The Computational Complexity of Nash Equilibria in Concisely Represented Games *

Grant Schoenebeck†
Computer Science Division
University of California, Berkeley
Berkeley, California 94720
grant@cs.berkeley.edu

Salil Vadhan‡
Division of Engineering & Applied Sciences
Harvard University
Cambridge, MA 02138
salil@eecs.harvard.edu

ABSTRACT

Games may be represented in many different ways, and different representations of games affect the complexity of problems associated with games, such as finding a Nash equilibrium. The traditional method of representing a game is to explicitly list all the payoffs, but this incurs an exponential blowup as the number of agents grows.

We study two models of concisely represented games: *circuit games*, where the payoffs are computed by a given boolean circuit, and *graph games*, where each agent's payoff is a function of only the strategies played by its neighbors in a given graph. For these two models, we study the complexity of four questions: determining if a given strategy is a Nash equilibrium, finding a Nash equilibrium, determining if there exists a pure Nash equilibrium, and determining if there exists a Nash equilibrium in which the payoffs to the players meet some given guarantees. In many cases, we obtain tight results, showing that the problems are complete for various complexity classes.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms

Theory, Economics

*Many of these results have appeared in the first author's undergraduate thesis [20]. A preliminary version of this paper has appeared on *ECCC* [19].

†Supported by NSF grant CCR-0133096.

‡Work done in part while a Fellow at the Radcliffe Institute for Advanced Study. Also supported by NSF grant CCR-0133096 and a Sloan Research Fellowship.

EC'06, June 11–15, 2006, Ann Arbor, Michigan.

Keywords

Nash equilibrium, concise games, graph games, circuit games, computational game theory

1. INTRODUCTION

A central topic at the interface of computer science and economics is understanding the complexity of computational problems involving equilibria in games. While these types of questions are already interesting (and often difficult) for standard two-player games presented in "bimatrix form" [16, 3, 9, 15], many of the current motivations for such games come from settings where there are many players (e.g. the Internet) or many strategies (e.g. combinatorial auctions). In n-player games and in games with many strategies, the representation of the game becomes an important issue. In particular, explicitly describing an n-player game in which each player has only two strategies requires an exponentially long representation (consisting of $N = n \cdot 2^n$ payoff values). Thus the complexity of this problem is more natural for games given by some type of *concise* representation, such as the graph games recently proposed by Kearns, Littman, and Singh [14].

Motivated by the above considerations, we undertake a systematic study of the complexity of Nash equilibria in games given by concise representations. We focus on two types of concise representations. The first are *circuit games*, where the game is specified by a boolean circuit computing the payoffs. Circuit games were previously studied in the setting of two-player zero-sum games, where computing (resp., approximating) the "value" of such a game is shown to be **EXP**-complete [7] (resp., $\mathbf{S_2P}$-complete [8]). They are a very general model, capturing essentially any representation in which the payoffs are efficiently computable. The second are *graph games* [14], where the game is presented by a graph whose nodes are the players and the payoffs of each player are a function only of the strategies played by each player's neighbor. (Thus, if the graph is of low degree, the payoff functions can be written very compactly). Kearns et al. showed that if the graph is a tree and each player has only two strategies, then approximate Nash equilibria can be found in polynomial time. Gotlobb, Greco, and Scarcello [12] recently showed that the problem of deciding if a degree-4 graph game has a pure-Nash equilibrium is **NP**-complete.

In these two models (circuit games and graph games), we study 4 problems:

1. IsNash: Given a game G and a randomized strategy profile θ, determine if θ is a Nash equilibrium in G,

2. ExistsPureNash: Given a game G, determine if G has a pure (i.e. deterministic) Nash equilibrium,

3. FindNash: Given a game G, find a Nash equilibrium in G, and

4. GuaranteeNash: Given a game G, determine whether G has a Nash equilibrium that achieves certain payoff guarantees for each player. (This problem was previously studied by [9, 3], who showed it to be **NP**-complete for two-player, bimatrix games.)

We study the above four problems in both circuit games and graphical games, in games where each player has only two possible strategies and in games where the strategy space is unbounded, in n-player games and in 2-player games, and with respect to approximate Nash equilibria for different levels of approximation (exponentially small error, polynomially small error, and constant error).

Our results include:

- A tight characterization of the complexity of all of the problems listed above *except for* FindNash, by showing them to be complete for various complexity classes. This applies to all of their variants (w.r.t. concise representation, number of players, and level of approximation). For the various forms of FindNash, we give upper and lower bounds that are within one nondeterministic quantifier of each other.

- A general result showing that n-player circuit games in which each player has 2 strategies are a harder class of games than standard two-player bimatrix games (and more generally, than the graphical games of [14]), in that there is a general reduction from the latter to the former which applies to most of the problems listed above.

Independent/Subsequent Results. Several researchers have independently obtained some results related to ours. Specifically, Daskalakis and Papadimitriou [5] give complexity results on concisely represented graphical games where the graph can be exponentially large (whereas we always consider the graph to be given explicitly), and Alvarez, Gabarro, and Serna [AGS05] give results on ExistsPureNash that are very similar to ours.

In addition, there have been several important related results subsequent to the original versions of our work [20, 19]. In particular, Goldberg and Papadimitriou [10] give a reduction from degree-d graph games to d^2-player normal-form games.[1] Their reduction uses a technique similar to one in the original version of this paper [19]. Furthermore, [10] gives a reduction from d-player games to degree-3 boolean graph games, strengthening our second bullet above. Each of these reductions creates a natural map between Nash equilibrium. By composing these reductions, they show that for any constant C, the problem of finding a Nash equilibrium in a game with at most C players, referred to as C-Nash, can be reduced to the case where $C = 4$, 4-Nash. Subsequently, with Daskalakis [4], they prove that for $C \geq 4$ C-Nash is complete for the class **PPAD** [16]. Daskalakis and Papadimitriou [6] and Chen and Deng [1] independently prove the same result holds for $C = 3$. Chen and Deng [2] have now proved the same result holds even for $C = 2$.

[1] A normal-form game is one where the payoffs are explicitly specified for each possible combination player strategies.

Organization. We define game theoretic terminology and fix a representation of strategy profiles in Section 2. Section 3 contains formal definitions of the concise representations and problems that we study. Section 4 looks at relationships between these representations. Sections 5 through 8 contain the main complexity results on IsNash, ExistsPureNash, FindNash, and GuaranteeNash.

Due to space constraints, many proof are omitted.

2. BACKGROUND AND CONVENTIONS

Game Theory. A *game* $\mathcal{G} = (\mathfrak{S}, \nu)$ with n agents, or players, consists of a set $\mathfrak{S} = \mathfrak{S}_1 \times \cdots \times \mathfrak{S}_n$ where $\mathfrak{S}_i$ is the *strategy space* of agent i, and a *valuation* or *payoff function* $\nu = \nu_1 \times \ldots \times \nu_n$ where $\nu_i : \mathfrak{S} \to \mathbb{R}$ is the valuation function of agent i. Intuitively, to "play" such a game, each agent i picks a strategy $s_i \in \mathfrak{S}_i$, and based on all players' choices realizes the payoff $\nu_i(s_1, \ldots, s_n)$.

For us, $\mathfrak{S}_i$ will always be finite and the range of ν_i will always be rational. An *explicit* representation of a game $\mathcal{G} = (\mathfrak{S}, \nu)$ is composed of a list of each $\mathfrak{S}_i$ and an explicit encoding of each ν_i. This encoding of ν consists of $n \cdot |\mathfrak{S}| = n \cdot |\mathfrak{S}_1| \cdots |\mathfrak{S}_n|$ rational numbers. An explicit game with exactly two players is call a *bimatrix* game because the payoff functions can be represented by two matrices, one specifying the values of ν_1 on $\mathfrak{S} = \mathfrak{S}_1 \times \mathfrak{S}_2$ and the other specifying the values of ν_2.

