POPL: G: Reasoning about Incentive Compatibility

Suguman Bansal
Rice University
suguman@rice.edu

Abstract

We study equilibrium computation in *regular repeated games*, a model of infinitely repeated, discounted, general-sum games where agent strategies are given by a finite automaton. Our goal is to compute, for subsequent querying and analysis, the set of all Nash equilibria in such a game. Our technical contribution is an algorithm for computing all Nash equilibria in a regular repeated games. We use our algorithm for automated analysis of properties such as incentive compatibility on complex multi-agent systems with reward maximizing agents, such as the Bitcoin protocol.

1. Introduction

Games [26], defined as multi-agent systems with reward maximizing agents, find applications in modeling and understanding properties in a wide variety of fields, such as economics [11, 13, 20, 25, 32], social sciences [8, 21, 22], biology [4, 9, 18, 33], etc. With the emergence of massive online protocols such as auctions, adopted by cyber giants such as Google Auctions and eBay, games have generated much interest for utilitarian research from within the computer science community. Presence of bugs in these protocols have tremendous impact on both, the protocol providers and its large number of users. Hence, the analysis of properties on such systems is crucial.

One concrete instance of the analysis of such protocols arises in the context of the *Bitcoin protocol*. Bitcoins are one of the most widely used on-line currency in today's time. The protocol by which Bitcoins are *mined* (minted) is called the Bitcoin protocol. Agents (miners) of the protocol receive rewards by mining more bitcoins. Earlier it was believed that miners receive the most reward when they mine following the honest policies of the protocol. Since the protocol rewarded miners proportional to the number of bitcoins they mined, it was considered to be a fair protocol. However, a surprising result by Eyal and Sirer disproved this misconception[12]. They proved *via rigorous manual analysis* that miners can receive greater rewards by following dishonest policies. A game is said to be *incentive compatible* if it ensures that reward maximizing agent will behave according to the honest policies of a game. Hence, Eyal and Sirer proved that the Bitcoin protocol is not incentive compatible.

Incentive compatibility is a desirable property in games, since it ensures that all agents behave honestly in the game. However, erroneous system design is common. Therefore, analysis of properties such as incentive compatibility is critical for good game design. Unfortunately, as demonstrated in the Bitcoin protocol case, even in today's times most of this analysis is performed manually. One cannot simply port results on verification of traditional systems in the context of games since their verification differs due to the presence of reward maximizing agents. More specifically, unlike in traditional systems, not all behaviors in a game need to be analyzed. In the analysis of games, one can safely ignore behaviors that do not conform with the reward maximizing objectives of agents. Behaviors that conform with agent objectives are called the *solution concepts* of the game.

Algorithmic analysis of solution concepts in games has been widely studied in *algorithmic game theory* [24]. In this field, complex games are studied as *repeated games* [1, 14, 17]. An important solution concept is that of *Nash equilibria* [23]. Intuitively, Nash equilibria is a state of a game in which no agent can increase its reward by changing its own behavior *only*.

There is a wide literature on modeling repeated games as finite state machines [19, 27, 28]. The problem of computing Nash equilibria in repeated games has received a lot of interest. The seminal work by Abreu, Pearce, and Stacchetti [2], and its followups [10], considered the problem of computing all or some equilibrium rewards rather than equilibria. The problem of efficiently computing a *single* equilibrium strategy in infinite games has been studied before [3, 15, 16]. There are also a few approaches to computing representations of *approximations* to the set of equilibria [5–7]. Except for work on finding one equilibrium, most other approaches do not guarantee crisp complexity bounds.

For our purpose of analysis in repeated games, we are interested in the problem of computing all Nash equilibria in a game. In this paper, we combine ideas from Game theory, and tools and techniques developed in Programming languages and Formal methods to realize this goal. First, we define a finite state machine based framework to model repeated games. We call our framework *regular repeated games*. Next, we develop an algorithm ComputeNash to compute and enumerate all Nash equilibria in a regular repeated game. Unlike previous approaches, our algorithm has a crisp complexity bound, and can determine all Nash equilibria in a regular repeated game. Lastly, as a proof of concept, we perform case studies on the Bitcoin protocol and repeated auctions, and corroborate previously known results on them via analysis by ComputeNash. The merit in our approach over earlier approaches for the analysis of games is that our analysis approach is automated, hence quicker and not prone to errors. For the Bitcoin protocol case, we first argue that the Bitcoin protocol can be viewed as a repeated game, and hence we can apply ComputeNash for its analysis.

This paper is organized as follows. Section 2 provides the necessary preliminary background from game theory and programming languages. Section 3 introduces and defines regular repeated games as a model for repeated games. Section 4 glosses over the description of ComputeNash. Lastly, Section 5 demonstrates how ComputeNash can be used for automatic analysis of incentive compatibly in the Bitcoin protocol and repeated auctions.

