Reasoning about Incentive Compatibility

Suguman Bansal
Rice University
suguman@rice.edu

Abstract

We study equilibrium computation in weighted regular games, a model of infinitely repeated, discounted, general-sum games where the space of agent strategies is given by a finite automaton. Our goal is to compute, for subsequent querying and analysis, the set of all pure-strategy Nash equilibria in such a game. Our technical contribution is an algorithm for computing an automaton representation of this set. We use our algorithm for checking properties of rational behavior such as incentive compatibility in complex multi-agent systems.

1. Introduction

Incentive compatibility, more commonly known as *truthfulness* of agents, is crucial to the success of most multi-agent systems. Incentive compatible systems are trusted to be fair to all agents despite the presence of *selfish agents* (self-reward maximizing). Consider auctions systems; agents (bidders) may be reluctant to bid at their actual valuation of the item to avoid overbidding. By bidding at a lower price, they act dishonestly. In an i-th price auction, the highest bidder wins, and pays the price of the i-th largest bid. Second-price auctions are incentive compatible, while first-price auctions are not. For long, conventional wisdom held that the Bitcoin protocol is incentive compatible. However, Eyal and Sirer recently disproved this via rigorous mathematical analysis [1] (See Section 2).

As demonstrated above, erroneous system design is common. Therefore, analysis of incentive compatibility, more generally rational behavior, is critical. Such systems can be modeled as non-zero sum games between selfish agents. From the PL perspective, they can be treated as *quantitative abstractions* of systems of selfish agents where quantitative utilities capture agent motives.

In this vein, we are interested in automating analysis of rational behavior. More specifically, we introduce *weighted regular games* to model such systems. We use techniques from formal methods to emulate rigorous mathematical analysis of these program abstractions, and compute the set of all *Nash equilibria*, a rational behavior solution concept in *non-zero sum games*.

2. Weighted Regular Games (WRG)

Non-zero sum games differ from zero-sum games, traditionally seen in verification and synthesis of software systems, in two major ways: (a). an agent need not always gain at the expense of another agent, and (b). agents receive rational rewards, called utility. We introduce WRGs as a model for non-zero sum games.

WRGs denote n-player games. Syntactically, WRGs form büchi automaton over n-tuple alphabets, called *action profile*. Each transition is labeled by at least one n-tuple, called the *weight tuple*. Each accepting word denotes a *strategy profile* in the game.

The i-th element of each action profile, weight profile and strategy profile denotes the *action* of the i-th agent, the rational value of *utility* attained by the i-th agent, and strategy of the i-th agent respectively.

Utility of agents over accepting runs is computed as the *discounted sum* of the agent's utilities along transitions on that run. The discounted sum of sequence A with discount factor d is given by $DS(A, d) = \Sigma_{i=0}^{\infty} a_i/d^i$.

A *Nash equilibrium* of a game is a strategy profile such that if any agent unilaterally changes its strategy, then it will receive lower aggregate utility. Since a strategy profile may have more that one accepting run, there may be multiple utilities for its agents. In this case, a strategy profile is in Nash equilibrium if it has at least one accepting run that is in Nash equilibrium.

Note, a protocol is proven to be non-incentive compatible if it has a Nash equilibria in which some agent plays dishonestly.

Illustrative example Consider a model for the Bitcoin protocol (See Figure 1), a 3-player game between a dishonest miner P_1, an honest miner P_2, and a scheduler P_3. Each miner continuously mines for a bitcoin (action s), and releases a bitcoin after it finds one (action r). P_2 immediately releases a bitcoin after finding one. P_1 may hoard (action h) a bitcoin after finding it to waste P_2's resources. For simplicity, we assume that P_1 can hoard a single bitcoin at any time. A miner finds a bitcoin only after P_3 rewards it. Actions 0, 1, and 2 denote that none, P_1, and P_2 have been rewarded respectively. Utilities along transitions are shown in Table 1.

3. Methodology

We have designed a provably correct algorithm to compute a büchi automaton for all Nash equilibria in a given WRG with discount factor $d \in \mathbb{N}$. This algorithm is exponential in number of states in the input WRG.

We first compute the set of all strategy profiles that are not in Nash equilibrium, then complement it to obtain all Nash equilibria. The exponential blow-up in comlexity arises from this step. Computation of all non-Nash equilibria involves two major constructions:

1. Construction of a *comparator automaton* that accepts a pair of bounded rational number sequences (A, B) iff $DS(A, d) > DS(B, d)$ when $d > 1, d \in \mathbb{N}$. For this, we use insights from arithmetic over numbers in base d, and that discounted sum of bounded sequences is bounded.
2. Construction of an automaton that accepts pair of strategy profiles (Π, Π') iff they are unilaterally deviated from each other for a single player P_i.

The idea is that if (Π, Π') are unilaterally deviated for P_i, and utility of P_i is greater on Π, then Π' is not in Nash equilibrium.