A *pure strategy* for an agent i is an element of $\mathfrak{S}_i$. A *mixed strategy* θ_i, or simply a *strategy*, for a player i is a random variable whose range is $\mathfrak{S}_i$. The set of all strategies for player i will be denoted Θ_i. A *strategy profile* is a sequence $\theta = (\theta_1, \ldots, \theta_n)$, where θ_i is a strategy for agent i. We will denote the set of all strategy profiles Θ. $\nu = \nu_1 \times \cdots \times \nu_n$ extends to Θ by defining $\nu(\theta) = \mathbb{E}_{s \leftarrow \theta}[\nu(s)]$. A *pure-strategy profile* is a strategy profile in which each agent plays some pure-strategy with probability 1. A k*-uniform* strategy profile is a strategy profile where each agent randomizes uniformly between k, not necessarily unique, pure strategies. The *support* of a strategy (or of a strategy profile) is the set of all pure-strategies (or of all pure-strategy profiles) played with nonzero probability.

We define a function $R_i : \Theta \times \Theta_i \to \Theta$ that replaces the ith strategy in a strategy profile θ by a different strategy for agent i, so $R_i(\theta, \theta'_i) = (\theta_1, \ldots, \theta'_i, \ldots, \theta_n)$. This diverges from conventional notation which writes (θ_{-i}, θ'_i) instead of $R_i(\theta, \theta'_i)$.

Given a strategy profile θ, we say agent i is in *equilibrium* if he cannot increase his expected payoff by playing some other strategy (giving what the other $n-1$ agents are playing). Formally agent i is in equilibrium if $\nu_i(\theta) \geq \nu_i(R_i(\theta, \theta'_i))$ for all $\theta'_i \in \Theta_i$. Because $R_i(\theta, \theta'_i)$ is a distribution over $R_i(\theta, s_i)$ where $s_i \in \mathfrak{S}_i$ and ν_i acts linearly on these distributions, $R_i(\theta, \theta'_i)$ will be maximized by playing some optimal $s_i \in \mathfrak{S}_i$ with probability 1. Therefore, it suffices to check that $\nu_i(\theta) \geq \nu_i(R_i(\theta, s_i))$ for all $s_i \in \mathfrak{S}_i$. For the same reason, agent i is in equilibrium if and only if each strategy in the support of θ_i is an optimal response. A strategy profile θ is a *Nash equilibrium* [17] if all the players are in equilibrium. Given a strategy profile θ, we say player i is in *ϵ-equilibrium* if $\nu_i(R_i(\theta, s_i)) \leq \nu_i(\theta) + \epsilon$ for all $s_i \in \mathfrak{S}_i$. A strategy profile θ is an *ϵ-Nash equilibrium* if all the players are in ϵ-equilibrium. A *pure-strategy Nash equilibrium* (respectively, a *pure-strategy ϵ-Nash equilibrium*) is a pure-strategy profile which is a Nash equilibrium (respectively, an ϵ-Nash equilibrium).

A two-player is called a *zero-sum* game if $\nu_1(s) = -\nu_2(s)$ for all $s \in \mathfrak{S}$. Each *zero-sum* game $\mathcal{G}$ has a *game value* $\nu(\mathcal{G})$ such that in any Nash equilibrium θ, $\nu(\mathcal{G}) = \nu_1(\theta) = -\nu_2(\theta)$.

Pennies is a 2-player game where $\mathfrak{S}_1 = \mathfrak{S}_2 = \{0, 1\}$, and $\nu_1(s_1, s_2) = 1$ if $s_1 = s_2$ and 0 otherwise. $\nu_2(s_1, s_2) = 1$ if $s_1 \neq s_2$ and 0 otherwise.

Pennies has a unique Nash equilibrium where both agents randomize uniformly between their two strategies. In any ϵ-Nash equilibrium of 2-player pennies, each player randomizes between each strategy with probability $\frac{1}{2} \pm 2\epsilon$.

Complexity Theory. A *promise-language* L is a pair (L^+, L^-) such that $L^+ \subseteq \Sigma^*$, $L^- \subseteq \Sigma^*$, and $L^+ \cap L^- = \emptyset$. We call L^+ the *positive instances*, and L^- the *negative instances*. An algorithm decides a promise-language if it accepts all the positive instances and rejects all the negative instances. Nothing is required of the algorithm if it is run on instances outside $L^+ \cup L^-$.

Because we consider approximation problems in this paper, which are naturally formulated as promise languages, **all complexity classes used in this paper are classes of promise languages.** We refer the reader to the recent survey of Goldreich [11] for about the usefulness and subtleties of working with promise problems.

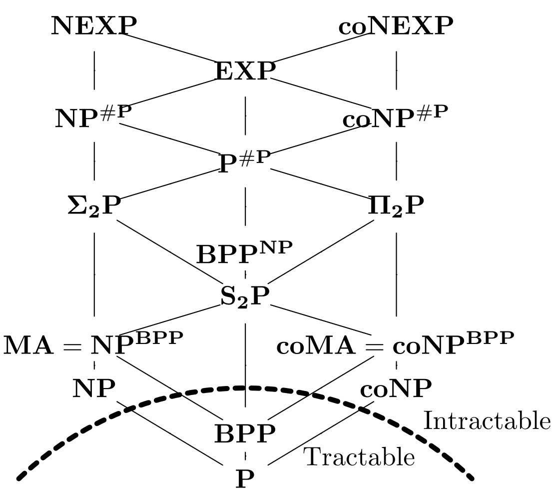

Figure 1: Relationships between complexity classes

Figure 1 shows the relationships between the complexity classes used in this paper. If a line connects two complexity classes in the figure, it indicates that the class lower on the page is contained in the complexity class higher on the page. **EXP** and **NEXP** are the classes of languages that can be decided in exponential time and nondeterministic exponential time, respectively. Both of these classes are provably not equal to **P**. **#P** is the class of **NP** counting functions. Functions in these class answer how many accepting computations a nondeterministic polynomial-time Turing machine has (rather than if one accepting computation exists, like **NP**). While **#P** contains **NP**, **#P** is generally thought to also contain much harder problems than **NP**. **BPP** is the class of languages that can be computed with by a randomized algorithm with two-sided error in polynomial time. **BPP** is generally considered a class of tractable problems. $\tilde{\mathbf{P}}$, called quasipolynomial time, contains languages that can be deterministically decided in time $2^{\text{poly}(\log(|x|))}$ on input x. $\tilde{\mathbf{P}}$, usually considered nearly tractable, clearly contains **P** and is contained by **EXP**. A promise language L is in $\mathbf{S_2P}$ if there exists a polynomial time computable and polynomially bounded relation $R \subset \Sigma^* \times \Sigma^* \times \Sigma^*$ such that:

1. If $x \in L^+$ then $\exists\, y$ such that $\forall\, z$, $R(x, y, z) = 1$.
2. If $x \in L^-$ then $\exists\, z$ such that $\forall\, y$, $R(x, y, z) = 0$.

If $\mathcal{C}$ and $\mathcal{D}$ are two complexity classes then the complexity class $\mathcal{C}^{\mathcal{D}}$ is the set of languages that can be decided in $\mathcal{C}$ augmented by oracle access to any particular language in $\mathcal{D}$. For example, a language is in $\mathbf{\Sigma_2 P} = \mathbf{NP^{NP}}$ if it can be decided by a nondeterministic polynomial-time Turning machine with oracle access to SAT.

A *search problem*, is specified by a relation $R \subseteq \Sigma^* \times \Sigma^*$ where given an $x \in \Sigma^*$ we want to either compute $y \in \Sigma^*$ such that $(x, y) \in R$ or say that no such y exists. When using a search problem as an oracle, it is required that *any* valid response from the oracle yields a correct answer.