2. Preliminaries

2.1 Repeated Games

Repeated games [17] are a model for multi-agent systems in which agents interact with each other infinitely often. In each round of interaction, each agent picks an *action*. These actions are picked *synchronously* by all agents i.e. in each round of interaction, agents

are not aware of what actions other agents will take in that round. However, agents become aware of actions taken by other agents in all previous rounds. Agents receive a *payoff* (or reward) based on the actions taken by each agent in an interaction. The payoff of an agent from a repeated game is computed by aggregating the rewards received by the agent in each round using the *discounted sum*. The discounted sum of a sequence of payoff $A = \{a_0, a_1, a_2 \dots\}$ with discount factor $d > 1$, denoted by $DS(A, d)$, is given by $\Sigma_{i=0}^{\infty} a_i/d^i$.

Each agent in a repeated games has its own *strategies*. Intuitively, a strategy can be thought of as a *guideline* for the agent on how to pick an action in a round based on actions taken by all agents in previous rounds. For example, in an auction one strategy for an agent is to bid truthfully, and another strategy is to never bid truthfully. All agents pick their *strategy* for the entire game only once in the beginning. Agents pick their strategies synchronously. *Strategy profile*, denoted by $\overline{\mathcal{M}}$, is a k-tuple of strategies $(\mathcal{M}^1, \dots, \mathcal{M}^k)$ where $\mathcal{M}^i$ is the strategy of the i-th agent.

Various *solution concepts* have been defined as a means to characterize which strategy an agent should pick in order to receive greater rewards from a game. This is important, since we assume that agents in multi-agent systems are reward-maximizing by nature. Two such solution concepts are: *Nash equilibria* and *best response strategy*.

Before defining these concepts, we introduce some necessary notation. For each agent $\mathcal{P}_i$ and its strategy $\mathcal{M}^{'}$, we define a strategy profile $\overline{\mathcal{M}}[i := \mathcal{M}^{'}]$ obtained by starting with $\overline{\mathcal{M}}$, and then switching the strategy of agent $\mathcal{P}_i$ to $\mathcal{M}^{'}$. Precisely, $\overline{\mathcal{M}}[i := \mathcal{M}^{'}]$ is defined as the tuple $(\mathcal{M}^1, \dots, \mathcal{M}^{i-1}, \mathcal{M}^{'}, \mathcal{M}^{i+1}, \dots, \mathcal{M}^k)$. $\overline{\mathcal{M}}[i = \emptyset]$ denotes the $k-1$ tuple obtained by removing the strategy for i-th agent.

A tuple of strategies $\overline{\mathcal{M}}$ is said to be in Nash equilibrium if no agent receives a greater payoff by *unilaterally* changing its strategy. More concretely, no agent $\mathcal{P}_i$ receives a greater payoff in $\overline{\mathcal{M}}[i = \mathcal{M}_*^i]$ as compared to that in $\overline{\mathcal{M}}$ where $\mathcal{M}_*^i$ is another strategy of $\mathcal{P}_i$.

A strategy $\mathcal{M}^i$ is said to be a best response strategy for agent $\mathcal{P}_i$ to $\overline{\mathcal{M}}[i = \emptyset]$ if for all other strategies $\mathcal{M}_*^i$ of $\mathcal{P}_i$, , $\mathcal{P}_i$ does not receive greater payoff on $\overline{\mathcal{M}}[i = \mathcal{M}_*^i]$ as opposed to that on $\overline{\mathcal{M}}[i = \mathcal{M}^i]$.

2.2 Büchi automata

A Büchi automaton [31] , denoted by $\mathcal{A}$, is a tuple $(S, \Sigma, \delta, Init, \mathcal{F})$ where S is a finite set of states, $Init \subseteq S$ is a non-empty set of initial states, Σ is a finite alphabet, $\delta \subseteq S \times \Sigma \times S$ is the transition relation, and $\mathcal{F} \subseteq S$ is a non-empty set of accepting states.

An automaton is called *deterministic* if for all states s, and all symbols $a \in \Sigma$, $|\{s' | (s, a, s') \in \delta\}| \leq 1$. Otherwise, it is called *non-deterministic*.

Let Σ^ω be the set of infinite words over Σ (similar notation is used for other alphabets). For $w = w_0 w_1 \dots \in \Sigma^\omega$, a *run* ρ of w in $\mathcal{A}$ is a sequence of transitions $\tau_0 \tau_1 \dots$ such that there is a sequence of states $s_\rho = s_0 s_1 \dots$ satisfying: (1) $s_0 = Init$, and (2) $\tau_i = (s_i, w_i, s_{i+1})$ for all i. Note, each sequence in a Büchi automaton is infinite.

Let $inf(\rho)$ be the set of states that occur infinitely often in s_ρ. We say w has an *accepting run* in $\mathcal{A}$ if there exists a run ρ of w such that $inf(\rho) \cap \mathcal{F} \neq \emptyset$. The automaton $\mathcal{A}$ *accepts* a word w if w has an accepting run in $\mathcal{A}$.

