Improved EGT-based Robustness Analysis of Negotiation Strategies in Multi-agent Systems via Model Checking

Songzheng Song*, Jianye Hao‡¶, Yang Liu*, Jun Sun‡, Ho-Fung Leung† and Jie Zhang*
*Nanyang Technological University Email: {songsz, yangliu, zhangj}@ntu.edu.sg
†The Chinese University of Hong Kong Email: lhf@cse.cuhk.edu.hk
‡Singapore University of Technology and Design Email: {jianye_hao,sunjun}@sutd.edu.sg

***Abstract*—Automated negotiations play an important role in various domains modeled as multi-agent systems where agents represent human users and adopt different negotiation strategies. Generally, given a multi-agent system, a negotiation strategy should be *robust* in the sense that most agents in the system have the incentive to choose it rather than other strategies. Empirical Game Theoretic (EGT) analysis is a game-theoretic analysis approach to investigate the robustness of different strategies based on a set of empirical results. In this work, we propose that model checking techniques can be adopted to improve EGT analysis for negotiation strategies. The dynamics of strategy profiles can be modeled as a Labeled Transition System using the counter abstraction technique. We define *single-agent best deviation* to represent the strategy deviations during negotiation, which focuses on each agent's best deviation benefit and is different from *best single-agent deviation* used in previous work. Two interesting properties in EGT analysis, *empirical pure strategy Nash equilibrium* and *best reply cycle*, are automatically verified to investigate the robustness of different strategies. For demonstration, the top six strategies from the Automated Negotiating Agents Competition (ANAC) 2010-2012 are studied in terms of their robustness performance. In addition to identifying the most robust strategies, we supply complete rankings among them in different settings. We show that model checking is applicable and efficient to perform robustness analysis of negotiation strategies.**

***Index Terms*—Automated Negotiation; Robustness Analysis; Empirical Game Theory; Model Checking;**

I. Introduction

Negotiations exist in many aspects of our daily life for resolving conflicts among different parties. In multi-agent systems, automated negotiation techniques can facilitate better negotiation outcomes by supporting humans when they are faced with complex negotiations. Several automated negotiation strategies have been proposed in different negotiation scenarios [11], [17], [28].

The most commonly adopted criterion for evaluating a negotiation strategy is its *efficiency*, i.e., the average payoff it can obtain under different negotiation scenarios against other negotiation strategies. One example is the annual *automated negotiating agents competition (ANAC)* [1], [2] which provides a general platform enabling different negotiation agents to be evaluated against a range of opponents under various realistic negotiation environments. The *efficiency* criterion in the competition corresponds to the expected payoff under the competition tournament over all participating agents averaged over all negotiation domains. However, the *efficiency* criterion does not address the *robustness* of the negotiation strategies in different negotiation scenarios, since it assumes that each agent's strategy is fixed (determined by its designer(s)) once it enters the competition. In practice, agents are free to choose any negotiation strategy available to them (e.g., from the public strategies developed in the ANAC). Besides, the human negotiators are able to observe their opponents' negotiation strategies, and may change strategies at any negotiation stage to improve their personal benefits. Given a strategy s, it is important to investigate whether a (human) negotiator adopting s has the incentive to unilaterally deviate to other available strategies under a particular negotiation tournament. Similarly, we may ask whether any (human) negotiator adopting other strategies is willing to switch to s under certain important negotiation domains. Because of its importance, the *robustness* criterion has been given attention [1], [35]. The essence of the robustness analysis is to apply game-theoretic analysis to investigate the dynamics of strategy changes of (rational) human negotiators during negotiation when they interact with different opponents.

However, the *robustness* of different negotiation strategies cannot be analyzed by applying standard game-theoretic approaches as there are an infinite number of possible negotiation strategies. One suitable approach is to adopt empirical game theoretic (EGT) analysis [1], [23], [35] which assumes that each agent only selects its strategy from a fixed set of strategies. Given a negotiation tournament consisting of n agents, each agent chooses one particular strategy from the set $\mathcal{S}$ of strategies, and this jointly constitutes a *strategy profile*. The outcomes for each strategy profile can be determined through simulation. However, the robustness analysis approach based on the EGT has two drawbacks. First, it assumes *best single-agent deviation*, i.e., only one agent with the largest payoff increase is allowed to change its strategy. In practice, each negotiating agent may change its strategy as long as its own benefit can be increased. Allowing all agents to update their strategies rationally can provide a more realistic analysis. Second, manual deviation analysis is highly nontrivial when

¶ Corresponding author

the number of agents/strategies is large since the number of strategy profiles increases exponentially with the number of agents/strategies.

Applying model checking [3], [10] for EGT analysis could be useful if the following challenges can be addressed:

- Accurate formal models are needed to represent the deviations of strategy profiles. Our assumption of allowing all agents to update their strategies in one strategy profile may significantly increase the analysis complexity.
- Dedicated model checking algorithms are needed for robustness analysis, since no such algorithms exist.
- The existence of multiple agents and strategies in the system can easily result in the state space explosion problem, which makes the verification infeasible, so an appropriate state space reduction technique is required.

In this work, we tackle the above challenges. First, we use the model checker PAT [29], which supports external libraries of imperative programming languages such as Java and C#, to model the complex dynamics of agents' possible strategy deviations during negotiations. Second, corresponding model checking algorithms are developed to exactly cover the properties needed for the *robustness* analysis. Third, we apply the counter abstraction technique [25] to reduce the state space of the models due to the symmetric property of negotiation problems. We apply the model checking approach to analyze the *robustness* of the top-eight strategies from ANAC 2012 and show that our approach can facilitate the robustness analysis process.

Compared with [1], [35], our contributions are fourfold:

1) We define the deviations of each strategy profiles based on each agent's benefits, which is more realistic than the definition of deviation used in [1] since each agent has its own incentive to get a better payoff, which is independent on other agents' interests.
2) We propose and apply a model checking approach to improve the robustness analysis via EGT, and generate the robustness ranking of different strategies. This approach could advance the strategy analysis and development for automated negotiation.
3) We apply the counter abstraction technique to reduce the model's state space due to the symmetric property existing in the negotiation, thus making analysis using model checking techniques feasible and efficient.
4) We have implemented the dedicated verification algorithms in PAT. We show the *robustness* of the state-of-the-art negotiation strategies, and rankings among them are given under different settings.

The paper is structured as follows. Section II surveys related work. Section III presents background. Section IV introduces the details of how to apply model checking to analyze the *robustness* of different negotiation strategies. An evaluation is reported in Section V and Section VI concludes the paper.

II. Related Work

We first review the application of model checking techniques to verify properties in multiagent systems.

Tadjouddine et al. [31] investigate the problem of automatically verifying the game-theoretic property of strategy-proofness for auction protocols. They consider the case of the Vickrey auction protocol and check the property of strategy-proofness using the model checker SPIN [20]. To solve the state space explosion problem, they apply two types of abstraction approaches: program slicing and abstract interpretation. Program slicing removes portions in the model irrelevant with respect to the property checked. Abstract interpretation maps the original strategy domain onto an abstract and less complex domain, and then performs model checking on the abstract model. By using these two methods, the authors show that strategy-proofness of a Vickrey auction can be automatically verified for any number of players.

Ballarini et al. [4] apply probabilistic model checking to automatically analyse the uncertainty existing in a two-agent negotiation game. In the game, one seller and one buyer bargain over a single item, and both players exhibit probabilistic behaviors based on the opponent's previous behavior. They model the dynamics of the two-player system as a discrete-time Markov chain (DTMC). They illustrate how to use the probabilistic model checker PRISM [18] to automatically analyse the probability that the players reach an agreement within each round of the game by specifying this property in probabilistic computation tree logic (PCTL) [13].