3. CONCISE REPRESENTATIONS AND PROBLEMS STUDIED

We now give formal descriptions of the problems which we are studying. First we define the two different representations of games.

Definition 3.1. *A* circuit game *is a game $\mathcal{G} = (\mathfrak{S}, \nu)$ specified by integers $k_1, \ldots, k_n$ and circuits $C_1, \ldots, C_n$ such that $\mathfrak{S}_i \subseteq \{0, 1\}^{k_i}$ and $C_i(s) = \nu_i(s)$ if $s_i \in \mathfrak{S}_i$ for all i or $C_i(s) = \perp$ otherwise.*

In a game $\mathcal{G} = (\mathfrak{S}, \nu)$, we write $i \propto j$ if $\exists s \in \mathfrak{S}, s_i' \in \mathfrak{S}_i$ such that $\nu_j(s) \neq \nu_j(R_i(s, s_i'))$. Intuitively, $i \propto j$ if agent i can ever influence the payoff of agent j.

Definition 3.2. *[14] A* graph game *is a game $\mathcal{G} = (\mathfrak{S}, \nu)$ specified by a directed graph $G = (V, E)$ where V is the set of agents and $E \supseteq \{(i, j) : i \propto j\}$, the strategy space $\mathfrak{S}$, and explicit representations of the function ν_j for each agent j defined on the domain $\prod_{(i,j) \in E} \mathfrak{S}_i$, which encodes the payoffs. A* degree-d graph game *is a graph game where the in-degree of the graph G is bounded by d.*

This definition was proposed in [14]. We change their definition slightly by using directed graphs instead of undirected ones (this only changes the constant degree bounds claimed in our results).

Note that any game (with rational payoffs) can be represented as a circuit game or a graph game. However, a degree-d graph game can only represent games where no one agent is influenced directly by the strategies of more than d other agents.

A circuit game can encode the games where each player has exponentially many pure-strategies in a polynomial amount of space. In addition, unlike in an explicit representation, there is no exponential blow-up as the number of agents increases. A degree-d graph game, where d is constant, also avoids the exponential blow-up as the number of agents increases. For this reason we are interested mostly in bounded-degree graph games.

We study two restrictions of games. In the first restriction, we restrict a game to having only two players. In the second restriction, we restrict each agent to having only two strategies. We will refer to games that abide by the former restriction as 2-*player*, and to games that abide by the latter restriction as *boolean*.

If we want to find a Nash equilibrium, we need an agreed upon manner in which to encode the result, which is a strategy profile. We represent a strategy profile by enumerating, by agent, each pure strategy *in that agent's* support and the probability with which the pure strategy is played. Each probability is given as the quotient of two integers.

This representation works well in bimatrix games, because the following proposition guarantees that for any 2-player game there exists Nash equilibrium that can be encoded in reasonable amount of space.

PROPOSITION 3.3. *Any 2-player game with rational payoffs has a rational Nash equilibrium where the probabilities are of bit length polynomial with respect to the number of strategies and bit-lengths of the payoffs. Furthermore, if we restrict ourselves to Nash equilibria θ where $\nu_i(\theta) \geq g_i$ for $i \in \{1, 2\}$ where each guarantee g_i is a rational number then either 1) there exists such a θ where the probabilities are of bit length polynomial with respect to the number of strategies and bit-lengths of the payoffs and the bit lengths of the guarantees or 2) no such θ exists.*

This proposition implies that for any bimatrix game there exists a Nash equilibrium that is at most polynomially sized with respect to the encoding of the game, and that for any 2-player circuit game there exists a Nash equilibrium that is at most exponentially sized with respect to the encoding of the game.

However, there exist 3-player games with rational payoffs that have no Nash equilibrium with all rational probabilities [18]. Therefore, we cannot hope to always find a Nash equilibrium in this representation. Instead we will study ϵ-Nash equilibrium when we are not restricted to 2-player games. The following result from [15] states that there is always an ϵ-Nash equilibrium that can be represented in a reasonable amount of space.

THEOREM 3.4. *[15] Let θ be a Nash equilibrium for an n-player game $\mathcal{G} = (\mathfrak{S}, \nu)$ in which all the payoffs are between 0 and 1, and let $k \geq \frac{n^2 \log(n^2 \max_i |\mathfrak{S}_i|)}{\epsilon^2}$. Then there exists a k-uniform ϵ-Nash equilibrium θ' where $|\nu_i(\theta) - \nu_i(\theta')| \leq \frac{\epsilon}{2}$ for $1 \leq i \leq n$.*

Recall that a k-uniform strategy profile is a strategy profile where each agent randomizes uniformly between k, not necessarily unique, pure strategies. The number of bits needed to represent such a strategy profile is $O((\sum_i \min\{k, |\mathfrak{S}_i|\}) \cdot \log k)$. Thus, Theorem 3.4 implies that for any that for any n-player game $(g_1, \ldots, g_n) = (\mathfrak{S}, \nu)$ in which all the payoffs are between 0 and 1, there exists an ϵ-Nash equilibrium of bit-length $\mathrm{poly}(n, 1/\epsilon, \log(\max_i |\mathfrak{S}_i|))$. There also is an ϵ-Nash equilibrium of bit-length $\mathrm{poly}(n, \log(1/\epsilon), \max_i |\mathfrak{S}_i|)$.

We want to study the problems with and without approximation. All the problems that we study will take as an input a parameter ϵ related to the bound of approximation. We define four types of approximation:

1a) EXACT**:** Fix $\epsilon = 0$ in the definition of the problem. [2]

1b) EXP-APPROX**:** input $\epsilon \geq 0$ as a rational number encoded as the quotient of two integers. [3]

2) POLY-APPROX**:** input $\epsilon > 0$ as 1^k where $\epsilon = 1/k$

3) CONST-APPROX**:** Fix $\epsilon > 0$ in the definition of the problem.

With all problems, we will look at only 3 types of approximation. Either 1a) or 1b) and both 2 and 3. With many of the problems we study, approximating using 1a) and 1b) yields identical problems. Since the notion of ϵ-Nash equilibrium is with respect to additive error, the above notions of approximation refer only to games whose payoffs are between 0 and 1 (or are scaled to be such). Therefore **we assume that all games have payoffs which are between 0 and 1** unless otherwise explicitly stated. Many times our constructions of games use payoffs which are not between 0 and 1 for ease of presentation. In such a cases the payoffs can be scaled.

Now we define the problems which we will examine.

DEFINITION 3.5. *For a fixed representation of games,* ISNASH *is the promise language defined as follows:*

> Positive instances*: $(\mathcal{G}, \theta, \epsilon)$ such that $\mathcal{G}$ is a game given in the specified representation, and θ is strategy profile which is a Nash equilibrium for $\mathcal{G}$.*
>
> Negative instances*: $(\mathcal{G}, \theta, \epsilon)$ such that θ is a strategy profile for $\mathcal{G}$ which is not an ϵ-Nash equilibrium.*

Notice that when $\epsilon = 0$ this is just the language of pairs $(\mathcal{G}, \theta)$ where θ is a Nash equilibrium of $\mathcal{G}$.

The the definition of ISNASH is only one of several natural variations. Fortunately, the manner in which it is defined does not affect our results and any reasonable definition will suffice. For example, we could instead define ISNASH where:

1. $(\mathcal{G}, \theta, \epsilon)$ a positive instance if θ is an $\epsilon/2$-Nash equilibrium of $\mathcal{G}$; negative instances as before.

