The Computational Complexity of Nash Equilibria in Concisely Represented Games

GRANT R. SCHOENEBECK, Princeton University
SALIL VADHAN, Harvard University

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 a player 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

Additional Key Words and Phrases: Nash equilibrium, concise games, graph games, circuit games, computational game theory

ACM Reference Format:
Schoenebeck, G. R. and Vadhan, S. 2012. The computational complexity of Nash equilibria in concisely represented games. ACM Trans. Comput. Theory 4, 2, Article 4 (May 2012), 50 pages.
DOI = 10.1145/2189778.2189779 http://doi.acm.org/10.1145/2189778.2189779

1. INTRODUCTION

In recent years, there has been a surge of interest at the interface between computer science and game theory. On one hand, game theory and its notions of equilibria provide a rich framework for modeling the behavior of selfish agents in the kinds of distributed or networked environments that often arise in computer science and offer mechanisms to achieve efficient and desirable global outcomes in spite of the selfish behavior. On the other hand, classical game theory ignores computational

---

An extended abstract is in *Proceedings of E-commerce 2006* [Schoenebeck and Vadhan 2006]. Many of these results first appeared in the first author's undergraduate thesis [Schoenebeck 2004].
G. R. Schoenebeck was supported by NSF grant CCR-0133096, NSF Graduate Student Fellowship, the US-Israel Binational Science Foundation Grant 2006060, and NSF grant CCF 0729137. S. Vadhan gratefully acknowledges that work was done in part while a Fellow at the Radcliffe Institute for Advanced Study, and was also supported by NSF grant CCR-0133096 and a Sloan Research Fellowship.
Authors' addresses: G. R. Schoenebeck, Computer Science Department, Princeton University, 35 Olden St., Princeton, NJ 08540; email: gschoene@cs.princeton.edu; S. Vadhan, School of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge, MA, 02138; email: salil@seas.harvard.edu.

DOI 10.1145/2189778.2189779 http://doi.acm.org/10.1145/2189778.2189779

considerations, and it is unclear how meaningful game-theoretic notions of equilibria are if they are infeasible to compute. Finally, game-theoretic characterizations of complexity classes have proved to be extremely useful even in addressing questions that a priori have nothing to do with games, of particular note being the work on interactive proofs and their applications to cryptography and hardness of approximation [Arora and Safra 1998; Arora et al. 1998; Feige et al. 1996; Goldreich et al. 1991; Goldwasser et al. 1989].

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 explicit "bimatrix" form [Conitzer and Sandholm 2008; Gilboa and Zemel 1989; Lipton et al. 2003; Megiddo and Papadimitriou 1991], 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). However, 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 [Kearns et al. 2001].

Motivated by these 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 [Feigenbaum et al. 1995] (resp., $\mathbf{S}_2\mathbf{P}$-complete [Fortnow et al. 2008]). They serve as a very general model, capturing essentially any representation in which the payoffs are efficiently computable. The second are *graph games* [Kearns et al. 2001], 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 neighbors. (Thus, if the graph is of low degree, the payoff functions can be written very compactly). Kearns et al. [2001] 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 [Gottlob et al. 2003] recently showed that the problem of deciding if a degree-4 graph game has a pure-Nash equilibrium is **NP**-complete.

In both of these 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.
(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 [Conitzer and Sandholm 2008; Gilboa and Zemel 1989], who showed it to be **NP**-complete for two-player, bimatrix games.)

We study these 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 the following.

— A tight characterization of the complexity of all of the listed problems 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 Kearns et al. [2001]), in that there is a general reduction from the latter to the former that applies to most of the listed problems.

Independent and Subsequent Results. Several researchers have independently obtained some results related to ours. Specifically, Daskalakis and Papadimitriou [2005a] 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 Àlvarez, Gabarró, and Serna [Àlvarez et al. 2005] give results on EXISTSPURENASH that are very similar to ours.

Subsequent to the original versions of our work [Schoenebeck 2004; Schoenebeck and Vadhan 2006], there have been several exciting new results that give tight characterizations of the complexity of FINDNASH (recall that this is the problem for which our upper and lower bounds do not match). First, Goldberg and Papadimitriou [2006] give a reduction for FINDNASH from degree-d graph games to d^2-player explicit (aka normal-form) games.[1] Their reduction uses a technique similar to one in our paper (which appeared in the original version [Schoenebeck and Vadhan 2006]). In addition, they give a reduction from d-player games to degree-3 graph games in which each player has two strategies, strengthening the second item in the preceding list. By composing these reductions, they show that for any constant C, FINDNASH in games with at most C players, can be reduced to FINDNASH in 4-player games.

Subsequently, Goldberg, Daskalakis, and Papadimitriou [Daskalakis et al. 2006] prove that for $C \geq 4$ FINDNASH in C-player games is complete for the class **PPAD** [Megiddo and Papadimitriou 1991]. Daskalakis and Papadimitriou [2005b] and Chen and Deng [2005] independently strengthen the result to hold for $C = 3$. Finally, Chen and Deng [2006] prove FINDNASH in 2-player, bimatrix games is **PPAD**-complete. Using the aforementioned reduction of Goldberg and Papadimitriou [2006], this also implies that FINDNASH in a degree-3 graph game in which each player has 2 strategies is **PPAD**-complete.

Furthermore, Chen, Deng, and Teng [Chen et al. 2006] show that even finding approximate Nash-equilibrium in bimatrix and graph games is **PPAD**-complete.

We will further discuss the relationship of these subsequent results to our work in the relevant sections.

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.

[1] A *normal-form* game is one where the payoffs are explicitly specified for each possible combination of player strategies. When there are two players, this is simply the standard *bimatrix* representation.

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 $\nu_i(s_1, \ldots, s_n)$. We call a game *constant-sum* if for any set of strategies, the sum of all players' payoffs is constant.[2]

For us, $\mathfrak{S}_i$ will always be finite and the range of ν_i will always be rational. An *explicit* (or *normal-form*) 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 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. If a strategy is rational, we represent it as a rational number (the quotient of two integers). We assume that all strategy profiles that are inputs to functions are rational (see Section 3 for a discussion of why this is reasonable). Some of our results, but not all, will carry over to the more general setting where the only thing assumed about the way that probabilities are represented in a strategy profile is that the first k bits of the decimal representation can be recovered in time polynomial in k.

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* [Nash 1951] 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 that is a Nash equilibrium (respectively, an ϵ-Nash equilibrium).

[2]Constant-sum games are computationally equivalent to zero-sum games.

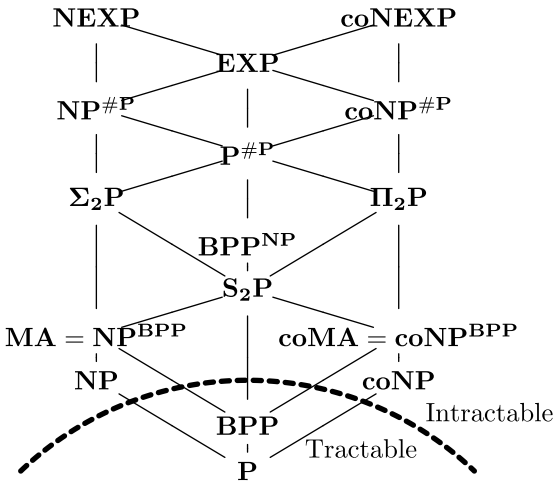

Fig. 1. Relationships between complexity classes.

Pennies is a 2-player game where $\mathfrak{s}_1 = \mathfrak{s}_2 = \{0, 1\}$, and the payoffs are as follows.

		Player 2	
		Heads	Tails
Player 1	Heads	(1, 0)	(0, 1)
	Tails	(0, 1)	(1, 0)

The first number in each ordered pair is the payoff of player 1 and the second number is the payoff to player 2.

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 $\frac{1}{2} \pm 2\epsilon$ (see Appendix A for details).

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 $L^+ \cup L^-$; effectively it is promised that the input is in $L^+ \cup L^-$.

Because we consider approximation problems in this article, which are naturally formulated as promise languages, *all complexity classes used in this article are classes of promise languages*. We refer the reader to the recent survey by Goldreich [2005] about the usefulness and subtleties of working with promise problems.

Figure 1 shows the relationships among the complexity classes used in this article. 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 classes answer how many accepting computations a nondeterministic polynomial-time Turing machine has (rather than whether or not an accepting computation exists, like **NP**). **#P** is generally thought to contain much harder problems than **NP**. **BPP** is the class of languages that

can be computed 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 some language in $\mathcal{D}$. For example, since SAT is **NP**-complete, a language is in $\Sigma_\mathbf{2}\mathbf{P} = \mathbf{NP}^{\mathbf{NP}}$ if and only if it can be decided by a nondeterministic polynomial-time Turing machine with oracle access to SAT.

A *search problem*, is specified by a relation, $R \subseteq \Sigma^* \times \Sigma^*$. We say that a Turing machine T *computes* R if given an $x \in \Sigma^*$, T either outputs $y \in \Sigma^*$ such that $(x, y) \in R$ or outputs the special character $\sqcup$ if no such y exists. We say that a nondeterministic Turing machine T *computes* a search problem R if, on input x, every computational path of $T(x)$ either outputs y such that $(x, y) \in R$ or fails. Moreover, if there exists a y such that $(x, y) \in R$ then at least one computational path of $T(x)$ must output such a y.

We say that a search problem R is in **FP** if there exists a polynomial time Turing Machine that computes the search problem specified by R. We similarly define $\mathbf{F\tilde{P}}$, **FNP**, $\mathbf{F\Sigma_2P}$, $\mathbf{FNP^{\#P}}$, $\mathbf{FNP^{BPP}} = \mathbf{FMA}$, and **FNEXP** as containing the search problems that can be computed with quasipolynomial time Turing machines, nondeterministic polynomial time Turing machines, nondeterministic polynomial time Turing machines with access to an **NP** oracle, nondeterministic polynomial time Turing machines with access to a **#P** oracle, nondeterministic polynomial time Turing machines with access to a **BPP** oracle, and nondeterministic exponential-time Turing machines respectively.[3] **TFP** is the family of total search problems R where $R \in \mathbf{FP}$ and for all x there exists y such that $(x, y) \in R$. The classes $\mathbf{TF\tilde{P}}$, **TFNP**, $\mathbf{TF\Sigma_2P}$, $\mathbf{TFNP^{\#P}}$, $\mathbf{TFNP^{BPP}}$, and **TFNEXP** are similarly defined. This will always be the case for us.

We define Karp and Cook reductions for (promise-)languages and search problems. We will be using Karp reductions unless we explicitly state otherwise.

Definition 2.1. (Karp Reductions) We say that promise-language L_1 reduces to promise-language L_2 via a logspace Karp reduction if there exists a logspace-computable function f such that $x \in L_1^+ \Rightarrow f(x) \in L_2^+$ and $x \in L_1^- \Rightarrow f(x) \in L_2^-$. We say that promise-language L_1 reduces to search problem R_2 via a logspace Karp reduction if there exists logspace-computable functions f and g such that for every mapping $h : \Sigma^* \to \Sigma^* \cup \{\sqcup\}$ that computes R_2, $g(h(f(x)))$ decides L_1. Finally, we say that search problem R_1 reduces to search problem R_2 via a logspace Karp reduction if there exist logspace computable functions f and g such that for every mapping $h : \Sigma^* \to \Sigma^* \cup \{\sqcup\}$ that computes R_2, $g(h(f(x)))$ computes R_1.

[3]This definition of **FNP** differs from the commonly used notation where it is required that deciding if $(x, y) \in R$ is polynomial time computable. Our definition yields a larger class (since we do not require that relation R be polynomial-time decidable), but it is morally capturing the same level of complexity, and generalizes to higher complexity class. If a relation R is in **FNP** according to our definition, then there is a relation R' such that deciding $(x, y) \in R'$ is polynomially decidable and such that R reduces to R' in the sense of Definition 2.1. Let $(x, y) \in R'$ if y describes a nonfailing computational path of the nondeterministic polynomial-time machine computing R. In this case R is no harder to compute than R'.

Definition 2.2. (Cook Reductions) We say that promise-language L_1 reduces to promise-language L_2 via a Cook reduction if there exists a polynomial time Turing machine with oracle access to L_2 that decides L_1. Likewise we say that promise-language L_1 reduces to search problem R_2 via a Cook reduction if there exists a polynomial time Turing machine with oracle access to R_2 that decides L_1.

When reducing to a search problem via an oracle, it is required that ANY valid response from the oracle yields a correct answer; similarly, ANY response for oracle queries outside $L^+ \cup L^-$ should yield a correct answer.

3. CONCISE REPRESENTATIONS AND PROBLEMS STUDIED

We now give formal descriptions of the problems that 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. We assume that all payoffs of ν are rational and that they are represented by C_i as the quotient of two integers.

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. [Kearns et al. 2001] 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. We assume that all payoffs of ν are rational and that they are represented as the quotient of two integers.

This definition was proposed in Kearns et al. [2001]. 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 directly influenced by the strategies of more than d 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, we restrict a game to having only two players. In the second, 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 a 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 $v_i(\theta) \geq g_i$ for $i \in \{1, 2\}$ where each guarantee g_i is a rational number, then either (1) there exists 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.*

PROOF SKETCH. If we are given the support of some Nash equilibrium, we can set up a polynomially sized linear program whose solution will be a Nash equilibrium in this representation, and so it is polynomially sized with respect to the encoding of the game. (Note that the support may not be easy to find, so this does not yield a polynomial time algorithm). If we restrict ourselves to Nash equilibria θ satisfying $v_i(\theta) \geq g_i$ as in the proposition, this merely adds additional constraints to the linear program. □

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 [Nash and Shapley 1950]. 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 [Lipton et al. 2003] states that there is always an ϵ-Nash equilibrium that can be represented in a reasonable amount of space.

THEOREM 3.4. *[Lipton et al. 2003] Let θ be a Nash equilibrium for an n-player game $\mathcal{G} = (\mathfrak{s}, v)$ 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 $|v_i(\theta) - v_i(\theta')| \leq \frac{\epsilon}{2}$ 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 $(g_1, \ldots, g_n) = (\mathfrak{s}, v)$ in which all the payoffs are between 0 and 1, there exists an ϵ-Nash equilibrium of bit-length $\text{poly}(n, 1/\epsilon, \log(\max_i |\mathfrak{s}_i|))$. There also is an ϵ-Nash equilibrium of bit-length $\text{poly}(n, \log(1/\epsilon), \max_i |\mathfrak{s}_i|)$.

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;
(1b) EXP-APPROX: input $\epsilon > 0$ as a rational number encoded as the quotient of two integers;
(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. We consider EXACT 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. We consider EXP-APPROX 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. With many of the problems we study, approximating using 1a) and 1b) yield identical problems.

Since the notion of ϵ-Nash equilibrium is with respect to additive error, the preceding 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 that are between 0 and 1 unless otherwise explicitly stated. Many times our constructions of games use payoffs that are not between 0 and 1 for ease of presentation. In such 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 a strategy profile that is a Nash equilibrium for $\mathcal{G}$.
Negative instances: $(\mathcal{G}, \theta, \epsilon)$ such that θ is a strategy profile for $\mathcal{G}$ that is not a 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) $(\mathcal{G}, \theta, \epsilon)$ is 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 a $(\epsilon + \delta)$-Nash equilibrium. δ is represented in the same way a ϵ.

Similar modifications can be made to Definitions 3.6, 3.7, and 3.9. The only result affected is the reduction in Corollary 4.6.

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 in the preceding, when dealing with FindNash in games with more than 2 players, we use Exp-Approx rather than Exact. This error ensures the

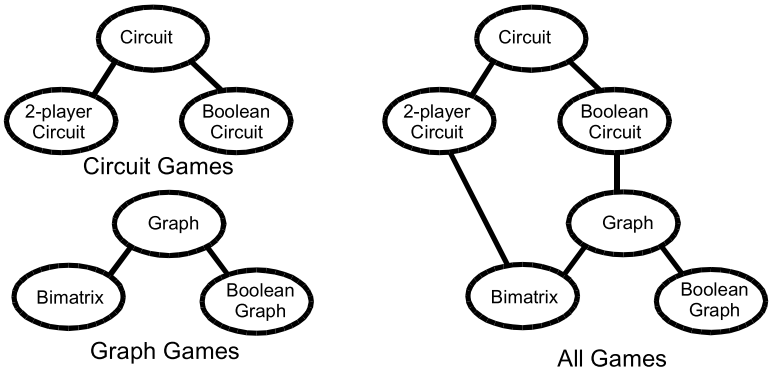

Fig. 2. Relationships between games.

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, $v_i(\theta) \geq g_i$.
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 $v_i(\theta) \geq g_i - \epsilon$.

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 sense 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. We will presently illustrate a reduction from graph games to Boolean strategy circuit games. This gives us the relationship in Figure 2.

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 $\mathcal{G}' = (\mathfrak{s}', \nu')$, with $n \le n' \le \sum_{i=1}^{n} |\mathfrak{s}_i|$, a logarithmic space function $f : \Theta \to \Theta'$, and a polynomial time function $g : \Theta' \to \Theta$ with the following properties.*[4]

(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 $\left(\lceil \log_2 k \rceil \cdot \epsilon\right)$-Nash equilibrium in $\mathcal{G}$ where $k = \max_i |\mathfrak{s}_i|$.*
(6) *— For every $\theta \in \Theta$, θ is a Nash equilibrium in $\mathcal{G}$ if and only if $f(\theta)$ is a Nash equilibrium in $\mathcal{G}'$.*
    — For every pure-strategy profile $\theta \in \Theta$, θ is an ϵ-Nash equilibrium if and only if $f(\theta)$ is an ϵ-Nash equilibrium.

Subsequent Results. The reductions in Goldberg and Papadimitriou [2006] and Daskalakis et al. [2008][5] are incomparable with our results. They show a reduction from degree-d graph games to d^2-player explicit (aka normal form) games[6] and a reduction from d-player games to degree-3 Boolean graph games. However, these reductions only apply to FINDNASH and only preserve the approximation to within a polynomial factor, whereas our reduction applies to all the problems we consider and only loses a logarithmic factor in the approximation error.

PROOF CONSTRUCTION OF $\mathcal{G}'$. 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. We denote the player at the root of the tree t_i as i.

There are $n' \le \sum_{i=1}^{n} |\mathfrak{s}_i|$ players in $\mathcal{G}'$, because the number of internal nodes in any tree is less than the number of leaves. $\mathfrak{s}' = \{\texttt{left}, \texttt{right}\}^{n'}$.

For each i, we can recursively define functions $\alpha_{i'} : \mathfrak{s}' \to \mathfrak{s}_i$ that associate pure strategies of agent i in $\mathcal{G}$ with each agent i' in t_i given a pure-strategy profile for $\mathcal{G}'$ as follows.

— if $s'_{i'} =$ `right` and the right child of i' is a leaf corresponding to a strategy $s_i \in \mathfrak{s}_i$, then $\alpha_{i'}(s') = s_i$.
— if $s'_{i'} =$ `right` and the right child of i' is another agent j', then $\alpha_{i'}(s') = \alpha_{j'}(s')$.
— If $s'_{i'} =$ `left`, $\alpha_{i'}(s')$ is similarly defined.

[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.
[5]The prejournal manuscript [Daskalakis et al. 2008] combines and expands the results of Goldberg and Papadimitriou [2006] and Daskalakis et al. [2006].
[6]An *explicit* game is one where the payoffs are explicitly specified for each possible combination of player strategies.

Notice each agent i' in the tree t_i is associated with a strategy of $\mathfrak{s}_i$ that is a descendant of i'. This implies that i is the only player in t_i that has the possibility of being associated with every strategy of agent i in $\mathcal{G}$.

Let s' be a pure-strategy profile of $\mathcal{G}'$ and let $s = (s_1, \ldots, s_n)$ be the pure-strategy profile of $\mathcal{G}$ where $s_i = \alpha_i(s')$. Then we define the payoff of an agent i' in t_i to be $v'_{i'}(s') = v_i(R_i(s, \alpha_{i'}(s')))$. So, the payoff to agent i' in tree t_i in $\mathcal{G}'$ is the payoff to agent i, in $\mathcal{G}$, playing $\alpha_{i'}(s')$ when the strategy of each other agent j is defined to be $\alpha_j(s')$.
By inspection, $\mathcal{G}'$ can be computed in log space.

We note for use in the following, that $\alpha_{i'} : \mathfrak{s}' \to \mathfrak{s}_i$ induces a map from Θ' (i.e. random variables on $\mathfrak{s}$) to Θ_i (i.e. random variables on $\mathfrak{s}_i$) in the natural way.

CONSTRUCTION OF $f : \Theta \to \Theta'$. Fix $\theta \in \Theta$. For each agent i' in tree t_i in $\mathcal{G}'$ let $L_{i'}, R_{i'} \subseteq \mathfrak{s}_i$ be the set of leaves in the left and right subtrees under node i' respectively. Now let $f(\theta) = \theta'$ where $\Pr[\theta'_{i'} = \texttt{left}] = \Pr[\theta_i \in L_{i'}] / \Pr[\theta_i \in L_{i'} \cup R_{i'}] = \Pr[\theta_i \in L_{i'} | \theta_i \in L_{i'} \cup R_{i'}]$.