Hao et al. [15] apply probabilistic model checking techniques to analyze stochastic dynamics among multiple learners. They focus on two strategies: *basic simple strategy* (BSS) and *extended simple strategies* (ESS), in the context of dispersion games. The system dynamics are modeled using DTMC and they propose to reduce models' state space by applying the counter abstraction technique. Two properties of the systems are considered: convergence and convergence rate. They show that these properties can be automatically verified using probabilistic model checking techniques.

Bordini et al. [5] review the problem of verifying multi-agent systems implemented in the language AgentSpeak using model checking techniques. They aim to automatically verify whether certain specifications are satisfied using existing model checkers. For this purpose, the original multi-agent system implemented in a BDI language AgentSpeak [27] is transformed into the formal language supported by current model checkers. They introduce a variant of the language AgentSpeak, AgentSpeak(F), which can be automatically transformed into Promela, the model specification language of SPIN [20]. They adopt a simplified form of BDI logic to specify the properties to be checked, which can be transformed into linear temporal logic (LTL), supported by existing model checkers. With the combination of these two techniques, the properties of a multi-agent system implemented in AgentSpeak can be automatically checked with existing model checkers. [36] transforms other agent-based languages such as Mable [36] into Promela and uses SPIN to perform model checking.

Several model checking approaches [9], [12], [24], [33] have been proposed to analyze different types of games and characterize their corresponding solution concepts. Góngora and Rosenblueth [12] propose a discrete-time Markov chain codification of a finite strategic game and an extension of probabilistic Computational-Tree Logic (PCTL) to quantify the expected cost. They characterize mixed strategy Nash equilibrium in finite strategic games. Mari et al. [24] propose two ordered binary design diagram (OBDD) based model checking algorithms for verifying Coalition Nash equilibrium and apply them to multiple administrative domains distributed systems. Vasconcelos and Haveusler [33] propose a first-order CTL based logic, Game Analysis Logic (GAL) to reason about extensive-form games with perfect information. They also develop a GAL-based model checker to compute the solution concept of Nash equilibrium and subgame perfect equilibrium. Chen et al. [9] develop a model checker, PRISM-games for stochastic games, based on the probabilistic model checker-PRISM. In PRISM-games, stochastic games are modeled in a probabilistic extension of the Reactive Module Language and the properties are specified in an extension of the well-known logic ATL. This model checker supports verifying properties in stochastic games and automatic synthesis of optimal strategies.

In this work, we apply a model checking approach to provide automatic analysis of the robustness of different negotiation strategies. First, apart from checking empirical pure strategy Nash equilibrium, analyzing robustness of strategies relies on checking *best reply cycle* and *basin of attraction*, which are different from analyzing traditional solution concepts such as Nash equilibrium. Second, the modeling of deviation of negotiation strategies involves complex real number operations and loops. Under the support of the external libraries based on the C# language, both complex real number operations and loops, not supported in other model checkers such as SPIN and PRISM, can be modeled using PAT.

III. Background

In this section, we recall some basic concepts and background used throughout the rest of the paper.

A. Negotiation Model

The ANAC is the annual competition which brings together researchers from the automated negotiation community [1], [2]. Following the settings adopted in ANAC, the basic negotiation form is bilateral negotiation, i.e., negotiations between two agents. The alternating-offers protocol is adopted to regulate the interactions between the negotiating agents, in which the agents take turns to exchange proposals. For each negotiation scenario, both agents can negotiate over multiple issues (items), and each item can have a number of different values. Let us denote the set of items as $\mathcal{M}$, and the set of values for each item $m_i \in \mathcal{M}$ as $\mathcal{V}_i$.[1] We define a negotiation outcome ω as a mapping from every item $m_i \in \mathcal{M}$ to a value $v \in \mathcal{V}_i$, and the negotiation domain is defined as the set Ω of all possible negotiation outcomes. For each negotiation outcome ω, we use $\omega(m_i)$ to denote the corresponding value of the item m_i in the negotiation outcome ω. It is assumed that the knowledge of the negotiation domain is known to both agents beforehand, and is not changed during the whole negotiation session.

For each negotiation outcome ω, different agents may have different preferences. Each agent i's preference can be modeled by a utility function u_i such that $\forall \omega \in \Omega$, it is mapped into a real-valued number in the range of [0,1], i.e., $u_i(\omega) \in [0,1]$. In practice, it is usually associated with certain cost in each negotiation. To account for cost, a real-time deadline is imposed on the negotiation process and each agent's actual utilities over the negotiation outcomes are decreased by a discounting factor δ over time. In the ANAC, each negotiation session is allocated 3 minutes, which is normalized into the range of [0,1], i.e., $0 \leq t \leq 1$. Formally, if an agreement is reached at time t before the deadline, each agent i's actual utility function $U_i^t(\omega)$ over this mutually agreed negotiation outcome ω is defined as follows:

$$U_i^t(\omega) = u_i(\omega)\delta^t \tag{1}$$

If no agreement is reached by the deadline, each agent i will obtain a utility of $ru_i^0\delta$, where ru_i^0 is agent i's private reservation value in the negotiation scenario. The agents will also obtain their corresponding reservation values if the negotiation is terminated before the deadline. Note that the agents' actual utilities over their reservation values are also discounted by the discounting factor δ over time t. We assume that the agents' preference information and their reservation values are private and cannot be accessed by their negotiating partners.

For example, consider a negotiation between the representatives from two companies (A and B) where company A wants to acquire company B. The companies negotiate five items: the price that company A pays to company B, the transfer of any intellectual property from B to A, any stock given to the founders in B, the terms of the employees' contract and the legal liability company A has to take. For each issue, there are a number of options available for both companies, and different companies may have different preferences for different options. For example, company A prefers to pay the minimum amount of money and stock to acquire company B, while B prefers the acquisition price to be as high as possible and a higher percentage of stock. For the issue of legal liability, both sides prefer the least legal liability. Any possible combination of the options for each issue can be considered as a proposal that any company can propose during the negotiation. One specific proposal could be that company A proposes to pay \$100,000.00 and give 5% of the stock to company B's founders, all intellectual property must be transferred to company A, the employees should have a 15% salary increase in the contract, and company A assumes all legal liabilities. Different companies might put different weights on different issues when they evaluate the outcomes, which are formally reflected in their utility functions.

The interaction between the negotiation agents is regulated by the alternating-offers protocol, in which the agents are allowed to take turns to exchange proposals. During each encounter, if it is agent i's turn to make a proposal, it can make a choice from the following three options:

- *Accept the offer from its negotiating partner*: In this case, the negotiation ends and an agreement is reached. Both agents will obtain the corresponding utilities according to Equation 1, where ω is the negotiation outcome to which they mutually agree.

- *Reject and propose a counter-offer to its negotiating partner*: In this case, the negotiation process continues and it is its negotiating partner's turn to make a counter-proposal provided that the deadline has not passed.
- *Terminate the negotiation*: In this case, the negotiation terminates and each agent i gets its corresponding utility based on its private reservation value with the initial value of ru_i^0. Note that their actual utilities are also decreased over time by the same discounting factor δ, i.e., $U_i^t = ru_i^0 * \delta^t$. The reservation value may be different for different negotiation parties and may vary in different negotiation domains.

The negotiation process terminates when any of the following conditions is satisfied: 1) the deadline is reached (*End*); 2) an agent chooses to terminate the negotiation before reaching deadline (*Terminate*); 3) an agent chooses to accept the negotiation outcome proposed by its negotiating partner (*Accept*).

B. Robustness Analysis using Empirical Game Theoretic Approach

Under the ANAC, the winner is the agent who receives the highest average payoffs in the specific competition tournament where each participation agent adopts a different strategy designed by different parties. According to the ANAC rule, once entering the competition, each agent's strategy cannot be changed during the tournaments, and the agents do not have the learning capability to adjust their negotiating strategy based on previous negotiations (each pairwise negotiation is a new start). However, in practice, agents are free to choose any strategy available, and it is important to analyze the *robustness* of different strategies. For example, it is interesting to investigate which strategy most agents would have the incentive to adopt if the agents are free to choose any strategy.