2. $(\mathcal{G}, \theta, \epsilon, \delta)$ is a positive instance if θ is an ϵ-Nash equilibrium; $(\mathcal{G}, \theta, \epsilon, \delta)$ is a negative instance if θ is not an $\epsilon + \delta$-Nash equilibrium. δ is represented in the same way as ϵ.

Similar modifications can be made to Definitions 3.6, 3.7, and 3.9. The only result affected is the reduction in Corollary 4.5.

[2]We use this type of approximation only when we are guaranteed to be dealing with rational Nash equilibrium. This is the case in all games restricted to 2-players and when solving problems relating to pure-strategy Nash equilibrium such as determining if a pure-strategy profile is a Nash equilibrium and determining if there exists a pure-strategy Nash equilibrium.

[3]We will only consider this in the case where a rational Nash equilibrium is not guaranteed to exist, namely in k-player games for $k \geq 3$ for the problems ISNASH, FINDNASH, and GUARANTEENASH.

Definition 3.6. *We define the promise language* IsPureNash *to be the same as* IsNash *except we require that, in both positive and negative instances, θ is a pure-strategy profile.*

Definition 3.7. *For a fixed representation of games,* ExistsPureNash *is the promise language defined as follows:*

> Positive instances*: Pairs $(\mathcal{G}, \epsilon)$ such that $\mathcal{G}$ is a game in the specified representation in which there exists a pure-strategy Nash equilibrium.*
>
> Negative instances*: $(\mathcal{G}, \epsilon)$ such that there is no pure-strategy ϵ-Nash equilibrium in $\mathcal{G}$.*

Note that Exact ExistsPureNash is just a language consisting of pairs of games with pure-strategy Nash equilibria.

Definition 3.8. *For a given a representation of games, the problem* FindNash *is a search problem where, given a pair $(\mathcal{G}, \epsilon)$ such that $\mathcal{G}$ is a game in a specified representation, a valid solution is any strategy-profile that is an ϵ-Nash equilibrium in $\mathcal{G}$.*

As remarked above, when dealing with FindNash in games with more than 2 players, we use Exp-Approx rather than Exact. This error ensures the existence of some Nash equilibrium in our representation of strategy profiles; there may be no rational Nash equilibrium.

Definition 3.9. *For a fixed representation of games,* GuaranteeNash *is the promise language defined as follows:*

> Positive instances*: $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ such that $\mathcal{G}$ is a game in the specified representation in which there exists a Nash equilibrium θ such that, for every agent i, $\nu_i(\theta) \geq g_i$.*
>
> Negative instances*: $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ such that $\mathcal{G}$ is a game in the specified representation in which there exist no ϵ-Nash equilibrium θ such that, for every agent i, $\nu_i(\theta) \geq g_i - \epsilon$.*

When we consider IsNash, FindNash, and GuaranteeNash in k-player games, $k > 2$, we will not consider Exact, but only the other three types: Exp-Approx, Poly-Approx, and Const-Approx. The reason for this is that no rational Nash equilibrium is guaranteed to exist in these cases, and so we want to allow a small rounding error. With all other problems we will not consider Exp-Approx, but only the remaining three: Exact, Poly-Approx, and Const-Approx.

4. RELATIONS BETWEEN CONCISE GAMES

We study two different concise representations of games: circuit games and degree-d graph games; and two restrictions: two-player games and boolean-strategy games. It does not make since to impose both of these restrictions at the same time, because in two-player, boolean games all the problems studied are trivial.

This leaves us with three variations of circuit games: circuit games, 2-player circuit games, and boolean circuit games. Figure 2 shows the hierarchy of circuit games. A line drawn between two types of games indicates that the game type higher in the diagram is at least as hard as the game type lower in the diagram in that we can efficiently reduce questions about Nash equilibria in the games of the lower type to ones in games of the higher type. (One caveat is that for 2-player circuit games we consider Exact but not Exp-Approx, and for circuit games we consider Exp-Approx but not Exact, and these models seem incomparable.)

This also leaves us with three variations of degree-d graph games: degree-d graph games, 2-player degree-d graph games, and boolean degree-d graph games. A 2-player degree-d graph game is simply a bimatrix game (if $d \geq 2$) so the hierarchy of games is as shown in Figure 2. (Again, the same caveat applies. For bimatrix games we consider Exact but not Exp-Approx, and for graph games and boolean graph games we consider Exp-Approx but not Exact, and these models seem incomparable.)

It is easy to see that given a bimatrix game, we can always efficiently construct an equivalent 2-player circuit game. It is also possible to construct a log space reduction from graph games of arbitrary degree to boolean circuit games, a result which we will state presently in Theorem 4.1 and its corollaries. This gives us the relationship in Figure 2.

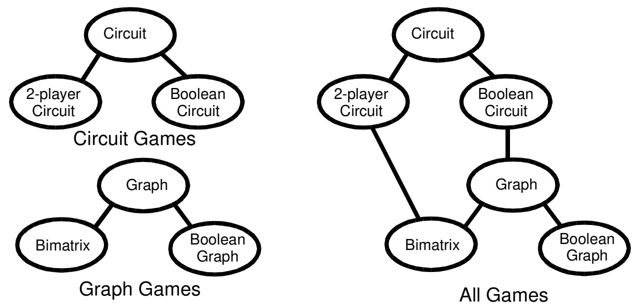

Figure 2: Relationships between games

Theorem 4.1. *Given an n-player graph game of arbitrary degree $\mathcal{G} = (G, \mathfrak{S}, \nu)$, in logarithmic space, we can create an n'-player Boolean circuit game $\mathcal{G}' = (\mathfrak{S}', \nu')$ where $n \leq n' \leq \sum_{i=1}^{n} |\mathfrak{S}_i|$ and a logarithmic space function $f : \Theta \to \Theta'$ and a polynomial time function $g : \Theta' \to \Theta$* [4] *with the following properties:*

1. *f and g map pure-strategy profiles to pure-strategy profiles.*
2. *f and g map rational strategy profiles to rational strategy profiles.*
3. *$g \circ f$ is the identity map.*
4. *For each agent i in $\mathcal{G}$ there an agent i in $\mathcal{G}'$ such that for any strategy profile θ of $\mathcal{G}$, $\nu_i(\theta) = \nu'_i(f(\theta))$ and for any strategy profile θ' of $\mathcal{G}'$, $\nu'_i(\theta') = \nu_i(g(\theta'))$.*
5. *If θ' is an ϵ-Nash equilibrium in $\mathcal{G}'$ then $g(\theta')$ is a $\lceil \log_2 k \rceil \cdot \epsilon$-Nash equilibrium in $\mathcal{G}$ where $k = \max_i |\mathfrak{S}_i|$.*
6. • *For every $\theta \in \Theta$, θ is a Nash equilibrium if and only if $f(\theta)$ is a Nash equilibrium.*

[4] More formally, we specify f and g by constructing, in space $O(\log(|\mathcal{G}|))$, a branching program for f and a circuit that computes g.

- *For every pure-strategy profile $\theta \in \Theta$, θ is an ϵ-Nash equilibrium if and only if $f(\theta)$ is and ϵ-Nash equilibrium.*

Proof Sketch: We briefly sketch the mapping of $\mathcal{G}$ to $\mathcal{G}'$. Given a graph game $\mathcal{G}$, to construct $\mathcal{G}'$, we create a binary tree t_i of depth $\log|\mathfrak{S}_i|$ for each agent i, with the elements of $\mathfrak{S}_i$ at the leaves of the tree. Each internal node in t_i represents an agent in $\mathcal{G}'$. The strategy space of each of these agents is {`left`, `right`}, each corresponding to the choice of a subtree under his node.