Note that if i' is an agent in t_i and some strategy s_i in the support of θ_i is a descendant of i', then this uniquely defines $\theta_{i'}$. However, for the other players this value is not defined because $\Pr[\theta_i \in L_{i'} \cup R_{i'}] = 0$. We define the strategy of the rest of the players inductively. The payoffs to these players are affected only by the mixed strategies associated to the roots of the other trees, i.e. $\{\alpha_j(\theta'), i \neq j\}$–which is already fixed–and the strategy to which they are associated. By induction, assume that the strategy to any descendant of a given agent i' is already fixed, now simply define $\theta'_{i'}$ to be the pure strategy that maximizes his payoff (we break a tie in some fixed but arbitrary manner so that each of these agents plays a pure strategy).

By inspection, this f can be computed in polynomial time given $\mathcal{G}$ and s, which implies that given $\mathcal{G}$, we can construct a circuit computing f in log space.

CONSTRUCTION OF $g : \Theta' \to \Theta$. Given a strategy profile θ' for $\mathcal{G}'$, we define $g(\theta') = (\alpha_1(\theta'), \ldots, \alpha_n(\theta'))$.

This can be done in log space because computing the probability that each pure strategy is played only involves multiplying a logarithmic number of numbers together, which is known to be in log space [Hesse et al. 2002]. This only needs to be done a polynomial number of times.

PROOF OF 1. If θ is pure-strategy profile, then for each agent i, there exists $s_i \in \mathfrak{s}_i$ such that $\Pr[\theta_i = s_i] = 1$. So all the agents in t_i that have s_i as a descendant must choose the child whose subtree contains s_i with probability 1—a pure strategy. The remaining agents merely maximize their payoffs, and so always play a pure strategy (recall that ties are broken in some fixed but arbitrary manner that guarantees a pure strategy).

$\alpha_{i'} : \mathfrak{s}' \to \mathfrak{s}_i$ maps pure-strategy profiles to pure-strategies, so $g(s') = (\alpha_1(s'), \ldots, \alpha_n(s'))$ does as well.

PROOF OF 2. For f we recall that if agent i' in tree t_i has a descendant in the support of θ_i, then $\Pr[f(\theta)_{i'} = \texttt{left}] = \Pr[\theta_i \in L_{i'}] / \Pr[\theta_i \in L_{i'} \cup R_{i'}]$ ($L_{i'}$ and $R_{i'}$ are as defined in the construction of f), so it is rational if θ is rational. The remaining agents always play a pure strategy.

For g we have $\Pr[g(\theta')_i = s] = \sum_{s' : \alpha_i(s') = s} \Pr[\theta' = s']$, which is rational if θ' is rational.

PROOF OF 3. Since $g(f(\theta)) = (\alpha_1(f(\theta)), \ldots, \alpha_n(f(\theta)))$, the claim $g \circ f = id$ is equivalent to the following lemma.

LEMMA 4.2. *The random variables $\alpha_i(f(\theta))$ and θ_i are identical.*

PROOF. We need to show that for every $s_i \in \mathfrak{s}_i$, $\Pr[\alpha_i(f(\theta)) = s_i] = \Pr[\theta_i = s_i]$. Fix s_i, let $i = i'_0, i'_1, \ldots, i'_k = s_i$ be the path from the root i to the leaf s_i in the tree t_i, let $dir_j \in \{\texttt{left}, \texttt{right}\}$ indicate whether i'_{j+1} is the right or left child of i'_j, and let $S_{i'}$ be the set of all leaves that are descendants of i'. Then

$$\begin{aligned}\Pr[\alpha_i(f(\theta)) = s_i] &= \prod_{j=0}^{k-1} \Pr[f(\theta)_{i'_j} = dir_j] = \prod_{j=0}^{k-1} \Pr[\theta_i \in S_{i'_{j+1}} | \theta_i \in S_{i'_j}] \text{ (by the definition of f)} \\ &= \Pr[\theta \in S_{i'_k} | \theta_i \in S_{i'_0}] \text{ (by Bayes' Law)} \\ &= \Pr[\theta_i = s_i] \text{ (because } S_{i'_k} = \{s_i\} \text{ and } S_{i'_0} = \mathfrak{s}_i). \end{aligned}$$

□

PROOF OF 4. We first show that $\nu'_i(\theta') = \nu_i(g(\theta'))$. Fix some $\theta' \in \Theta'$.

$$\begin{aligned}\nu'_i(\theta') &= \nu_i(R_i((\alpha_1(\theta'), \ldots, \alpha_n(\theta')), \alpha_i(\theta'))) \text{ (by definition of } \nu'_i) \\ &= \nu_i(\alpha_1(\theta'), \ldots, \alpha_n(\theta')) = \nu_i(g(\theta')) \text{ (by definition of } g).\end{aligned}$$

Finally, to show that $\nu'_i(f(\theta)) = \nu_i(\theta)$, fix $\theta \in \Theta$ and let $\theta' = f(\theta)$. By what we have just shown

$$\nu'_i(\theta') = \nu_i(g(\theta')) \Rightarrow \nu'_i(f(\theta)) = \nu_i(g(f(\theta))) = \nu_i(\theta).$$

The last equality comes from the fact that $g \circ f = id$.

PROOF OF 5. Fix some ϵ-Nash equilibrium $\theta' \in \Theta'$ and let $\theta = g(\theta')$.

We must show that $\nu_i(\theta)$ is within $\lceil \log_2 k \rceil \cdot \epsilon$ of the payoff for agent i's optimal response. To do this we show by induction that $\nu_i(R_i(\theta, \alpha_{i'}(\theta')) \geq \nu_i(R_i(\theta, s_i)) - d\epsilon$ for all s_i that are descendants of agent i' in tree t_i, where d is the depth of the subtree with agent i' at the root. We induct on d. The result follows by taking $i' = i$, and noting that $R_i(\theta, \alpha_i(\theta')) = \theta$ and $d \leq \lceil \log_2 k \rceil$.

We defer the base case and proceed to the inductive step. Consider some agent i' in tree t_i such that the subtree of i' has depth d. i' has two strategies, $\{\texttt{left}, \texttt{right}\}$. Let $E_{i'} = \nu_{i'}(\theta') = \nu(R_i(\theta, \alpha_i(\theta'))$ be the expect payoff of i', and let $Opt_{i'}$ be the maximum of $\nu(R_i(\theta, s_i))$ over $s_i \in \mathfrak{s}_i$ that are descendants of i'. Similarly define E_l, Opt_l, E_r, and Opt_r for the left subtree and right subtree of i' respectively. We know $E_{i'} \geq \max\{E_l, E_r\} - \epsilon$ because otherwise i' could do ϵ better by playing `left` or `right`. By induction $E_l \geq Opt_l - (d-1)\epsilon$ and $E_r \geq Opt_r - (d-1)\epsilon$. Finally, putting this together, we get that

$$E_{i'} \geq \max\{E_l, E_r\} - \epsilon \geq \max\{Opt_l, Opt_r\} - (d-1)\epsilon - \epsilon = Opt_{i'} - d\epsilon.$$

The proof of the base case, $d = 0$, is the same except that instead of employing the inductive hypothesis, we note that there is only one pure strategy in each subtree and so it must be optimal.

PROOF OF 6. Fix some strategy profile $\theta \in \Theta$ and let $\theta' = f(\theta)$. Let θ be a Nash equilibrium and let i' be an agent in t_i that has a descendant that is a pure strategy in the support of θ_i. All the strategies in the support of $\alpha_{i'}(\theta')$ are also in the support of θ_i; but, all the strategies in the support of θ_i are optimal and therefore agent i' cannot do better. All of the remaining agents are in equilibrium because they are playing an optimal strategy by construction. Conversely, if $f(\theta)$ is a Nash equilibrium, then $g(f(\theta))$ is also (by Part 5). But by Part 3, $g(f(\theta)) = \theta$, and therefore θ is a Nash equilibrium.

Let θ be a pure-strategy ϵ-Nash equilibrium for $\mathcal{G}$. Fix some agent i, and let $s_i \in \mathfrak{s}$ be such that $\Pr[\theta_i = s_i] = 1$. Then any agent in t_i that does not have s_i as a descendant plays optimally in $f(\theta)$. If agent i' does have s_i as a descendant then according to $f(\theta)$, agent i' should select the subtree containing s_i with probability 1. Assume without loss

of generality this is in the right subtree. If agent i' plays `right`, as directed by $f(\theta)$, his payoff will be $v_i(\theta)$. If he plays `left`, his payoff will be $v_i(R_i(\theta, s_i'))$, where s_i' is the strategy that α assigns to the left child of i'. But $v_i(\theta) + \epsilon \geq v_i(R_i(\theta, s_i'))$ because θ is an ϵ-Nash equilibrium.

Now say that $f(\theta)$ is a pure-strategy ϵ-Nash equilibrium for $\mathcal{G}'$, where $\theta \in \Theta$ is a pure-strategy profile. Fix some agent i, and let $s_i \in \mathfrak{s}$ be such that $\Pr[\theta_i = s_i] = 1$. If s_i is an optimal response to θ, then agent i is in equilibrium. Otherwise, let $s_i' \neq s_i$ be an optimal response to θ. Then let i' be the last node on the path from i to s_i in the tree t_i such that i' has s_i' as a descendant. By definition of f and v', agent i' gets payoff $v_i(R_i(\theta, s_i)) = v_i(\theta)$, but would get payoff $v_i(R_i(\theta, s_i'))$ if he switched strategies (because the nodes off of the path from i to s_i in the tree t_i play optimally). Yet $f(\theta)$ is an ϵ-Nash equilibrium, and so we conclude that these differ by less than ϵ, and thus agent i is in an ϵ-equilibrium in $\mathcal{G}$. □

COROLLARY 4.3. *There exist Boolean games without rational Nash equilibria.*

PROOF. We know that there is some 3-player game $\mathcal{G}$ with rational payoffs but no rational Nash equilibrium [Nash and Shapley 1950]. Applying the reduction in Theorem 4.1 to this game results in a Boolean game $\mathcal{G}'$. If θ' were some rational Nash equilibrium for $\mathcal{G}'$, then, by parts 2 and 5 of Theorem 4.1, $g(\theta')$ would be a rational Nash equilibrium for $\mathcal{G}$. □

COROLLARY 4.4. *With* EXP-APPROX *and* POLY-APPROX*, there is a log space reduction from graph game* EXISTSPURENASH *to Boolean circuit game* EXISTSPURENASH.

PROOF. Given an instance $(\mathcal{G}, \epsilon)$ where $\mathcal{G}$ is a graph game, we construct the corresponding Boolean circuit game $\mathcal{G}'$ as in Theorem 4.1, and then solve EXISTSPURENASH for $(\mathcal{G}', \epsilon/\log_2 k)$.

We show that $(\mathcal{G}, \epsilon)$ is a positive instance of EXISTSPURENASH if and only if $(\mathcal{G}', \epsilon/\log_2 k)$ is also. Say that $(\mathcal{G}, \epsilon)$ is a positive instance of EXISTSPURENASH. Then $\mathcal{G}$ has a pure-strategy Nash equilibrium θ, and, by Parts 1 and 6 of Theorem 4.1, $f(\theta)$ will be a pure-strategy Nash equilibrium in $\mathcal{G}'$. Now say that $(\mathcal{G}', \epsilon/\lceil\log_2 k\rceil)$ is not a negative instance of EXISTSPURENASH. Then there exists a pure-strategy profile θ' that is an $\epsilon/\log_2 k$-Nash equilibrium in $\mathcal{G}'$. If follows from Part 5 of Theorem 4.1 that $g(\theta')$ is a pure-strategy ϵ-Nash equilibrium in $\mathcal{G}$. □

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

COROLLARY 4.5. *With* EXP-APPROX *and* POLY-APPROX*, there is a log space reduction from graph game* FINDNASH *to Boolean circuit game* FINDNASH.

PROOF. Given an instance $(\mathcal{G}, \epsilon)$ of n-player graph game FINDNASH we transform $\mathcal{G}$ into a Boolean circuit game $\mathcal{G}'$ as in Theorem 4.1. Then we can solve FINDNASH for $(\mathcal{G}', \epsilon/\lceil\log_2 k\rceil)$, where k is the maximum number of strategies of any agent to obtain an $(\epsilon/\lceil\log_2 k\rceil)$-Nash equilibrium θ' for $\mathcal{G}'$, and return $g(\theta')$, which is guaranteed to be an ϵ-Nash equilibrium of $\mathcal{G}$ by Part 5 of Theorem 4.1. $\mathcal{G}'$ and $g(\theta')$ can be computed in log space. □

COROLLARY 4.6. *With* EXP-APPROX *and* POLY-APPROX*, there is a log space reduction from graph game* GUARANTEENASH *to Boolean circuit game* GUARANTEENASH.

PROOF. Given an instance $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ of graph game GUARANTEENASH we transform $\mathcal{G}$ into a Boolean circuit game $\mathcal{G}'$ as in Theorem 4.1. Then we can solve

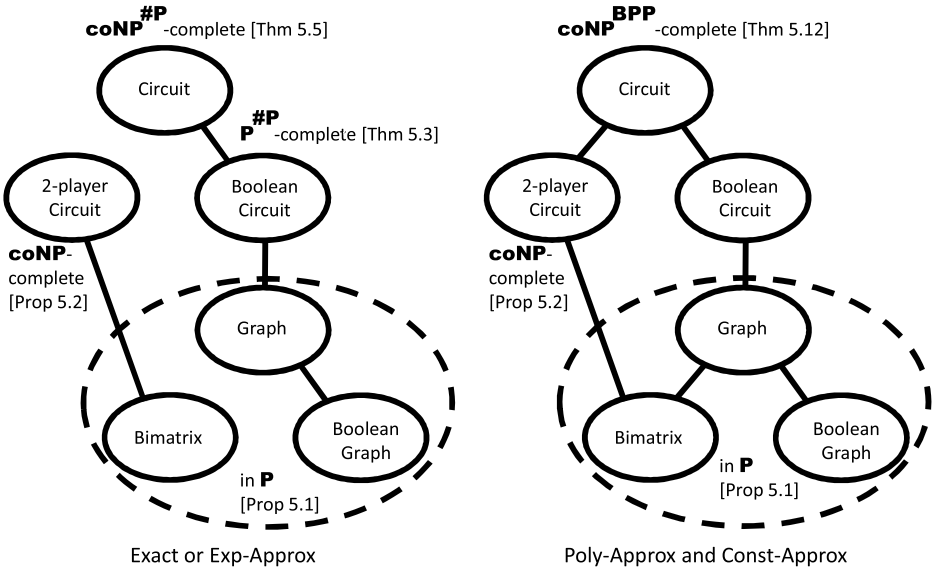

Fig. 3. Summary of ISNASH results.

GUARANTEENASH for $(\mathcal{G}', \epsilon/\lceil \log_2 k \rceil, (g_1, \ldots, g_n, 0, \ldots, 0)$, where k is the maximum number of strategies of any agent. So that we require guarantees for the agents at the roots of the trees, but have no guarantee for the other agents.

We show that if $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a positive instance of GUARANTEENASH, then so is $(\mathcal{G}', \epsilon/\lceil \log_2 k \rceil, (g_1, \ldots, g_n, 0, \ldots, 0))$. Say that $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a positive instance of GUARANTEENASH. Then there exists some Nash Equilibrium of $\mathcal{G}$, θ, such that $v_i(\theta) \geq g_i$ for each agent i. But then by Parts 6 and 4 of Theorem 4.1 respectively, $f(\theta)$ is a Nash Equilibrium of $\mathcal{G}'$ and $v_i'(f(\theta)) = v_i(\theta) \geq g_i$ for each agent i of $\mathcal{G}$ and corresponding agent i of $\mathcal{G}'$.

Say that $(\mathcal{G}', \epsilon/\lceil \log_2 k \rceil, (g_1, \ldots, g_n, 0, \ldots, 0))$ is not a negative instance of GUARANTEENASH. Then there exists some $(\epsilon/\lceil \log_2 k \rceil)$-Nash equilibrium θ' of $\mathcal{G}'$ such that $v_i'(\theta') > g_i - \epsilon/\lceil \log_2 k \rceil$ for each agent i at the root of a tree. But then by Parts 5 and 4 of Theorem 4.1 respectively, $g(\theta')$ is an ϵ-Nash Equilibrium of $\mathcal{G}$ and $v_i(g(\theta)) = v_i'(\theta') \geq g_i - \epsilon/\lceil \log_2 k \rceil \geq g_i - \epsilon$.

$\mathcal{G}'$ can be computed in log space. □

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.

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 (under 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.

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

PROOF. Given an instance $(\mathcal{G}, \theta, \epsilon)$, where $\mathcal{G}$ is a graph game, θ is an ϵ-Nash equilibrium if and only $v_i(\theta) + \epsilon \geq v_i(R_i(\theta, s_i))$ for all agents i and for all $s_i \in \mathfrak{s}_i$. But there are only polynomially many of these inequalities, and we can compute $v_i(\theta)$ and $v_i(R_i(\theta, s_i))$ in polynomial time. □

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* $\epsilon < 1$.

PROOF. In a 2-player circuit game, Exact IsNash is in **coNP** because given a pair $(\mathcal{G}, \theta)$, we can prove θ is not a Nash equilibrium by guessing an agent i and a strategy s_i', such that agent i can do better by playing s_i'. Then we can compute if $v_i(R_i(\theta, s_i')) > v_i(\theta) + \epsilon$. This computation is in **P** because θ is in the input, represented as a list of the probabilities of each strategy in the support of each player. The same remains true if $\mathcal{G}$ is restricted to any constant number of agents.

It is **coNP**-hard because even in a one-player game we can offer an agent a choice between a payoff of 0 and the output of a circuit C with an input chosen by the player. If the agent settling for a payoff of 0 is a Nash equilibrium, then C is unsatisfiable. Notice that in this game, the range of payoffs is 1, and as long as $\epsilon < 1$, the hardness result will still hold. □

In the previous proof, we obtain the hardness result by making one player choose between many different strategies, 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 under Cook reductions.*

PROOF. We first show 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 $i > 1$. We claim that θ is a Nash equilibrium if and only if $C \in$MajoritySat. All agents besides agent 1 are in equilibrium, so it is a Nash equilibrium if the first player cannot do better by changing his strategy. Currently his expected payoff is $\frac{m}{2^n}$, where m is the number of satisfying assignments of C. If he changes his strategy to 0, his expected payoff will be $\frac{1}{2}$. He must change his strategy if and only if $\frac{1}{2} > \frac{m}{2^n} + \epsilon$.

Now we show that determining if $(\mathcal{G}, \theta, \epsilon) \in$ IsNash is in $\mathbf{P}^{\#\mathbf{P}}$. θ is an ϵ-Nash equilibrium if $v_i(\theta) + \epsilon \geq v_i(R_i(\theta, s_i')) \; \forall \, i \, \forall \, s_i' \in \{0, 1\}$. There are only $2n$ of these equations to check. For any strategy profile θ, we can compute $v_i(\theta)$ as follows.

$$v_i(\theta) = \sum_{s_1 \in supp(\theta_1), \cdots, s_n \in supp(\theta_n)} C_i(s_1, s_2, \ldots, s_n) \prod_{j=1}^{n} \Pr[\theta_j = s_j], \tag{1}$$

where C_i is the circuit that computes v_i. Computing such sums up to polynomially many bits of accuracy can easily be done in $\mathbf{P}^{\#\mathbf{P}}$. □

Remark 5.4. In the same way we can show that, given an input $(\mathcal{G}, \theta, \epsilon, \delta)$ where ϵ and δ are encoded as in Exp-Approx, it is in $\mathbf{P}^{\#\mathbf{P}}$ to differentiate between the case when θ is an ϵ-Nash equilibrium in $\mathcal{G}$ and the case where θ is not a $(\epsilon + \delta)$-Nash equilibrium in $\mathcal{G}$.

Theorem 5.5. *Circuit game* Exp-Approx IsNash *is* $\mathbf{coNP}^{\#\mathbf{P}}$*-complete.*

We first use a definition and a lemma to simplify the reduction.

Definition 5.6. #CircuitSat is the function, which given a circuit C, computes the number of satisfying assignments to C.

It is known that #CircuitSat is #**P**-complete. It follows that every problem in $\mathbf{NP}^{\#\mathbf{P}}$ can be solved in **NP** with an oracle to #CircuitSat. However, using nondeterminism, we can reduce the number of oracle queries to one (which enables us to avoid the need for a Cook reduction, as was used in Theorem 5.3).

Lemma 5.7. *Any language* $L \in \mathbf{NP}^{\#\mathbf{P}}$ *is recognized by a nondeterministic polynomial-time TM that has all its nondeterminism up front, makes only one* #CircuitSat *oracle query, encodes a guess for the oracle query result in its nondeterminism, and accepts only if the oracle query guess encoded in the nondeterminism is correct.*

Proof. Let $L \in \mathbf{NP}^{\#\mathbf{P}}$ and let M be a nondeterministic polynomial-time TM with access to a #CircuitSat oracle that decides L. Then if M fails to satisfy the statement of the lemma, we build a new TM M' that does the following.