Büchi automata are also known to be closed under set-theoretic union, intersection, and complementation over Σ^ω. Languages accepted by these automata are called *ω-regular languages*.

3. Regular repeated games (RRG)

We define *Regular repeated games* as a formal model for repeated games. Regular repeated games builds on top of the well established Rubinstein's model for repeated games [27].

Formally, a *regular repeated game* $\mathcal{G}$ is given by a set of k agents, where the i-th agent $\mathcal{P}_i$ consists of a finite set $Action(i)$ of *actions*, and a finite set $Strategy(i)$ of *strategies*. A strategy $\mathcal{M}$ for $\mathcal{P}_i$ is Büchi automaton with weights. Figure 1 shows a strategy for the agent $\mathcal{P}_i$. Each transition of the automaton represents one round of the repeated game. Transitions occur over the alphabet of $(a, (a)_{-i})$ where $a \in Action(i)$ is the action of the agent $\mathcal{P}_i$, and $(a)_{-i} \in \prod_{j \neq i} Action(j)$ is the *environment action*. Agent $\mathcal{P}_i$ receives a reward along each transition. In Figure 1, transition $q_0 \xrightarrow{(C,\mathbf{D}),0} q_1$ denotes that when the agent acts C, and the environment acts **D**, then the agent receives a reward of 0.

Intuitively, accepting runs of a strategy $\mathcal{M}$ offer an agent-level view of executions of the game. Specifically, consider a transition $(q, (a)_{-i}, a_i, p, q')$ that appears at the j-th position of an accepting run of $\mathcal{M}$. This means that there is a *possible* execution of the game in which: (i) the i-th agent is at state q immediately before the j-th round of the game; (ii) in this round, the i-th agent and its environment synchronously perform the actions a_i and $(a)_{-i}$, respectively; (iii) the concurrent action leads agent $\mathcal{P}_i$ to transition to state q' at the end of this round, and receive payoff p.

Strategy profiles Now we define the semantics of interactions between agents $\mathcal{P}_1, \dots, \mathcal{P}_k$ that constitute a game $\mathcal{G}$.

A *strategy profile* $\overline{\mathcal{M}}$ of $\mathcal{G}$ is a tuple of strategies $(\mathcal{M}^1, \dots, \mathcal{M}^k)$, where $\mathcal{M}^i \in Strategy(i)$ for each $\mathcal{P}_i$. Intuitively, $\overline{\mathcal{M}}$ captures a scenario in which $\mathcal{P}_i$ follows the strategy $\mathcal{M}^i$. Strategies of agents $\mathcal{M}^i$ synchronize on their actions, and the environment's actions to form the automaton corresponding to the strategy profile. In other words, the strategy profile is the synchronized product of its component strategies.

We illustrate the construction of a strategy profile from its component strategies in Figures 2, 3, 4. Figure 2 and Figure 3 depict two strategies, $\mathcal{M}^1$ and $\mathcal{M}^2$ respectively, for agents in a 2-agent game. Figure 4 illustrates the strategy profile $(\mathcal{M}^1, \mathcal{M}^2)$. Transition $t_1 \xrightarrow{(D,C),(4,0)} t_2$ is present in the strategy profile since transitions $q \xrightarrow{(D,\mathbf{C}),4} q$ and $s_1 \xrightarrow{(C,\mathbf{D}),0} s_2$ are present in strategies $\mathcal{M}^1$ and $\mathcal{M}^2$ respectively.

Transition $t_1 \xrightarrow{(D,C),(4,0)} t_2$ denotes that when agents $\mathcal{P}_1$ and $\mathcal{P}_2$ take actions D and C, the receive a payoff of 4, and 0 respectively. Suppose $\rho : \tau_0 \tau_1 \tau_2 \dots$ denotes an accepting sequence of transitions (or a run) in the profile s.t. the payoffs received by agent $\mathcal{P}_i$ are given by $P : p_0^i p_1^i p_2^i \dots$, then the payoff of $\mathcal{P}_i$ from the game, $\mathbf{P}^i(\rho) = DS(P, d) = \Sigma_{j=0}^{\infty} p_j^i / d^j$.

3.1 Solution concepts in regular repeated games

In this section, we define the solution concepts of Nash equilibria and best response strategy for repeated regular games.

Definition 1. *A strategy profile $\overline{\mathcal{M}}$ is a* Nash equilibrium *if for each agent $\mathcal{P}_i$, for each strategy $\mathcal{M}_*^i \in Strategy(i)$, there exists a run $\overline{\rho} \in \overline{\mathcal{M}}$ such that for all other runs $\overline{\rho'} \in \overline{\mathcal{M}}[i := \mathcal{M}_*^i]$, $\boldsymbol{P}^i(\overline{\rho}) \geq \boldsymbol{P}^i(\overline{\rho'})$.*

A run $\overline{\rho} \in \overline{\mathcal{M}}$ is a *non-Nash run* if there exists another strategy profile $\overline{\mathcal{M}}'$ s.t. the strategy profiles differ in strategy of exactly one agent, say $\mathcal{P}_i$, and there exists a run $\overline{\rho'} \in \overline{\mathcal{M}}'$, s.t. $\mathbf{P}^i(\overline{\rho}) < \mathbf{P}^i(\overline{\rho'})$. A run is a *Nash run* otherwise. A strategy profile $\overline{\mathcal{M}_*}$ that demonstrates that profile $\overline{\mathcal{M}}$ is not in Nash equilibria is called a *witness* of $\overline{\mathcal{M}}$.