The strategy of each agent in t_i "points" to another agent or strategy (the root of the right or left subtree) further down in the tree t_i. By following the paths induced by these points, each node in the tree t_i can be associated with some leaf at the bottom of the tree t_i and, therefore, with some strategy $s_i \in \mathfrak{S}_i$ from the game $\mathcal{G}$. Let s be such that s_j is the strategy associated with the agent at the root of t_j. Then we define the payoff of an agent i' in t_i to be $\nu'_{i'} = \nu_i(R_i(s, s'_i))$ where s'_i is the strategy associated with agent i'. □

Corollary 4.2. *There exist boolean games without rational Nash equilibria.*

Corollary 4.3. *With* Exp-Approx *and* Poly-Approx*, there is a log space reduction from graph game* ExistsPureNash *to boolean circuit game* ExistsPureNash

We do not mention IsNash or IsPureNash because they are in **P** for graph games (see Section 5.)

Corollary 4.4. *With* Exp-Approx *and* Poly-Approx*, there is a log space reduction from graph game* FindNash *to boolean circuit game* FindNash.

Corollary 4.5. *With* Exp-Approx *and* Poly-Approx*, there is a log space reduction from graph game* GuaranteeNash *to boolean circuit game* GuaranteeNash.

The new results in [10] change things slightly. This result gives a polynomial-time reduction from Exact and Exp-Approx FindNash in degree-d graph games to degree-3 boolean graph games as long as d is constant.

5. ISNASH AND ISPURENASH

In this section, we study the problem of determining whether a given strategy profile is a Nash equilibrium. Studying this problem will also help in studying the complexity of other problems.

5.1 IsNash

A summary of the results for IsNash is shown in Figure 3. The various complexity classes are described in Section 2 under the heading of Complexity Theory.

Notice that with Poly-Approx and Const-Approx everything works much as with Exp-Approx and Exact, but #**P**, counting, is replaced by **BPP**, approximate counting.

IsNash is in **P** for all graph games. When allowing arbitrarily many players in a boolean circuit game, IsNash becomes $\mathbf{P}^{\#\mathbf{P}}$-complete (via Cook reductions). When allowing exponentially many strategies in a 2-player circuit game, it becomes **coNP**-complete. IsNash for a generic circuit game combines the hardness of these 2 cases and is $\mathbf{coNP}^{\#\mathbf{P}}$-complete.

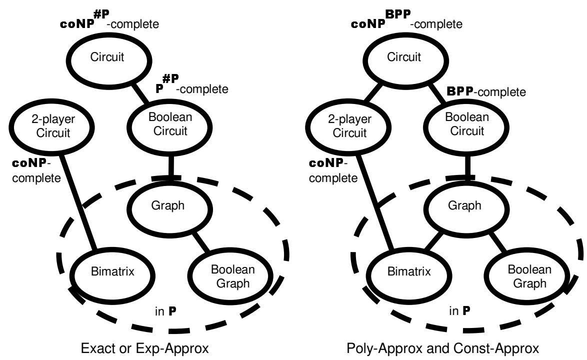

Figure 3: Summary of IsNash Results

Proposition 5.1. *In all approximation schemes, graph game* IsNash *is in* **P**.

Proposition 5.2. *In all approximation schemes, 2-player circuit game* IsNash *is* **coNP***-complete. Furthermore, it remains in* **coNP** *for any constant number of players, and it remains hard as long as approximation error $\epsilon < 1$.*

In the previous proof, we obtain the hardness result by making one player choose between many different strategies, and thus making him assert something about the evaluation of each strategy. We will continue to use similar tricks except that we will often have to be more clever to get many strategies. Randomness provides one way of doing this.

Theorem 5.3. *Boolean circuit game* Exp-Approx IsNash *is* $\mathbf{P}^{\#\mathbf{P}}$*-complete via Cook reductions.*

Proof Sketch: We show here that it is $\mathbf{P}^{\#\mathbf{P}}$-hard. We reduce from MajoritySat, which is $\mathbf{P}^{\#\mathbf{P}}$-complete under Cook reductions. A circuit C belongs to MajoritySat if it evaluates to 1 on at least half of its inputs.

Given a circuit C with n inputs (without loss of generality, n is even), we construct an $(n+1)$-player boolean circuit game. The payoff to agent 1 if he plays 0 is $\frac{1}{2}$, and if he plays 1 is the output of the circuit, $C(s_2, \ldots, s_{n+1})$, where s_i is the strategy of agent i. The payoffs of the other agents are determined by a game of pennies (for details see Section 2) in which agent i plays against agent $i+1$ where i is even.

Let $\epsilon = 1/2^{n+1}$, and let θ be a mixed strategy profile where $\Pr[\theta_1 = 1] = 1$, and $\Pr[\theta_i = 1] = \frac{1}{2}$ for $i > 1$. It can be verified that θ is a Nash equilibrium if and only if $C \in$MajoritySat. □

Theorem 5.4. *Circuit game* Exp-Approx IsNash *is* $\mathbf{coNP}^{\#\mathbf{P}}$*-complete.*

Even if we allow just one agent in a boolean circuit game to have arbitrarily many strategies, then the problem becomes $\mathbf{coNP}^{\#\mathbf{P}}$-complete.

We now look at the problem when dealing with Poly-Approx and Const-Approx.

Theorem 5.5. *With* Poly-Approx *and* Const-Approx*, boolean circuit game* IsNash *is* **BPP***-complete.*[5] *Furthermore, this holds for any approximation error $\epsilon < 1$.*

[5]Recall that all our complexity classes are promise classes, so this is really **prBPP**.

Proof Sketch: Boolean circuit game IsNash with polynomially small approximation is in **BPP** because given an instance $(\mathcal{G}, \theta, \epsilon)$, for each agent i and each strategy $s_i \in \{0, 1\}$ we can use random sampling to distinguish the following two possibilities in probabilistic polynomial time: whether $\nu_i(\theta) \geq \nu_i(R_i(\theta, s_i))$ or $\nu_i(\theta) + \epsilon < \nu_i(R_i(\theta, s_i))$.

The hardness result is very similar to the proof in Theorem 5.3. The key difference is that here we reduce from the **BPP**-complete promise language of deciding if a given a circuit C accepts fewer than 1/3 or more than 2/3 of its inputs. □

Theorem 5.6. *With* Poly-Approx *and* Const-Approx, *circuit game* IsNash *is* $\mathbf{coNP}^{\mathbf{BPP}} = \mathbf{coMA}$*-complete. Furthermore, this holds for any approximation error* $\epsilon < 1$.

Note that when approximating IsNash, it never made a difference whether we approximated by a polynomially small amount or by any constant amount less than 1.

5.2 IsPureNash

In this section we will study a similar problem: IsPureNash. In the case of non-boolean circuit games, IsPureNash is **coNP**-complete. With the other games examined, IsPureNash is in **P**.

Proposition 5.7. *With any approximation scheme, circuit game and 2-player circuit game* IsPureNash *is* **coNP**-*complete. Furthermore, it remains hard for any approximation error* $\epsilon < 1$.

Proposition 5.8. *With any approximation scheme, Boolean circuit game* IsPureNash *is* **P**-*complete, and remains so even for one player and any approximation error* $\epsilon < 1$.

Proposition 5.9. *With any approximation scheme, graph game* IsPureNash *is in* **P** *for any kind of graph game.*

6. EXISTENCE OF PURE-STRATEGY NASH EQUILIBRIA

We now will use the previous relationships to study the complexity of ExistsPureNash. Figure 4 gives a summary of the results.