(1) Use nondeterminism to
   — guess nondeterminism for M;
   — guess all oracle results for M;
   — guess the oracle query result for M'.
(2) Simulate M using guessed nondeterminism for M and assuming that the guessed oracle results for M are correct. Each time an oracle query is made, record the query and use the previously guessed answer.
(3) Use one oracle query (as described in the following) to check if the actual oracle results correspond correctly with the guessed oracle results.
(4) Accept if all of the following occurred.
   — the simulation of M accepts;
   — the actual oracle query results of M correctly correspond with the guessed oracle results of M;
   — the actual oracle query result of M' correctly corresponds with the guessed oracle result of M';
   otherwise reject.

It is straightforward to check that, if M' decides L, then M' fulfills the requirements of the Lemma.

Now we argue that M has an accepting computation if and only if M' does also. Say that a computation is accepted on M. Then the same computation where the oracle answers are correctly guessed will be accepted on M'. Now say that an computation is accepted by M'. This means that all the oracle answers were correctly guessed, and that the simulation of M accepted; so this same computation will accept on M.

Finally, we need to show that Step 3 is possible. That is that we can check whether all the guessed oracle query results are correct with only one query to a #CIRCUITSAT oracle. Specifically, we need to test if circuits $C_1, \ldots, C_k$ with $n_1, \ldots, n_k$ inputs, respectively, have $q_1, \ldots, q_k$ satisfying assignments, respectively. For each circuit C_i create a new circuit C'_i by adding $\sum_{j=1}^{i-1} n_j$ dummy variables to C_i. Then create a circuit C that takes as input an integer i and a bit string X of size $\sum_{j=1}^{k} n_j$, as follows.

(1) If the last $\sum_{j=i+1}^{k} n_j$ bits of X are not all 0, then $C(i, X) = 0$.
(2) Otherwise, $C(i, X) = C'_i(X)$, where we use the first $\sum_{j=1}^{i} n_j$ bits of X as an input to C'_i.

The circuit C has $\sum_{i=1}^{k} \left(q'_i \cdot 2^{n_1+n_2+\cdots+n_{i-1}}\right)$ satisfying assignments where q'_i is the number of satisfying assignments of C_i. Note that this number together with the n_i's uniquely determines the q'_i's. Therefore it is sufficient to check if the number of satisfying assignments of C equals $\sum_{i=1}^{k} (q_i \cdot 2^{n_1+n_2+\cdots+n_{i-1}})$. □

COROLLARY 5.8. *Any language $L \in \mathbf{coNP}^{\#\mathbf{P}}$ is recognized by a conondeterministic polynomial-time TM that has all its nondeterminism up front, makes only one* #CIRCUITSAT *oracle query, encodes a guess for the oracle query result in its nondeterminism, and rejects only if the oracle query guess encoded in the nondeterminism is correct.*

PROOF. Say $L \in \mathbf{coNP}^{\#\mathbf{P}}$, then the compliment of L, $\bar{L}$, is in $\mathbf{NP}^{\#\mathbf{P}}$. We can use Lemma 5.7 to design a TM M as in the statement of Lemma 5.7 that accepts $\bar{L}$. Create a new TM M' from M, where M' runs exactly as M accept switches the output. Then M' is a nondeterministic polynomial-time TM that has all its nondeterminism up front, makes only one #CIRCUITSAT oracle query, and rejects only if an oracle query guess encoded in the nondeterminism is correct. □

PROOF OF THEOREM 5.5. First we show that given an instance $(\mathcal{G}, \theta, \epsilon)$, it is in $\mathbf{coNP}^{\#\mathbf{P}}$ to determine if θ is a Nash equilibrium. If θ is not a Nash equilibrium, then there exists an agent i with a strategy s_i such that $v_i(R_i(\theta, s_i)) > v_i(\theta)$. As in the proof of Theorem 5.3 (see Equation 1), we can check this in $\#\mathbf{P}$ (after nondeterministically guessing i and s_i).

To prove the hardness result, we first note that by Corollary 5.8 it is sufficient to consider only conondeterministic Turing machines that make only one query to a $\#\mathbf{P}$-oracle. Our oracle will use the $\#\mathbf{P}$-complete problem #CIRCUITSAT, so given an encoding of a circuit, the oracle will return the number of satisfying assignments.

Given a $\mathbf{coNP}^{\#\mathbf{P}}$ computation with one oracle query to #CIRCUITSAT, we create a circuit game with $1+2q(|x|)$ agents where $q(m)$ is a polynomial which provides an upper bound on the number of input gates in the circuit that is fed to the #CIRCUITSAT oracle on input string x of length m. Agent 1 can either play a string s_1, which is interpreted as containing the nondeterminism to the computation and an oracle result, or he can play some other strategy $\emptyset$. The rest of the agents, agent 2 through agent $2q(|x|) + 1$, have a strategy space $\mathfrak{s}_i = \{0, 1\}$.

The payoff to agent 1 on the strategy $s = (s_1, s_2, \ldots, s_{2q(|x|)+1})$ is 0 if $s_1 = \emptyset$, and otherwise is $1 - f(s_1) - g(s)$, where $f(s_1)$ is the polynomial-time function checking

if the computation and oracle-response specified by s_1 would cause the conondeterministic algorithm to accept, and $g(s)$ is a function to be constructed below such that $E_{s_2,\ldots,s_{2q(|x|)+1}}[g(s)] = 0$ if s_1 contains the correct oracle query and $E_{s_2,\ldots,s_{2q(|x|)+1}}[g(s)] \geq 1$ otherwise, where the expectations are taken over $s_2, \ldots, s_{2q(|x|)+1}$ chosen uniformly at random. The rest of the agents, agent 2 through agent $2q(|x|) + 1$, receive payoff 1 regardless.

This ensures that if agent 1 plays $\emptyset$ and the other agents randomize uniformly, this is a Nash equilibrium if there is no rejecting computation and is not even a $1/2$-Nash equilibrium if there is a rejecting computation. If there is a rejecting computation then the first player can just play that computation and his payoff will be 1. If there is no rejecting computation, then either $f(s_1) = 1$ or contains an incorrect query result, in which case $E_{s\to\theta}[g(s)] \geq 1$. If either the circuit accepts or his query is incorrect, then the payoff will always be at most 0.

Now we construct $g(s_1, s_2, \ldots, s_{2q(|x|)+1})$. Let C, a circuit, be the oracle query determined by the nondeterministic choice of s_1, let k be the guessed oracle results, and let $S_1 = s_2 s_3 \ldots s_{q(|x|)+1}$, and $S_2 = s_{q(|x|)+2} s_{q(|x|)+3} \ldots s_{2q(|x|)+1}$. For convenience we will write g in the form $g(k, C(S_1), C(S_2))$.

$$\begin{aligned} g(k,1,1) &= k^2 - 2^{n+1}k + 2^{2n} \\ g(k,0,1) &= g(k,1,0) = -2^n k + k^2 \\ g(k,0,0) &= k^2. \end{aligned}$$

Now let m be the actual number of satisfying assignments of C. Then, if agent 2 through agent $2q(|x|) + 1$ randomize uniformly over their strategies,

$$\begin{aligned} &\mathbb{E}[g(k, C(S_1), C(S_2))] \\ &= (m/2^n)^2 g(k,1,1) + 2(m/2^n)(1-(m/2^n))g(k,0,1) + (1-(m/2^n))^2 g(k,0,0) \\ &= 2^{2n}(m/2^n)^2 - 2^{n+1}(m/2^n)k + k^2 = (m-k)^2. \end{aligned}$$

So if $m = k$ then $E[g] = 0$, but if $m \neq k$ then $E[g] \geq 1$. In the preceding game, the range of payoffs is not bounded by any constant, so we scale $\mathcal{G}$ to make all payments in $[0, 1]$ and adjust ϵ accordingly. □

Notice that 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 5.9. *With* POLY-APPROX *and* CONST-APPROX*, Boolean circuit game* ISNASH *is* **BPP***-complete.*[7] *Furthermore, this holds for any approximation error* $\epsilon < 1$.

PROOF. We start by showing Boolean circuit game POLY-APPROX ISNASH is in **BPP**. Given an instance $(\mathcal{G}, \theta, \epsilon)$, for each agent i and each strategy $s_i \in \{0, 1\}$, we use random sampling of strategies according to θ to distinguish the following two possibilities in probabilistic polynomial time.

— $v_i(\theta) \geq v_i(R_i(\theta, s_i))$, OR
— $v_i(\theta) + \epsilon < v_i(R_i(\theta, s_i))$

(We will show how we check this in a moment.) If it is a Nash equilibrium then the first case is true for all agents i and all $s_i \in \{0, 1\}$. If it is not an ϵ-Nash equilibrium, then the second case is true for some agents i and some $s_i \in \{0, 1\}$. So, it is enough to be able to distinguish these cases with high probability.

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

Now the first case holds if $v_i(\theta) - v_i(R_i(\theta, s_i)) \geq 0$ and the second case holds if $v_i(\theta) - v_i(R_i(\theta, s_i)) \leq -\epsilon$. We can view $v_i(\theta) - v_i(R_i(\theta, s_i))$ as a random variable with the range $[-1, 1]$ and so, by a Chernoff bound, averaging a polynomial number of samples (in $1/\epsilon$), the chance that the deviation will be more than $\epsilon/2$ will be exponentially small, and so the total chance of an error in the $2n$ computations is $< \frac{1}{3}$ by a union bound.

Remark 5.10. In the same way we can show that, given an input $(\mathcal{G}, \theta, i, k, \epsilon)$ where $\mathcal{G}$ is a circuit game, θ is a strategy profile of $\mathcal{G}$, ϵ is encoded as in POLY-APPROX, it is in **BPP** to differentiate between the case when $v_i(\theta) \geq k$ and $v_i(\theta) < k - \epsilon$.

Remark 5.11. *Also, in this way we can show that, given an input $(\mathcal{G}, \theta, \epsilon, \delta)$ where $\mathcal{G}$ is a Boolean circuit game, θ is a strategy profile of $\mathcal{G}$, and ϵ and δ are encoded as in* POLY-APPROX*, it is in* **BPP** *to differentiate between the case when θ is an ϵ-Nash equilibrium in $\mathcal{G}$ and the case where θ is not a $(\epsilon + \delta)$-Nash equilibrium in $\mathcal{G}$.*

We now show that Boolean circuit game CONST-APPROX ISNASH is **BPP**-hard. Fix some $\epsilon < 1$. Let $\delta = \min\{\frac{1-\epsilon}{2}, \frac{1}{4}\}$.

We create a reduction as follows. Given a language L in **BPP** there exists an algorithm $A(x, r)$ that decides if $x \in L$ using coin tosses r with two-sided error of at most δ. Now create $\mathcal{G}$ with $|r| + 1$ agents. The first player gets paid $1 - \delta$ if he plays 0, or the output of $A(x, s_2 s_3 \ldots s_n)$ if he plays 1. All the other players have a strategy space of $\{0, 1\}$ and are paid 1 regardless. Let the strategy profile θ be such that $\Pr[\theta_1 = 1] = 1$ and $\Pr[\theta_i = 1] = \frac{1}{2}$ for $i > 1$.

Each of the players besides the first player are in equilibrium because they always receive their maximum payoff. The first player is in equilibrium if $\Pr[A(x, s_2 s_3 \ldots s_n)] \geq 1 - \delta$, which is true if $x \in L$. However, if $x \notin L$, then $v_1(\theta) = \Pr[A(x, s_2 s_3 \ldots s_n)] < \delta$, but $v_1(R_1(\theta, 0)) = 1 - \delta$. So agent 1 could do better by $v_1(R_1(\theta, 0)) - v_1(\theta) > 1 - \delta - \delta \geq \epsilon$. □

THEOREM 5.12. *With* POLY-APPROX *and* CONST-APPROX*, circuit game* ISNASH *is* $\mathbf{coNP^{BPP}} = \mathbf{coMA}$*-complete. Furthermore, this holds for any approximation error $\epsilon < 1$.*

ACAPP, the *Approximate Circuit Acceptance Probability Problem* is the promise-language where positive instances are circuits that accept at least 2/3 of their inputs, and negative instances are circuits that reject at least 2/3 of their inputs. ACAPP is in **prBPP** and any instances of a **BPP** problem can be reduced to an instance of ACAPP.

LEMMA 5.13. *Any language $L \in \mathbf{NP^{BPP}}$ is recognized by a nondeterministic polynomial-time TM that has all its nondeterminism up front, makes only one* ACAPP *oracle query, encodes an oracle query guess in its nondeterminism, and accepts only if the oracle query guess is correct.*

PROOF. The proof is exactly the same as that for Lemma 5.8 except that we now need to show that we can check arbitrarily many **BPP** oracle queries with only one query.

Because any **BPP** instance can be reduced to ACAPP, we can assume that all oracle calls are to ACAPP. We are given circuits $C_1, \ldots, C_n$ and are promised that each circuit C_i either accepts at least 2/3 of its inputs, or accepts at most 1/3 of its inputs. Without loss of generality, we are trying to check that each circuit accepts at least 2/3 of its inputs (simply negate each circuit that accepts fewer than 1/3 of its inputs). Using boosting, we can instead verify that circuits $C'_1, \ldots, C'_n$ each reject on fewer than $\frac{1}{2^{n+1}}$

of their inputs. So simply AND the C'_i circuits together to create a new circuit C'', and send C'' to the **BPP** oracle. Now if even one C_i does not accept 2/3 of its inputs, then C'_i will accept at most a $\frac{1}{2^{n+1}}$ faction of inputs. So also, C'' will accept at most a $\frac{1}{2^{n+1}}$ faction of inputs. But if all the C_i circuits accept at least 2/3 of their inputs, then each C'_i will accept at least a $1 - \frac{1}{2^{n+1}}$ faction of its inputs. So C'' will accept at least a $1 - \frac{n}{2^{n+1}} > 2/3$ fraction of its inputs. □

LEMMA 5.14. *Any language $L \in$ **coNP**$^{\mathbf{BPP}}$ is decided by conondeterministic TM that only uses one* **BPP** *oracle query to* ACAPP*, has all its nondeterminism up front, encodes an oracle query guess in its nondeterminism, and rejects only if the oracle query guess is correct.*

PROOF. This corollary follows from Lemma 5.13 in exactly the same way as Corollary 5.8 followed from Lemma 5.7. □

PROOF OF THEOREM 5.12. First we show that circuit game POLY-APPROX ISNASH is in **coNP**$^{\mathbf{BPP}}$. Say that we are given an instance $(\mathcal{G}, \theta, \epsilon)$. We must determine if θ is a Nash equilibrium or if it is not even an ϵ-Nash equilibrium.

To do this, we define a promise language L with the following positive and negative instances.

$$L^+ = \{((\mathcal{G}, \theta, \epsilon), (i, s'_i)) : s'_i \in \mathfrak{s}_i,\ v_i(R_i(\theta, s'_i)) \leq v_i(\theta)\}$$
$$L^- = \{((\mathcal{G}, \theta, \epsilon), (i, s'_i)) : s'_i \in \mathfrak{s}_i,\ v_i(R_i(\theta, s'_i) > v_i(\theta) + \epsilon\}$$

Now if for all pairs (i, s'_i), $((\mathcal{G}, \theta, \epsilon), (i, s'_i)) \in L^+$, then θ is a Nash equilibrium of $\mathcal{G}$, but if there exists (i, s'_i), such that $((\mathcal{G}, \theta, \epsilon), (i, s'_i)) \in L^-$, then θ is not an ϵ-Nash equilibrium of $\mathcal{G}$. But $L \in$ **BPP** because, by Remark 5.10, as we saw in the proof of Theorem 5.9, we can just sample $v_i(\theta) - v_i(R_i(\theta, s'_i)) = \mathbb{E}_{s \leftarrow \theta}[v_i(s) - v_i(R_i(s, s'_i))]$ to see if it is ≥ 0 or $< -\epsilon$.

Remark 5.15. In the same way we can show that, given an input $(\mathcal{G}, \theta, \epsilon, \delta)$, where $\mathcal{G}$ is a circuit game, θ is a strategy profile of $\mathcal{G}$, and ϵ and δ are encoded as in POLY-APPROX, it is in **coNP**$^{\mathbf{BPP}}$ to differentiate between the case when θ is an ϵ-Nash equilibrium in $\mathcal{G}$ and the case where θ is not an $(\epsilon + \delta)$-Nash equilibrium in $\mathcal{G}$.

Now we show that circuit game CONST-APPROX ISNASH is **coNP**$^{\mathbf{BPP}}$-hard. The proof is similar to the proof of Theorem 5.5

Fix $\epsilon < 1$ and let $\delta = \min\{\frac{1-\epsilon}{2}, \frac{1}{4}\}$. Given a **coNP**$^{\mathbf{BPP}}$ computation with one oracle query, we create a circuit game with $q(|x|) + 1$ agents, where q is some polynomial which we will define later. Agent 1 can either play a string s_1 that is interpreted as containing the nondeterminism to be used in the computation and an oracle answer, or he can play some other strategy $\emptyset$. The other agents, agent 2 through agent $q(|x|) + 1$, have strategy space $\mathfrak{s}_i = \{0, 1\}$.

The payoff to agent 1 is δ for $\emptyset$, and $1 - \max\{f(s_1), g(s)\}$ otherwise, where $f(s_1)$ is the polynomial time function that we must check, and $\mathbb{E}_{s_2,\ldots,s_{q(|x|)+1}}[g(s)] > 1 - \delta$ if the oracle guess is incorrect, and $\mathbb{E}_{s_2,\ldots,s_{q(|x|)+1}}[g(s)] < \delta$ of the oracle guess is correct. The other agents are paid 1 regardless.

We claim that if agent 1 plays $\emptyset$ and the other agents randomize uniformly, this is a Nash equilibrium if there is no rejecting computation and is not even a δ-Nash equilibrium if there is a failing computation.

In the first case, if the first agent does not play $\emptyset$, either the computation will accept and his payoff will be 0, or the computation will reject but the guessed oracle results will be incorrect and his expected payoff will be

$$1 - \max\{f(s_1), g(s)\} = 1 - \max\{0, g(s)\} = 1 - \mathbb{E}[g(s)] > 1 - (1 - \delta) = \delta,$$

so he would earn at least that much by playing $\emptyset$.

In the latter case where there is a failing computation, by playing that and a correct oracle result, agents 1's payoff will be $1 - \max\{f(s_1), g(s)\} > 1 - \delta$. If we compare this to what he would be paid for playing $\emptyset$, we see that it is greater by $[1 - \delta] - [\delta] \geq \epsilon$.

Now we define $g(s)$. Let C_{s_1} be the circuit corresponding to the oracle query in s_1, and let, $C_{s_1}^{(k)}$ be the circuit corresponding to running C_{s_1} k times, and taking the majority vote. We define $g(s) = 0$ if $C_{s_1}^{(k)}(s_2 s_3 \cdots s_{q(|x|)})$ agrees with the oracle guess in s_1 and $g(s) = 1$ otherwise. Now if the oracle result is correct, then the probability that $C_{s_1}^{(k)}(s_2 s_3 \cdots s_{q(|x|)})$ agrees with it is $1 - 2^{-\Omega(k)}$, and if the oracle results are incorrect, the probability that $C_{s_1}^{(k)}(s_2 s_3 \cdots s_{q(|x|)})$ agrees with the oracle results (in s_1) is $2^{-\Omega(k)}$, so by correctly picking k, $g(s)$ will have the desired properties. Define $q(|x|)$ accordingly. □

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 nonBoolean circuit games, IsPureNash is **coNP**-complete. With the other games examined, IsPureNash is in **P**.

Proposition 5.16. *With any approximation scheme, circuit game, and 2-player circuit game* IsPureNash *is* **coNP***-complete. Furthermore, it remains hard for any approximation error* $\epsilon < 1$.

Proof. The proof is the same as that for Proposition 5.2: in the reduction for the hardness result, θ is always a pure-strategy profile. It is in **coNP** because it is a more restricted class of problems than circuit game IsPureNash, which is in **coNP**. □

Proposition 5.17. *With any approximation scheme, Boolean circuit game* IsPureNash *is* **P***-complete, and remains so even for one player and any approximation error* $\epsilon < 1$.

Proof. It is in **P** because each player has only one alternative strategy, so there are only polynomially many possible deviations, and the payments for any given strategy can be computed in polynomial time.