In robustness analysis of negotiation strategies, the standard game-theoretic approach is not applicable because it explicitly considers all possible strategies and there are an infinite number to consider. Empirical Game Theoretic (EGT) analysis [23] is a game-theoretic analysis approach based on a set of empirical results and can be used to investigate the robustness of the strategies. EGT assumes that each agent only selects its strategy from a fixed set of strategies and the outcomes for each strategy profile can be determined through simulation. This technique has been successfully applied in addressing questions about robustness of different strategies in previous years' trading agent competitions [23] and negotiation strategies in ANAC 2011 [1].

In EGT analysis, a fixed set of negotiation strategies, S, is given. Each agent can select any strategy from S as its negotiation strategy. For each bilateral negotiation (p, p'), the corresponding payoff $U_p(p, p')$ received by agent p is determined as its average payoff over all possible domains against its current opponent p', which can be obtained through empirical simulation (available from the ANAC website). The average payoff of an agent in any given tournament is determined by averaging its payoff obtained in all bilateral negotiations against all other agents in the tournament. Specifically, for a given tournament involving a set $\mathcal{P}$ of agents, the payoff $U_p(\mathcal{P})$ obtained by agent p can be calculated as follows:

$$U_p(\mathcal{P}) = \frac{\sum_{p' \in \mathcal{P}, p' \neq p} U_p(p, p')}{\mid \mathcal{P} \mid -1} \tag{2}$$

where $U_p(p, p')$ represents the corresponding average payoff of agent p in the bilateral negotiation against agent p'. Note that agent p and p' can use either the same or different strategies. Based on Equation 2, we can determine the corresponding payoff profile for any given tournament. For each agent, there may exist multiple candidate strategies that it has the incentive to deviate to (i.e., multiple single-agent deviations exist for one agent), but here we only consider the best deviation available to that agent in terms of maximizing its deviation benefit, which is called *single-agent best deviation*. Our definition is different from *best single-agent deviation* defined in [1]. In [1], among all agents, only the one obtaining the largest payoff increase can deviate to another strategy, which is not realistic when all agents have the incentive to receive a larger payoff via updating their strategies. Therefore, one strategy profile in our setting can have multiple single-agent best deviations enabled, each of which corresponds to a unique agent.

Given a strategy profile under a negotiation tournament, if no agent has the incentive to unilaterally deviate from its current strategy, then this strategy profile is called an *empirical pure strategy Nash equilibrium*[2]. It is also possible for the agents to adopt mixed strategies and thus we can define the concept of *empirical mixed strategy Nash equilibrium* accordingly. However, in practice people are risk-averse and may like to be represented by a strategy with predictable behaviors instead of a probabilistic one [26]. Therefore we only consider empirical pure strategy Nash equilibria in our analysis following [1]. It is possible that a game may have no empirical pure strategy Nash equilibrium.

Another useful concept for analyzing the *stability* of the strategy profiles is *best reply cycle* [37], which is a subset of strategy profiles. Since we allow that one strategy profile can have multiple outgoing transitions, our best reply cycle is a subset of strategy profiles which are strongly connected instead of just a cycle. For any strategy profile within this subset, there is no single-agent best deviation path leading to any profile outside this subset. In a best reply cycle, all single-agent best deviation paths starting from any strategy profile within itself must lead to another strategy profile inside the cycle, and there must exist paths between any two strategy profiles in this subset.

Both empirical pure strategy Nash equilibrium and best reply cycle can be considered as two different interpretations of empirical stable sets to evaluate the *stability* of different strategy profiles. We evaluate the *robustness* of a strategy using the concept of *basin of attraction* of a stable set [34]. The basin of attraction of a stable set is the percentage of strategy profiles which can lead to this stable set through a series of single-agent best deviations. A negotiation strategy s is considered to be *robust* if it belongs to a stable set with a large basin of attraction [1], [34]. In other words, if there exists a large proportion of initial strategy profiles which, through a series of single-agent best deviations, can eventually lead to a stable set containing strategy s, then strategy s is highly robust in the

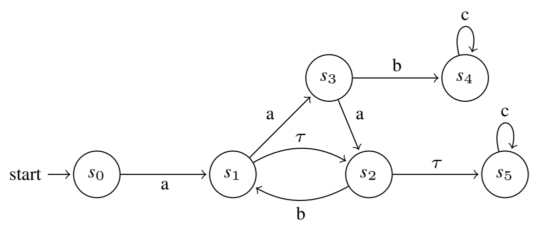


Fig. 1. Example: LTS $\mathcal{L}$

long run, since the strategy s can always have the opportunity of being adopted eventually if the tournament is sufficiently repeated due to single-agent best deviations.

In summary, we apply the EGT analysis to identify both empirical pure strategy Nash equilibrium and best reply cycle and evaluate the robustness of negotiation strategies via the concept of basin of attraction.

C. PAT

The model checker we use is PAT, [21], [22], [29], a stand-alone model checking framework, which has a user-friendly editor, simulator and verifier. It is implemented in C# and supports various operating systems such as Windows, Linux and Mac OS. The modeling language of PAT is called CSP# [30], which can be used to capture the deviations between strategy profiles in negotiations. CSP# is based on Tony Hoare's CSP [19], and it combines low-level programs, e.g., sequence programs defined in a simple imperative language or any imported libraries[3] of imperative programming languages such as Java and C#, with high-level specifications. It supports shared variables as well as abstract events, making it both state-based and event-based. The semantic model of CSP# is Labeled Transition Systems (LTS), whose formal definition is as follows.

Definition 1: A Labeled Transition System $\mathcal{L}$ is a tuple $(S, init, Act, T, AP, L)$ where S is a finite set of states; and $init \in S$ is an initial state;[4] Act is an alphabet; $T \subseteq S \times Act \times S$ is a labeled transition relation; AP is a set of atomic propositions and L: $S \to 2^{AP}$ is a labeling function.

A set of states $C \subseteq S$ is called *connected* in $\mathcal{L}$ iff $\forall s, s' \in C$, there is a finite path $\pi = \langle s_0, s_1, \cdots, s_n \rangle$ satisfying $s_0 = s \wedge s_n = s' \wedge \forall i \in [0, n], s_i \in C$. Strongly Connected Components (SCCs) are those maximal sets of states which are mutually connected. An SCC is called *trivial* if it just has one state without a self-loop. An SCC is *nontrivial* iff it is not trivial. If an SCC does not have outgoing transitions to states outside it, it is called *bottom* SCC (BSCC). Note that one state can only be in one SCC. In other words, SCCs are disjoint.

A simple LTS $\mathcal{L}$ is shown in Fig. 1. τ indicates invisible events in the system. In this figure, $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$, and s_0 is the initial state. $Act = \{a, b, c\}$ and T is denoted in the figure. For example, $(s_1, \tau, s_2) \in T$, meanwhile $(s_2, b, s_1) \in T$. There are three nontrivial SCCs: $\{s_1, s_2, s_3\}, \{s_4\}$ and $\{s_5\}$, and the latter two are BSCCs.

Based on the semantic model, PAT provides a discrete-event simulator. In simulation, system models follow the operation semantics in order to guarantee that each step reflects a meaningful execution of the system. Users could choose automatic simulation, which means the simulator will randomly execute the model and generate random states, or manual simulation, which allows users to choose the next event from the current enabled events. Through simulation, users can visually check how the model executes step by step, which is useful in system design and analysis, especially when there are undesired executions found in verification. Simulation is a complement to verification and it makes debugging more convenient.

Compared with simulation, automatic verification provides accurate results of whether a property is satisfied in a system because it performs exhaustive exploration of the state space. Compared with other exploration methods such as SAT solvers, model checkers have their formal modeling languages, which can be easily used to build accurate desired models. Also, automated verification algorithms in model checking can efficiently and completely analyze the dynamics of the targeted systems. The properties verified in this paper are well-known problems in the model checking domain; therefore it is reasonable to select model checking as the suitable technique for our analysis and verification.