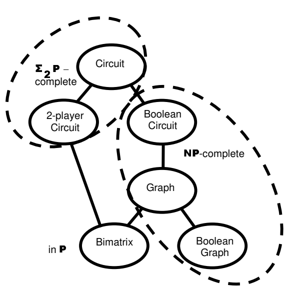

Figure 4: Summary of ExistsPureNash **Results**

The hardness of these problems is directly related to the hardness of IsPureNash. We can always solve ExistsPureNash with one more non-deterministic alternation because we can nondeterministically guess a pure-strategy Nash equilibrium, and then check that it is correct. Recall that in the case of non-boolean circuit games, IsPureNash is **coNP**-complete. With the other games examined, IsPureNash is in **P** (but is only proven to be P-hard in the case of boolean circuit games; see Subsection 5.2). As shown in Figure 4, with the exception of bimatrix games, this strategy of nondeterministically guessing and then checking is the best that one can do.

We first note that ExistsPureNash is an exceedingly easy problem in the bimatrix case because we can enumerate over all the possible pure-strategy profiles and check whether they are Nash equilibria.

ExistsPureNash is interesting because it is a language related to the Nash equilibrium of bimatrix games that is not **NP**-complete. One particular approach to the complexity of finding a Nash equilibrium is to turn the problem into a language. Both [9] and [3] show that just about any reasonable language that one can create involving Nash equilibrium in bimatrix games is **NP**-complete; however, ExistsPureNash is a notable exception. Our results show that this phenomenon does not extend to concisely represented games.

Theorem 6.1. *Circuit game* ExistsPureNash *and* 2*-player circuit game* ExistsPureNash *are* $\mathbf{\Sigma_2 P}$*-complete with any of the defined notions of approximation. Furthermore, it remains hard as long as approximation error* $\epsilon < 1$.

For graph games, it was recently shown by Gottlob, Greco, and Scarcello [12] that ExistsPureNash is **NP**-complete, even restricted to graphs of degree 4. Below we strengthen their result by showing this also holds for *boolean* graph games, for graphs of degree 3, and for any approximation error $\epsilon < 1$.

Theorem 6.2. *For boolean circuit games, graph games, and boolean graph games using any of the defined notions of approximation* ExistsPureNash *is* **NP**-*complete. Moreover, the hardness result holds even for degree-d boolean graph games for any* $d \geq 3$ *and for any approximation error* $\epsilon < 1$.

Proof Sketch: Here, to show the hardness result, we reduce from CircuitSat which is **NP**-complete. Given a circuit C (without loss of generality every gate in C has total degree ≤ 3; we allow unary gates), we design the following game: For each input of C and for each gate in C, we create player with the strategy space $\{\texttt{true}, \texttt{false}\}$. We call these the *input agents* and *gate agents* respectively, and call the agent associated with the output gate the *judge*. We also create two additional agents P_1 and P_2 with strategy space $\{0, 1\}$.

We now define the payoffs. Each input agent is rewarded 1 regardless. Each gate agent is rewarded 1 for correctly computing the value of his gate and is rewarded 0 otherwise.

If the judge plays `true` then the payoffs to P_1 and P_2 are always 1. If the judge plays `false` then the payoffs to P_1 and P_2 are the same as the game pennies–P_1 acting as the first player, P_2 as the second.

It is now straightforward to check that a pure strategy Nash equilibria exists if and only if C is satisfiable. □

The first thing to notice is that like IsPureNash this problem does not become easier with approximation, even if we allow as much approximation as possible without the problem becoming trivial. Also, similarly to IsPureNash, any reasonable definition of approximation would yield the same results.

7. FINDING NASH EQUILIBRIA

Perhaps the most well-studied of these problems is the complexity of finding a Nash equilibria in a game. In the bimatrix case FindNash is known to be **P**-hard but unlikely to be **NP**-hard. There is something elusive in categorizing the complexity of finding something we know is there. Such problems, including finding Nash equilibrium, are studied by [16].

Recently, [15] showed that if we allow constant error, the bimatrix case FindNash is in quasipolynomial time (i.e. $\tilde{\mathbf{P}}$). The results are summarized in Figure 5.

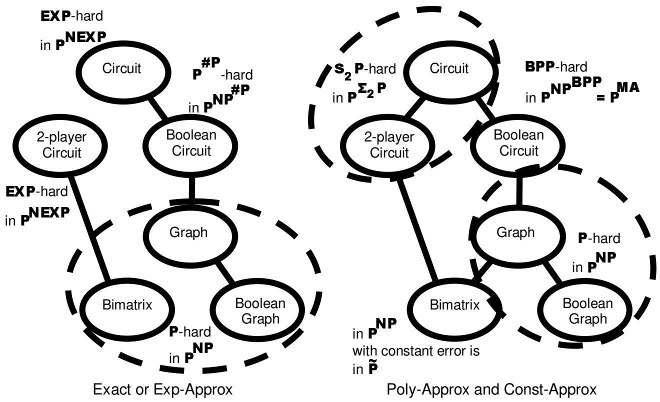

Figure 5: Summary of FindNash Results

In all types of games, there remains a gap of knowledge of less than one alternation. This comes about because to find a Nash equilibrium we can simply guess a strategy profile and then check whether it is a Nash equilibrium. It turns out that in all the types of games, the hardness of FindNash is at least as hard as IsNash (although we do not have a generic reduction between the two). Circuit game and 2-player circuit game Poly-Approx and Const-Approx FindNash are the only cases where the gap in knowledge is less than one alternation. Subsequent to our work, it was shown in [4] that, in graph game and boolean graph games, Exact and Exp-Approx FindNash are complete for the class **PPAD**. Also, subsequent to our work, it was shown in [2] that the same is true for bimatrix games.

In a circuit game, there may be exponentially many strategies in the support of a Nash equilibrium or the bit length of the probability that a particular strategy is played may be exponentially large. In either case, it would take exponentially long just to write down a Nash equilibrium. In order to avoid this problem, when we are not assured the existence of a polynomially sized Nash equilibrium (or ϵ-Nash equilibrium), we will prove hardness results not with FindNash, but with FindNashSimple. FindNashSimple is an easier promise language version of FindNash that always has a short answer.

Definition 7.1. *For a fixed representation of games,* FindNashSimple *is the promise language defined as follows:*

> Positive instances*:* $(\mathcal{G}, i, s_i, k, \epsilon)$ *such that* $\mathcal{G}$ *is a game given in the specified representation, and in every* ϵ*-Nash equilibrium* θ *of* $\mathcal{G}$*,* $\Pr[\theta_i = s_i] \geq k$*.*
>
> Negative instances*:* $(\mathcal{G}, i, s_i, k, \epsilon)$ *such that* $\mathcal{G}$ *is a game given in the specified representation, and in every* ϵ*-Nash equilibrium* θ *of* $\mathcal{G}$*,* $\Pr[\theta_i = s_i] < k$*.*

FindNashSimple is easier than FindNash in that a FindNash algorithm can be used to obtain FindNashSimple algorithm of similar complexity, but the converse is not clear.

Theorem 7.2. *2-player circuit game* Exact FindNashSimple *is* **EXP***-hard, but can be computed in polynomial time with an* **NEXP** *oracle. However, if it is* **NEXP***-hard, it implies that* **NEXP** *is closed under complement.*

Theorem 7.3. *Circuit game* Exp-Approx FindNashSimple *is* **EXP***-hard, but is in* **NEXP***. However, if it is* **NEXP***-hard, it implies that* **NEXP** *is closed under complement. The* **EXP***-hardness holds even for circuit games with 6 players.*

Proof Sketch: We show here that circuit game Exp-Approx FindNashSimple is **EXP**-hard. We reduce from SuccinctCircuitValue. Given a succinctly represented circuit C, we construct an instance of FindNashSimple based upon a 6-player game $\mathcal{G} = (\mathfrak{S}, \nu)$.

