it (by performing *concede*), claim the opposite (by performing *claim* with the opposite content) or ask for justification (by performing *why*).

The *effect rules* defines effects of actions. Due to performing some action a set of commitments (public declarations) of the player can change. The result of an action is a change in the commitments set of the player.

To define a dialogue game, we must specify these three sets of rules, which allow to determine available moves for each player at every point of the dialogue. We want to verify some properties of the dialogue games by the means of model checking method.

Main approaches in this matter combine bounded model checking (BMC) with symbolic verification using ordered binary decision diagrams (BDDs) [5] or propositional logic (SAT) [14].

One of our inspirations is MCMAS - the Model Checker for Multi-Agent Systems [12, 17]. Verification is very important for multi-agent systems, which are intended to capture complex properties of large, distributed, autonomous systems. MCMAS uses the Interpreted Systems Programming Language (ISPL) dedicated to MAS characterization.

Given a MAS specification, MCMAS verifies a set of specified formulae. The mentioned specification is provided using a dedicated language (ISPL), which allows to describe agents and their behaviour using variables and Boolean expressions. The evaluation of formulae uses algorithms based on Ordered Binary Decision Diagrams (OBDDs). MCMAS shows witnesses for true formulae and counterexamples for false formulae. It provides many modalities e.g. CTL operators, epistemic operators, and so on. In ISPL, there are two kinds of agents: standard agents, and the optional environment agent, which represents common properties of the system. Each agent is characterised by: a set of local states, a set of actions, a protocol describing which actions can be performed by an agent in a given local state, and an evolution function, describing change of the local state of the agent according to the current local state and other agents actions.

The main reason ISPL was not sufficient in our case is the domain we are considering. Probably the most important difference between muti-agent systems and dialogue games is a need for referring to the history of the dialogue. In dialogue games, the history of moves is often crucial for the decision about a current move. Our rules are based not only on the last/next action of the agent but potentially on the whole history of the dialogue between players.

Thanks to adding the possibility to specify our own sets of specific types, we can base some decision (e.g. about current move or change in variables) on the content of such a set, and indirectly on the history of the dialogue. We provided two additional variable types (Action and Player). In ISPL we cannot refer to the other agent's variables, there is a special agent called Environment to convey the communication between agents. In our solution there is no such a need, the players can make some of their variables directly visible for the other players. While maintaining the general structure of the ISPL input file and most of the operators, we extended ISPL to suit our needs of expressing emotions and players' preferences.

Our second inspiration was the Dialogue Game Description Language (DGDL), and also its refined version - DGDL+, which is a language for describing dialectical games [20]. Dialectical games are one of the kinds of multi-player argumentative dialogue games, which provide argumentative behaviours. The DGDL determines whether a game description is syntactically correct. This language was created for describing the features of some commonly used dialectical games and also to be able to describe other games. DGDL takes into account the most important aspects of dialogue games: moves per turn, turn organization, dialogue magnitude, move types, move content, openers, stores, store contents, store visibility, move legality, move effects, participants, roles, and rules.

This language was created to play the specified game using Dialogue Game Execution Platform and it does not allow to specify or verify its properties. Despite large possibilities, it does not suit our goals, since we need the possibility of expressing properties of the game and verifying them by the means of the model checking and we are interested not only in the dialogue game simulation. We would be able just to describe players' emotions but we would not be able to verify its emotion-based properties.

Since we wanted to extend an existing approach, not to create a new one, we decided to enrich the framework, which was created for verification in the first place, MCMAS. Certainly, we derive from the both systems and DGDL is still our object of interest because of the great flexibility in describing dialogue games.

3 Model

We start out by defining a mathematical model for argumentation dialogue games, which uses the concept of interpreted systems and Kripke structures. Obtained Kripke structure and model checking techniques allow us to perform automatic verification of dialogue protocols. In this model formulas of a modal logic adequate to express properties of dialogues are interpreted. This is an example of the dialogue game, which can be realized and verified by our tool. The tool itself is supposed to be universal and take any dialogue game described in GERDL as an input.