We chose PAT here because compared with other well-known model checkers such as SPIN, PRISM and MCMAS, PAT has a more expressive modeling language. Through imported C# libraries, PAT can model complex data structures and algorithms, such as Equation (1) or even calculus formulae if needed. Besides, modeling the algorithm and data structure directly in other model checkers would significantly increase the state space, e.g., implementing a hashing function.

Two aspects are important in verification with a model checker. One is the properties it can support, and the other is the efficiency of the verification algorithms. In the following, we review two widely used properties in PAT, which are related to our requirements in this work.

- Deadlock Checking: in system execution, it is possible that no more actions can be executed at some state. It is meaningful to check whether there are deadlock states existing in the system, and the verification is similar to reachability checking.
- LTL Checking: sometimes, properties may need temporal information. In PAT, we choose LTL properties to capture this kind of requirement. LTL formulae in PAT can be built from not only atomic state propositions but also events so that it is called SE-LTL [7]. It is very expressive and suitable for PAT since our language is both event-based and state-based. LTL verification in PAT follows the classic automata-based approach [3].

IV. Robustness Analysis using Model Checking

In this section, we describe the modeling and verification details of the deviations of strategy profiles.

A. Modeling with Counter Abstraction

Given a set of agents denoted as N and a set of negotiation strategies denoted as S, the n-agent negotiation problem can be modeled as a strategic form game. Formally it can be represented as a tuple $\langle N, (S_i), (U_i)\rangle$ where

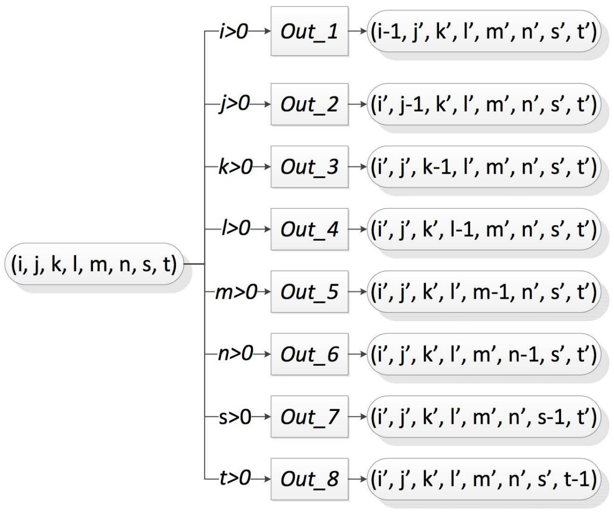
- $N = \{a_1, a_2, \ldots, a_n\}$ is the set of agents.
- S_i is the set of negotiation strategies available to a_i.
- U_i is the utility function of agent i, where $U_i(\mathcal{P})$ corresponds to the average payoff a_i receives, given the set of agents involved in the current negotiation is $\mathcal{P}$, which can be calculated according to Equation 2.

In this paper, we assume that each agent shares the same set of actions, i.e., $S_1 = S_2 = \ldots = S_n = S$. Given the negotiation strategy $s_i \in S$ of each agent (process) a_i, we denote the global state $\alpha = (v, \langle s_1, \ldots, s_n\rangle) = (v, O_i)$, which is the combination of the valuations of the global variables v^5 and the chosen strategies of all agents (or the game outcome O_i). If global variables are ignored, then each global state represents a unique strategy profile in the system. The transition relation T is built based on the single-agent best deviation, i.e., an agent may change its current strategy to another one according to the maximal deviation benefit. Thus the formal model representing the dynamics of the negotiations can be automatically constructed and is uniquely determined.

However, the state space of the model can be very large and thus hinders the efficiency of analyzing the model, due to the explosion of the combination of all agent processes' strategy choice. For example, consider the case of n agents and k strategies; without taking the global variables into consideration, the number of possible combinations of all agent processes' local states is k^n. To handle the state explosion problem, we can take advantage of the intrinsic agent-symmetric property of the negotiation game. According to Equation 2, we can see that the expected utility of an agent is obtained by averaging over all payoffs from negotiating with the rest of the agents in the tournament. Also the payoff from bilateral negotiation for each pair of agents only depends on the specific negotiation strategies of agents, which is not affected by the identities of agents. Thus each agent's utility over a particular outcome is only determined by the number of agents choosing each strategy, and is not influenced by the identities of the agents. Accordingly, in the robustness analysis, there is no need to record the specific strategy adopted by each agent, and it will not affect the overall analysis results.

As a result, we adopt the counter abstraction technique [25] to reduce the state space in our model. In model checking, counter abstraction is a special case of symmetry reduction. If a system is composed of several behaviorally similar processes, and the process ID has no effect on the behavior of the whole system, we can abstract its state space by grouping the processes based on in which local state they reside. For example, suppose there are 3 behaviorally similar processes residing in a system. Instead of saying "process 1 is in state s, process 2 is in state t and process 3 is in state s", we simply say "two processes are in state s and one process is in state t". In this way, the state space can be reduced by exploiting the state space symmetry.

In EGT analysis, the agents always choose strategies from the same strategy group and the identities are not important; therefore this abstraction technique can be applied. We only need to consider how many agents choose each strategy at the same time. Previously, given two states in which the numbers of agents choosing each strategy are the same, but the identities of the agents choosing the same strategy are different, they are defined as different states. Now they can be merged into a single state. For example, consider an EGT analysis with 5 agents and 3 strategies and two possible global states $s = (v, \langle s_1, s_1, s_2, s_2, s_3\rangle)$ and $s' = (v, \langle s_1, s_2, s_2, s_3, s_1\rangle)$. We only need to keep track of the number of agents choosing each strategy, i.e., we have $f(s_1) = 2, f(s_2) = 2, f(s_3) = 1$, where $f(s)$ records the number of agents choosing strategy s, and thus the two original global states are reduced to a single state (v, f). Thus, we can reduce the state space of multi-agent negotiation models.



(i, j, k, l, m, n, s, t)
i>0
Out_1
(i-1, j', k', l', m', n', s', t')
j>0
Out_2
(i', j-1, k', l', m', n', s', t')
k>0
Out_3
(i', j', k-1, l', m', n', s', t')
l>0
Out_4
(i', j', k', l-1, m', n', s', t')
m>0
Out_5
(i', j', k', l', m-1, n', s', t')
n>0
Out_6
(i', j', k', l', m', n-1, s', t')
s>0
Out_7
(i', j', k', l', m', n', s-1, t')
t>0
Out_8
(i', j', k', l', m', n', s', t-1)

Fig. 2. One Step of the Negotiations

Next, we use the 8-agent 8-strategy negotiations, which is the setting adopted in the final round of the ANAC, to show the formal modeling of the system. The group of strategies is denoted as $S = \{s_1, s_2, \cdots, s_8\}$. Since the counter abstraction is used, we do not record the individual agent's strategy. Instead, states representing the overall strategies distribution between agents are defined. We take one step of the dynamic behaviors of the system as an example.

In Fig 2, the arrows indicate the direction of the state transition. (i, j, k, l, m, n, s, t) is used to represent a strategy profile in the system. i means currently there are i agents choosing s_1 and m means there are m agents choosing s_5. The sum of these integers should be 8. This combination is a state in the corresponding LTS. For the whole system, given a strategy profile, there are at most 8 enabled outgoing transitions, because each transition corresponds to the single-agent best deviation for each individual strategy. Therefore, there are 8 outgoing arrows from state (i, j, k, l, m, n, s, t) displayed in the figure.