It is **P**-hard even in a one-player game because, given a circuit C with no inputs (an instance of CircuitValue, which is known to be **P**-hard), we can offer an agent a choice between a payoff of 0 and the output of the circuit C. If the agent settling for a payoff of 0 is a Nash equilibrium, then C then must evaluate to 0. Notice that in this game, the range of payoffs is 1, and as long as $\epsilon < 1$, the hardness result will still hold. □

Proposition 5.18. *With any approximation scheme, graph game* IsPureNash *is in* **P** *for any kind of graph game.*

Proof. In all these representations, given a game $\mathcal{G}$, there are only a polynomial number of players, and each player has only a polynomial number of strategies. To check that s is an ϵ-Nash equilibrium, one has to check that for all agents i and

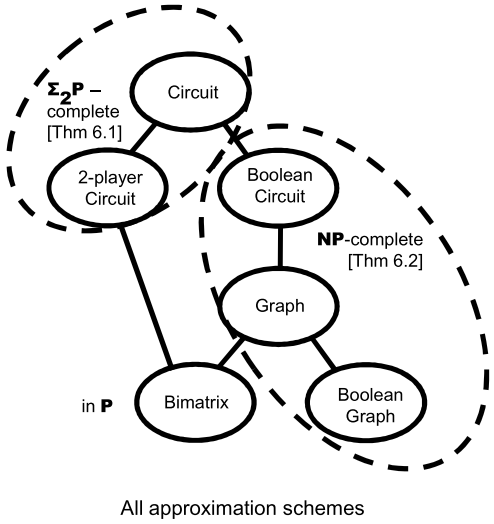

Fig. 4. Summary of EXISTSPURENASH results.

strategies $s_i' \in \mathfrak{S}_i$, $v_i(s) \geq v_i(R_i(s, s_i))$. But as mentioned there are only polynomially many of these strategies and each can be evaluated in polynomial time. □

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.

The hardness of these problems is directly related to the hardness of ISPURENASH. We can always solve EXISTSPURENASH with one more nondeterministic alternation because we can nondeterministically guess a pure-strategy profile, and then check that it is pure-strategy Nash equilibrium. Recall that in the case of nonBoolean 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 this strategy of nondeterministically guessing and then checking is the best that one can do. However, for bimatrix games the problem still remains in **P** because there are only a polynomial number of possible guesses.

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 Gilboa and Zemel [1989] and Conitzer and Sandholm [2008] 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* Σ_2**P***-complete with any of the defined notions of approximation. Furthermore, it remains hard as long as approximation error* $\epsilon < 1$ *and even in zero-sum games.*

PROOF. Membership in $\Sigma_2\mathbf{P}$ follows by observing that the existence of a pure-strategy Nash equilibrium is equivalent to the following $\Sigma_2\mathbf{P}$ predicate.

$$\exists s \in \mathfrak{s}, \forall\, i, s_i' \in \mathfrak{s}_i \ \left[\nu_i(R_i(s, s_i')) \leq \nu_i(s) + \epsilon\right].$$

To show it is $\Sigma_2\mathbf{P}$-hard, we reduce from the $\Sigma_2\mathbf{P}$-complete problem

$$\text{QCIRCUITSAT}_2 = \{(C, k_1, k_2) : \exists x_1 \in \{0,1\}^{k_1}, \forall x_2 \in \{0,1\}^{k_2}\ C(x_1, x_2) = 1\},$$

where C is a circuit that takes $k_1 + k_2$ Boolean variables. Given an instance (C, k_1, k_2) of QCIRCUITSAT$_2$, create 2-player circuit game $\mathcal{G} = (\mathfrak{s}, \nu)$, where $\mathfrak{s}_i = \{0,1\}^{k_i} \cup \{\perp\}$.

Player i has the choice of playing a strategy $x_i \in \{0,1\}^{k_i}$ or a strategy $x_i = \perp$. The payoffs for the first player are as follows.

		Player 2	
		x_2	$\perp$
Player 1	x_1	$C(x_1, x_2)$	1
	$\perp$	1	0

Payoffs of player 1

If both players play an input to C, then player 1 gets paid the results of C on these inputs. If both play $\perp$, the payoff to the first player is 0. If one player plays an input to C, and the other plays $\perp$, then the first player receives 1.

For every pure strategy profile, we define the payoff of the second player to be $\nu_2(s) = 1 - \nu_1(s)$. Since the sum of the payoffs is constant, this is a game constant-sum game.[8]

Now we show that this construction indeed gives a reduction from QCIRCUITSAT$_2$ to 2-player Circuit Game EXISTSPURENASH. Suppose that $(C, k_1, k_2) \in$QCIRCUITSAT$_2$. Then there is an $x_1 \in \{0,1\}^{k_1}$ such that $\forall x_2 \in \{0,1\}^{k_2}$, $C(x_1, x_2) = 1$, and we claim $(x_1, 0^{k_2})$ is a pure-strategy Nash equilibrium. Player 1 receives a payoff of 1 and so cannot do better. Player 2 will get payoff 0 no matter what $s_2 \in \mathfrak{s}_2$ he plays, and so is indifferent.

Now suppose that $(C, k_1, k_2) \notin$QCIRCUITSAT$_2$, i.e., $\forall x_1, \exists x_2\ C(x_1, x_2) = 0$. Then we want to show there does not exist a pure-strategy ϵ-Nash equilibrium. We show that for any strategy profile (s_1, s_2) either player 1 or player 2 can do better by changing strategies.

Fix some pure-strategy profile $s = (s_1, s_2)$. Either $\nu_1(s) = 0$ or $\nu_2(s) = 0$. We first examine the case that $\nu_1(s) = 0$. If $s_2 \in \{0,1\}^{k_2}$, then player 1 can increase his payoff by playing $s_1' = \perp$. Alternatively if $s_2 = \perp$, then player 1 can increase his payoff by playing $s_1' \in \{0,1\}^{k_1}$.

Now say that $\nu_2(s) = 0$. If $s_1 \in \{0,1\}^{k_2}$, then player 2 can increase his payoff by playing $s_2' \in \{0,1\}^{k_2}$ such that $C(s_1, s_2') = 0$; such an s_2' must exist by assumption. Alternatively if $s_1 = \perp$, then player 2 can increase his payoff by playing $s_2' = \perp$ such that $y_2 = \neg y_1$ and receive payoff 1.

Because the payoffs are only 0 and 1, the increase is always 1 and so the hardness results holds for any $\epsilon < 1$. □

For graph games, it was recently shown by Gottlob, Greco, and Scarcello [Gottlob et al. 2003] that EXISTSPURENASH is **NP**-complete, even restricted to graphs of degree 4. In the following, we strengthen their result by showing this also holds for BOOLEAN graph games, for graphs of degree 3, and for any approximation error $\epsilon < 1$.

[8]Recall that constant-sum games are computationally equivalent to zero-sum games.

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. We first show that Boolean circuit game EXACT EXISTSPURENASH is in **NP**. Then, by Theorem 4.1, EXACT EXISTSPURENASH is in **NP** for graph games as well. Adding approximation only makes the problem easier. Given an instance $(\mathcal{G}, \epsilon)$, we can guess a pure-strategy profile θ. Let $s \in \mathfrak{s}$ such that $\Pr[\theta = s] = 1$. Then, for each agent i, in polynomial time we can check that $v_i(s) \geq v_i(R_i(s, s_i')) - \epsilon$ for all $s_i' \in \{0, 1\}$. There are only polynomially many agents, so this takes at most polynomial time.

Now we show that EXISTSPURENASH is also **NP**-hard, even in degree-3 Boolean graph games with CONST-APPROX for every $\epsilon < 1$. 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 a player with the strategy space {`true`, `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.

We claim that pure strategy Nash equilibria only exist when C is satisfiable. Say C is satisfiable and let the input agents play a satisfying assignment, and let all the gate agents play the correct value of their gate, given the input agents' strategies. Because it is a satisfying assignment, the judge plays `true`, and so every agent—the input agents, the gate agents, P_1, and P_2—receive a payoff of 1, and are thus in a Nash equilibrium.

Say C is not satisfiable. The judge cannot play `true` in any Nash equilibrium. For, to all be in equilibrium, the gate agents must play the correct valuation of their gate. Because C is unsatisfiable, no matter what pure-strategies the input agents play, the circuit will evaluate to false, and so in no equilibrium will the judge play `true`. But if the judge plays `false`, then P_1 and P_2 are playing pennies against each other, and so there is no pure-strategy Nash equilibrium.

Because the only payoffs possible are 0 and 1, if any agent is not in equilibrium, he can do at least 1 better by changing his strategy. So there does not exist a pure-strategy ϵ-Nash equilibrium for any $\epsilon < 1$.

Note that the in-degree of each agent is at most 3 (recall that we count the agent himself if he influences his own payoff), and that the total degree of each agent is at most 4. □

Like ISPURENASH, EXISTSPURENASH does not become easier with approximation, even if we approximate as much 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

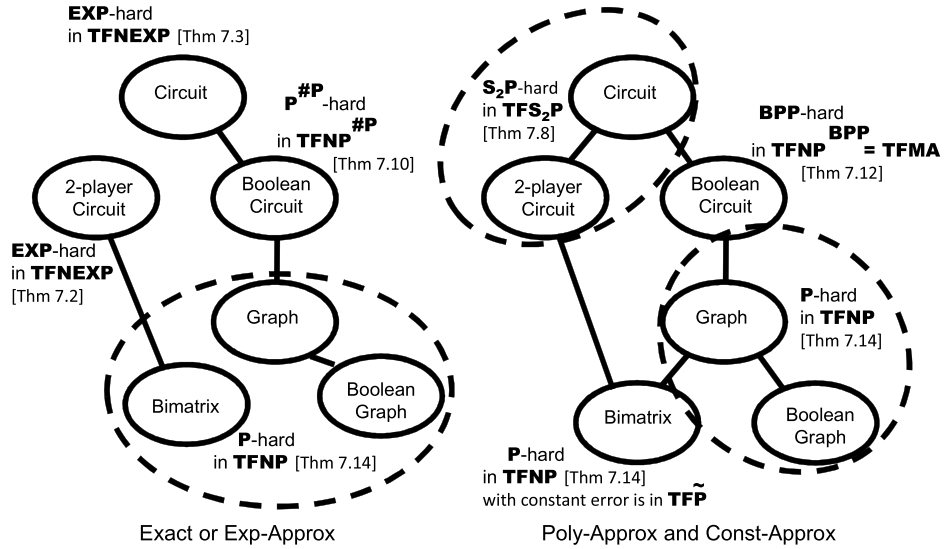

Fig. 5. Summary of FINDNASH results.

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 Megiddo and Papadimitriou [1991]. Recently, Lipton et al. [2003] showed that if we allow constant error, the bimatrix case FINDNASH can be computeded in quasipolynomial time (i.e. $\mathbf{TF\tilde{P}}$). Our results are summarized in Figure 5.

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 Work. Exciting recent results have exactly characterized the complexity of FINDNASH in graph games and bimatrix games. Specifically, it was shown by Daskalakis, Goldberg, and Papadimitriou [Daskalakis et al. 2006] that, in graph games and Boolean graph games, EXACT and EXP-APPROX FINDNASH are complete for the class **PPAD** introduced in Megiddo and Papadimitriou [1991]. Later, it was shown by Chen and Deng [2006] that the same is true for bimatrix games. Chen, Deng, and Teng [Chen et al. 2006] showed that even POLY-APPROX FINDNASH is complete for the class **PPAD**. We note that these results leave open the complexity of FINDNASH in circuit games with any approximation, and in graphical and bimatrix games with constant approximation.

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 reduces to FINDNASH (via a Karp reduction), but the converse is not clear.

THEOREM 7.2. *2-player circuit game* EXACT FINDNASHSIMPLE *is* **EXP***-hard, but is in* **NEXP** ∩ **coNEXP**. *Moreover, 2-player circuit game* EXACT FINDNASH *is in* **TFNEXP**.

In the proof we will reduce from a problem called GAMEVALUE. A 2-player game is a *zero-sum game* if $\nu_1(s) = -\nu_2(s)$ for all $s \in \mathfrak{s}$. By the von Neumann min-max theorem, for every 2-player zero-sum game there exists a value $\nu(\mathcal{G})$, such that in any Nash equilibrium θ of $\mathcal{G}$, $\nu_1(\theta) = -\nu_2(\theta) = \nu(\mathcal{G})$. Moreover, it is known that, given a 2-player circuit game $\mathcal{G}$, it is **EXP**-hard to decide if $\nu(\mathcal{G}) \geq 0$ [Feigenbaum et al. 1995].

PROOF OF THEOREM 7.2. We reduce from 2-player circuit game GAMEVALUE. Say we are given such a zero-sum game $\mathcal{G} = (\mathfrak{s}, \nu)$ and we want to decide if $\nu(\mathcal{G}) \geq 0$. Without loss of generality, assume the payoffs are between $\pm 1/2$. We construct a game $\mathcal{G}' = (\mathfrak{s}', \nu')$ as follows. $\mathfrak{s}'_1 = \mathfrak{s}_1 \cup \{\emptyset\}$, $\mathfrak{s}'_2 = \mathfrak{s}_2 \cup \{\emptyset\}$, and the payoffs are

		Player 2	
		s_2	$\emptyset$
Player 1	s_1	$1 + \nu_1(s_1, s_2)$	0
	$\emptyset$	0	1

Payoffs of player 1 in $\mathcal{G}'$

		Player 2	
		s_2	$\emptyset$
Player 1	s_1	$\nu_2(s_1, s_2)$	1
	$\emptyset$	1	0

Payoffs of player 2 in $\mathcal{G}'$.

We claim that if $\nu(\mathcal{G}) \geq 0$, then $(\mathcal{G}', 2, \emptyset, 1/2)$ is a positive instance of FINDNASHSIMPLE, and if $\nu(\mathcal{G}) < 0$, then $(\mathcal{G}', 2, \emptyset, 1/2)$ is a negative instance of FINDNASHSIMPLE. Fix θ', a Nash equilibrium for $\mathcal{G}'$. Let $p_i = \Pr[\theta'_i \in \mathfrak{s}_i]$. It is straightforward to check that $p_1, p_2 \neq 0, 1$. Let θ be the strategy profile where θ_i is distributed as θ'_i given that $\theta'_i \in \mathfrak{s}_i$. This is well defined because $p_1, p_2 \neq 0$. Also, θ is a Nash equilibrium of $\mathcal{G}$ because if either player could increase their payoff in $\mathcal{G}$ by deviating from θ, they could also increase their payoff in $\mathcal{G}'$.

We will now relate $1 - p_2$, the probability that agent 2 plays $\emptyset$, to $\nu(\mathcal{G})$. The expected payoff to player 1 is $p_1 p_2(1+\nu(\mathcal{G}))+(1-p_1)(1-p_2)$, which we can view as a function of p_1. Because agent 1 is in an equilibrium, he must play p_1 to maximize this function. This can only happen at the end points ($p_1 = 0$ or 1) or when the derivative with respect to p_1 is 0. We have already observed that no Nash equilibrium occurs when $p_1 = 0$ or 1, so one must occur when the derivative is 0. The derivative with respect to p_1 is

$p_2(1+\nu(G))-(1-p_2)$. And so $p_2(1+\nu(G))-(1-p_2) = 0 \Rightarrow p_2 = \frac{1}{2+\nu(G)} \Rightarrow 1-p_2 = 1-\frac{1}{2+\nu(G)}$. Therefore, if $\nu(\mathcal{G}) \geq 0$ then in any Nash equilibrium θ', $\Pr[\theta'_2 = \emptyset] \geq \frac{1}{2}$; but if $\nu(\mathcal{G}) < 0$ then in any Nash equilibrium θ', $\Pr[\theta'_2 = \emptyset] < \frac{1}{2}$.

Now we show that 2-player circuit game EXACT FINDNASHSIMPLE is in **NEXP** $\cap$ **coNEXP**. Given input $(\mathcal{G}, i, s_i, k, \epsilon)$, the nondeterminism guesses a strategy profile θ that is at most exponentially long, and then checks whether it is a Nash equilibrium in **EXP**. The machine will output "positive" if θ is a Nash Equilibrium of $\mathcal{G}$ and $\Pr[\theta_i = s_i] \geq k$. The machine will output "negative" if θ is a Nash Equilibrium of $\mathcal{G}$ and $\Pr[\theta_i = s_i] < k$. The machine will fail if θ is not an exact Nash Equilibrium of $\mathcal{G}$. (The machine ignores the parameter ϵ.)

If the input is a positive instance of FINDNASHSIMPLE, then the machine will always output "positive" or fail. Similarly, if the input is a negative instance of FINDNASHSIMPLE, then the machine will always output "negative" or fail. By Proposition 3.3, an exact Nash equilibrium that can be encoded in exponential space always exists in a 2-player circuit game. If the nondeterminism guesses this strategy profile, then the machine will not fail.

2-player circuit game EXACT FINDNASH is in **TFNEXP**, because given input $(\mathcal{G}, \epsilon)$, the nondeterminism guesses a strategy profile θ that is at most exponentially long, and then checks whether it is a Nash equilibrium in **EXP**. By Proposition 3.3, a Nash equilibrium that can be encoded in exponential space always exists in a 2-player circuit game.

Because the description of the strategy-profile that was guessed may be exponential in length, we cannot simply use our result from ISNASH (Proposition 5.2) to show that we can determine if θ is a Nash equilibrium. However, it is not hard to see that we can verify this in a straightforward manner by computing, for each agent, $i \in \{1, 2\}$, $\nu_i(\theta)$, and $\nu_i(R_i(\theta, s_i))$ for all $s_i \in \mathfrak{s}_i$. □

This result is analogous to the bimatrix case [Megiddo and Papadimitriou 1991]; everything scales up by an exponential factor.

The problem becomes more tedious when we add exponentially small error. The difficulty is that we only know GAMEVALUE is hard to solve exactly. Because we introduce an element of approximation, we cannot use the same reduction in a straightforward manner. The reductions from **EXP** to GAMEVALUE used in Fortnow et al. [2008] and Feigenbaum et al. [1995] require an error bound that is at least doubly exponentially small.

THEOREM 7.3. *Circuit game* EXP-APPROX FINDNASHSIMPLE *is* **EXP***-hard, but is in* **NEXP** $\cap$ **coNEXP**. *The* **EXP***-hardness holds even for circuit games with 6 players. Moreover, Circuit game* EXP-APPROX FINDNASH *is* **TFNEXP**.

Definition 7.4. Let C' be a circuit where every gate is numbered, has at most 2 inputs, and is of the type ZERO, ONE, AND, OR, or NOT. Furthermore let the output gate of C' be labeled 0. We say that a circuit C encodes a circuit C' if $C(i)$ outputs $input_1(i), input_2(i), type(i)$, where $input_1(i)$ is the label of the first gate input to gate i, $input_2(i)$ is the label of the second gate input to gate i, and $type(i)$ is the type of gate i. C will output the special character $\perp$ if i is not the label of a gate of C', or in place of $input_1(i)$ and/or $input_2(i)$ if gate i does not have 2 input gates.

SUCCINCTCIRCUITVALUE is the language that consists of circuits C such that the circuit C encodes a circuit C' and circuit C' evaluates to true.

It will not be important to us how exactly C encodes C' the encodings work, but only that SUCCINCTCIRCUITVALUE is an **EXP**-complete language.

PROOF. We first prove 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 $\mathcal{G} = (\mathfrak{s}, v)$.

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 of the enforcing agents and the computing agents are designed so that in any ϵ-Nash equilibrium, each computing agent must play g or $\bar{g}$ with probability at least $1/N - \epsilon$. The payoffs of the computing agents are also designed so that each computing agent must play a strategy that corresponds with a correct computation of C. That is, if g evaluates to true, each computing agent must play g with probability close to $\frac{1}{N}$ and $\bar{g}$ with probability close to 0; and if g evaluates to false, vice versa.

Let $B = \frac{2\epsilon+N^2}{\epsilon}\frac{N-1}{N-2}$. We define the payoffs of the enforcer agents as follows.

s_{e_i}	s_{c_i}	v_{e_i}
g	$g, \bar{g}$	$-B$
g	$\neq g, \bar{g}$	$\frac{B}{N-1}$

We define the payoffs of the computing agents as follows (t will be defined momentarily).

$$\begin{aligned} v_{c_1}(s) &= t(s_{c_1}, s_{c_2}, s_{c_3}) - v_{e_1}(s) \\ v_{c_2}(s) &= t(s_{c_2}, s_{c_3}, s_{c_1}) - v_{e_2}(s) \\ v_{c_3}(s) &= t(s_{c_3}, s_{c_1}, s_{c_2}) - v_{e_3}(s) \end{aligned}$$

We now define $t(s_{c_i}, s_{c_j}, s_{c_k})$, which will always be either $-N^2$ or 0. Define g to be the gate such that $s_{c_i} \in \{g, \bar{g}\}$. Let gates g_1 and g_2 be the two inputs of g. If g_1 is a constant, then for this definition ignore s_{c_j} and instead use the value of the constant. Do likewise for g_2 and s_{c_k}.

Then $t(s_{c_i}, s_{c_j}, s_{c_k}) =$

(1) $-N^2$ if $s_{c_j} \in \{g_1, \bar{g_1}\}$ (or g_1 is a constant) and $s_{c_k} \in \{g_2, \bar{g_2}\}$ (or g_2 is a constant) and $g(s_{c_j}, s_{c_k}) \neq s_{c_i}$, where $g(s_{c_j}, s_{c_k})$ is the output of circuit g using s_{c_j} and s_{c_k} (or the respective constants) as the inputs.
(2) 0 Otherwise

Let $\epsilon = \frac{1}{100 \cdot N^2}$.