Let G be the set of gates in C and let $N = |G|$. We create 3 *computing agents*: c_1, c_2, and c_3; and we create 3 *enforcing agents*: e_1, e_2, and e_3. Each computing agent has the strategy set $\mathfrak{S}_{c_i} = \{g, \bar{g} : g \in G\}$. Each enforcing agent has the strategy set $\mathfrak{S}_{e_i} = \{g : g \in G\}$. The payoffs are designed so that in any ϵ-Nash equilibrium the enforcing agent e_i forces the computing agent c_i to play g or $\bar{g}$ (for each $g \in G$) with probability at least $1/N - \epsilon$. In addition, the payoff of computing agent c_i is designed so that, if the other two computing agents play strategies associated to the input gates of the gate associated with the strategy of c_i, agent c_i is penalized for playing the incorrect output of the gate (using the strategies of the other computing agents as inputs).

We can then show inductively that in any ϵ-Nash equilibrium θ, no agent c_i plays a strategy corresponding to the incorrect evaluation of a gate with probability greater than 2ϵ.

Given this, we can solve an instance of SuccinctCircuitValue on an instance C by querying FindNashSimple on the instance $(\mathcal{G}, c_1, o, 2\epsilon, \epsilon)$, where o is the output gate, and returning the same answer. $\square$

A technique similar to this proof was used in [10] to show a reduction from degree-d graph games to d^2 player normal form games. The basic idea is as follows, given a degree-d graph game $\mathcal{G}$, color the vertices of the graph corresponding to $\mathcal{G}$ so that no vertex has two neighbors of the same color (this requires at most d^2 colors). Now create a computing agent and an enforcing agent for each color. Similar to the above construction, each enforcing agent forces the corresponding computing agent to nearly uniformly randomize between the strategies spaces of each player associated with his color in the graph game.

Circuit game FindNash become easier when we approximate. The main reason is that now, by Theorem 3.4, there always exists a Nash equilibrium with a polynomially sized support. Thus we can guess an ϵ-Nash equilibrium and, using a result like IsNash, test that it is such. Unlike the exponential case, here the complexity is at most one alternation more than the complexity of the corresponding IsNash problem.

THEOREM 7.4. *Circuit game and 2-player circuit game* POLY-APPROX *and* CONST-APPROX FINDNASH *are* $\mathbf{S_2P}$*-hard but can be computed by a polynomial-time algorithm with access to a* $\mathbf{\Sigma_2P}$ *oracle.*

This hardness result, as well as that of Theorem 7.2, is based on a reduction to GAMEVALUE, which is known to be **EXP**-complete [7] to compute exactly and $\mathbf{S_2P}$-complete to approximate. The next two hardness results use a different general approach, the hardness of ISNASH.

Unlike EXISTSPURENASH, FINDNASH is a lot harder in boolean circuit games than in graph games. This is because of the hardness of ISNASH in boolean circuit games.

THEOREM 7.5. *Boolean circuit game* EXP-APPROX FINDNASH *is* $\mathbf{P}^{\#\mathbf{P}}$*-hard via cook reductions but can be computed in polynomial time given an* $\mathbf{NP}^{\#\mathbf{P}}$ *oracle.*

PROOF. It is in $\mathbf{NP}^{\#\mathbf{P}}$ because there always exists a polynomial length solution, and by Theorem 5.3 we can verify a solution in $\#\mathbf{P}$. The hardness result is very similar to that of Theorem 5.3. Given a circuit C with n inputs, we build a similar $n+1$ player game that offers agent 1 a choice of payment of $\frac{1}{2} - 1/2^{n+1}$ or the output of the circuit. For sufficiently small ϵ, if the circuit evaluates to true a majority of the time, agent 1 must choose the output of the circuit with high probability. ■

In the previous result, the hardness comes from the hardness of ISNASH, so it is not surprising that boolean circuit game FINDNASH becomes easier when we introduce approximation.

THEOREM 7.6. *Boolean circuit game* POLY-APPROX *and* CONST-APPROX FINDNASH *are* **BPP***-hard, but can be computed in polynomial time with an oracle to* $\mathbf{NP}^{\mathbf{BPP}} = \mathbf{MA}$.

Proof Sketch: It is in $\mathbf{NP}^{\mathbf{BPP}}$ because there exists a solution with a polynomial length description, and by Theorem 5.5 the validity of any solution can be check in **BPP**.

We create the same game as in the reduction of Theorem 5.5, but with two differences: First, instead of creating $|r|$-players (where r is the randomness required) that play pennies against each other, we create $2k \cdot |r|$ such players for some polynomially large k that we will specify later. Secondly, when we calculate the payoff to the first player for choosing the circuit, we obtain an input to the circuit by taking XORs of the bits played by $|r|$ disjoint subsets of the remaining players, of size k each.

Now because the $2k \cdot |r|$ remaining players must play a ϵ-Nash equilibrium, they must randomize over their inputs so that they play 0 with probability $\in [1/2 - 2\epsilon, 1/2 + 2\epsilon]$. The bits from the strategies of these players are fully independent, so by XORing we get a string of $|r|$ bits that is exponentially close to uniform.

The payoff to agent 1 for playing $\emptyset$ is $\frac{1}{2}$, but the payoff for choosing the circuit will by exponentially close to the probability that a random input satisfies the circuit. □

Finally, we show the complexity for graph games.

THEOREM 7.7. *With any type of approximation, graph game and boolean graph game* FINDNASH *can be computed in polynomial time with access to an* **NP** *oracle, but neither is* **NP***-hard unless* $\mathbf{NP} = \mathbf{coNP}$*. Furthermore, graph game and boolean graph game* FINDNASH *are* **P***-hard, even when restricted to boolean graphs of degree* ≥ 3.

8. EXISTENCE OF NASH EQUILIBRIA WITH GUARANTEED PROPERTIES

Because FINDNASH is a search problem where a solution is guaranteed to exist, it is hard to define a nontrivial language from it. It is possible to create languages from FINDNASH by adding additional constraints on the equilibrium. For example: does there exists a Nash equilibrium where each player is paid at least x amount? does there exists a Nash equilibrium with social welfare x? or does there exists a Nash equilibrium in which player 1 does not play some strategy s_1? It turns out that in the bimatrix case, for almost any constraint the language ends up being **NP**-complete [3, 9].[6] GUARANTEENASH is one such a problem. In our results, each GUARANTEENASH problem is complete for the class that was the upper bound for the same instance of FINDNASH. Figure 6 shows a summary of the results.

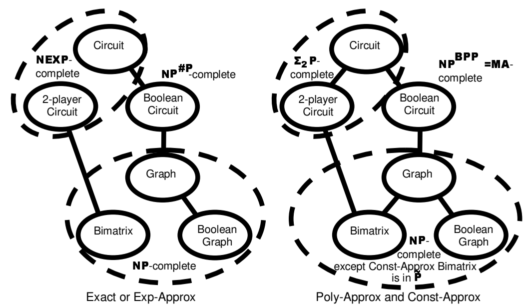

Figure 6: Summary of GUARANTEENASH **Results**

THEOREM 8.1. *Circuit game* EXP-APPROX GUARANTEENASH *and 2-player circuit game* EXACT GUARANTEENASH *are* **NEXP***-complete.*

This result relies heavily on proof techniques of [3]. It follows from their proof that EXACT GUARANTEENASH is NP-complete in bimatrix games. We observe that the result scales up exponentially and also holds even for EXP-APPROX in this case.