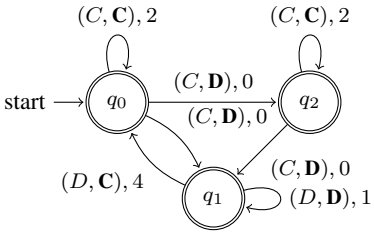

Figure 1: Strategy in a 2-agent RRG

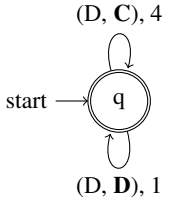

Figure 2: Strategy $\mathcal{M}^1$

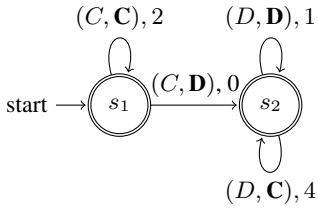

Figure 3: Strategy $\mathcal{M}^2$

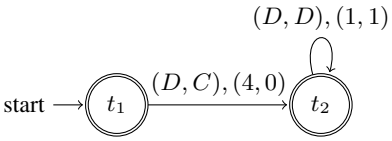

Figure 4: Strategy profile $(\mathcal{M}^1, \mathcal{M}^2)$

Definition 2. *A strategy $\mathcal{M}^i_*$ is said to be a* best response *to the environment action w.r.t. $\mathcal{P}_i$, $\overline{\mathcal{M}}[i = \emptyset]$, if for all other strategies $\mathcal{M}^i \in Strategy(i)$, there exists a run $\overline{\rho} \in \overline{\mathcal{M}}[i = \mathcal{M}^i_*]$ s.t. for all runs $\overline{\rho'} \in \overline{\mathcal{M}}[i = \mathcal{M}^i]$, $\boldsymbol{P}^i(\rho) \geq \boldsymbol{P}^i(\rho')$.*

We make the following interesting observation between Nash equilibria and the best response strategy in an RRG.

Lemma 1. *Let $\mathcal{G}$ be a 2-agent game. Suppose $\mathcal{P}_2$ receives a constant payoff in each strategy. Then, $(\mathcal{M}^1, \mathcal{M}^2)$ is in Nash equilibria in $\mathcal{G}$ iff $\mathcal{M}^1$ is a best response to the environment w.r.t $\mathcal{P}_1$.*

3.2 Size of regular repeated games

Definition 1 and Definition 2 are both defined for/on strategy profiles. Therefore, in all analysis that follows, we consider strategy profiles, *not* strategies, to be the fundamental unit for analysis. Hence we denote the size of a game w.r.t. the number and size of strategy profiles in the game.

Let SP denote the set of all strategy profiles in a given game $\mathcal{G}$. We define $|\mathcal{G}| = \Sigma_{S \in SP}|S|$, where strategy profiles are represented in their automaton form. We observe that the size of a strategy profile is proportional to the product of the size of each component strategy. Let $S_i \in Strategy(i)$ denote the largest strategy for $\mathcal{P}_i$. Then the size of each strategy profile S, $|S|$, is given by $\emptyset(\Pi_{i=1}^k |S_i|)$. Also the number of strategy profiles in game $|\mathcal{G}|$ is given by $\emptyset(\Pi_{i=1}^k |Strategy(i)|)$. Together this implies $|\mathcal{G}| = \Sigma_{S \in SP}|S| \implies |\mathcal{G}| = \emptyset(\Pi_{i=1}^k |Strategy(i)|) \cdot \emptyset(\Pi_{i=1}^k |S_i|)$.

4. Methodology

We design a provably correct algorithm ComputeNash to compute and enumerate all the Nash equilibria in a given regular repeated games. The input regular repeated games is represented by the set of strategies for each agent in the game.

ComputeNash executes by determining the existence of a witness for each strategy profile. The algorithm deems a profile to be in Nash equilibria if it fails to find a witness for it, else the profile is not in Nash equilibria.

The crux of ComputeNash lies in the procedure that checks if a profile W is a witness of another profile P. In this step ComputeNash is required to compare the payoff (computed as the discounted sum) of agents along the infinite executions in W and P. To this end, we construct a *comparator automaton*. The comparator automaton accepts a pair of bounded rational number sequences (A, B) iff $DS(A, d) < DS(B, d)$ when $d > 1$. This construction uses insights from arithmetic over numbers in base d, and the fact that discounted sum of bounded sequences is bounded.