CLAIM 7.5. *In any ϵ-Nash equilibrium θ, for any $i \in \{1, 2, 3\}$ and any $g \in G$,* $\Pr[\theta_{c_i} = g, \bar{g}] \geq 1/N - \epsilon$.

PROOF OF CLAIM. Say not, then for some agent c_i and gate g, $\Pr[\theta_{c_i} = g, \bar{g}] < 1/N - \epsilon$. We show that in such a case, agent c_i can do ϵ better by changing his strategy.

Let $g = \arg\min_{g \in G} \Pr[\theta_{c_i} = g, \bar{g}]$, and let $p = 1/N - \Pr[\theta_{c_i} = g, \bar{g}]$. By playing g, e_i will make

$$-(1/N - p)B + \left(\frac{N-1}{N} + p\right)\frac{B}{N-1} = pB - p\frac{B}{N-1} = p\frac{N-2}{N-1}B$$

So e_i has a strategy to make at least $p\frac{N-2}{N-1}B$. But θ is a Nash equilibrium, so $v_{e_i}(\theta) \geq p\frac{N-2}{N-1}B - \epsilon$. But this means that

$$v_{c_i}(\theta) = t(\theta_{c_i}, \theta_{c_j}, \theta_{c_k}) - v_{e_i}(\theta) \leq -v_{e_i}(\theta) \leq -p\frac{N-2}{N-1}B + \epsilon,$$

because it is always the case that $t(\theta_{c_1}, \theta_{c_3}, \theta_{c_4}) \leq 0$. If θ'_{c_i} is the mixed strategy where agent c_i randomizes uniformly over all 2N strategies, $v_{c_i}(R_{c_i}(\theta, \theta'_{c_i})) \geq -N^2$. This is because here $v_{e_i}(R_{c_i}(\theta, \theta'_{c_i})) = 0$ no matter what θ_{e_i} is and $t(s_{c_i}, s_{c_j}, s_{c_k}) \geq -N^2$ regardless of the inputs.

Because this is an ϵ-Nash equilibrium,

$$v_{c_i}(R_{c_i}(\theta, \theta'_{c_i})) - v_{c_i}(\theta) \leq \epsilon \Rightarrow -N^2 - [-p\frac{N-2}{N-1}B + \epsilon] \leq \epsilon \Rightarrow p \leq \frac{2\epsilon + N^2}{\frac{N-2}{N-1}B} \Rightarrow p \leq \epsilon.$$

The last step follows because $B = \frac{2\epsilon+N^2}{\epsilon}\frac{N-1}{N-2}$.

When C is correctly evaluated, each gate $g \in G$ evaluates to either true of false. If gate g evaluates to true, we call the strategy g *correct*. If gate g evaluates to false, we call the strategy $\bar{g}$ *correct*. If a strategy is not correct, we say it is *incorrect*. For any gate g, define g^* to be the the correct strategy for g and define $\bar{g}^*$ to be the incorrect strategy for g. In $\mathfrak{s}_{c_i}$ there are N correct strategies, and N incorrect strategies.

CLAIM 7.6. *In any ϵ-Nash equilibrium θ, no agent c_i plays an incorrect strategy with probability $\geq 2\epsilon$.*

PROOF OF CLAIM. The proof follows by induction. We defer the base case. Fix a gate $g \in G$ with input gates g_1, g_2. Now fix a computing agent c_i. By induction, assume that g_1 and g_2 are played incorrectly by each computing agent with probability less than 2ϵ. (By Claim 7.5 this requires that each computing agent plays correctly with probability greater than $1/N - 3\epsilon$).

We now show that the claim is true by showing that the expected payoff of c_i for playing the correct strategy of gate g is always more than $\frac{1}{2}$ better than his payoff for playing the incorrect strategy for gate g. Therefore, if agent c_i is playing an incorrect strategy for gate g with probability $\geq 2\epsilon$, agent c_i could do more than ϵ better by playing the correct strategy for gate g whenever he had previously played the incorrect strategy. Note that strategies played for g, g_1, and g_2 which are used in computing payoffs are always independent because t takes its three inputs from three different agents.

We first compute $v_{c_i}(R_{c_i}(\theta, g^*))$. First note that $t(g^*, s_{c_j}, s_{c_k}) = -N^2$ only if $s_{c_j} = \bar{g}_1^*$ or $s_{c_k} = \bar{g}_2^*$, and each of these events happens with probability less then 2ϵ by induction. So

$$\begin{aligned} v_{c_i}(R_{c_i}(\theta, g^*)) &= t(g^*, \theta_{c_j}, \theta_{c_k}) - v_{e_i}(R_{c_i}(\theta, g^*)) \\ &\geq -N^2 \cdot \Pr[\theta_{c_j} = \bar{g}_1^*] - N^2 \cdot \Pr[\theta_{c_k} = \bar{g}_2^*] - v_{e_i}(R_{c_i}(\theta, g^*)) \\ &> -4N^2\epsilon - v_{e_i}(R_{c_i}(\theta, g^*)) \end{aligned}$$

We next compute $v_{c_i}(R_{c_i}(\theta, \bar{g}^*))$. First note that $t(\bar{g}^*, s_{c_j}, s_{c_k}) = -N^2$ whenever $s_{c_j} = g_1^*$ and $s_{c_k} = g_2^*$ and each of these events is independent and happens with probability greater than $1/N - 3\epsilon$, by induction. So

$$\begin{aligned} v_{c_i}(R_{c_i}(\theta, \bar{g}^*)) &= t(g^*, \theta_{c_j}, \theta_{c_k}) - v_{e_i}(R_{c_i}(\theta, \bar{g}^*)) \\ &\leq -N^2 \cdot \Pr[\theta_{c_j} = g_1^* \text{ and } \theta_{c_k} = g_2^*] - v_{e_i}(R_{c_i}(\theta, \bar{g}^*)) \\ &\leq -N^2(1/N - 3\epsilon)^2 - v_{e_i}(R_{c_i}(\theta, g^*)) \end{aligned}$$

Finally we show that $v_{c_i}(R_{c_i}(\theta, g^*)) - v_{c_i}(R_{c_i}(\theta, \bar{g}^*)) > 1/2$.

$$\begin{aligned} v_{c_i}(R_{c_i}(\theta, g^*)) - v_{c_i}(R_{c_i}(\theta, \bar{g}^*)) &> -4N^2\epsilon - v_{e_i}(R_{c_i}(\theta, g^*)) \\ &\quad -[-N^2(1/N - 3\epsilon)^2 - v_{e_i}(R_{c_i}(\theta, \bar{g}^*))] \\ (\text{because } v_{e_i}(R_{c_i}(\theta, g^*)) = v_{e_i}(R_{c_i}(\theta, \bar{g}^*))) &\geq -4N^2\epsilon + 1 - 6\epsilon N + 9\epsilon^2 N^2 \\ (\text{because } \epsilon = \frac{1}{100 \cdot N^2}) &\geq 1/2. \end{aligned}$$

It remains to show the case where g_1, g_2, or both are actually constants in the circuit. The preceding analysis remains the same. Before we claimed, by induction, that the probability that θ_{c_j} equals the correct solution (g_i or $\bar{g}_i$) is $> 1 - 3\epsilon$ and the probability that θ_{c_j} equals the incorrect solution (g_i or $\bar{g}_i$) is $< 2\epsilon$. If g_1 is a constant input, the way that we have defined t, θ_{c_j} will always equal the correct input, and will never equal the incorrect, so the aforementioned assumptions remain valid.

Now 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. By Claim 7.6, if C evaluates to true, in any ϵ-Nash equilibrium, c_1 will play $\bar{o}$ with probability less than 2ϵ and thus by Claim 7.5 will play o with probability at least $1/N - 3\epsilon$. If C evaluates to false, by Claim 7.6, in any ϵ-Nash equilibrium, c_1 will play o with probability less than 2ϵ.

Now we show that circuit game EXP-APPROX EXACT FINDNASHSIMPLE is in **NEXP** $\cap$ **coNEXP**. Given input $(\mathcal{G}, i, s_i, k, \epsilon)$, the nondeterminism guesses a strategy profile θ that is at most exponentially long, and then checks whether θ is an ϵ Nash equilibrium in **EXP**. The machine will output "positive" if θ is an ϵ Nash Equilibrium of $\mathcal{G}$ and $\Pr[\theta_i = s_i] \geq k$. The machine will output "negative" if θ is an ϵ Nash Equilibrium of $\mathcal{G}$ and $\Pr[\theta_i = s_i] < k$. The machine will fail if θ is not an ϵ Nash Equilibrium of $\mathcal{G}$.

If the input is a positive instance of FINDNASHSIMPLE, then the machine will always output "positive" or fail. Similarly, if the input is a negative instance of FINDNASHSIMPLE, then the machine will always output "negative" or fail. By Theorem 3.4 there always exists an ϵ-equilibrium that uses at most exponential space. If the nondeterminism guesses this strategy profile, then the machine is guaranteed to output either "positive" or "negative."

Circuit game EXP-APPROX EXACT FINDNASH is in **TFNEXP** because given input $(\mathcal{G}, \epsilon)$ the nondeterminism guesses a strategy profile θ that is at most exponentially long, and then checks whether it is a Nash equilibrium in **EXP**. By Theorem 3.4, there always exists an ϵ-equilibrium that uses at most exponential space.

Because the description of the strategy-profile that was guessed may be exponential in length, we cannot simply use our result from ISNASH (Proposition 5.2) to show that we can determine if θ is a Nash equilibrium. However, it is not hard to see that we can verify this in a straightforward manner by computing, for each agent i, $v_i(\theta)$ and $v_i(R_i(\theta, s_i))$ for all $s_i \in \mathfrak{s}_i$. □

In subsequent work, a technique similar to this proof was used in Goldberg and Papadimitriou [2006] 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 previous 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.

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

Definition 7.7. A promise language L is in $\mathbf{S_2P}$ if there exists 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 $\forall\, y$, $R(x, y, z) = 0$.

THEOREM 7.8. *Circuit game and 2-player circuit game* Poly-Approx *and* Const-Approx FindNash *are* $\mathbf{S_2P}$*-hard but are in* $\mathbf{TF\Sigma_2P}$.

We first prove a technical lemma that will later free us from messy computations.

LEMMA 7.9. *Let $\mathcal{G} = (\mathfrak{s}, \nu)$ be a two-player, zero-sum game, and create a new game $\mathcal{G}' = (\mathfrak{s}', \nu')$ where $\mathfrak{s}'_1 = \mathfrak{s}_1 \cup \{\emptyset\}$, $\mathfrak{s}'_2 = \mathfrak{s}_2$, and*

$$\begin{aligned} \nu'_i(s'_1, s'_2) &= \nu_i(s_1, s_2) && \text{if } s'_1 \in \mathfrak{s}_1 \\ \nu'_i(s'_1, s'_2) &= \alpha && \text{if } s'_1 = \emptyset. \end{aligned}$$

If $\nu(\mathcal{G}) < \alpha - 4\epsilon$, then in any ϵ-Nash equilibrium θ' of $\mathcal{G}'$, $\Pr[\theta'_i \in \mathfrak{s}_i] < 1/2$. Also, if $\alpha + 4\epsilon < \nu(\mathcal{G})$, then in any ϵ-Nash equilibrium θ' of $\mathcal{G}'$, $\Pr[\theta'_i \in \mathfrak{s}_i] > 1/2$.

PROOF. For the sake of contradiction, suppose that $\nu(\mathcal{G}) < \alpha - 4\epsilon$ and θ' is an ϵ-Nash equilibrium of $\mathcal{G}$ such that $p = \Pr[\theta'_i \in \mathfrak{s}_i] \geq \frac{1}{2}$. Let θ_1 denote the probability distribution over $\mathfrak{s}'_1$ of θ'_1 given that $\theta'_1 \in \mathfrak{s}_1$. θ_1 is well defined because $p > 0$.

Player 2's payoff is

$$p(\nu_2(R_1(\theta', \theta_1))) + (1 - p)(\alpha).$$

However, player 2 can attain a payoff of

$$p(-\nu(\mathcal{G})) + (1 - p)(\alpha).$$

by playing an optimal strategy in $\mathcal{G}$. Because θ' is an ϵ-Nash equilibrium, the difference of these two values is at most ϵ.

$$\begin{aligned} p(-\nu(\mathcal{G})) + (1 - p)(\alpha) & \\ - \left[p(\nu_2(R_1(\theta', \theta_1))) + (1 - p)(\alpha)\right] &\leq \epsilon \\ \Rightarrow -\nu_2(R_1(\theta', \theta_1)) &\leq \epsilon/p + \nu(\mathcal{G}) \\ \Rightarrow \nu_1(R_1(\theta', \theta_1)) &\leq \epsilon/p + \nu(\mathcal{G}). \end{aligned}$$

In the last step $-\nu_2(R_1(\theta', \theta_1)) = \nu_1(R_1(\theta', \theta_1))$ because $\mathcal{G}$ is a zero-sum game.

Because player 1 can always receive α by playing $\emptyset$, he receives at least $\alpha - \epsilon$ in any ϵ-Nash equilibrium. This implies that

$$\begin{aligned} \alpha - \epsilon &\leq \nu_1(\theta') \\ &= p(\nu_1(R_1(\theta', \theta_1))) + (1 - p)(\alpha) \\ &\leq p\left[\epsilon/p + \nu(\mathcal{G})\right] + (1 - p)(\alpha) && \text{because } \nu_1(R_1(\theta', \theta_1)) \leq \epsilon/p + \nu(\mathcal{G}) \\ &< p\left[\epsilon/p + (\alpha - 4\epsilon)\right] + (1 - p)(\alpha) && \text{because } \nu(\mathcal{G}) < \alpha - 4\epsilon \\ &\leq \alpha - \epsilon. \end{aligned}$$

So we have found our contradiction. The other implication follows in a very similar manner. □

PROOF OF THEOREM 7.8. We first show that circuit game POLY-APPROX FINDNASH can be computed in $\mathbf{TF}\Sigma_2\mathbf{P}$.

Our machine will first guess an $\epsilon/2$ Nash equilibrium and then verify the correctness of its guess by checking if the guessed strategy profile is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium.

By Theorem 3.4 in every n-player game $\mathcal{G}$, there exists a k-uniform $\epsilon/2$-Nash equilibrium, where $k = 3n^2 \ln(n^2 \max_i\{|\mathfrak{s}_i|\})/(\epsilon/2)^2$. This is polynomially bounded with respect to the encoding of $\mathcal{G}$ as a circuit game and $|\epsilon|$ where ϵ is represented as in POLY-APPROX.

By Theorem 5.12 and Remark 5.15 we can check with a co-nondeterministic polynomial-time machine with oracle access to **BPP** (using the alternate definition 2 of ISNASH in Definition 3.5) if the guessed equilibrium is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium. However, $\mathbf{coNP}^{\mathbf{BPP}} = \mathbf{coMA} \subseteq \Sigma_2\mathbf{P}$ by Babai and Moran [1988]. Thus an $\exists\, \mathbf{coNP}^{\mathbf{BPP}}$-predicate can be replaced by an $\exists\, \Sigma_2\mathbf{P}$-predicate = $\Sigma_2\mathbf{P}$-predicate.

We now show that 2-player circuit game CONST-APPROX FINDNASH is $\mathbf{S}_2\mathbf{P}$-hard. We first follow the proof of Fortnow et al. [2008], which shows that approximating GAMEVALUE in 2-player circuit games is $\mathbf{S}_2\mathbf{P}$-hard in order to make a game with value either 1 or -1. Then we employ Lemma 7.9.

Recall that if a language is in $\mathbf{S}_2\mathbf{P}$ then there exists a polynomially balanced and polynomially decidable predicate φ such that $x \in L^+ \Rightarrow \exists y, \forall z\ \varphi(x, y, z) = 1$ and $x \in L^- \Rightarrow \exists z, \forall y\ \varphi(x, y, z) = 0$. Let $p(|x|)$ be a polynomial that bounds the lengths of y and z.

Let L be a promise language in $\mathbf{S}_2\mathbf{P}$, now construct a game G' so that, given an ϵ-Nash equilibrium to $\mathcal{G}'$, we can determine if a given x is in L^+ or L^-. Given an x, construct an instance of FINDNASH $(\mathcal{G}', \epsilon)$ as follows.

First, let $\mathcal{G}$ be the 2-player circuit game $\mathcal{G} = (\mathfrak{s}, \nu)$ where $\mathfrak{s}_i = \{$ strings of length $\leq p(|x|)\}$ and

$$\nu_1(s_1, s_2) = -\nu_2(s_1, s_2) = \varphi(x, s_1, s_2).$$

Let $\epsilon < \frac{1}{4}$.

If $x \in L^+$, the first player has a strategy s_1 such that whatever strategy $s_2 \in \mathfrak{s}_2$ player 2 plays, $\varphi(x, s_1, s_2)$ evaluates to true. So player 1 has a strategy that guarantees him a payoff of 1. On the other hand, if $x \in L^-$, the second player has a strategy that guarantees him a payoff of 1.

We create a new game $\mathcal{G}'$ as in Lemma 7.9. $\mathcal{G}' = (\mathfrak{s}', \nu')$, where $\mathfrak{s}'_1 = \mathfrak{s}_1 \cup \{\emptyset\}$, $\mathfrak{s}'_2 = \mathfrak{s}_2$, and

— $\nu'_i(s_1, s_2) = \nu_i(s_1, s_2)$ if $s_1 \in \mathfrak{s}_i$
— $\nu'_1(\emptyset, s_2) = \nu_2(\emptyset, s_2) = 0$.

Then if $x \in L^+$, $\nu(\mathcal{G}) = 1$ and so because $0 + 4\epsilon < \nu(\mathcal{G})$ by Lemma 7.9 in any ϵ-Nash equilibrium θ' of $\mathcal{G}'$, $\Pr[\theta' = \emptyset] < 1/2$. However, if $x \in L^-$, $\nu(\mathcal{G}) = -1$ and so because $\nu(\mathcal{G}) < 0 - 4\epsilon$ by Lemma 7.9 in any ϵ-Nash equilibrium θ' of $\mathcal{G}'$, $\Pr[\theta' = \emptyset] \geq 1/2$. □

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 [Feigenbaum et al. 1995] to compute exactly and $\mathbf{S}_2\mathbf{P}$-complete to approximate. The next two hardness results use a different general approach. The hardness of these problems is derived from the hardness of ISNASH.

We could have obtained the result that Circuit Game FINDNASH is **coMA**-hard by using a proof similar to that of Theorem 7.10, based on the hardness of ISNASH. However it is known that $\mathbf{coMA} \subseteq \mathbf{S}_2\mathbf{P}$, so the preceding is a stronger result.

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 7.10. *Boolean circuit game* EXP-APPROX FINDNASH *is* $\mathbf{P}^{\#\mathbf{P}}$*-hard under Cook reductions but is in* $\mathbf{FNP}^{\#\mathbf{P}}$.

PROOF. We first show that Boolean circuit game EXP-APPROX FINDNASH can be computed by a nondeterministic polynomial time machine with oracle access to #**P**. Such a machine will first guess an $\epsilon/2$ Nash equilibrium and then verify the correctness of its guess by checking if the guessed strategy profile is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium.

By Theorem 3.4 in every n-player game $\mathcal{G}$, there exists an $\epsilon/2$-Nash equilibrium that can be encoded in polynomial space.

By Theorem 5.3 and Remark 5.4 we can check in polynomial time with oracle access to #**P** (using the alternate definition 2 of ISNASH in Definition 3.5) if the guessed equilibrium is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium.

The proof of the hardness result is very similar to that of Theorem 5.3. Again, 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 , we construct an $n+1$-player Boolean circuit game. The payoffs to agent 1 are as follows:

— $\frac{1}{2} - \left(\frac{1}{2}\right)^{n+1}$ for playing 0;
— the output of the circuit $C(s_2, \ldots, s_{n+1})$, where s_i is the strategy of agent i, for playing 1.

The payoff of the other agents is 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 $\epsilon = \frac{1}{2n} \cdot \left(\frac{1}{2}\right)^{n+2}$.

Now we claim it is possible to determine whether a majority of the inputs satisfy C by checking player 1's strategy in any ϵ-Nash equilibrium. If C belongs to MAJORITYSAT, then $\Pr[\theta_1 = 0] < 1/2$; if C does not belong to MAJORITYSAT then $\Pr[\theta_1 = 0] \geq 1/2$.

Say that a majority of the inputs are accepted and let θ be an ϵ-Nash equilibrium for $\mathcal{G}$. By Theorem A.1, in pennies to obtain an ϵ-Nash equilibria, it is necessary that each player plays each strategy with probability $\in [1/2 - 2\epsilon, 1/2 + 2\epsilon]$. That is, for each $i = 2, \ldots, n+1$, the random variable θ_i has statistical distance at most 2ϵ from a uniform random bit. This implies that the joint distribution $(\theta_2, \ldots, \theta_{n+1})$ has statistical distance at most $2\epsilon \cdot n$ from U_n. Thus, $\left|\mathbb{E}[C(\theta_2, \ldots, \theta_{n+1})] - \mathbb{E}[C(U_n)]\right| \leq 2\epsilon n = (1/2)^{n+2}$.