First, we assume that the set of players of a dialogue game consists of two players: White (W) and Black (B), $Pl = \{W, B\}$. To each player $p \in Pl$, we assign a set of possible local states L_p and a set of actions Act_p. Player's local state $l_p \in L_p$ consists of the player's *commitments* and *emotions*, $l_p = (C_p, E_p)$.

Players' commitments are elements of a fixed topic language, which allows expressing the content of locutions. They are understood as public declarations of players but we do not assume their honesty and truthfulness. Thus, C_p are sets of such expressions. These sets may be subject to change after a player's action. Formally we assume a finite set $FORM$ of expressions which can be used as a content of a locution and thereby express some commitment of a player. We do not assume that this set is closed under logical or material implication and it can contain conflict expressions.

Emotions which we consider are *fear*, *disgust*, *joy*, *sadness*, and *anger*, and their strength (intensity) is represented by natural numbers from the set $\{1, 2, \ldots, 10\}$. Thus, E_p is a 5-tuple consisting of five values, which may also change after a certain action.

Every action from Act_p can influence participant's commitments and emotions. Every action (except null action) is synonymous with locution expressed by the specific player. The most commonly used locutions are e.g. *claim* - some statement, *concede* - confirmation, *since* - justification, *why* - the request for justification, $retract$ - revocation, and $question$ about some fact. We extended thislist by adding two, which we believe allow better describe dialogues and changes in the intensity of players' emotions. These are *scold* and *nod*. They express reprimand and approval, respectively. Results of locutions are determined by *evolution function*.

Next, Act denotes a subset of the Cartesian product of the players' actions and the global action $a \in Act$ is a pair of actions $a = (a_W, a_B)$, where $a_W \in Act_W$, $a_B \in Act_B$ and at least one of these actions is the empty action. This means that players cannot speak at the same time and reply to his own moves. Thus, the empty action is performed alternately by players W and B.

Also, we need to order performed global actions and indicate which actions correspond with which ones and therefore we define *double-numbered global actions* set $Num_2Act = \mathbb{N} \times \mathbb{N} \times Act$. and *numbered global actions* set $Num_1Act = \mathbb{N} \times Act$.

A global state g is a triple consisting of dialogue history and players' local states corresponding to a snapshot of the system at a given time point $g = (d(g), l_W(g), l_B(g))$, $g \in G$ where G is the set of global states. Given a global state g, we denote by $d(g)$ a sequence of moves executed on a way to state g and by $l_p(g)$ - the local state of player p in g.

An *interpreted system* for a dialogue game is a tuple $IS = (I, \{L_p, Act_p\}_{p \in Pl})$ where $I \subseteq G$ is the set of initial global states.

We defined also *legal answer function* $F_{LA} : Num_2Act \rightarrow 2^{Num_1Act}$, which maps a double-numbered action to the set of possible numbered actions. This function is symmetrical for both players and determines for every action a set of legal actions which can be performed next.

The actions executed by players are selected according to a *protocol function* $Pr : G \rightarrow 2^{Num_2Act}$, which maps a global state g to the set of possible double-numbered global actions. The protocol is a crucial element of the model since it gives strict rules which determine the behaviour of players.

To express how locutions and their contents affect players' emotions during the dialogue we define a function, which determines the change of intensity of emotions: $EMOT_p : Act_p \times Emotion_p \rightarrow Emotion_p$ where $p \in Pl$ and $Emotion_p$ is a set of all possible 5-tuples for emotions, i.e., $Emotion_p = \{(n_1, \ldots, n_5) : n_i \in \{1, \ldots, 10\} \wedge i \in \{1, \ldots, 5\}\}$.

Finally, we define *global* (partial) *evolution function* $t : G \times Num_2Act \rightarrow G$, which determines results of actions. This function is symmetrical for both players and defines results of actions.

The details of the legal answer function, protocol function and global evolution function were presented in our earlier papers.