We demonstrate how ComputeNash determines if W is a witness of P on the simple case when both strategy profiles consist of a single accepting run. Suppose profiles P and W are given as in Figure 5 and Figure 6 respectively. First, we take the product $\hat{\mathcal{A}}^{prod}$ of P with W (See Figure 7). Suppose, the discount factor $d = 2$. Let $\mathcal{A}_{\succ_{DS(2)}}$ denote the comparator automaton for $d = 2$. We take the intersection of $\hat{\mathcal{A}}^{prod}$ with $\mathcal{A}_{\succ_{DS(2)}}$ w.r.t. both agents. This intersection is empty for $\mathcal{P}_1$ since for the reward of $\mathcal{P}_1$ on P is greater than that on W, as shown by $DS(\{4, 1, 1 \dots\}, 2) > DS(\{2, 2, \dots\}, 2)$. But the reward of agent $\mathcal{P}_2$ is greater on W as $DS(\{0, 1, 1 \dots\}, 2) < DS(\{2, 2, \dots\}, 2)$. Therefore, $\hat{\mathcal{A}}^{prod} \cap^*_2 \mathcal{A}_{\succ_{DS(2)}}$ is non-empty (See Figure 8). This proves that the run in profile P is a non-Nash run. Since there is a single run in P, this also proves that profile W is a witness of P. Hence, P is not in Nash equilibria.

In the case when profile P has multiple accepting runs, concluding that P is not in Nash equilibria is not as straightforward. To conclude that P is not in Nash equilibria, we have to ensure that every accepting run in P is a non-Nash run. A combination of automata-theoretic operations, involving taking the projection of $\hat{\mathcal{A}}^{prod} \cap^*_i \mathcal{A}_{\succ_{DS(d)}}$ along the first component to construct $\overline{Nash}$ (Figure 9), and then equating $\overline{Nash}$ with profile P, are employed to ensure that P is not in Nash equilibria.

The worst case complexity of ComputeNash is polynomial in the size of the game when the game is deterministic (corresponds to the single accepting run strategy profiles), and exponential in the size of the game otherwise. The exponential blow up occurs due to one automata-theoretic equivalence step, which is known to be exponential in the worst case. However, in practice our algorithm works much faster, as we will demonstrate in Section 5.

We combine the algorithm for ComputeNash and the result of Lemma 1 to design an algorithm for computing and enumerating all best response strategies for an agent $\mathcal{P}_i$. To compute the best response strategies of an agent $\mathcal{P}_i$, we assign zero payoff on all transitions of all strategies of all agents except for agent $\mathcal{P}_i$, and then run ComputeNash over the modified game.

5. Results

To demonstrate the practical utility of ComputeNash for automatically verifying incentive compatibility in systems such as auctions and the Bitcoin protocol, we implemented a prototype of ComputeNash in Python. We employed existing tools, GOAL [29] and RABIT-Reduce [30], for operations over Büchi automata. Each experiment was conducted on a single core of a 12 core 2.2GHz Intel Xeon processor with 64 GB RAM.

In the following, we discuss the model, and experimental results for two case studies: Bitcoin protocol, and repeated auctions.

5.1 Bitcoin Protocol

Bitcoins are one of the most widely used on-line currency in today's time. The protocol by which Bitcoins are *mined* (minted) is called the Bitcoin protocol. Agents (miners) of the protocol receive rewards by mining more Bitcoins. Our first observation is that since the Bitcoin protocol is a multi-agent system with reward maximizing agents, it is a repeated game. Given this insight, we first model the Bitcoin protocol as a regular repeated game. Next, we perform

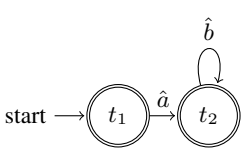

Figure 5: Profile P

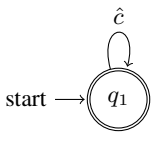

Figure 6: Profile W

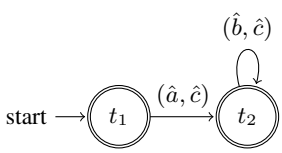

Figure 7: $\hat{\mathcal{A}}^{prod} = P \times W$

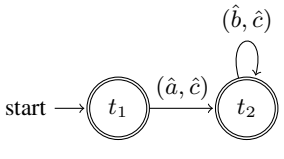

Figure 8: $Witness = \hat{\mathcal{A}}^{prod} \cap^*_2 \mathcal{A}_{\succ_{DS(2)}}$

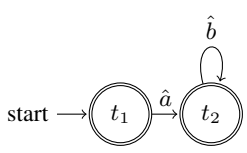

Figure 9: $\overline{Nash}$

$$\hat{a} = ((D, C), (4, 0)), \hat{b} = ((D, D), (1, 1)), \hat{c} = ((C, C), (2, 2))$$

automated analysis of the Bitcoin game via ComputeNash. We observed that in one of the Nash equilibrium returned by the algorithm, an agent adhered to dishonest policies in the protocol. This proves that the Bitcoin protocol is not incentive compatible.