So the payoff to agent 1 for playing 0 is $\frac{1}{2} - \left(\frac{1}{2}\right)^{n+1}$ and for playing 1 is $\mathbb{E}[C(s_2, \ldots, s_{n+1})] \geq 1/2 - \left(\frac{1}{2}\right)^{n+2}$. So by playing $s_1 = 1$, agent 1 expects to do better by at least $1/2 - \left(\frac{1}{2}\right)^{n+2} - [1/2 - \left(\frac{1}{2}\right)^{n+1}] = \left(\frac{1}{2}\right)^{n+2} > 2\epsilon$. So the following claim shows that $\Pr[\theta_1 = 0] < 1/2$.

CLAIM 7.11. *Let* θ *be an* ϵ*-Nash equilibrium. If there exists a strategy* $s_i \in \mathfrak{s}_i$ *such that* $v_i(R_i(\theta, s_i)) \geq v_i(R_i(\theta, s_i')) + 2\epsilon$ *for all* $s_i' \in \mathfrak{s}_i$, $s_i' \neq s_i$, *then* $\Pr[\theta_i = s_i] \geq 1/2$.

PROOF OF CLAIM. For the sake of contradiction, assume that θ is an ϵ-Nash equilibrium, where $\Pr[\theta_i = s_i] < 1/2$. Let $v = \max_{s_i' \in \mathfrak{s}_i, s_i' \neq s_i} v_i(R_i(\theta, s_i'))$. $v_i(\theta) < \frac{1}{2} v_i(R_i(\theta, s_i)) + \frac{1}{2} v \leq \frac{1}{2} v_i(R_i(\theta, s_i)) + \frac{1}{2}(v_i(R_i(\theta, s_i)) - 2\epsilon) = v_i(R_i(\theta, s_i)) - \epsilon$. So by changing his strategy to s_i, agent i could do ϵ better. Therefore θ is not an actual ϵ-Nash equilibrium.

Now say that C is not a member of MAJORITYSAT and θ is a Nash equilibrium for $\mathcal{G}$. We will show that $\Pr[\theta_1 = 0] \geq 1/2$. By the same reasoning, in any ϵ-Nash equilibrium $|\mathbb{E}[C(s_2, \ldots, s_{n+1})] - \mathbb{E}[C(U_n)]| \leq \left(\frac{1}{2}\right)^{n+2}$.

So the payoff to agent 1 for playing 0 is $\frac{1}{2} - \left(\frac{1}{2}\right)^{n+1}$ and for playing 1 is $\mathbb{E}[C(s_2, \ldots, s_{n+1})] \leq 1/2 - \left(\frac{1}{2}\right)^{n} + \left(\frac{1}{2}\right)^{n+2}$. So by playing $s_1 = 0$, agent 1 expects to do better by at least $1/2 - \left(\frac{1}{2}\right)^{n+1} - [1/2 - \left(\frac{1}{2}\right)^{n} + \left(\frac{1}{2}\right)^{n+2}] = \left(\frac{1}{2}\right)^{n+2} > 2\epsilon$. So by Claim 7.11, $\Pr[\theta_1 = 0] \geq 1/2$. □

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 7.12. *Boolean circuit game* POLY-APPROX *and* CONST-APPROX FINDNASH *are* **BPP**-*hard, but can be computed in* $\mathbf{TFNP^{BPP}} = \mathbf{TFMA}$.

PROOF. We show that Boolean circuit game POLY-APPROX FINDNASH can be computed in nondeterministic polynomial time with an oracle to **BPP**.

Such a machine will first guess an $\epsilon/2$ Nash equilibrium and then verify the correctness of its guess by checking if the guessed strategy profile is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium.

By Theorem 3.4 in every Boolean circuit game $\mathcal{G}$, there exists an $\epsilon/2$-Nash equilibrium that can be encoded in polynomial space.

By Theorem 5.9 and Remark 5.11 we can check in **BPP** (using the alternate definition 2 of ISNASH in Definition 3.5) whether the guessed equilibrium is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium.

We now show that Boolean circuit game CONST-APPROX ISNASH is **BPP**-hard. Given a **BPP** language L and an instance x, we create a game so that we can tell whether $x \in L$ by looking at the first agent's strategy in any $\frac{1}{100}$-Nash equilibrium.

We create a reduction as follows. Given a language L in **BPP** there exists an algorithm $A(x, r)$ that decides if $x \in L$ using coin tosses r with two-sided error of at most $\frac{1}{100}$. Let $n = |r|$ and let $k = \lceil \log_{25} 100n \rceil$.

Now create $\mathcal{G}$ with $n \cdot k + 1$ agents. Each player has a strategy space of $\{0, 1\}$. Let $w = w_1 w_2 \ldots w_n$, where $w_i = XOR(s_{(i-1)k+2}, s_{(i-1)k+3}, \ldots, s_{i \cdot k+1})$. The first player gets paid

— $1/2$ if he plays 0;
— the output of $A(x, w)$. if he plays 1.

All the other players play pennies against each other. So agent i plays pennies with agent $i + 1$ where i is even. Let $\epsilon = 1/100$.

We claim that if $x \in L$, then $\Pr[\theta_1 = 0] < 1/2$ in any ϵ-Nash equilibrium, and that if $x \notin L$, then $\Pr[\theta_1 = 0] \geq 1/2$ in any ϵ-Nash equilibrium.

Say that $x \in L$ and that θ is an ϵ-Nash equilibrium for $\mathcal{G}$. By Theorem A.1, in order to be in an ϵ-equilibrium, all players but the first, must randomize between their two strategies, playing 0 with probability $\in [1/2 - 2\epsilon, 1/2 + 2\epsilon]$. The bits from the strategies of agents 2 through $n \cdot k + 1$ are fully independent, and so by the next claim, if we XOR k of them together, the resulting bit is within $(4\epsilon)^k = 1/(100n)$ of being uniform.

CLAIM 7.13. *Let* $X_1, \ldots, X_n$ *be independent random variables where,* $X_i \in \{0, 1\}$ *and* $\Pr[X_i = 0] \in [1/2 - \epsilon, 1/2 + \epsilon]$. *Let* $X = XOR(X_1, \ldots, X_n)$, *then* $\Pr[X = 0] \in [1/2 - (2\epsilon)^n, 1/2 + (2\epsilon)^n]$.

PROOF OF CLAIM. First create variables $Y_i = 2X_i - 1$ (so that $Y_i \in \{-1, 1\}$ and if $X_i = 0$ then $Y_i = -1$ and if $X_i = 1$ then $Y_i = 1$). $\mathbb{E}[Y_i] = 2\mathbb{E}[X_i] - 1$ and so $-2\epsilon \leq \mathbb{E}[Y_i] \leq 2\epsilon$. Let $Y = \prod_{i=1}^{n} Y_i$. It is straightforward to check that $Y = 2X - 1$. So $\frac{(\mathbb{E}[Y]+1)}{2} = \mathbb{E}[X]$.

But

$$\left|\mathbb{E}[Y]\right| = \prod_{i=1}^{n} \left|\mathbb{E}[Y_i]\right| \leq \prod_{i=1}^{n} |2\epsilon| = (2\epsilon)^n .$$

So $\Pr[X = 1] = \mathbb{E}[X] \in [1/2 - \frac{(2\epsilon)^n}{2}, 1/2 + \frac{(2\epsilon)^n}{2}]$.

Because each input, w_i, to the circuit is within $1/(100n)$ of uniform, their joint distribution is within $1/100$ of uniform. So

$$\left|\mathbb{E}[A(x, w)] - \mathbb{E}[A(x, U_n)]\right| \leq \frac{1}{100},$$

where U_n is the uniform distribution over strings of length n. So if player 1 plays 0, his payoff is $1/2$. But if player 1 plays 1, his payoff is

$$\mathbb{E}[A(x, w)] \geq \mathbb{E}[A(x, U_n)] - \frac{1}{100}, \geq \frac{98}{100}.$$

Therefore, because agent 1 expects to do better by $\frac{98}{100} - \frac{1}{2} \geq 2\epsilon$ by playing 1, by Claim 7.11, $\Pr[\theta_1 = 0] < 1/2$.

Say $x \notin L$ and θ is a ϵ-Nash equilibrium of $\mathcal{G}$. Then by the same reasoning

$$\left|\mathbb{E}[A(x, w)] - \mathbb{E}[A(x, U_n)]\right| \leq \frac{1}{100}.$$

So the payoff to agent 1 for playing 0 is $\frac{1}{2}$, but the payoff to player 1 for playing 1 is

$$\mathbb{E}[A(x, w)] \leq \mathbb{E}[A(x, U_n)] + \frac{1}{100} \leq \frac{2}{100}.$$

Therefore, because agent 1 expects to do better by $\frac{1}{2} - \frac{2}{100} > 2\epsilon$ by playing 0, by Claim 7.11, $\Pr[\theta_1 = 0] \geq 1/2$. □

Finally, we show the complexity for graph games.

THEOREM 7.14. *With any type of approximation, graph game and Boolean graph game* FINDNASH *is in* **FNP**, *but neither is* **NP**-*hard under Cook reductions unless* **NP** = **coNP**. *Furthermore, graph game and Boolean graph game* FINDNASH *are* **P**-*hard, even when restricted to Boolean graphs of degree* ≥ 3.

PROOF. We show that graph game EXP-APPROX FINDNASH can be computed in nondeterministic polynomial time. Such a machine will first guess an $\epsilon/2$ Nash equilibrium and then verify the correctness of its guess by checking if the guessed strategy profile is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium .

By Theorem 3.4, in every graph game $\mathcal{G}$, there exists an $\epsilon/2$-Nash equilibrium that can be encoded in polynomial space.

By Proposition 5.1 we can check in polynomial time (using the alternate definition 2 of ISNASH in Definition 3.5) if the guessed equilibrium is an $\epsilon/2$-Nash equilibrium or is not even an ϵ-Nash equilibrium.

To show the hardness result, we reduce from CIRCUITVALUE. Given a circuit C, we construct a game $\mathcal{G}$ with an agent for each gate in C. Each agent has possible strategies $\{0, 1\}$ and is paid 1 for correctly computing the output of his gate (with respect to

the strategies of the agents that correspond to the inputs to his gate), and is paid 0 otherwise. Let $\epsilon = 1/100$.

We call the strategy of the agent associated with gate g *correct* if it corresponds with the output of the gate in an evaluation of C. The unique Nash equilibrium of G is where each player plays the correct strategy.

CLAIM 7.15. *In any ϵ-Nash equilibrium, each player must play the correct strategy with probability $\geq 1 - 2\epsilon$.*

PROOF OF CLAIM. We proceed by induction, but we defer the base case. Assume that the two agents associated with the inputs, gates to a particular gate g, play the correct pure strategy with probability $\geq 1-2\epsilon$. Let v be the payoff the agent associated with gate g when he plays his correct strategy. We know that $v \geq (1-2\epsilon)^2$ because if both of the input agents play their correct strategies, then the agent associated with g will receive a payoff of 1 when he plays his correct strategy. If g plays the opposite strategy his payoff will be $1 - v$. Now say that g plays the opposite strategy with probability p. Because he is in an ϵ-equilibrium, we know that $(1-p)v + p(1-v) + \epsilon \geq v$ because he can get paid v if he just plays the correct strategy all the time. By simple arithmetic, this implies that

$$p \leq \frac{\epsilon}{2v-1} \leq \frac{\epsilon}{2(1-2\epsilon)^2 - 1} \text{ (by what we know of } v) \leq 2\epsilon \left(\text{by inspection when } \epsilon \leq \frac{1}{100} \right).$$

The base case consists of those agents connected directly to the constant gates. However, if we view the constant gates as agents who always tell the truth, then the previous argument applies.

Therefore, in any ϵ-Nash equilibrium, each player must play the strategy corresponding with the correct valuation of the circuit with probability $\geq 1 - 2\epsilon$.

So by looking at the strategy in an ϵ-Nash equilibrium of the agent at the output gate, we can correctly deduce the value of the circuit.

If graph game FINDNASH were **NP**-hard under cook reductions, it would also be **coNP**-hard under cook reductions. However, this would imply **coNP** $\subseteq$ **NP**, because in **NP** we could simulate the polynomial-time algorithm with oracle access to FINDNASH, guessing and verifying FINDNASH oracle query results. Indeed, this is true for any problem in **TFNP** [Megiddo and Papadimitriou 1991]. □

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 exist a Nash equilibrium where each player is paid at least x amount? Does there exist a Nash equilibrium with social welfare x? Does there exist 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 [Conitzer and Sandholm 2008; Gilboa and Zemel 1989].[9] 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.

[9]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.

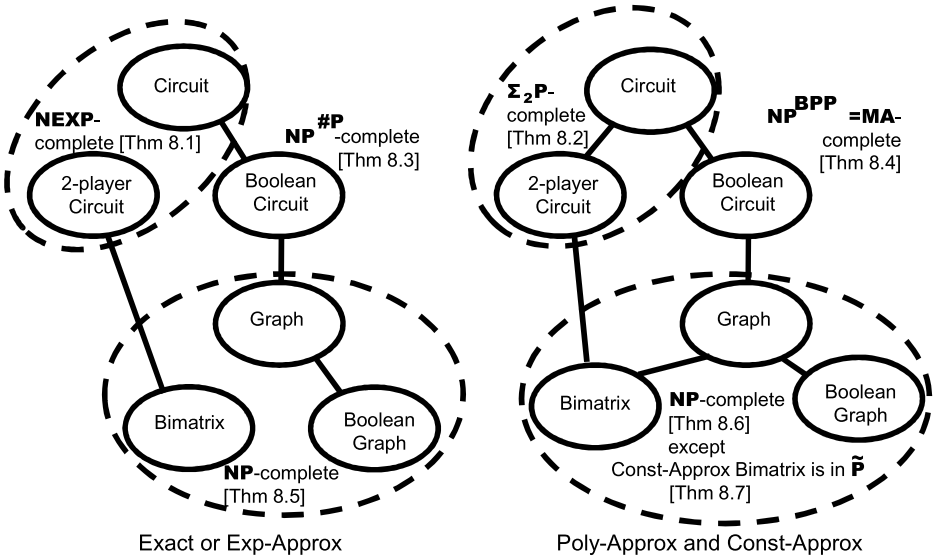

Fig. 6. Summary of GUARANTEENASH results.

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

PROOF. We first show that 2-player circuit game EXACT GUARANTEENASH is in **NEXP**. Given instance $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$, guess a strategy profile θ, of at most exponential length, and then check whether θ is a Nash equilibrium that meets the guarantees.

The correctness of this algorithm follows from Proposition 3.3, which tells us that if a Nash equilibrium that meets the guarantees exists, then one exists that is at most exponentially large.

To check that θ is a Nash equilibrium, we need only check that $v_i(\theta) \geq v_i(R_i(\theta_i', s_i))$ for all agents i and for all $s_i \in \mathfrak{s}_i$. Because there are only 2 agents, and only an exponential number of strategies, there are only exponentially many of these inequalities. To check that θ meets the guarantees, we need only check that $v_i(\theta) \geq g_i$ for both agents. Therefore, it is enough to show that we can compute v_i in **EXP**. But

$$v_i(\theta) = \sum_{s_1 \in \mathfrak{S}_1} \sum_{s_2 \in \mathfrak{S}_2} \left[\Pr[\theta_1' = s_1] \cdot \Pr[\theta_2' = s_2] \cdot v_i(s_1, s_2)\right].$$

All values that are multiplied or summed have at most exponential bit size, thus $v(\theta)$ can be computed in **EXP**.

We next show that circuit game EXP-APPROX GUARANTEENASH is in **NEXP**. Given instance $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ we guess a $\frac{n^2 \log(n^2 \max_i |s_i|)}{\epsilon^2}$-uniform strategy profile θ. We then check whether θ is an ϵ-Nash equilibrium that is within ϵ of meeting the guarantees. If it is, we accept, otherwise we reject.

The correctness of this algorithm follows from Theorem 3.4, which states that if there exists a Nash equilibrium that meets the guarantees, then there will exist a $\frac{n^2 \log(n^2 \max_i |s_i|)}{\epsilon^2}$-uniform ϵ-Nash equilibrium that gets within $\epsilon/2$ of the guarantees.

To check that θ is an ϵ-Nash equilibrium, we need only check that $v_i(\theta) \geq v_i(R_i(\theta_i', s_i)) + \epsilon$ for all agents i and for all $s_i \in \mathfrak{s}_i$. Because there are only a polynomial

number of agents, and only an exponential number of strategies, there are only exponentially many of these inequalities. To check that θ meets the guarantees, we need only check that $v_i(\theta) \geq g_i - \epsilon$ for at most polynomially many agents. Therefore, it is enough to show that we can compute v_i in **EXP**. But

$$v_i(\theta) = \sum_{s_1 \in \mathfrak{S}_i} \cdots \sum_{s_n \in \mathfrak{S}_n} \left[\prod_{i=1}^{n} \left(\Pr[\theta_i' = s_1] \right) v_i(s_1, \ldots, s_n) \right].$$

All values that are multiplied or summed have polynomial bit size (because it is a k-uniform strategy profile), so the product of n of them is still polynomial. And the sum of exponentially many, is exponential. Thus $v(\theta)$ can be computed in **EXP**.

We now show that 2-player circuit game GUARANTEENASH with exponentially small error is **NEXP**-hard. We use a reduction and analysis very similar to the one of Conitzer and Sandholm [2008] except that instead of reducing from 3SAT, we reduce from the **NEXP**-complete problem SUCCINCT 3SAT, and we keep track of approximation errors in the analysis. (Conitzer and Sandholm consider a different notion of inapproximation. They show it is **NP**-complete to approximate the optimum social welfare obtained by any EXACT Nash Equilibrium to within a constant factor. This result is incomparable with ours.)

Given a succinct representation of a Boolean formula φ in conjunctive normal form with the set of variables V and the set of clauses C, let $N = |V|$ be the number of variables, and let the set $L = \{x, \bar{x} : x \in V\}$ be the set of literals. We treat L, the set of literals, as formally distinct from V, the set of variables,[10] and define a function $v : L \to V$ such that $v(x) = v(\bar{x}) = x$. We construct the 2-player circuit game $G = (\mathfrak{S}, v)$, where $\mathfrak{S}_1 = \mathfrak{S}_2 = V \cup C \cup L$, so that if φ is satisfiable and $l_1, \ldots, l_N$ are literals that satisfy the formula (exactly one for each variable), then the strategy where each player randomizes uniformly between those N literals is a Nash equilibrium where the expected payoff to each player is $N - 1$; however, if φ is not satisfiable, then no ϵ-Nash equilibrium with payoffs to each player of at least $N - 1 - \epsilon$ exists.

Define $v_1(s_1, s_2) = v_2(s_2, s_1)$ as follows.

(1) $v_1(l_1, l_2) = N - 1$, where $l_1 \neq \bar{l}_2$ for all $l_1, l_2 \in L$. This will ensure each player gets a high payoff for playing the aforementioned strategy.
(2) $v_1(l, \bar{l}) = N - 4$ for all $l \in L$. This will ensure that each player does not play a literal and its negation.
(3) $v_1(v, l) = 0$, where $v(l) = v$, for all $v \in V$, $l \in L$. This, along with rule 4, ensures that for each variable v, each agent plays either l or $\bar{l}$ with probability at least $1/N$, where $v(l) = v(\bar{l}) = v$.
(4) $v_1(v, l) = N$, where $v(l) \neq v$, for all $v \in V$, $l \in L$.
(5) $v_1(l, x) = N - 4$, where $l \in L$, $x \in V \cup C$. This, along with rules 6 and 7, ensures that if both players do not play literals, then the payoffs cannot meet the guarantees.
(6) $v_1(v, x) = N - 4$ for all $v \in V$, $x \in V \cup C$.
(7) $v_1(c, x) = N - 4$ for all $c \in C$, $x \in V \cup C$.
(8) $v_1(c, l) = 0$ where $l \in c$ for all $c \in C$, $l \in L$. This, along with rule 9, ensures that for each clause c, each agent plays a literal in the clause c with probability least $1/N$.
(9) $v_1(c, l) = N$, where $l \notin c$ for all $c \in C$, $l \in L$.

Let $\epsilon = 1/2N^3$ and let the guarantee to each player be $N - 1$.

First we show that if φ is satisfiable, then there exists a Nash equilibrium with the guarantees. Say that $l_1, \ldots, l_N$ are literals that satisfy the formula (exactly one for each

[10] So that $|V \cup L| = 3N$.

variable). Then the strategy where each player randomizes uniformly between those N literals is a Nash equilibrium where the expected payoff to each player is $N-1$. The expected payoff to each player is $N-1$ because they will always be playing l_1 and l_2, where $l_1 \neq \bar{l}_2$ and $l_1, l_2 \in L$. Second, there are only two rules that pay out more than $N-1$: $\nu_1(c,l) = N$, where $l \notin C$ for all $c \in C$, $l \in L$ and $\nu_1(v,l) = N$ where $v(l) \neq v$, for all $v \in V$, $l \in L$. However, if agent i deviates and plays any clause c, the other player will play a literal in that clause c with probability $1/N$ because he randomizes between literals in a satisfying assignment. So in this case, agent i's payoff is at most $1/N \cdot 0 + (N-1)/N \cdot N = N-1$, and so agent i is no better off than before. Similarly no matter what variable an agent deviates to, his opponent plays a corresponding literal with probability $1/N$.