Let us take the uppermost arrow in the figure as an example. This is a potential transition which means one agent choosing s_1 tries to deviate to another strategy according to the best payoff it can achieve. If this transition can happen, i must be

Algorithm 1: Deciding single-agent best deviation

```
input  : intArray strategy[8];
output: int $\mathcal{T}$;
1  Initialize $\mathcal{T}$ = 0;
2  Initialize $utility\_one[i] = 0$, $i = 1, 2, ..., 8$
3  Initialize $utility[i][j]$ = the corresponding expected payoff
   of strategy $i$ against strategy $j$.
4  for $i = 1 \to 8$ do
5      for $j = 1 \to 8$ do
6          Let $num$ = strategy[$j$];
7          if $j = 1$ then
8              $num$ = $num$ - 1;
9          if $num > 0$ then
10             for $k = 1 \to num$ do
11                 utility_one[$i$] = utility_one[$i$]+utility[$i$][$j$];
12 Let $max$ = 0.0;
13 for $i = 1 \to 8$ do
14     if $max < utility\_one[i]$ then
15         $max$ = utility_one[$i$];
16         $\mathcal{T}$ = $i$;
17 return $\mathcal{T}$;
```

positive; otherwise no agent can abandon s_1. A label $i > 0$ is used on the arrow to represent this constraint. Assume this condition is true; then there are some agents currently choosing s_1, and one of them may have the incentive to change s_1 to another strategy which can mostly increase its payoff. Which strategy will be the agent's new choice? The answer is decided via the negotiation procedure represented by *Out*_1, in which *Out*_1 means there will be an agent replacing its current strategy from s_1 to another. The details of the negotiation procedure *Out_i* ($i = 1, 2, ..., 8$) is described later. Afterwards, a new state *(i-1, j', k', l', m', n', s', t')* will be generated since a new strategy profile is obtained. Because one agent abandons s_1, then i becomes *i-1*. One of the other 7 integers will increase by 1 according to the result of *Out*_1.

After defining the transition rules of an individual state, another question is what the initial distribution of the strategies is among the agents, i.e., the initial strategy profile. This is important since it affects the executions thereafter. For example, $(8, 0, 0, 0, 0, 0, 0, 0)$ and $(0, 0, 0, 0, 0, 0, 0, 8)$ have totally different behaviors in their future executions because of the different payoff of different strategies. For robustness analysis, we should consider all possible scenarios of the system, i.e., it is better that the initial states cover all strategies profiles. Since PAT only supports one initial state, we manually add a virtual (non-existing) state as the initial state which is guaranteed to transit to all possible strategy profiles. This extra state does not affect the robustness analysis. In this way, all possible strategy profiles have been considered as one possible initial state of the system.

We illustrate the algorithms used in the model to decide single-agent best deviation. Here we use algorithm *Out*_1 as

Algorithm 2: High-level modeling in PAT

```
input  : intArray strategy[8];
output: int $\mathcal{T}$;
1  $P(i, j, k, l, m, n, o, p) = [i > 0$
2  $\&\&call(IdeviateToJ, i, j, k, l, m, n, o, p)]DeviationIJ \to$
   $P(i-1, j+1, k, l, m, n, o, p)$
3  //transition 1: modeling the case of deviating from the
   first strategy to the second one
4  []
5  $[i > 0\&\&call(IdeviateToK, i, j, k, l, m, n, o, p)]$
   $DeviationIK \to P(i-1, j, k+1, l, m, n, o, p)$
6  //transition 2. modeling the case of deviating from the
   first strategy to the third one.
7  [] ...
```

an example shown in Alg. 1, in which we want to check which strategy has a better payoff compared with s_1. In Alg. 1, the input is the array *strategy*[8] which represents the current strategy profile, i.e., the i^{th} element in this array indicates the number of agents choosing strategy s_i. The output is the strategy that an agent adopting strategy s_1 will be switched to under the single-agent best deviation principle. The array *utility_one* is the average utility that an agent adopting s_1 can obtain during the tournament and is initialized to 0 (Line 2). The array *utility* stores the corresponding payoffs obtained for each strategy when it negotiates against others (Line 3). Lines 4 to 11 calculate the average utility that any agent currently adopting s_1 can obtain during the tournament by deviating to other strategies according to Equation (2). Next the best deviation choice for any agent currently adopting s_1 is calculated in Lines 12-16 and $\mathcal{T}$ will be returned. In this algorithm, $\mathcal{T}$ can be 1 to 8. Actually if $\mathcal{T} = 1$, it means there is no need for agents using s_1 to change their choice, and thus no transition from the current state (i, j, k, l, m, n, s, t) will be generated. For simplicity, the transition generated by *Out*_1 in Fig. 2 ignores this case by assuming i will always be decreased by 1.

Strategy updating for each agent in Alg. 1 is nontrivial for traditional model checkers because of the existence of loops and real-valued matrices. In PAT, the single-agent best deviation algorithm for each strategy is implemented as a *C#* library and is imported into the PAT model. Each algorithm corresponds to one function which can be called directly in the PAT model following the syntax of call(functionname, parameter1, parameter2, ..., parameter8), where parameter*i* corresponds to the number of agents choosing strategy *i*. One code fragment of the high level modeling in PAT is shown in Algorithm 2. Process P models the current state of the system in terms of the number of agents choosing each strategy denoted as $i, j, ...p$. We model possible system state transitions as guarded processes, which are then combined together using the choice operator []. The function *IdeviateToJ* is one implementation of the single-agent best deviation (Algorithm 1) in C# and is called to determine whether the corresponding process will be executed or not.

B. Properties Verification

From the analysis of Section III-B, empirical pure strategy Nash equilibrium, best reply cycle and the resulting basin of attraction should be verified. In the following we describe the corresponding model checking algorithms used to check suitable properties.

1) Empirical Pure Strategy Nash Equilibrium: The existence of empirical pure strategy Nash equilibrium means there exist some states that all agents in the negotiation setting will keep their strategies, so that no outgoing transition exists from these states. From the viewpoint of model checking, these states are *deadlock* states in the system. On the other hand, each state in the system is corresponding to a strategy profile; therefore each deadlock state existing in the model indicates the agents will not change their mind, and this state should satisfy empirical pure strategy Nash equilibrium. As a result, deadlock checking can be used to check whether empirical pure strategy Nash equilibrium exists in the system. In traditional deadlock checking, the verification algorithm stops whenever a deadlock state is found, and returns a counterexample. Otherwise no deadlock state exists. In our setting, we need to find all deadlock states instead of one in order to analyze the robustness. So we modify the traditional deadlock checking algorithm to capture all deadlock states, if there are any.

2) Best Reply Cycle: Best reply cycle describes the scenario where there are several states composing a group, and these states do not have outgoing transitions to states outside this group. In this case, states in the group cannot reach states which correspond to empirical pure strategy Nash equilibrium since they can always transit to other states. From the viewpoint of model checking, the desired groups are nontrivial BSCCs in the state space. It is trivial to show that all nontrivial BSCCs should be best reply cycle. Therefore our target is to find all nontrivial BSCCs. SCC searching is widely used in model checking techniques, especially for LTL verification. Tarjan's SCC searching algorithm [32] can be applied here to find all nontrivial SCCs, and BSCCs are restored as our targets.

3) Basin of Attraction: Based on the above two properties, the stability of different strategy profiles can be decided. All empirical pure strategy Nash equilibria and best reply cycles compose the stable sets of the negotiation system. Basin of attraction of a stable set is defined as the percentage of strategy profiles which can lead to itself through a series of single-agent best deviations, i.e., the percentage of the states in the system reaching the deadlock states or BSCCs. It can be calculated based on the results of the above two properties, since the total number of states reaching each stable set can be recorded. One strategy is robust if and only if it belongs to stable sets with large basin of attraction.