It is worth noting that the same result was earlier proved by Eyal and Sirer [12]. However, unlike our approach, theirs was via manual analysis of the system. This precisely exhibits the potential of our approach. By careful modeling of such systems as regular repeated games, one can perform foolproof automated analysis of complex systems, and derive important results on them.

In the rest of this section, we describe our model of the Bitcoin protocol. We compute the set of Nash equilibria on the Bitcoin game using ComputeNash, which demonstrates that acting dishonestly can favor miners.

Game Model We follow Eyal and Sirer's model in that our model for the Bitcoin protocol is a 3-agent repeated game between an honest miner, a dishonest miner, and a scheduler. Unlike their model, we model the scheduler as non-deterministic, while they model it is probabilistic.

Both miners continuously mine at the tip of a linear chain of blocks, which is represented by action s to denote search operation. The scheduler performs one of the three actions, n, o, or t indicating whether none, $\mathcal{P}_1$ or $\mathcal{P}_2$ receives the block as reward. Miners perform action r to release the information of receiving reward from the scheduler. Miners do not expend their resources on an already mined block. Therefore, following honest policies each agent declares receiving a block by performing r immediately after receiving it. However, if a miner adopts the dishonest policies, then it will hoard the block h to waste resources of the other miner. We limit the hoarding capacity of a dishonest miner to a single block.

In our model of the protocol, we assume that $\mathcal{P}_1$ is honest, while $\mathcal{P}_2$ is dishonest. Then, actions of $\mathcal{P}_1$ and $\mathcal{P}_2$ are s, r and s, r, and h respectively. Correspondingly, miners have two strategies, one in which they act honestly, called Honest (Figure 10), and the other in which they act dishonestly, called Dishonest (Figure 11).

Under the Honest strategy, both miners begin with searching for a mine in state q_1. State q_1 denotes the state where neither agent has received a mine. When the honest miner receives a mine, control shifts to state q_2 if the dishonest miner doesn't hoard a block, and to state q_4 otherwise. Control shifts to state q_3 when the dishonest agent receives its first block. From this state, the dishonest agent may choose to release the mine immediately or hoard it. Due to its limited hoarding capacity, the dishonest agent, control shifts to state q_5 if it receives more blocks. At q_5 the dishonest miner is forced to release all but one block. If the honest miner receives a block while the dishonest miner hoards one, control shifts to state q_4. If both miners release their mine at the same time, then the block is non-deterministically rewarded to one miner by the scheduler.

Dishonest strategy for miners is illustrated in Figure 11.

Rewards The scheduler is a passive agent, and does not receive any reward. A miner receives reward of 1 if it releases a block in a state other than q_4 and 2 for releasing a block from state q_4. The higher reward of 2 indicates the success of the dishonest miner in wasting other miners resources – a tactic that a dishonest miner would pursue. For all other actions, the reward is 0 for miners. The dotted transition denote the non-deterministic reward when the scheduler non-deterministically chooses between which agent to reward.

Experiment and Result We assign $Strategy(1) = \{\mathsf{Honest}\}$, and $Strategy(2) = \{\mathsf{Honest}, \mathsf{Dishonest}\}$. ComputeNash returned $\{(\mathsf{Honest}, \mathsf{Dishonest})\}$ on the game. Since the miner plays dishonestly in the Nash equilibria, this game is not incentive compatible. ComputeNash returned result in less than 30 sec on the game.

5.2 Repeated Auctions

An auction protocol is said to be incentive compatible if the best response strategy of agents is to bid at their true valuation of the item. In this section we employ the algorithm for computing best response strategies in a game to determine whether an auction protocol is incentive compatible or not. We prove via automated analysis that the repeated first-price auction is not incentive compatible, while repeated second-price auction is incentive compatible. Our results conform with previously known literature on these auctions.

We model an auction as a 2-agent game. Our objective is to determine the best response strategies for the first bidder. We illustrate strategies of the bidders in Figure 12. In $\mathsf{All}(v_1, v_2)$, $\mathcal{P}_1$ bids at all values between 0 and v_1. In $\mathsf{Always}(v_1, v_2)$, $\mathcal{P}_1$ bids at v_1 only. $\mathsf{MostlyDishonest}(v_1, v_2)$ and $\mathsf{MostlyHonest}(v_1, v_2)$ denote strategies in which $\mathcal{P}_1$ may not always bid at v_1. In $\mathsf{MostlyDishonest}(v_1, v_2)$, $\mathcal{P}_i$ bids at values lesser than v_1 infinitely often. In $\mathsf{MostlyHonest}(v_1, v_2)$, $\mathcal{P}_i$ bids at v_1 infinitely often. In all of these strategies $\mathcal{P}_2$ can bid at all values ranging from 0 to v_2. We define $Strategy(1) = \{$ $\mathsf{All}(v_1, v_2)$, $\mathsf{All}(v_1 - 1, v_2)$, $\mathsf{Always}(v_1, v_2)$, $\mathsf{MostlyDishonest}(v_1, v_2)$, $\mathsf{MostlyHonest}(v_1, v_2)$ $\}$, and $Strategy(2) = \{\mathsf{All}(v_2, v_1)\}$. Both our auction experiments completed in ~ 3 muns