The application of the model checking requires a *model* of the system under consideration. We associate with the given interpreted system a *Kripke structure*, that is the basis for the application of model checking. A Kripke structure is defined as a tuple $M = (G, Act, T, I)$ consisting of a set of global states G, a set of actions Act (in our approach Num_2Act), a set of initial states $I \subseteq G$, a transition relation $T \subseteq G \times Act \times G$ such that T is left-total. Relation T is defined as follows $(g, a, g') \in T$ *iff* $g' \in t(g, a)$. By T^* we will denote the relation $T^* \subseteq G \times G$ defined as follows: $(g', g) \in T^*$ if there exist $a_1, a_2, \ldots, a_n \in Num_2Act$ and there exists $g_1, g_2, \ldots, g_{n-1}$ such that $(g', a_1, g_1) \in T$, $(g_{n-1}, a_n, g) \in T$, and for $i = 1, 2, \ldots, n-2$ it holds that $(g_i, a_{i+1}, g_{i+1}) \in T$.

Also, we need to formulate *properties* of the dialogue protocol to be checked, we present a suitable language in the next section.

4 Description language and future work

We introduced the Game with Emotional Reasoning Description Language (GERDL) [18] designed to describe a dialogue game on the input of the framework and perform a model checking of the properties of this dialogue game. A dialogue game is specified by a set of players and some attributes of the game background (e.g. turns' rules, commitments). In the mentioned language, we specify the players by declaring the locutions available to this player, his public and private variables and protocol and evolution function. We also describe allowed commitments (public declarations) of players and which states are initial. As last, we specify properties of the dialogue game to be verified.

We assume that the locutions are speech acts uttered during a dialogue. Players' commitments are elements of a fixed topic language, which allows expressing the content of locutions. The commitments are public and each player is aware of other player's utterances. For each player, there are boolean variables representing all possible commitments. If the player has committed to something, then the corresponding variable is true, otherwise - it is false. We also presented some notations in GERDL, which allow to describe relations between contents of the consecutive locutions.

In our system, we have six types of variables: boolean, enumeration, bounded integer, set, Action and Player (the last two with additional attributes). In the dialogue game specification, some predefined variables are available which simplify the system description.

Specification of the protocol function is composed of the conditions, which are Boolean formulae over global states, and after each condition, there is a list of actions which can be used, if a condition is fulfilled. If there are many matching rules, actions from all of them are considered available.

An evolution function description consists of a set of assignments of variables (accessible for the specific player) and an enabling condition, which is a Boolean formula over variables of the player, public variables of the other players and actions of all players. Some of these rules can be interpreted as properties of the dialogue game itself and the other ones rather as elements of player's profile.

To specify the properties of dialogue systems to be verified, we express them in language based on CTL logic introduced by Emerson and Clarke [4] enriched with commitment and emotion components (presented at [7]). Usually, properties we want to verify are used to reason about desirable behaviour of the system, e.g. the formula $\mathrm{AG}(COM_p(\alpha) \Rightarrow \mathrm{E}(true\ \mathrm{U} \neg COM_p(\alpha)))$ expresses that even if a player p has committed to α at some point, then during the dialogue he can change his mind and retract this commitment. Formula $COM_p(\varphi)$ expresses that φ is in the set of commitments of player p (φ is not a formula of this language, but a part of a separate structure in which we express the uttered sentences), the formula $\mathrm{E}(\alpha \mathrm{U} \beta)$ means that on some path β eventually occurs and that α holds continuously until then, and the formula $\mathrm{AG}(\alpha)$ means that "for all states on all paths" α holds.

Semantics of the above logic is given by means of interpreted systems. By a *computation* in a Kripke structure $M = (G, Act, T, I)$ we understand a possibly infinite sequence of states $\pi = (g_0, g_1, \ldots)$ such that there exists an action a_m for which $(g_m, a_m, g_{m+1}) \in T$ for each $m \in \mathbb{N}$, i.e., g_{m+1} is the result of applying the transition relation T to the global state g_m, and the action a_m.