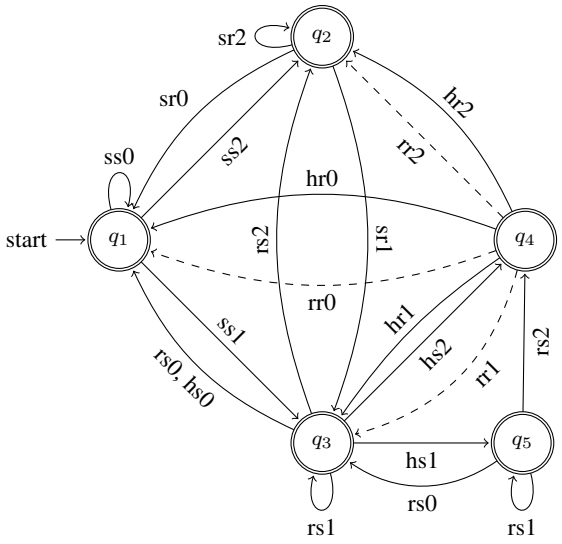

Figure 1: Bitcoin Protocol Game Model. xyz denotes action profile (x, y, z). Weight Tuples defined as in Table 1.

From q_1, q_2, q_3, and q_5

Action	r	s, h, 0, 1, 2
Action Utility	1	0

From q_4

Action Profile	$(h, r, *)$	$(r, r, *)$
Weight Tuple	(0,1,0)	(2,0,0) or (0,1,0)

Table 1: Utilities for agents in Figure 1

Note: 1. P_1 receives greater utility from q_4 since it wastes P_2's resources.
2. Two weight tuples on $(r, r, *)$ is due to tie break.

4. Results

We implemented a prototype of our algorithm in Python using existing tools for Büchi operations (GOAL [2], RABIT-Reduce [3]).

We computed the set of all Nash equilibria in the Bitcoin Protocol. The resultant automaton contained 13 states and 57 transitions. Clearly, this cannot be computed manually. From this automaton, we concluded that the Bitcoin Protocol is not incentive compatible. We made similar observations for various auction system. Additionally, we re-discovered, and made some new observations on the Nash equilibria in the classical Iterated Prisoner Dilemma game.

5. Related Work

There is a large literature on non-zero sum infinite games with discounted-sum payoffs as a standard model [4–6]. The seminal work by Abreu, Pearce, and Stacchetti [7], and its followups [8], are restricted to computing all or some equilibrium payoffs rather than strategy profiles.

The problem of efficiently computing a *single* equilibrium strategy in infinite games has been studied before [9–11]. There are also a few approaches to computing representations of *approximations* to the set of equilibria [12–14]. Except for work on finding one equilibrium, most other approaches do not guarantee crisp complexity bounds.

References

[1] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*, pp. 436–454, Springer, 2014.

[2] GOAL, "GOAL." `http://goal.im.ntu.edu.tw/wiki/`.

[3] RABIT-REDUCE, "Rabit-reduce." `http://www.languageinclusion.org/`.

[4] G. J. Mailath and L. Samuelson, *Repeated games and reputations: Long-running relationships*. Oxford University Press, 2006.

[5] D. Abreu, "On the theory of infinitely repeated games with discounting," *Econometrica: Journal of the Econometric Society*, pp. 383–396, 1988.

[6] D. Fudenberg and E. Maskin, "The folk theorem in repeated games with discounting or with incomplete information," *Econometrica: Journal of the Econometric Society*, vol. 54, no. 3, pp. 533–554, 1986.

[7] D. Abreu, D. Pearce, and E. Stacchetti, "Toward a theory of discounted repeated games with imperfect monitoring," *Econometrica: Journal of the Econometric Society*, pp. 1041–1063, 1990.

[8] M. B. Cronshaw, "Algorithms for finding repeated game equilibria," *Computational Economics*, vol. 10, no. 2, pp. 139–168, 1997.

[9] M. L. Littman and P. Stone, "A polynomial-time nash equilibrium algorithm for repeated games," *Decision Support Systems*, vol. 39, no. 1, pp. 55–66, 2005.

[10] G. Andersen and V. Conitzer, "Fast equilibrium computation for infinitely repeated games," in *Proc. of AAAI*, pp. 53–59, 2013.

[11] K. L. Judd, S. Yeltekin, and J. Conklin, "Computing supergame equilibria," *Econometrica: Journal of the Econometric Society*, pp. 1239–1254, 2003.

[12] A. Burkov and B. Chaib-draa, "An approximate subgame-perfect equilibrium computation technique for repeated games," in *Proc. of AAAI*, pp. 729–736, 2010.

[13] K. Berg and M. Kitti, "Equilibrium paths in discounted supergames," tech. rep., Working paper, 2012.

[14] K. Berg and M. Kitti, "Computing equilibria in discounted $2\times$ 2 supergames," *Computational Economics*, vol. 41, no. 1, pp. 71–88, 2013.