Repeated first-price auctions The auction rule for $\mathcal{P}_1$, when $\mathcal{P}_1$ and $\mathcal{P}_2$ bid at u_1 and u_2 respectively, is given as follows:

$$FirstPriceAuction(u_1, u_2) = \begin{cases} 1 & \text{if } u_1 > u_2 \\ 0 & \text{if } u_1 < u_2 \\ 0 \text{ or } 1 & \text{if } u_1 == u_2 \end{cases}$$

The best response strategies for $\mathcal{P}_1$ turned out to be $\mathsf{All}(v_1, v_2)$, $\mathsf{All}(v_1 - 1, v_2)$, $\mathsf{Always}(v_1, v_2)$, $\mathsf{MostlyDishonest}(v_1, v_2)$, and $\mathsf{MostlyHonest}(v_1, v_2)$. $\mathsf{All}(v_1 - 1, v_2)$ and $\mathsf{MostlyDishonest}(v_1, v_2)$ as best response strategies indicate that $\mathcal{P}_1$ receives highest reward even when she always bids at values lower than her true valuation. This proves that the first-price auction is not incentive compatible.

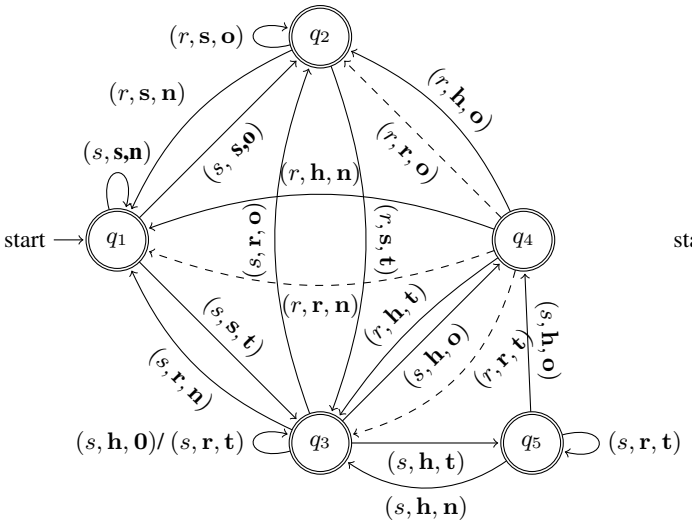

Figure 10: Honest strategy in Bitcoin protocol. Payoffs explained in text.

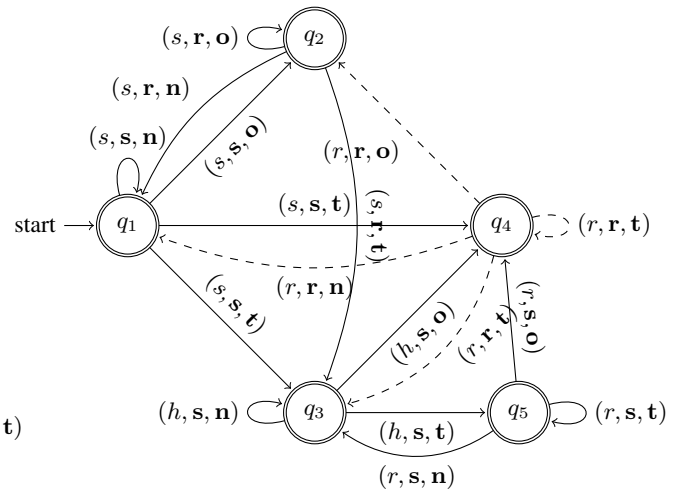

Figure 11: Dishonest strategy in Bitcoin protocol. Payoffs explained in text.

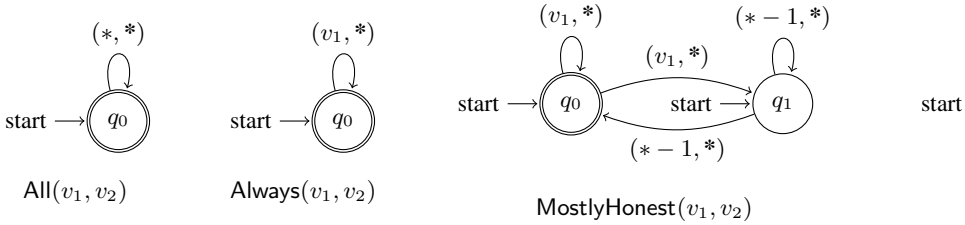

Figure 12: Auction Strategies for $\mathcal{P}_1$ in 2-agent auction. True valuation of $\mathcal{P}_1$ = v_1, True valuation of $\mathcal{P}_2$ = v_2. Gradient: $* : \{0, 1 \ldots v_1\}$, $* - 1 : \{0, 1 \ldots v_1 - 1\}$, * : $\{\mathbf{0}, \mathbf{1} \ldots \mathbf{v_2}\}$. For payoff, see the corresponding auction rule.