V. Evaluation

In this section, we apply our approach to perform robustness analysis on the top strategies chosen from ANAC 2010-2012 using PAT [21], [22], [29]. We perform the negotiation competition among all strategies in three ANAC finals and choose the top 6 as our target:

- *Gahboninho* (**G**): the *Gahboninho* strategy learns to predict whether the opponent is concessive or not based on the past negotiation experience (the proposals of the opponent), and adjusts its behavior based on the prediction. If the opponent is predicted to be concessive, it will behave in a very selfish way by giving up almost no utility; Otherwise, it will adapt itself to minimize losses by giving up more utility to its opponent.
- *HardHeaded* (**H**): the *HardHeaded* strategy gradually increases its concession degree to the opponent as negotiation continues. It always proposes the bid which is estimated to give the highest utility to its opponent from the set of bids over which the *HardHeaded* agent's utility is higher than its acceptable threshold.
- *IAMhaggler2011* (**I**): the *IAMhaggler2011* strategy employs a Gaussian process regression technique to predict the opponent's behavior and determines its optimal concession strategy based on its predictions and the time constraints.
- *AgentLG* (**A**): the *AgentLG* strategy divides the negotiation period into three different stages and makes more concessions to the opponent as the negotiation moves into the next stage. At the same time, it learns the preference function of the opponent from the negotiation history and always proposes the offer that is both higher than its current acceptance threshold and also is expected to give the highest utility to the opponent.
- *CUHKAgent* (**C**): the *CUHKAgent* strategy employs the non-exploitation point to adaptively adjust the appropriate time to stop exploiting the negotiating partner and also predicts the optimal offer for the negotiating partner based on a reinforcement-learning based approach [14], [16].
- *OMACAgent* (**O**): the *OMACAgent* predicts the utilities of its opponent's future counter-offers using wavelet decomposition and cubic smoothing spline techniques. Based on the prediction of the future utilities that the opponent is willing to offer, the OMAC agent adaptively adjusts its concession rate [8].

The set of top strategies are abbreviated as $\mathcal{S} = \{\mathbf{G, H, I, A, C, O}\}$. We use the notation $\mathcal{P}$ to represent the set of agents participating in the negotiation.

Given any two negotiation strategies, their corresponding payoffs from the bilateral negotiation are obtained by averaging over all negotiation domains used in the ANAC. The detailed payoff matrix for all possible bilateral negotiations between these top strategies is given in Table I and will be used as the basis for performing the robustness analysis. Next, multiple experiments are conducted under different numbers of agents to analyze the robustness of these strategies and illustrate the effectiveness of our approach.

A. Bilateral Negotiations among Six Possible Strategies

In the context of bilateral negotiations, there exist 2 agents, i.e., $|\mathcal{P}| = 2$, and the strategy set $\mathcal{S}$ is defined as above. Each agent can choose any strategy from the set $\mathcal{S}$ during negotiation. Through the counter abstraction approach, the state space of the negotiation model is reduced from $|\mathcal{S}|^{|\mathcal{P}|} = 6^2 = 36$

TABLE I
Payoff matrix for the top six negotiation strategies in ANAC 2010-2012 average over all domains (For each strategy profile, only the row agent's payoff is given since the game is symmetric.)

$U(p,p')$	**G**	**H**	**I**	**A**	**C**	**O**
G	0.6803	0.5202	0.8120	0.5801	0.5892	0.5545
H	0.6620	0.5995	0.7567	0.5694	0.6041	0.5485
I	0.6219	0.5641	0.7149	0.5946	0.4704	0.4915
A	0.7092	0.5900	0.7870	0.5675	0.6638	0.5612
C	0.7397	0.6388	0.8261	0.5515	0.5971	0.5897
O	0.6973	0.6275	0.7705	0.5513	0.6053	0.5711

TABLE II
Robustness ranking of strategies in bilateral negotiations.

Strategy	**G**	**H**	**I**	**A**	**C**	**O**
Ranking	5th	5th	1st	1st	1st	4th

to $\binom{|\mathcal{P}|+|\mathcal{S}|-1}{|\mathcal{S}|-1} = 21$. This reduction can help to reduce the verification time of PAT. The verification results are:

- Verification time: 0.1 second.
- Deadlock states: there are no such states.
- BSCCs: there is only one nontrivial BSCC existing in the system: $(0,0,0,1,1,0) \rightarrow (0,0,1,1,0,0) \rightarrow (0,0,1,0,1,0) \rightarrow (0,0,0,1,1,0)$.
- States reaching deadlock or BSCCs: 21 states reach the BSCC; no state reaches deadlock states.

Under bilateral negotiations, there is no empirical pure strategy Nash equilibrium and there exists only one best reply cycle, i.e., $(0,0,0,1,1,0) \rightarrow (0,0,1,1,0,0) \rightarrow (0,0,1,0,1,0) \rightarrow (0,0,0,1,1,0)$. Here 0 and 1 indicate the number of agents choosing each strategy, and the order of these numbers in one state is consistent with the strategies order listed in $\mathcal{S}$. The basin of attraction of this cycle is 100%, i.e., for all possible initial strategy profiles, there always exists a single-agent best deviation path which can lead to one of the strategy profiles within this cycle. Strategies **I, A** and **C** are contained in strategy profiles within the cycle, which indicates that **I, A** and **C** are very robust against other strategies since agents will be willing to deviate to them no matter what their initial strategies are. EGT analysis in [1] has the same result in this scenario. In other words, both approaches generate the same best reply cycle because each strategy profile in this cycle only has one outgoing transition.

Next, we investigate the overall ranking of these top strategies according to their robustness. We adopt the elimination mechanism used in [6]. Different from [6] which eliminates the worst player, we gradually eliminate the most robust strategies available to the agents in the experiment, and try to find the most robust strategies in the remaining ones. Note that the robustness of one strategy may be related with some opponents' performance. Note that the ranking of a given strategy may be affected by the presence, or absence, of other strategies, and may depend on the number of agents.

According to the existence of best reply cycle, we can conclude that strategies **I**, **A**, and **C** rank the first in all strategies in the current system. These three strategies are removed from $\mathcal{S}$, and the robustness analysis is conducted with the remaining three strategies. Step by step, the overall robustness ranking of these top strategies in bilateral negotiations is listed in Table II. **O** has the average robustness, while **G** and **H** are relatively not so robust. Therefore, the agents having the last two strategies will try to change their choices to get better payoff.

In most cases a tournament setting involves more than two agents. Therefore, the following checks whether the number of agents could affect the robustness of the strategies.

TABLE III
Robustness ranking of strategies in six-agent negotiations.

Strategy	**G**	**H**	**I**	**A**	**C**	**O**
Ranking	1st	6th	1st	1st	1st	5th

B. Six-agent Negotiations among Six Possible Strategies

We increase the number of agents to six over the set $\mathcal{S}$, i.e., $|\mathcal{P}| = 6$. The total number of strategy profiles can be reduced from $|\ \mathcal{S}\ |^{|\mathcal{P}|} = 6^6 = 46656$ to $\binom{|\mathcal{P}|+|\mathcal{S}|-1}{|\mathcal{S}|-1} = 462$ considering the symmetry of the negotiation. The verification results from PAT are as follows:

- Verification time: 0.4 seconds.
- Deadlock states: there is no such states.
- BSCCs: there is only one nontrivial BSCC existing in the system: (2,0,0,2,2,0), (2,0,0,3,1,0), (1,0,0,4,1,0), (1,0,1,4,0,0), (2,0,1,3,0,0), (2,0,1,2,1,0), (2,0,1,1,2,0), (1,0,1,2,2,0), (0,0,1,3,2,0), (1,0,0,3,2,0), (0,0,0,4,2,0), (1,0,1,3,1,0), (0,0,1,4,1,0), (0,0,0,5,1,0), (0,0,1,5,0,0).
- States reaching deadlock or BSCCs: 462 states reach the BSCC; no state reaches deadlock states.

There is only one best reply cycle. Here these states compose a real BSCC instead of a single cycle, which is generated via our single-agent best deviation based approach and is different from the result under the definition of [1]. For instance, state (2, 0, 1, 3, 0, 0) can go to (2, 0, 0, 3, 1, 0) and (2, 0, 1, 2, 1, 0), and all these three states are in the BSCC. On the contrary, applying the method in [1] only generates a best reply cycle $(1,0,1,4,0,0) \rightarrow (2,0,1,3,0,0) \rightarrow (2,0,0,3,1,0) \rightarrow (1,0,0,4,1,0) \rightarrow (1,0,1,4,0,0)$. Thus we are able to provide more realistic robustness analysis and some important information is lost in the previous anlaysis.