THEOREM 8.2. *Circuit game and 2-player circuit game* POLY-APPROX *and* CONST-APPROX GUARANTEENASH *are* $\mathbf{\Sigma_2P}$*-complete.*

Proof Sketch: We show that 2-player circuit game CONST-APPROX GUARANTEENASH is $\mathbf{\Sigma_2P}$-hard. We reduce from QCIRCUITSAT$_2$, which is $\mathbf{\Sigma_2P}$-complete. QCIRCUITSAT$_2$ = $\{(C, k_1, k_2) : \exists x \in \{0,1\}^{k_1}, \forall y \in \{0,1\}^{k_2}\ C(x,y) = 1\}$ where C is a circuit that takes $k_1 + k_2$ boolean variables. Given an instance (C, k_1, k_2) create the 2-player circuit game $\mathcal{G} = (\mathfrak{s}, \nu)$, where $s_i = \left(\{0,1\}^{k_1} \times \{0,1\}^{k_2}\right) \cup \{\emptyset\}$. The payoffs to $\mathcal{G}$ will be designed so that if there exists an $x_0 \in \{0,1\}^{k_1}$ such that $C(x_0, y) = 1$ for all $y \in \{0,1\}^{k_2}$, then a Nash equilibrium is for each player to play strategies of the form (x_0, y) (for any $y \in \{0,1\}^{k_2}$) with probability 1. However, if no such x_0 exists, the only ϵ-Nash equilibrium will be to play $\emptyset$ most of the time.

We will only define the payoffs for the first player because the payoffs are symmetric, that is $\nu_1(s_1, s_2) = \nu_2(s_2, s_1)$.

1. $x_1 \neq x_2$, $\nu_1((x_1, y_1), (x_2, y_2)) = 0$

[6]Note that our results show that EXISTSPURENASH was an exception to this rule. It was trivial in bimatrix games, but at least **NP**-hard in every other setting.

2. $\nu_1((x,y_1),(x,y_2)) =$
   - $1-\gamma$ if $C(x,y_1) = C(x,y_2) = 1$
   - 0 if $C(x,y_1) = 1$ and $C(x,y_2) = 0$,
   - 1 if $C(x,y_1) = 0$ and $C(x,y_2) = 1$,
   - $\frac{1}{2}$ if $C(x,y_1) = C(x,y_2) = 0$
3. $\nu_1(\emptyset,\emptyset) = \gamma$
4. $\nu_1((x_1,y_1),\emptyset) = 0$
5. $\nu_1(\emptyset,(x_2,y_2)) = 1-\gamma$

Let $\epsilon = \frac{1}{100}$, $\gamma = \frac{1}{10}$, and $g_i = 1-\gamma$. It is relatively straightforward to check that $(C,k_1,k_2) \in$ QCircuitSat if and only if there exists a Nash equilibrium that meets the guarantees.

We now argue that Circuit game Poly-Approx GuaranteeNash is in $\mathbf{\Sigma_2 P}$. First note that $\mathbf{\Sigma_2 P^{BPP}} = \mathbf{\Sigma_2 P}$ because $\mathbf{coNP^{BPP}} = \mathbf{coMA} \subseteq \mathbf{\Sigma_2 P}$. Thus an $\exists\, \mathbf{coNP^{BPP}}$-predicate can be replaced by an $\exists\, \mathbf{\Sigma_2 P}$-predicate $= \mathbf{\Sigma_2 P}$-predicate. Now the result follows because we can guess such an equilibrium and check, in $\mathbf{coNP^{BPP}}$, that it is a Nash equilibrium and meets the guarantees. □

Theorem 8.3. *Boolean circuit game* Exp-Approx GuaranteeNash *is* $\mathbf{NP^{\#P}}$*-complete.*

Theorem 8.4. *Boolean circuit game* Poly-Approx *and* Const-Approx GuaranteeNash *are* $\mathbf{NP^{BPP}} = \mathbf{MA}$*-complete.*

Theorem 8.5. *Graph game and boolean graph game* GuaranteeNash *is* $\mathbf{NP}$*-complete for all levels of approximation. The results hold even when restricted to degree d graphs, $d \geq 3$.*

Conitzer and Sandholm [3] showed that Exact GuaranteeNash is NP-complete in bimatrix games. We observe that the same holds even for Poly-Approx:

Theorem 8.6. *[3] Bimatrix* Exact *and* Poly-Approx GuaranteeNash *are* $\mathbf{NP}$*-complete.*

Theorem 8.7. *Bimatrix* Const-Approx GuaranteeNash *is in* $\tilde{\mathbf{P}}$.

Proof. Given an instance $(\mathcal{G},\epsilon,(g_1,\ldots,g_n))$ simply look through all the k-uniform strategies, where $k = \frac{4\log(4\max_i|\mathfrak{S}_i|)}{\epsilon^2}$ for a strategy profile that is an ϵ-Nash equilibrium where the payoffs to players are within $\epsilon/2$ of their guarantees. There are only a quasipolynomial number of k-uniform strategies and checking each strategy takes only polynomial time. If such a strategy is found, accept, otherwise reject.

If there is no ϵ-Nash equilibrium within ϵ of the guarantees, surely the algorithm will not find one. However, if there exists some Nash equilibrium θ that pays off each player his guaranteed amount, then by Theorem 3.4 there will exist a k-uniform ϵ-Nash equilibrium θ' that is within $\epsilon/2$ of the guarantees, and so the algorithm will find it. ■

Acknowledgments

We thank Eli Ben-Sasson, Adam Klivans, Ryan O'Donnell, Rocco Servedio, and Amir Shpilka for many discussions about algorithmic aspects of Nash equilibria which informed this work. We thank Mike Kearns, Christos Papadimitriou, David Parkes, and Avi Pfeffer for helpful pointers and advice, and we thank Saurabh Sanghvi for fruitful discussions during the initial stages of this work.

9. REFERENCES

[1] X. Chen and X. Deng. 3-Nash is PPAD-complete. Technical Report TR05-134, ECCC, Nov. 2005.

[2] X. Chen and X. Deng. Settling the complexity of 2-player Nash-equilibrium. Technical Report TR05-140, ECCC, Dec. 2005.

[3] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *IJCAI*, 2003.

[4] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. In *38th STOC*, 2006. To appear.

[5] C. Daskalakis and C. Papadimitriou. The complexity of games on highly regular graphs. In *13th ESA*, 2005.

[6] C. Daskalakis and C. Papadimitriou. Three-player games are hard. Technical Report TR05-139, ECCC, Nov. 2005.

[7] J. Feigenbaum, D. Koller, and P. Shor. A game-theoretic classification of interactive complexity classes. In *10th CCC*, pages 227–237, 1995.

[8] L. Fortnow, R. Impagliazzo, V. Kabanets, and C. Umans. On the complexity of succinct zero-sum games. In *20th CCC*, 2005.

[9] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1:80–93, 1989.

[10] P. W. Goldberg and C. H. Papadimitriou. Reducibility among equilibrium problems. In *38th STOC*, 2006. To appear.

[11] O. Goldreich. On promise problems (a survey in memory of Shimon Even [1935–2004]). Technical Report TR05-018, ECCC, Feb. 2005.

[12] G. Gottlob, G. Greco, and F. Scarcello. Pure Nash equilibria: hard and easy games. In *TARK '03: Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, pages 215–230, New York, NY, USA, 2003. ACM Press.

[13] W. Hesse, E. Allender, and D. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. of Computer and System Sci.*, 65:695–716, 2002.

[14] M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *UAI*, pages 253–260, 2001.

[15] R. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *ACM Conference on Electronic Commerce*, 2003.

[16] N. Megiddo and C. H. Papadimitriou. A note on total functions, existence theorems, and computational complexity. *Theoretical Computer Sci.*, 81(1):317–324, 1991.

[17] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[18] J. F. Nash and L. S. Shapley. A simple three-person poker game. *Contributions to the Theory of Games*, 1(24):105–116, 1950.

[19] G. Schoenebeck and S. Vadhan. The computational complexity of Nash equilibria in concisely represented games. Technical Report TR05-052, ECCC, May 2005.

[20] G. R. Schoenebeck. The complexity of finding Nash equilibria in succinctly represented games. Undergraduate Thesis, Harvard University, 2004.