Repeated second-price auctions The auction rule for $\mathcal{P}_1$, when $\mathcal{P}_1$ and $\mathcal{P}_2$ bid at u_1 and u_2 respectively, is given as follows:

$$VickeryAuction(u_1, u_2) = \begin{cases} u_1 - u_2 & \text{if } u_1 \geq u_2 \\ 0 & \text{otherwise} \end{cases}$$

The best response strategies for $\mathcal{P}_1$ turned out to be $\mathsf{Always}(v_1, v_2)$, and $\mathsf{All}(v_1, v_2)$. Therefore, $\mathcal{P}_1$ does not receive high reward by always bidding at lower than v_1. This proves that repeated second-price auctions are incentive compatible.

6. Conclusion

In this paper, we introduced a new framework for repeated games, regular repeated games. We discussed algorithms with crisp complexity to compute all Nash equilibria and best response strategies for games in this framework. We utilized these algorithms to perform automated analysis of incentive compatibility on complex multi-agent systems with reward maximizing agents.

Acknowledgements

I would like to acknowledge Swarat Chaudhuri and Kuldeep S. Meel, my collaborators on this work.

References

[1] Dilip Abreu. On the theory of infinitely repeated games with discounting. *Econometrica*, pages 383–396, 1988.

[2] Dilip Abreu, David Pearce, and Ennio Stacchetti. Toward a theory of discounted repeated games with imperfect monitoring. *Econometrica*, pages 1041–1063, 1990.

[3] Garrett Andersen and Vincent Conitzer. Fast equilibrium computation for infinitely repeated games. In *Proc. of AAAI*, pages 53–59, 2013.

[4] David Basanta, Haralambos Hatzikirou, and Andreas Deutsch. Studying the emergence of invasiveness in tumours using game theory. *The European Physical Journal B*, 63(3):393–397, 2008.

[5] Kimmo Berg and Mitri Kitti. Computing equilibria in discounted 2× 2 supergames. *Computational Economics*, 41(1):71–88, 2013.

[6] Kimmo Berg, Mitri Kitti, et al. Equilibrium paths in discounted supergames. Technical report, Working paper, 2012.

[7] Andriy Burkov and Brahim Chaib-draa. An approximate subgame-perfect equilibrium computation technique for repeated games. In *Proc. of AAAI*, pages 729–736, 2010.

[8] Colin Camerer. *Behavioral game theory*. New Age International, 2010.

[9] Andrew M Colman. *Game theory and its applications: in the social and biological sciences*. Psychology Press, 2013.

[10] Mark B Cronshaw. Algorithms for finding repeated game equilibria. *Computational Economics*, 10(2):139–168, 1997.

[11] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[12] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[13] James W Friedman. *Game theory with applications to economics*. Oxford University Press New York, 1990.

[14] Drew Fudenberg and Eric Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica*, 54(3):533–554, 1986.

[15] Kenneth L Judd, Sevin Yeltekin, and James Conklin. Computing supergame equilibria. *Econometrica*, pages 1239–1254, 2003.

[16] Michael L Littman and Peter Stone. A polynomial-time nash equilibrium algorithm for repeated games. *Decision Support Systems*, 39(1):55–66, 2005.

[17] George J. Mailath and Larry Samuelson. *Repeated games and reputations: Long-running relationships*. Oxford University Press, 2006.

[18] Matthieu Manceny, A Lackmy, C Chettaoui, F Delaplace, and Cours Monseigneur Romero. Application of game theory to gene networks analysis. 2005.

[19] Robert E Marks. *Repeated games and finite automata*. Australian Graduate School of Management, University of New South Wales, 1990.

[20] Paul R Milgrom and Robert J Weber. A theory of auctions and competitive bidding. *Econometrica*, pages 1089–1122, 1982.

[21] James D Morrow. *Game theory for political scientists*. Princeton University Press Princeton, NJ, 1994.

[22] Herve Moulin. *Game theory for the social sciences*. NYU press, 1986.

[23] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[24] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[25] Martin J Osborne and Ariel Rubinstein. *Bargaining and markets*. Academic press, 1990.

[26] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[27] Ariel Rubinstein. Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory*, 39(1):83–96, 1986.

[28] Herbert A Simon. A behavioral model of rational choice. *The quarterly journal of economics*, pages 99–118, 1955.

[29] GOAL. GOAL. `http://goal.im.ntu.edu.tw/wiki/`.

[30] RABIT-REDUCE. Rabit-reduce. `http://www.languageinclusion.org/`.

[31] Wolfgang Thomas, Thomas Wilke, et al. *Automata, logics, and infinite games: a guide to current research*, volume 2500. Springer Science & Business Media, 2002.

[32] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton university press, 2007.

[33] Jörgen W Weibull. *Evolutionary game theory*. MIT press, 1997.