In the interpreted systems terminology, a computation is a part of a run. A k-computation is a computation of length k. For a computation $\pi = (g_0, g_1, \ldots)$, let $\pi(k) = g_k$, and $\pi_k = (g_0, \ldots, g_k)$, for each $k \in \mathbb{N}$. By $\Pi(g)$ we denote the set of all the infinite computations starting at g in M, whereas by $\Pi_k(g)$ the set of all the k-computations starting at g.

In our earlier paper [18], we specified the GERDL input file of the specific dialogue game. The protocol description (corresponding to protocol function in our model) is used to reflect the rules of the given type of dialogues. The sample file presented there reflects the Prakken-inspired dialogue game. In our system, we assume that in order to verify dialogue game properties the designer should define the specification according to some existing or created dialogue game. Since our goal is the verification of some emotion-based properties, the designer's work should be based on the cooperation with the psychologists. Their help can be invaluable both at the protocol definition stage and at the property design stage. The protocol design is domain-dependent, e.g. there will be a lot of differences between parent-child and employer-employee dialogues.

The possible found counterexamples are dialogues compliant with the given protocol, but not having the verified property.

For each player, we declared variables responsible for modeling emotions. Since our modeled dialogue game focuses not only on the commitments of the players but also on their emotions, some of the rules of evolution functions represent properties of the dialogue game itself (e.g. changes in the sets of commitments) and the other ones can be understood as elements of player's emotional profile (e.g. changes in the emotions levels of the specific player). In contrast to the presented in [18] sample file, actual files can be much larger since the rules of evolution function of each player expressing his specific behaviour are usually more numerous in order to prevent too many possible dialogues.

Next step of our work on dialogue games verification is focused on the design and the implementation of model checker that will use presented in this paper GERDL file as the input file and verify whether specified properties are true. The first step of this task is to encode local and global states as well as actions as quantifier-free first-order logic formulae. The same kind formula symbolically encodes also the transition relation. Also, it seems that the verification method will be different, MCMAS uses OBDDs and we plan to use SMT.

5 Conclusion

The main goal of our work is the specification and semantic verification of protocols for dialogue games with a fixed protocol. It can highlight some of the mechanisms that exist in dialogues, and which have their grounds in our emotions.

The GERDL is essential to describe the verified dialogue game on the input of our framework, which will be focused on model checking of the properties of this dialogue game. We can describe the most important features of dialogue games we wanted to capture in our specification. To represent the verified properties of the protocol we used the extension of CTL logic with commitment and emotion modalities (formulated in [7]). Our talk will be devoted to our concepts of the Boolean encodings of the transition relation and the problems encountered during this process. We want to discuss available verification methods, their advantages and disadvantages for our case. We belive that the results of the model checking of dialogue games will show how important is the role of emotions in the argumentative discourse.

Acknowledgment. The research by Kacprzak was supported by a grant S/WI/1/2014 from Bialystok University of Technology and founded from the resources for research by Ministry of Science and Higher Education.