Obviously, the basin of attraction of our BSCC is 100%. Next, we rank the strategies in the six-agent negotiation scenario. Strategies **G, I, A** and **C** are contained in strategy profiles within the BSCC, which indicates that in the six-agent scenario, **G** is also very robust. **G, I, A** and **C** are removed and the robustness analysis is conducted with the remaining strategies **H** and **O**. The ranking information is listed in Table III. We conclude that in this six-agent scenario, **H** is relatively not so robust.

C. Ten-agent Negotiations among Six Possible Strategies

We investigate a ten-agent negotiation scenario over the set $\mathcal{S}$, i.e., $|\mathcal{P}| = 10$. The total number of strategy profiles considered can be reduced from $|\mathcal{S}|^{|\mathcal{P}|} = 6^{10} = 60466176$ to $\binom{|\mathcal{P}|+|\mathcal{S}|-1}{|\mathcal{S}|-1} = 3003$ considering the symmetry of the negotiation. The verification results from PAT are:

TABLE IV
ROBUSTNESS RANKING OF STRATEGIES IN TEN-AGENT NEGOTIATIONS.

Strategy	**G**	**H**	**I**	**A**	**C**	**O**
Ranking	1st	5th	6th	1st	3rd	3rd

TABLE V
ROBUSTNESS RANKING OF STRATEGIES IN THE TWENTY-AGENT NEGOTIATIONS.

Strategy	**G**	**H**	**I**	**A**	**C**	**O**
Ranking	1st	5th	6th	1st	3rd	3rd

- Verification time: 2.8 seconds.
- Deadlock states: there is one deadlock state (3, 0, 0, 7, 0, 0).
- BSCCs: there is no nontrivial BSCC existing in the system.
- States reaching deadlock or BSCCs: all 3003 states reach the deadlock state.

Because there is only one deadlock state in the system, we conclude that one empirical pure strategy Nash equilibrium exists in the ten-agents negotiation scenario: (3, 0, 0, 7, 0, 0). When 3 agents choose **G** and 7 agents choose **A**, no further strategy deviation will happen. In this case, **G** and **A** are the most robust strategies. This result is consistent with the outcome obtained via the method in [1]. The basin of attraction of this equilibrium is 100%.

Again, we get the robustness ranking of these strategies in this ten-agent scenario (Table IV). **G** and **A** have the best robustness; **C** and **O** have average performance and **H** and **I** are not so robust.

D. Twenty-agent Negotiations among Six Possible Strategies

We further increase the number of agents to twenty, therefore $|\mathcal{P}| = 20$. The total number of strategy profiles considered can be reduced from $\mid \mathcal{S} \mid^{|\mathcal{P}|} = 6^{20} = 3.66E15$ to $\binom{|\mathcal{P}|+|\mathcal{S}|-1}{|\mathcal{S}|-1} = 53130$ considering the symmetry of the negotiation. The verification results from PAT are:

- Verification time: 83.8 seconds.
- Deadlock states: there is one deadlock state (6, 0, 0, 14, 0, 0).
- BSCCs: there is no nontrivial BSCC existing in the system.
- States reaching deadlock or BSCCs: all 53130 states reach the deadlock state.

Similar to the ten-agent scenario, the twenty-agent case only has one empirical pure strategy Nash equilibrium: (6, 0, 0, 14, 0, 0). Again, **G** and **A** are the most robust strategies. This result is also consistent with the outcome obtained via the method in [1]. The basin of attraction of this equilibrium is 100%, i.e., all strategy profiles can reach this equilibrium via single-agent best deviation paths.

The robustness ranking of these six strategies under twenty-agent negotiation scenarios is listed in Table V, which is consistent with the ranking in ten-agent negotiation scenario.

The robustness rankings may be different when different numbers of agents participate in the negotiation. This is reasonable since whether one agent will change its strategy depends on its opponents' situation. We summarize the rankings in Fig. 3. Strategy **A** always has the best robustness no matter how many agents participate in the negotiation. **G** is also very robust when more than 6 agents are involved. By contrast, **I** is attractive when the number of agents is small, but it is not so robust if more agents are taken into consideration. **C** has stable performance since it is always in the top 3 in the robustness ranking. **O** always ranks between 3rd to 5th, which means it is not very attractive. **H** is most likely to be abandoned by agents in this top six strategies competition. These rankings indicate that different strategies have different robustness in various scenarios; thus our analysis results provide valuable recommendations and reference for selecting the best strategy in terms of robustness within different negotiation contexts.

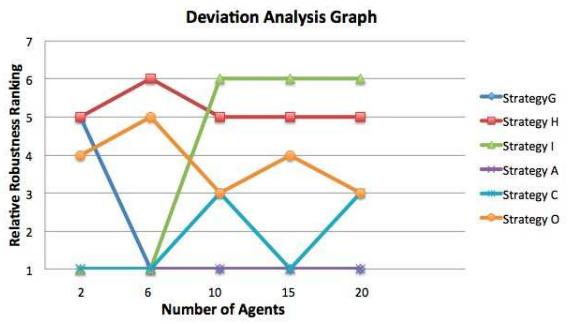


Fig. 3. Deviation Analysis Graph for All Strategy Profiles

VI. CONCLUSION

In this work, we proposed to improve the EGT analysis via model checking approach to analyze the *robustness* of negotiation strategies in a general multi-agent system. This approach guarantees the automaton, efficiency and completeness of the analysis procedure. The dynamics of strategy profiles are modeled in a formal modeling language. In order to reduce the state space, the counter abstraction technique is applied in the modeling. Afterwards, properties representing empirical pure strategy Nash equilibrium, best reply cycle and the corresponding basin of attraction are formally verified by PAT. The approach is effective and efficient in analyzing these properties, and it can also be used to generate the overall robustness rankings of the strategies.

The current robustness analysis relies on the assumption of single-agent best deviation, which only allows each agent to switch to the strategy that could bring it the highest possible deviation benefit on average. However, it might be more practical to consider probabilistic deviation under which each negotiator chooses to switch to other strategies with certain probability since the actual deviation benefit for each strategy could be uncertain. Meanwhile, although the counter abstraction technique has a remarkable effect in reducing the state space of multi-agent systems, it requires that all agents are identical. In many scenarios such as auctions, the dynamics of agents may be different. Thus another interesting direction is to explore how to handle the cases with heterogeneous agents.

VII. Acknowledgement

This work is supported by “Formal Verification on Cloud” project under Grant No: M4081155.020 and “Bring the Advanced Model Checking Techniques to Real-world Problems” project under Grant No: M4011178.020. The work presented was partially supported by CUHK Research Committee Funding (Direct Grants) Project Code EE13379.

Notes

[1]Here $\mathcal{V}_i$ can be either discrete values or continuous real values.

[2]This concept is similar to the concept of *pure strategy Nash equilibrium* in classical game theory, but it is called empirical pure strategy Nash equilibrium since the analysis is based on empirical results.

[3]Via external libraries, CSP# supports any user-defined data type.

[4]Without losing generality, we assume there is only one initial state in the system.

[5]Here the global variables refer to all variables defined in the model apart from the local variables $(s_1, \ldots, s_n)$ storing the strategy choices for each agent.

References

[1] T. Baarslag, K. Fujita, E. H. Gerding, K. Hindriks, T. Ito, N. R. Jennings, C. Jonker, S. Kraus, R. Lin, V. Robu, and C. R. Williams. Evaluating practical negotiating agents: Results and analysis of the 2011 international competition. *Artificial Intelligence Journal*, 198:73–103, May 2013.