Now we show that if φ is not satisfiable, in any ϵ-Nash equilibrium at least one player fails to receive an expected payoff of $N-1-\epsilon$. Unless both players are playing a literal, the maximum sum of the outcomes is $2N-4$. We cannot be in this case with probability greater than ϵ, because otherwise the payoffs will sum to less than $2N-2-2\epsilon$. So both players play elements of L with probability $> 1-\epsilon$.

Now assume that the probability that agent i plays l or $\bar{l}$ for some specific l is less than $1/N - \epsilon - \frac{2\epsilon}{N}$ ($\geq 1/N - 2\epsilon$). Then the expected value for the other player, agent j, to play $v(l)$ is at least $(1/N - \epsilon - \frac{2\epsilon}{N}) \cdot 0 + \epsilon \cdot 0 + (1 - \epsilon - (1/N - \epsilon - \frac{2\epsilon}{N}))N = N - 1 + 2\epsilon$ (the first term is when agent i plays l or $\bar{l}$, the second is when agent i does not play a literal, and the third term is when agent i plays a literal $\neq l, \bar{l}$). So either agent j can do ϵ better by changing his strategy or he is already receiving $N-1+\epsilon$ and so the other player does not meet his guarantee (recall the sum of payoffs is at most $2N-2$).

Now we show that for each pair of literals, there is one that is played with probability $\geq 1/N - 2\epsilon - 1/N^2$, while the other is played with probability less than $1/N^2$.

If one player plays l and the other one $-l$, then the sum of payoffs is $2N-8$ and so this must happen also with probability $\leq \epsilon/3$, otherwise at least one player will fail to meet his guarantee. Without loss of generality, assume that player 1 plays l more than $\bar{l}$. For the sake of contradiction, assume player 1 plays l with probability less than $1/N - (1/N^2 + 2\epsilon)$ and so plays $\bar{l}$ with probability more than $1/N^2$. (Recall that each player plays either l or $\bar{l}$ with probability at least $1/N - \epsilon - \frac{2\epsilon}{N} \geq 1/N - 2\epsilon$.) Either the other agent plays l with probability less than $1/N^2$ or plays $\bar{l}$ with probability greater than $1/N - (1/N^2 + 2\epsilon)$. In either case, the two players play both l and $\bar{l}$ with probability $[1/N - (1/N^2 + 2\epsilon)][1/N^2] = 1/N^3 - 1/N^4 - 2/N^6 \geq \frac{1}{2N^3} = \epsilon$. This cannot happen. So player 1 must play l with probability greater than $1/N - (1/N^2 + 2\epsilon)$ and by a symmetric argument so must player 2. By the same argument, each must play $\bar{l}$ with probability less than $1/N^2$.

So in any ϵ-Nash equilibrium that meets the guarantees, we can create a correspondence between literals and truth assignments. We say that a literal is true if it is played more often than its negation. However, if φ is not satisfiable, it means that for the corresponding assignment, there exists at least one clause with no satisfying literal. Now by changing his strategy to that clause, agent i will expect to receive a payoff of N whenever the other player, agent j, plays a literal that is not in that clause. Agent j plays a literal with probability $> 1-\epsilon$, and there only 3 literals in the clause, each of which agent j plays with probability $\leq 1/N^2$. By changing his strategy, agent i will receive at least $(1 - \epsilon - 3/N^2)N > N - 1 + 2\epsilon$. So either agent i can do ϵ better by changing his strategy or he is already receiving $N-1+\epsilon$ and so the other player does not meet his guarantee (recall the sum of payoffs is at most $2N-2$). □

THEOREM 8.2. *Circuit game and 2-player circuit game* POLY-APPROX *and* CONST-APPROX GUARANTEENASH *are* $\Sigma_2\mathbf{P}$*-complete.*

PROOF. We first show that circuit game POLY-APPROX GUARANTEENASH is in $\Sigma_2\mathbf{P}^{\mathbf{BPP}}$. Given an instance $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$, we nondeterministically guess a polynomially small strategy profile θ. Then we test whether θ is an $\epsilon/2$-Nash equilibrium that is within $\epsilon/2$ of meeting the guarantees or whether θ is either not an ϵ-Nash equilibrium or fails to be within ϵ of the guarantees. In the former case, we accept, in the latter case we reject.

We now argue the correctness of the algorithm. If $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a positive instance of GUARANTEENASH, then there exists a Nash equilibrium with the guaranteed properties in $\mathcal{G}$. By Theorem 3.4 there exists an $\epsilon/2$-Nash equilibrium θ that can be represented in polynomial space, where the payoffs of each player are within $\epsilon/2$ of the guarantees. So the algorithm will accept upon guessing θ.

If $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a negative instance of GUARANTEENASH, then there does not exist any ϵ-Nash equilibrium within ϵ of meeting the guaranteed properties. So whatever strategy profile θ the algorithm guesses, either θ will fail to be an ϵ-Nash equilibrium or θ will fail to be within ϵ of the guarantees. Therefore the algorithm will always reject θ.

It is now left to show that in $\mathbf{coNP}^{\mathbf{BPP}}$ we can tell whether θ is an $\epsilon/2$-Nash equilibrium that is within $\epsilon/2$ of meeting the guarantees or whether θ is either not an ϵ-Nash equilibrium or fails to be within ϵ of the guarantees. Note that by Remark 5.15 we can verify whether θ is an $\epsilon/2$-Nash equilibrium or not even an ϵ-Nash equilibrium in $\mathbf{coNP}^{\mathbf{BPP}}$. Also, in $\mathbf{BPP}$ we can test whether $v_i(\theta) \geq g_i - \epsilon/2$ or $v_i(\theta) < g_i - \epsilon$ by Remark 5.10. Therefore we can test whether all these properties hold or at least one fails to hold in $\mathbf{coNP}^{\mathbf{BPP}}$.

Finally, recall from the proof of Theorem 7.8 that $\Sigma_2\mathbf{P} = \Sigma_2\mathbf{P}^{\mathbf{BPP}}$.

We now show that 2-player circuit game CONST-APPROX GUARANTEENASH is $\Sigma_2\mathbf{P}$-hard. We reduce from QCIRCUITSAT_2, which is $\Sigma_2\mathbf{P}$-complete.

$$\text{QCIRCUITSAT}_2 = \{(C, k_1, k_2) : \exists x \in \{0,1\}^{k_1}, \forall y \in \{0,1\}^{k_2}\ C(x, y) = 1\},$$

where C is a circuit that takes $k_1 + k_2$ Boolean variables. Given such an instance (C, k_1, k_2), create 2-player circuit game $\mathcal{G} = (\mathfrak{s}, v)$, 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, $v_1(s_1, s_2) = v_2(s_2, s_1)$.

(1) $x_1 \neq x_2$, $v_1((x_1, y_1), (x_2, y_2)) = 0$
(2) $v_1((x, y_1), (x, y_2)) =$
  — $1 - \gamma$ 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) $v_1(\emptyset, \emptyset) = \gamma$
(4) $v_1((x_1, y_1), \emptyset) = 0$
(5) $v_1(\emptyset, (x_2, y_2)) = 1 - \gamma$

Let $\epsilon = \frac{1}{100}$, $\gamma = \frac{1}{10}$, and $g_i = 1 - \gamma$.

We now show that if $(C, k_1, k_2) \in \text{QCIRCUITSAT}$ then there exists a Nash equilibrium that meets the guarantees and if $(C, k_1, k_2) \notin \text{QCIRCUITSAT}$ then no ϵ-Nash equilibrium in which each player is paid within ϵ of his guarantees exists. Let

$(C, k_1, k_2) \in$QCIRCUITSAT, then there exist some x_0 such that for all y, $C(x_0, y) = 1$. Let θ be the strategy profile where both agents play $(x_0, 0^{k_2})$ with probability 1. Now the payoff to each agent is $1 - \gamma$ and it is easy to see that this is a Nash equilibrium.

Now suppose that $(C, k_1, k_2) \notin$QCIRCUITSAT. We must show that no ϵ-Nash equilibrium gets within ϵ of the guarantees. For the sake of contradiction, assume that such a strategy profile θ exists. We first note that for both players to get within ϵ of their guarantees, the sum of the payoffs to the agents must be greater than $2 - 2\gamma - 2\epsilon \geq 2 - 4\gamma$.

We claim that $\Pr[\theta_1 = (x, y_1) \wedge \theta_2 = (x, y_2)$ such that $C(x, y_1) = C(x, y_2) = 1] > 1 - 4\gamma$. The maximum sum of payoffs for any strategy profile is $2 - 2\gamma$. If both agents do not agree on the x component of the strategy and do not both play pairs (x, y) that satisfy C, then the maximum sum of their payoffs will be 1. If this happens with probability more than 4γ, the sum of the payoffs will be at most $(1 - 4\gamma) \cdot (2 - 2\gamma) + 4\gamma \cdot 1 = 2 - 6\gamma + 4\gamma^2 < 2 - 4\gamma$. So this cannot happen in an ϵ-Nash equilibrium that meets the guarantees.

However, because $(C, k_1, k_2) \notin$QCIRCUITSAT, for any $x \in \{0, 1\}^{k_1}$ there exists some y such that $C(x, y) = 0$. We claim that if agent 1 unilaterally changes his strategy to θ_1' so that every time he had played a strategy (x, y), where $C(x, y) = 1$ in θ_1, he now plays a strategy (x, y'), where $C(x, y') = 0$ in θ_1', then $\nu_1(R_1(\theta, \theta_1')) > \nu_1(\theta) + \epsilon$. Agent 1 will always be paid at least as much, and whenever in θ the strategies were such that $s_1 = (x, y_1)$ and $s_2 = (x, y_2)$, where $C(x, y_1) = C(x, y_2) = 1$, the strategies in θ_2 will instead be $s_1 = (x, y_1')$ and $s_2 = (x, y_2)$, where $C(x, y_1) = 0$ and $C(x, y_2) = 1$. In this case agent 1 will receive γ more than before. However, this happens with probability $> 1 - 4\gamma$. Therefore his payoff will increase by $\gamma - 4\gamma^2 > \epsilon$. So there is no ϵ-Nash equilibrium where each agent comes within ϵ of his guarantees. □

THEOREM 8.3. *Boolean circuit game* EXP-APPROX GUARANTEENASH *is* $\mathbf{NP}^{\#\mathbf{P}}$*-complete.*

PROOF. We first show that Boolean circuit game EXP-APPROX GUARANTEENASH is in $\mathbf{NP}^{\#\mathbf{P}}$. Given an instance $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$, we nondeterministically guess a polynomially small strategy profile θ. Then we test whether θ is an $\epsilon/2$-Nash equilibrium that is within $\epsilon/2$ of meeting the guarantees or whether θ is either not an ϵ-Nash equilibrium or fails to be within ϵ of the guarantees. In the former case we accept, in the latter case we reject.

We now argue the correctness of the algorithm. If $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a positive instance of GUARANTEENASH, then there exists a Nash equilibrium with the guaranteed properties in $\mathcal{G}$. By Theorem 3.4 there exists an $\epsilon/2$-Nash equilibrium θ that can be represented in polynomial space where the payoffs of each player are within $\epsilon/2$ of the guarantees. So the algorithm will accept upon guessing θ.

If $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a negative instance of GUARANTEENASH, then there does not exist any ϵ-Nash equilibrium with the guaranteed properties. So whatever strategy profile θ the algorithm guesses, either θ will fail to be an ϵ-Nash equilibrium or θ will fail to be within ϵ of the guarantees. Therefore the algorithm will always reject θ.

It is now left to show that in $\mathbf{NP}^{\#\mathbf{P}}$ we can tell whether θ is an $\epsilon/2$-Nash equilibrium that is within $\epsilon/2$ of meeting the guarantees or whether θ is either not an ϵ-Nash equilibrium or fails to be within ϵ of the guarantees. We can do this by using a $\#\mathbf{P}$ oracle to compute ν as in Equation 1 (in proof of Theorem 5.3) to within a polynomial number of bits of accuracy. Therefore in $\mathbf{P}^{\#\mathbf{P}}$ we can test whether $\nu_i(\theta) + \epsilon/2 \geq \nu_i(R_i(\theta, s_i))$ or $\nu_i(\theta) + \epsilon < \nu_i(R_i(\theta, s_i))$ for every agent i and $s_i \in \{0, 1\}$ and can test whether $\nu_i(\theta) \geq g_i + \epsilon/2$ or $\nu_i(\theta) < g_i - \epsilon$ for every agent i.

We now show that Boolean circuit game EXP-APPROX GUARANTEENASH is $\mathbf{NP}^{\#\mathbf{P}}$-hard. Say that we have a language $L \in \mathbf{NP}^{\#\mathbf{P}}$. By Corollary 5.7 there exists a

nondeterministic TM M that computes L, which makes only one query calls to a #CIRCUITSAT oracle, has all its nondeterminism at the beginning, and only accepts computations where the correct oracle query result is encoded in the nondeterminism. Let $f(|x|)$ be a polynomial that bounds the length of a string needed to encode the nondeterminism of M, let $g(|x|)$ (without loss of generality even) be a polynomial that bounds the number of inputs to the circuit queried by M, and let y be a string of bits that encodes the nondeterminism used in M on a particular run.

Given an input x we construct a Boolean game $\mathcal{G}$ with the following agents: $f(|x|)$ agents $y_1, \ldots, y_{f(|x|)}$, called y *agents*, $f(|x|)$ agents $y'_1, \ldots, y'_{f(|x|)}$, called y' *agents*, $g(|x|)$ agents $z_1, \ldots, z_{g(|x|)}$, called z *agents*, and agents $\mathcal{J}_1$, $\mathcal{J}_2$, and $\mathcal{J}_3$, called the *judges*.

Let the string $y = s_{y_1} s_{y_2} \ldots s_{y_{f(|x|)}}$ encode the nondeterminism of M, and let C be the circuit sent to the oracle query using the nondeterminism encoded in y, let k be the oracle query guess encoded by y, let m be the actual number of satisfying assignments of C, and let n be the number of inputs to C.

The payoffs are as follows.

y agents. Agent y_i is paid 1 regardless.

y' agents. Agent y'_i receives payoff 1 if his strategy is the same as y_i's and 0 otherwise.

z agents. The z agents are paid according to a game of pennies (see Section 2). Agent z_i plays pennies against agent z_{i+1}, where i is odd.

agent $\mathcal{J}_1$. $\mathcal{J}_1$ receives payoff $\frac{k+\frac{1}{2}}{2^n}$ if he plays 0 and $C(s_{z_1} s_{z_2} \ldots s_{z_k})$ otherwise.

agent $\mathcal{J}_2$. $\mathcal{J}_2$ receives payoff $\frac{k-\frac{1}{2}}{2^n}$ if he plays 0 and $C(s_{z_1} s_{z_2} \ldots s_{z_k})$ otherwise.

agent $\mathcal{J}_3$. $\mathcal{J}_3$ receives payoff 1 if $\mathcal{J}_1$ plays 0, $\mathcal{J}_2$ plays 1, and M run on input x with the nondeterministic choices encoded by y accepts assuming that the query result encoded by y is correct. Otherwise, $\mathcal{J}_3$ receives 0.

We guarantee that $\mathcal{J}_3$ and all the y'_i be paid 1. We make no guarantees to the other players. Let $\epsilon = 1/(200 \cdot f(|x|) \cdot g(|x|) \cdot 2^n)$.

Now we show that if $x \in L$, then there exists a Nash equilibrium in G with these guarantees, and if $x \notin L$, then there exists no ϵ-Nash equilibrium in G within ϵ of these guarantees.

Say $x \in L$. Then there exists a nondeterministic guess $y = y_1 y_2 \cdots y_{f(|x|)}$ such that M accepts x run with the nondeterminism encoded by y, and the query result encoded by y is correct. We claim that the strategy profile θ is a Nash equilibrium that meets the guarantees, where θ is the strategy profile where: $s_{y_i} = s_{y'_i} = y_i$; $s_{\mathcal{J}_1} = 0$, $s_{\mathcal{J}_2} = 1$, $s_{\mathcal{J}_3} = 0$, and the z agents randomize uniformly between their two strategies. We first show that in θ, each agent is in equilibrium playing θ. The y agents and the y' agents are in equilibrium because they all receive payoff 1. The z agents are because they are playing the unique equilibrium strategy of pennies. $\mathcal{J}_1$ is in equilibrium because he now receives $\frac{k+\frac{1}{2}}{2^n}$ and playing 1 yields a payoff of $C(s_{z_1} s_{z_2} \ldots s_{z_k})$, which has expectation $\frac{m}{2^n}$. However, because y encodes a valid query guess, $k = m$. Similarly, $\mathcal{J}_2$ currently receives payoff $C(s_{z_1} s_{z_2} \ldots s_{z_k})$, which is expected to be $\frac{m}{2^n} = \frac{k}{2^n}$ and would receive only $\frac{k-\frac{1}{2}}{2^n}$ by changing his strategy. Finally, $\mathcal{J}_3$'s payoff is independent of his strategy and so he is also in equilibrium.

The y' agents all receive their guarantees of 1. $\mathcal{J}_3$ also receives his guarantee of 1 because $s_{\mathcal{J}_1} = 0$, $s_{\mathcal{J}_2} = 1$, and running M on x with the nondeterminism encoded by y results in an accepting computation.

Say $x \notin L$, then there exists no ϵ-Nash equilibrium within ϵ of the guarantees. For the sake of contradiction, assume that an ϵ-Nash equilibrium θ exists in which

each agent is within ϵ of his guarantees. We note that each y agent must play some particular strategy with probability greater than $1-\epsilon$ (if y_i does not, then y_i' cannot attain a payoff of at least $1-\epsilon$). Let $\bar{s}_{y_i}$ be the strategy agent y_i plays with probability $\geq 1-\epsilon$ in θ, and let $\bar{y} = \bar{s}_{y_1}\bar{s}_{y_2}\ldots\bar{s}_{y_{f(|x|)}}$. By union bound, $\Pr[\theta_{y_1}\theta_{y_2}\ldots\theta_{y_{f(|x|)}} = \bar{y}] \geq 1-f(|x|)\cdot\epsilon$. Because $\epsilon < \frac{1}{100\,f(|x|)}$, $\bar{y}$ is played with probability at least $99/100$.

Also, by Theorem A.1, $\Pr[\theta_{z_i} = 0] \in [1/2 - 2\epsilon, 1/2 + 2\epsilon]$. $\mathbb{E}[C(\theta_{z_1}\theta_{z_2}\ldots\theta_{z_n})]$ is within $2\epsilon \cdot n \leq 1/(100 \cdot 2^n)$ of $m/2^n$.

Now because $x \notin L$ either $\bar{y}$ encodes a rejecting computation on M, or the query result of $\bar{y}$ is incorrect. In the former case, J_3 receives payoff 0 whenever $\bar{y}$ is played, and so cannot receive more than $1/100$. In the latter case, $k \neq m$. If $k < m$, then agent J_1 will receive $\frac{k+1/2}{2^n}$ for playing 0, but will receive at least $\frac{m}{2^n} - \frac{1}{100\cdot 2^n}$ for playing 1. Because $[\frac{m}{2^n} - \frac{1}{100\cdot 2^n}] - [\frac{k+1/2}{2^n}] > 2\epsilon$ by Claim 7.11 $\Pr[\theta_{J_1} = 1] \geq \frac{1}{2}$ and so J_3's payoff will be at most $1/2 < 1-\epsilon = g_{J_3} - \epsilon$. A symmetric argument handles the case where $k > m$. □

THEOREM 8.4. *Boolean circuit game* POLY-APPROX *and* CONST-APPROX GUARANTEENASH *are* $\mathbf{NP}^{\mathbf{BPP}} = \mathbf{MA}$*-complete.*

PROOF. We first show that Boolean circuit game POLY-APPROX GUARANTEENASH is in $\mathbf{NP}^{\mathbf{BPP}}$. Given an instance $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$, we nondeterministically guess a polynomially small strategy profile θ. Then we test whether θ is an $\epsilon/2$-Nash equilibrium that is within $\epsilon/2$ of meeting the guarantees or whether θ is either not an ϵ-Nash equilibrium or fails to be within ϵ of the guarantees. In the former case we accept, in the latter case we reject.

We now argue the correctness of the algorithm. If $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a positive instance of GUARANTEENASH, then there exists a Nash equilibrium with the guaranteed properties in $\mathcal{G}$. By Theorem 3.4 there exists an $\epsilon/2$-Nash equilibrium θ that can be represented in polynomial space where the payoffs of each player are within $\epsilon/2$ of the guarantees. So the algorithm will accept upon guessing θ.

If $(\mathcal{G}, \epsilon, (g_1, \ldots, g_n))$ is a negative instance of GUARANTEENASH, then there does not exist any ϵ-Nash equilibrium within ϵ of meeting the guaranteed properties. So whatever strategy profile θ the algorithm guesses, either θ will fail to be an ϵ-Nash equilibrium or θ will fail to be within ϵ of the guarantees. Therefore the algorithm will always reject θ.