References

1. Ch. Baier and J.P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
2. K. Budzynska, M. Kacprzak, A. Sawicka, and O. Yaskorska. *Dialogue Dynamics: Formal Approach*. IFS PAS, 2015.
3. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, 2001.
4. E. A. Emerson and E. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.
5. A. V. Jones and A. Lomuscio. Distributed BDD-based BMC for the verification of multi-agent systems. In W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen, editors, *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, 2010, Volume 1-3*, pages 675–682. IFAAMAS, 2010.
6. M. Kacprzak, M. Dziubinski, and K. Budzynska. Strategies in dialogues: A game-theoretic approach. In S. Parsons, N. Oren, C. Reed, and F. Cerutti, editors, *Computational Models of Argument - Proc. of COMMA 2014, Scottish Highlands, UK, 2014*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 333–344. IOS Press, 2014.
7. M. Kacprzak, K. Rzenca, A. Sawicka, A. Zbrzezny, and K. Zukowska. A formal model of an argumentative dialogue in the management of emotions, Poznan Reasoning Week, L&C 2016/14th ArgDiap/QuestPro 2016 Abstracts. http://poznanreasoningweek.files.wordpress.com/2016/09/prw2016abstracts.pdf, 2016. [Online; accessed 05-03-2017].
8. M. Kacprzak and A. Sawicka. Identification of formal fallacies in a natural dialogue. *Fundam. Inform.*, 135(4):403–417, 2014.
9. M. Kacprzak, A. Sawicka, and A. Zbrzezny. Dialogue systems: Modeling and prediction of their dynamics. In A. Abraham, K. Wegrzyn-Wolska, A. Ella Hassanien, V. Snásel, and A. M. Alimi, editors, *Proc. of AECIA 2015, Villejuif (Paris-sud), France, 2015*, volume 427 of *Advances in Intelligent Systems and Computing*, pages 421–431. Springer, 2015.
10. M. Kacprzak, A. Sawicka, and A. Zbrzezny. Towards model checking argumentative dialogues with emotional reasoning (extended abstract). In *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming*, pages 257–268, 2016.
11. M. Kacprzak and O. Yaskorska. Dialogue protocols for formal fallacies. *Argumentation*, 28(3):349–369, 2014.
12. A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 19(1):9–30, 2017.
13. A. Meski, W. Penczek, M. Szreter, B. Wozna-Szczesniak, and A. Zbrzezny. BDD-versus SAT-based bounded model checking for the existential fragment of linear temporal logic with knowledge: algorithms and their performance. *Autonomous Agents and Multi-Agent Systems*, 28(4):558–604, 2014.

14. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundam. Inform.*, 55(2):167–185, 2003.
15. W. Penczek, B. Wozna-Szczesniak, and A. Zbrzezny. Towards sat-based BMC for LTLK over interleaved interpreted systems. *Fundam. Inform.*, 119(3-4):373–392, 2012.
16. H. Prakken. Models of persuasion dialogue. In *Argumentation in AI*, pages 281–300. Springer, 2009.
17. F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. *Journal of Applied Logic*, 5(2):235 – 251, 2007. Logic-Based Agent Verification.
18. Anna Sawicka, Magdalena Kacprzak, and Andrzej Zbrzezny. A novel description language for two-agent dialogue games. In Lech Polkowski, Yiyu Yao, Piotr Artiemjew, Davide Ciucci, Dun Liu, Dominik Slezak, and Beata Zielosko, editors, *Rough Sets - International Joint Conference, IJCRS 2017, Olsztyn, Poland, July 3-7, 2017, Proceedings, Part II*, volume 10314 of *Lecture Notes in Computer Science*, pages 466–486. Springer, 2017.
19. J. Visser, F. Bex, C. Reed, and B. Garssen. Correspondence between the pragma-dialectical discussion model and the argument interchange format. *Studies in Logic, Grammar and Rhetoric*, 23(36):189–224, 2011.
20. S. Wells and C. A. Reed. A domain specific language for describing diverse systems of dialogue. *J. Applied Logic*, 10(4):309–329, 2012.
21. O. Yaskorska, K. Budzynska, and M. Kacprzak. Proving propositional tautologies in a natural dialogue. *Fundam. Inform.*, 128(1-2):239–253, 2013.
22. A. M. Zbrzezny, B. Wozna-Szczesniak, and A. Zbrzezny. Smt-based bounded model checking for weighted epistemic ECTL. In F. C. Pereira, P. Machado, E. Costa, and A. Cardoso, editors, *Progress in Artificial Intelligence - 17th EPIA 2015, Coimbra, Portugal, 2015. Proc.*, volume 9273 of *Lecture Notes in Computer Science*, pages 651–657. Springer, 2015.
23. A. M. Zbrzezny, A. Zbrzezny, and F. Raimondi. Efficient model checking timed and weighted interpreted systems using SMT and SAT solvers. In *Agent and Multi-Agent Systems: Technology and Applications, 10th KES International Conference, KES-AMSTA 2016, Puerto de la Cruz, Tenerife, Spain, 2016, Proceedings*, pages 45–55, 2016.