[2] T. Baarslag, K. Hindriks, C. Jonker, S. Kraus, and R. Lin. The first automated negotiating agents competition (anac 2010). *New Trends in Agent-Based Complex Automated Negotiations*, pages 113–135, 2010.

[3] C. Baier and J. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[4] P. Ballarini, M. Fisher, and M. Wooldridge. Uncertain agents verification through probabilistic model-checking. In *SASEMAS'09*, Lecture Notes in Computer Science, pages 162–174, 2009.

[5] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. *AAMAS*, 12:239–256, 2006.

[6] B. Bouzy and M. Métivier. Multi-agent learning experiments on repeated matrix games. In *Proceedings of ICML'10*, pages 119–126, 2010.

[7] S. Chaki, E. M. Clarke, J. Ouaknine, N. Sharygina, and N. Sinha. State/Event-Based Software Model Checking. In *Proceedings of IFM'04*, pages 128–147, 2004.

[8] S. Q. Chen and G. Weiss. An efficient and adaptive approach to negotiation in complex environments. In *Proceedings of ECAI12*, pages 228–233, 2012.

[9] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A model checker for stochastic multi-player games. In *Proceedings of TACAS'13*, volume 7795, pages 185–191, 2013.

[10] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[11] P. Faratin, C. Sierra, and N. R. Jennings. Using similarity criteria to make negotiation trade-offs. *Artifical Intelligence*, 142(2):205–237, 2003.

[12] P. A. Gongora and D. A. Rosenblueth. A characterization of mixed-strategy nash equilibria in pctl augmented with a cost quantifier. *Computational Logic in Multi-Agent Systems*, 6214:158–177, 2010.

[13] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:102–111, 1994.

[14] J. Y. Hao and H. F. Leung. Abines: An adaptive bilateral negotiating strategy over multiple items. In *Proceedings of IAT'12*, pages 95 – 102, 2012.

[15] J. Y. Hao, S. Z. Song, Y. Liu, J. Sun, L. Gui, J. S. Dong, and H. F. Leung. Probabilistic model checking multi-agent behaviors in dispersion games using counter abstraction. In *Proceedings of PRIMA'12*, pages 16–30, 2012.

[16] J. Y. Hao, S.Z., H.F. Leung, and Z. Ming. An efficient and robust negotiating strategy in bilateral negotiations over multiple items. *Engineering Applications of Artificial Intelligence*, 34:45–57, 2014.

[17] K. Hindriks and D. Tykhonov. Opponent modeling in auomated multi-issue negotiation using bayesian learning. In *Proceedings of AAMAS'08*, pages 331–338, 2008.

[18] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *Proceedings of TACAS06*, pages 441–444, 2006.

[19] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[20] Gerard J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23:279–295, 1997.

[21] Y. Liu, J. Sun, and J.S. Dong. Developing model checkers using pat. In *Proceedings of 8th International Symposium Automated Technology for Verification and Analysis (ATVA 2010)*, volume 6252, pages 371–377, 2010.

[22] Y. Liu, J. Sun, and J.S. Dong. Pat 3: An extensible architecture for building multi-domain model checkers. In *Proceedings of International Symposium on Software Reliability Engineering (ISSRE)*, pages 190–199, 2011.

[23] J. Estelle M. P. Wellman, S. Singh, Y. Vorbeychik, and V. Soni. Strategic interactions in a supply chain game. *Computational Intelligence*, 21(1):1–26, 2005.

[24] F. Mari, I. Melatti, I. Salvo, E. Tronci, L. Alvisi, A. Clement, and H. Li. Model checking coalition nash equilibria in mad distributed systems. In *Stabilization, Safety, and Security of Distributed Systems*, pages 531–546. Springer, 2009.

[25] A. Pnueli, J. Xu, and L. Zuck. Liveness with (0,1,∞)-counter abstraction. In *Proceedings of the 14st International Conference on Computer Aided Verification (CAV'02)*, pages 107–122, 2002.

[26] J. W. Pratt. Risk aversion in the small and in the large. *Econometrica*, 32:122–136, 1964.

[27] A.S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In *Proceeding of MAAMAW'96*, pages 42–55, 1996.

[28] S. Saha, A. Biswas, and S. Sen. Modeling opponent decision in repeated one-shot negotiations. In *Proceedings of 4th international joint conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, pages 397–403, 2005.

[29] J. Sun, Y. Liu, J. S. Dong, and J. Pang. PAT: Towards Flexible Verification under Fairness. In *Proceedings of the 21st International Conference on Computer Aided Verification (CAV'09)*, pages 709–714, 2009.

[30] J. Sun, Y. Liu, J.S. Dong, and C.Q. Chen. Integrating specification and programs for system modeling and verification. In *Proceedings of TASE'09*, pages 127–135, 2009.

[31] E. M. Tadjouddine, F. Guerin, and W. Vasconcelos. Abstraction for model checking game-theoretical properties of auction(short paper). In *Proceedings of AAMAS'08*, pages 1613–1616, 2008.

[32] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

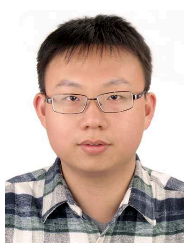
[33] D. R. Vasconcelos and E. H. Haeusler. Reasoning about games via a first-order modal model checking approach. In *Proceedings of SBMF07 10th Brazilian Symposium on Formal Methods*, 2007.

[34] P. Vytelingum, D. Cliff, and N.R. Jennings. Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172(14):1700–1729, 2008.

[35] C. R. Williams, V. Robu, E. H. Gerding, and N. R. Jennings. Using gaussian processes to optimise concession in complex negotiations against unknown opponents. In *Proceedings of IJCAI'11*, pages 432–438, 2011.

[36] M. Wooldridge, M. Fisher, M. P. Huget, and S. Parsons. Model checking multi-agent systems with mable. In *Proceedings of AAMAS'02*, pages 952–959, 2002.

[37] H. Yong. The evolution of conventions. *Econometrica*, 61(1):57–84, 1993.



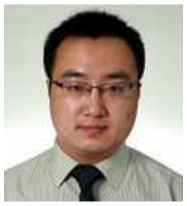
Songzheng Song received his PhD degree in Computer Science from National University of Singapore in 2013. After that, he worked as a postdoctoral fellow in Nanyang Technological University for one year and now he works in Autodesk as a software engineer. His current research interests focus on formal methods and multiagent systems.

Jianye Hao received his PhD degree in Computer Science and Engineering from The Chinese University of Hong Kong in 2013. Since then, he works as a SUTD-MIT postdoctoral Fellow at Massachusetts Institute of Technology (MIT) and Singapore University of Technology and Design (SUTD). His current research interests focus on multiagent systems and network security.

Yang Liu received Bachelor and PhD degrees in computing science from National University of Singapore (NUS) in 2005 and 2010, and continued with his post doctoral work in NUS. Since 2012, he joined Nanyang Technological University as an Assistant Professor. His research focuses on software engineering, formal methods and security, and particularly specializes in software verification using model checking techniques, which led to the development of a state-of-theart model checker, Process Analysis Toolkit.

SUN, Jun received Bachelor and PhD degrees in computing science from National University of Singapore (NUS) in 2002 and 2006. Since 2010, he joined Singapore University of Technology and Design (SUTD) as an Assistant Professor. Jun's research interests include software engineering, formal methods and cyber-security.

Ho-fung Leung received his BSc and MPhil degrees in Computer Science from The Chinese University of Hong Kong, and his PhD degree from University of London with Diploma of Imperial College in Computing from Imperial College London. He is currently a Professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong. His research interests cover various aspects centring around artificial intelligence and the web, including ontologies, intelligent agents, self-organising systems, complex systems, and social computing.



Jie Zhang obtained Ph.D. in Cheriton School of Computer Science from University of Waterloo, Canada, in 2009. He is currently an Assistant Professor of the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include Artificial Intelligence, Multiagent System and Trust Management, and his papers have been published by top journals and conferences and won several best paper awards.