It is now left to show that in **BPP** we can tell whether θ is an $\epsilon/2$-Nash equilibrium that is within $\epsilon/2$ of meeting the guarantees or whether θ is either not an ϵ-Nash equilibrium or fails to be within ϵ of the guarantees. Note that by Remark 5.11 we can verify whether θ is an $\epsilon/2$-Nash equilibrium or not even an ϵ-Nash equilibrium in **BPP**. By Remark 5.10 in **BPP** we can determine if $v_i(\theta) \geq g_i - \frac{\epsilon}{2}$ or $v_i(\theta) \geq g_i - \epsilon$. Therefore we can test whether all these properties hold or at least one fails to hold using calls to a **BPP** oracle.

We now show that Boolean circuit game CONST-APPROX GUARANTEENASH is $\mathbf{NP}^{\mathbf{BPP}}$-hard. Say that we have a language $L \in \mathbf{NP}^{\mathbf{BPP}}$. By Lemma 5.13 there exists a nondeterministic TM M, which computes L, which makes only one query call to a **BPP** oracle for the problem ACAPP, has all its nondeterminism at the beginning, and only accepts computations where the correct oracle query is encoded in the nondeterminism. Let $f(|x|)$ be a polynomial that bounds the length of a string needed to encode the nondeterminism of M, let $g(|x|)$ (without loss of generality even) be a polynomial that bounds the number of inputs to the circuit queried by M, let y be a string of bits that encodes the nondeterminism used in M on a particular run, and let $r = \lceil \log_{1/(4\epsilon)} 100g(|x|) \rceil$.

Given an input x, we construct a Boolean game $\mathcal{G} = (\mathfrak{s}, \nu)$ with the following agents: $f(|x|)$ agents $y_1, \ldots, y_{f(|x|)}$ called y *agents*, $f(|x|)$ agents $y'_1, \ldots, y'_{f(|x|)}$ called y' *agents*, $r \cdot g(|x|)$ agents $z_1, \ldots, z_{r \cdot g(|x|)}$ called z *agents*, and agents J_1 and J_2 called the *judges*.

Let the string $y = s_{y_1} s_{y_2} \ldots s_{y_{f(|x|)}}$ encode the nondeterminism of M, and let C be the circuit sent to the oracle query using the nondeterminism encoded in y, let $k \in \{0, 1\}$ be the oracle query guess encoded by y, let $m \in \{0, 1\}$ be the correct query response when C is queried, let n be the number of inputs to C, and let $w = w_1 w_2 \ldots w_n$ where $w_i = XOR(s_{z_{(i-1)r+1}}, s_{z_{(i-1)r+2}}, \ldots, s_{z_{i \cdot r}})$.

The payoffs are as follows.

y agents. Agent y_i is paid 1 regardless.
y′ agents. Agent y'_i receives payoff 1 if his strategy is the same as y_i's and 0 otherwise.
z agents. The z agents are paid according to a game of pennies (see Section 2). Agent z_i plays pennies against agent z_{i+1} where i is odd.
agent J_1. J_1 receives payoff $\frac{1}{2}$ if he plays 0 and $C(w)$ if he plays 1.
agent J_2. J_2 receives payoff 1 if J_1 plays k and M run on input x with the nondeterministic choices encoded by y accepts assuming that the query result encoded by y is correct. Otherwise, J_2 receives 0.

We guarantee that J_2 and all the y'_i be paid 1. We make no guarantees to the other players. Let $\epsilon = \frac{1}{800 \cdot f(|x|) \cdot g(|x|)}$.

Now we show that if $x \in L$, then there exists a Nash equilibrium in G with these guarantees, and if $x \notin L$, then there exists no ϵ-Nash equilibrium in G within ϵ of these guarantees.

Say $x \in L$. Then there exists a nondeterministic guess $y = y_1 y_2 \cdots y_{f(|x|)}$ such that M accepts x run with the nondeterminism encoded by y and the query result encoded by y is correct. We claim that the strategy profile θ is a Nash equilibrium that meets the guarantees, where θ is the strategy profile where: $s_{y_i} = s_{y'_i} = y_i$; $s_{J_1} = m$, $s_{J_2} = 1$, and the z agents randomize uniformly between their two strategies. We first show that in θ, each agent is in equilibrium. The y agents and the y' agents are in equilibrium because they all receive payoff 1. The z agents are because they are playing the unique equilibrium strategy of pennies. J_1 is in equilibrium because if $m = 0$, then C accepts at most $\frac{1}{3}$ of its inputs. The XOR of uniformly random bits is uniformly random so $\mathbb{E}[C(w)] = \mathbb{E}[C(U_n)] \leq \frac{1}{3}$ (where U_n is the uniform distribution over n-bit strings). So J_1 does better by playing $s_{J_1} = 0 = m$. If $m = 1$, a similar argument works. Finally, J_2's payoff is independent of his strategy, so he is also in equilibrium.

The y' agents all receive their guarantees of 1. J_2 also receives his guarantee of 1 because $s_{J_1} = m = k$ (because the oracle query result encoded by y is correct) and running M on x with the nondeterminism encoded by y results in an accepting computation.

Say $x \notin L$, then there exists no ϵ-Nash equilibrium within ϵ of the guarantees. For the sake of contradiction, assume that an ϵ-Nash equilibrium θ exists in which each agent is within ϵ of his guarantees. We note that each y agent must play some particular strategy with probability greater than $1 - \epsilon$ (if y_i does not, then y'_i cannot attain a payoff of at least $1 - \epsilon$). Let $\bar{s}_{y_i}$ be the strategy agent y_i plays with probability $\geq 1 - \epsilon$ in θ, and let $\bar{y} = \bar{s}_{y_1} \bar{s}_{y_2} \ldots \bar{s}_{y_{f(|x|)}}$. By a union bound, $\Pr[\theta_{y_1} \theta_{y_2} \ldots \theta_{y_{f(|x|)}} = \bar{y}] \geq 1 - f(|x|) \cdot \epsilon$. Because $\epsilon < \frac{1}{100 f(|x|)}$, $\bar{y}$ is played with probability at least $99/100$.

Also $\Pr[\theta_{z_i} = 0] \in [1/2 - 2\epsilon, 1/2 + 2\epsilon]$ by Theorem A.1. So by Claim 7.13 each bit w_i, which is the XOR of r such bits, is within $(4\epsilon)^r \leq 1/100n$ of being uniform, and their joint distribution is within $1/100$ of uniform. So $\mathbb{E}[C(w)]$ is within $1/100$ of $\mathbb{E}[C(U_n)]$.

Now because $x \notin L$, either y encodes a rejecting computation on M, or the query result of $\bar{y}$ is incorrect. In the former case, J_3 receives payoff 0 whenever $\bar{y}$ is played, and so cannot receive more than $1/100$. In the latter case, $k \neq m$. If $k = 0$ and $m = 1$, then agent J_1 will receive $\frac{1}{2}$ for playing 0, but will receive $\mathbb{E}[C(w)] \geq \mathbb{E}[C(U_n)] - \frac{1}{100} \geq \frac{2}{3} - \frac{1}{100}$ for playing 1. Because $[\frac{2}{3} - \frac{1}{100}] - [\frac{1}{2}] > 2\epsilon$ by Claim 7.11 $\Pr[\theta_{J_1} = 1] \geq \frac{1}{2}$ and so J_3's payoff will be at most $1/2 < 1 - \epsilon$. A symmetric argument handles the case where $k = 1$ and $m = 0$. □

THEOREM 8.5. *Graph game and Boolean graph game* GUARANTEENASH *is* **NP***-complete for all levels of approximation. The results hold even when restricted to degree d graphs, $d \geq 3$.*

PROOF. Graph game GUARANTEENASH is in **NP** because we can guess a strategy profile $\frac{n^2 \log(n^2 \max_i |\mathfrak{S}_i|)}{\epsilon^2}$-uniform strategy profile θ and test in polynomial time whether θ is an ϵ-Nash equilibrium where each player is payed within ϵ of his guarantees. If it is, accept; if not, reject. This algorithm works because by Theorem 3.4, if there exits a Nash equilibrium that meets the guarantees, then there exists a $\frac{n^2 \log(n^2 \max_i |\mathfrak{S}_i|)}{\epsilon^2}$-uniform ϵ-Nash equilibrium that gets within $\epsilon/2$ of the guarantees.

To show that it is **NP**-hard, we reduce from CIRCUITSAT. Given a circuit C we create an instance of Boolean graph game GUARANTEENASH, $(\mathcal{G}, \epsilon, (1, \ldots, 1))$. We create $\mathcal{G}$ with the following agents: 2 *input agents* x and x' for each input x to C and a *gate agent* g for each gate g in the circuit. Each agent has a strategy space of $\{0, 1\}$.

Each input agent x is paid 1 regardless. Each input agent, x' is paid 1 only if it plays the same strategy as x, the other input agent that represents the same circuit input. Except for the gate agent associated with the output gate, each gate agent g is paid 1 for correctly computing the output of his gate (with respect to the strategies of the agents that correspond to the inputs of his gate), and is paid 0 otherwise. If an input x to the circuit is an input to a gate g, then the agent associated with the gate g receives his payoff according to x's strategy (not x''s strategy). The output agent gets paid 1 only if he both correctly computes the output of his gate and that value is 1. Let $\epsilon = 1/100$.

We now show that if $C \in$SAT, then there exists a Nash equilibrium that meets the guarantees, but if $C \notin$SAT, then there exist no ϵ-Nash equilibrium that comes within ϵ of meeting the guarantees.

Say C has a satisfying assignment. Then the strategy profile where all input agents play a strategy that corresponds to some satisfying assignment, and the gate agents correctly evaluate their gates is a Nash equilibrium that meets the guarantees. It is a Nash equilibrium because each agent receives a payoff of 1, and so cannot do better. The input agents receive 1 because x and x' always play the same strategy. Each gate agent besides the output agent is paid 1 because he correctly computes the output of his gate with respect to the strategies of the two inputs. The output gate agent correctly computes the output of his gate with respect to the strategies of the two inputs; moreover, because this is a satisfying assignment, the output he computes is 1, and so he also receives a payoff of 1.

If C has no satisfying assignment, then there exists no ϵ-Nash equilibrium that comes within ϵ of the guarantees. Every ϵ-Nash equilibrium of $\mathcal{G}$ that is within ϵ of the guarantees corresponds to some evaluation of the circuit. By induction we show that every player in the game (with the exception of the gate agent associated with the output gate) plays a pure strategy, that corresponds with some evaluation of the circuit with probability $> 1 - 2\epsilon$.

The base case is the input gates. Every input agent x must play some strategy with probability $\geq 1 - \epsilon$, otherwise, his strategy will not agree with x'''s strategy with probability $\geq 1 - \epsilon$ no matter what x' plays, and so x'''s payoff will be less than $1 - \epsilon$.

Given that the input agents each play some strategy the majority of the time, we can define a correct strategy for gate agent g. We call the strategy of the gate agent g *correct* if it corresponds with the output of the gate g in an evaluation of C using the strategy that agent x plays the majority of the time as the input to gate x.

We claim that in any ϵ-Nash equilibrium, each gate agent besides the output agent must play the correct strategy with probability $\geq 1 - 2\epsilon$. We proceed by induction. The base case has already been proven. The inductive step is exactly as in the proof of Claim 7.15.

Therefore, in any qualifying Nash equilibrium, each player (beside the output gate player) must play a strategy corresponding with the correct valuation of the circuit; but because there is no satisfying assignment, the agent assigned to the output node, will not get a payoff close to 1. For in the correct valuation, his gate evaluates to 0, but if he plays 0, he is paid nothing; so the best he can do is play 1. Because each of the agents corresponding to the input gates plays the correct strategy with probability $\geq 1-2\epsilon$, and the output gate receives nothing when they both play the correct strategy, the most that the output agent can be paid is $4\epsilon < 1 - \epsilon$. □

Conitzer and Sandholm [2008] showed that EXACT GUARANTEENASH is NP-complete in bimatrix games. We observe that the same holds even for POLY-APPROX.

THEOREM 8.6. *[Conitzer and Sandholm 2008] Bimatrix* EXACT *and* POLY-APPROX GUARANTEENASH *are* **NP***-complete.*

PROOF. The hardness proof is exactly the same as the proof of Theorem 8.1 except now N is polynomial in the size of the input instead of exponential.

It is in **NP** because we can guess a polynomially-sized strategy profile, θ, and then in polynomial time check that it is a Nash equilibrium that satisfies the guarantees. By Proposition 3.3, if such a Nash equilibrium exits, then there exists one of at most polynomial size. □

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. □

APPENDIX

A. ANALYSIS OF PENNIES

The game of pennies $\mathcal{G} = (\mathfrak{s}, \nu)$ involves 2 players. $\mathfrak{s}_1 = \mathfrak{s}_2 = \{0, 1\}$ and the payoffs are as follows.

		Player 2	
		Heads	Tails
Player 1	Heads	(1, 0)	(0, 1)
	Tails	(0, 1)	(1, 0)

Pennies has a unique Nash equilibrium where both agents randomize uniformly between their two strategies. If the second player does not randomize equally between his strategies, player 1's best strategy is to play, with probability 1, the strategy that player 2 plays more often. Similarly if player 1 does not randomize equally, player 2 does the opposite of what player 1 plays most often. So the only Nash equilibrium is when both players randomize equally between their two options.

The following theorem gives us an idea of what constitutes an ϵ-Nash equilibrium in the game of pennies.

THEOREM A.1. *In any ϵ-Nash equilibrium of pennies, each player randomizes between each strategy with probability $\frac{1}{2} \pm 2\epsilon$.*

PROOF. Say that player 1 plays 1 with probability p and player 2 plays 1 with probability q. Then the payoff to player 1 is $pq + (1-p)(1-q)$. Now let $p = \frac{1}{2} + \delta$ and $q = \frac{1}{2} + \delta'$. If agent 1 plays a pure strategy, his payoff will be either q or $1-q$. In any ϵ-Nash equilibrium it must be that $pq + (1-p)(1-q) + \epsilon \geq \max\{q, 1-q\} \Rightarrow \max\{q, 1-q\} - [pq + (1-p)(1-q)] \leq \epsilon$.

Say that $\delta' \geq 0$. Then we get

$$\frac{1}{2} + \delta' - \left[\left(\frac{1}{2} + \delta\right)\left(\frac{1}{2} + \delta'\right) + \left(\frac{1}{2} - \delta\right)\left(\frac{1}{2} - \delta'\right)\right] \leq \epsilon \Rightarrow \delta' - 2\delta\delta' \leq \epsilon \Rightarrow \delta' \leq \frac{\epsilon}{1-2\delta}.$$

Similarly, if $\delta' \leq 0$. Then we get

$$\frac{1}{2} - \delta' - \left[\left(\frac{1}{2} + \delta\right)\left(\frac{1}{2} + \delta'\right) + \left(\frac{1}{2} - \delta\right)\left(\frac{1}{2} - \delta'\right)\right] \leq \epsilon \Rightarrow -\delta' - 2\delta\delta' \leq \epsilon \Rightarrow -\delta' \leq \frac{\epsilon}{1+2\delta}.$$

Doing the same thing for agent 2 with $\delta > 0$:

$$\frac{1}{2} + \delta - \left[\left(\frac{1}{2} + \delta\right)\left(\frac{1}{2} - \delta'\right) + \left(\frac{1}{2} - \delta\right)\left(\frac{1}{2} + \delta'\right)\right] \leq \epsilon \Rightarrow \delta + 2\delta\delta' \leq \epsilon \Rightarrow \delta \leq \frac{\epsilon}{1+2\delta'}.$$

And now with $\delta < 0$:

$$\frac{1}{2} - \delta - \left[\left(\frac{1}{2} + \delta\right)\left(\frac{1}{2} - \delta'\right) + \left(\frac{1}{2} - \delta\right)\left(\frac{1}{2} + \delta'\right)\right] \leq \epsilon \Rightarrow -\delta + 2\delta\delta' \leq \epsilon \Rightarrow -\delta \leq \frac{\epsilon}{1-2\delta'}.$$

Now by substitution and algebraic manipulation, we can see that these conditions require that $|\delta|, |\delta'| < 2\epsilon$. □

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 that 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. We would also like to thank the anonymous referees for helpful comments.

REFERENCES

ÀLVAREZ, C., GABARRÓ, J., AND SERNA, M. 2005. Pure Nash equilibria in games with a large number of actions. In *Mathematical Foundations of Computer Science 2005*, J. Jedrzejowicz and A. Szepietowski Eds., Lecture Notes in Computer Science Series, vol. 3618, Springer Berlin, 95–106.

ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M. 1998. Proof verification and the hardness of approximation problems. *J. ACM 45*, 3, 501–555.

ARORA, S. AND SAFRA, S. 1998. Probabilistic checking of proofs: A new characterization of NP. *J. ACM 45*, 1, 70–122.

BABAI, L. AND MORAN, S. 1988. Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes. *J. Comput. Syst. Sci. 36*, 254–276.

CHEN, X. AND DENG, X. 2005. 3-Nash is PPAD-complete. Tech. rep. TR05-134, Electronic Colloquium on Computational Complexity.

CHEN, X. AND DENG, X. 2006. Settling the complexity of two-player Nash equilibrium. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, Los Alamitos, CA, 261–272.

CHEN, X., DENG, X., AND TENG, S.-H. 2006. Computing Nash equilibria: Approximation and smoothed complexity. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 603–612.

CONITZER, V. AND SANDHOLM, T. 2008. New complexity results about Nash equilibria. *Games Econ. Behav. 63*, 2, 621–641.

DASKALAKIS, C., GOLDBERG, P. W., AND PAPADIMITRIOU, C. H. 2006. The complexity of computing a Nash equilibrium. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (FOCS)*.

DASKALAKIS, C., GOLDBERG, P. W., AND PAPADIMITRIOU, C. H. 2008. The complexity of computing a Nash equilibrium. *SIAM J. Comput. 39*, 1, 195–259.

DASKALAKIS, C. AND PAPADIMITRIOU, C. 2005a. The complexity of games on highly regular graphs. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*.

DASKALAKIS, C. AND PAPADIMITRIOU, C. 2005b. Three-player games are hard. Tech. rep. TR05-139, Electronic Colloquium on Computational Complexity.

FEIGE, U., GOLDWASSER, S., LOVÁSZ, L., SAFRA, S., AND SZEGEDY, M. 1996. Interactive proofs and the hardness of approximating cliques. *J. ACM 43,* 2, 268–292.

FEIGENBAUM, J., KOLLER, D., AND SHOR, P. 1995. A game-theoretic classification of interactive complexity classes. In *Proceedings of the 10th Annual IEEE Conference on Computational Complexity (CCC)*. 227–237.

FORTNOW, L., IMPAGLIAZZO, R., KABANETS, V., AND UMANS, C. 2008. On the complexity of succinct zero-sum games. *Computat. Complex. 17,* 3, 353–376.

GILBOA, I. AND ZEMEL, E. 1989. Nash and correlated equilibria: Some complexity considerations. *Games Econ. Behav. 1*, 80–93.

GOLDBERG, P. W. AND PAPADIMITRIOU, C. H. 2006. Reducibility among equilibrium problems. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (FOCS)*. 21–23

GOLDREICH, O. 2005. On promise problems (a survey in memory of Shimon Even [1935–2004]). Tech. rep. TR05-018, Electronic Colloquium on Computational Complexity.

GOLDREICH, O., MICALI, S., AND WIGDERSON, A. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM 38*, 3, 691–729.

GOLDWASSER, S., MICALI, S., AND RACKOFF, C. 1989. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18*, 1, 186–208.

GOTTLOB, G., GRECO, G., AND SCARCELLO, F. 2003. Pure Nash equilibria: Hard and easy games. In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*. ACM Press, New York, NY, 215–230.

HESSE, W., ALLENDER, E., AND BARRINGTON, D. 2002. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci. 65*, 695–716.

KEARNS, M., LITTMAN, M. L., AND SINGH, S. 2001. Graphical models for game theory. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. 253–260.

LIPTON, R., MARKAKIS, E., AND MEHTA, A. 2003. Playing large games using simple strategies. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*.

MEGIDDO, N. AND PAPADIMITRIOU, C. H. 1991. A note on total functions, existence theorems, and computational complexity. *Theoret. Comput. Sci. 81*, 1, 317–324.

NASH, J. 1951. Non-cooperative games. *Annals Math. 54*, 2, 286–295.

NASH, J. F. AND SHAPLEY, L. S. 1950. A simple three-person poker game. *Contrib. Theor. Games 1*, 24, 105–116.

SCHOENEBECK, G. 2004. The complexity of finding Nash equilibria in succinctly represented games. Undergraduate thesis, Harvard University.

SCHOENEBECK, G. AND VADHAN, S. 2006. The computational complexity of Nash equilibria in concisely represented games. In *Proceedings of the 7th ACM Conference on Electronic Commerce (EC)*. ACM, New York, NY, 270–279.

Received October 2009; revised March 2012; accepted March 2012