The complexity of game dynamics: BGP oscillations, sink equlibria, and beyond

Alex Fabrikant* Christos H. Papadimitriou†

July 6, 2007

Abstract

We settle the complexity of a well-known problem in networking by establishing that it is PSPACE-complete to tell whether a system of path preferences in the BGP protocol [25] can lead to oscillatory behavior; one key insight is that the BGP oscillation question is in fact one about Nash dynamics. We show that the concept of *sink equilibria* proposed recently in [11] is also PSPACE-complete to analyze and approximate for graphical games. Finally, we propose a new equilibrium concept inspired by game dynamics, *unit recall equilibria*, which we show to be close to universal (exists with high probability in a random game) and algorithmically promising. We also give a relaxation thereof, called *componentwise unit recall equilibria*, which we show to be both tractable and universal (guaranteed to exist in every game).

*Computer Science Division, University of California at Berkeley. Research supported by the Fannie and John Hertz Foundation Fellowship. email: alexf@cs.berkeley.edu

†Computer Science Division, University of California at Berkeley. Research supported by NSF grant CCR-0121555 and CCF-0515259 and a grant from Microsoft Research. email: christos@cs.berkeley.edu

1 Introduction

The recent proof that finding a Nash equilibrium is an intractable problem [6, 4] is a serious complexity-theoretic criticism of game theory's most prominent solution concept. In its aftermath, it is natural to seek alternatives. The *approximate Nash equilibrium* is perhaps the most obvious target, and it is indeed being pursued vigorously [19, 7, 17] while negative results about it begin to surface as well [5, 7, 9].

What does one look for in a solution concept?

1. First it must be *natural, compelling, convincing, realistic* as a behavioral prediction. Here the mixed Nash equilibrium does not do too well, as it postulates deliberate randomization whose only goal is the creation of a flat objective encouraging similar randomization by others — arguably not the most natural behavior. The correlated equilibrium [1, 23] fares even worse, with its requirement of a trusted randomizer. But the pure Nash equilibrium is, of course, ideal from this point of view. Incidentally, see [10] for an extensive commentary on what it means for an equilibrium concept to be realistic in the context of the Internet.

2. It must be *guaranteed to exist*, an obvious desideratum for a behavioral prediction, one that ensures universal applicability. The mixed Nash equilibrium and, as a corollary, the correlated equilibrium do well here, but not pure Nash.

3. Finally (and this is an important conceptual contribution of computer science to this field) for a solution concept to be a credible prediction of the behavior of a group of agents *it must be computationally tractable.* This criterion separates the mixed Nash equilibrium from the correlated and the pure Nash one. But if we consider multi-player succinctly representable games (as we have to, if we want to be true to our ambition of using game theory to understand the Internet and electronic commerce), only the correlated equilibrium survives (most of the time) [23].

The second criterion, universality, easily presents itself for many solution concepts based on the dynamics of the gameplay — every game starts somewhere, and goes from there. An interesting idea along these lines, which has been proposed and pursued recently [11] in the context of the "price of anarchy", is that of *sink equilibria*. It is an extension of the pure Nash equilibrium, in cases where it does not exist, to its natural dynamic generalization: *the ergodic states of the Nash dynamics*. Recall that the Nash dynamics [8] is a directed graph on the game's state space (set of pure strategy profiles, see Section 2), with an edge (s, s') signifying that s and s' agree on all players except for player i, and player i has a better payoff in s' than in s (in a more restrictive version, s' represents i's best response to s). The pure Nash equilibria are the sinks of this graph. In their absence, we may want to look at the sink connected components, and postulate that a distribution on them is the desired solution. Two alternatives emerge: (1) We could focus on one sink component and consider its steady-state distribution of the corresponding Markov chain, assuming that the chain starts there, or (2) consider the steady-state distribution of the Markov chain associated with the Nash dynamics, perhaps assuming a uniform initial distribution (a natural alternative which, to our knowledge, has not been pursued explicitly).

In this paper we point out that this approach is hindered by devastating complexity-theoretic obstacles: Even though the distribution (and the expected payoffs) of sink equilibria are easy to calculate for normal-form games, it is PSPACE-complete to say anything nontrivial about them

in succinct games — arguably the most important case (Theorem 2.2). Our proof for the case of graphical games is a novel simulation of a space-bounded Turing machine by the Nash dynamics of such a game.

An explicit ambition of the area of algorithmic game theory is to use insights from the interface between game theory and computation in order to understand the Internet. We next present an important (and unexpected) applied result of this sort, related to *BGP oscillations*. Autonomous systems (ASes) comprising the Internet route traffic originating in their domain or in the domains of other ASes using a proverbially complex protocol called *the Border Gateway Protocol (BGP)* [25, 27]. BGP allows each AS to specify arbitrary preferences on the paths to a destination, and to selectively offer such paths to others. Possibly because of the complexity of these preferences and offers (reflecting obscure agreements and opaque interests), BGP is known to occasionally cause *oscillations* in the Internet, that is, lengthy, drawn-out sequences of route reassignments by many AS's. This is a very disruptive, and consequently well-studied, phenomenon [15, 18, 28, 21]. A formal approach to BGP oscillations was initiated by [15]; they defined the *stable paths problem* as an abstraction of the phenomenon. They defined *safety* as the absence of oscillatory possibilities, and gave a necessary condition for safety (the presence of a stable path configuration, known to be NP-complete to test) and a sufficient condition for safety (the absence of a *dispute wheel,* a set of circular preferences), which is coNP-complete to test. But no necessary and sufficient condition had been known.

In Section 3 we point out that *the BGP oscillation problem is in fact also a problem of Nash dynamics* in a new kind of succinct game (Proposition 3.1). The ASes are players, strategies correspond to choice of a next hop, and BGP preferences define (in a succinct way) the utility of the resulting graph for each of the players. Safety is tantamount to the game having *only* pure Nash equilibria, and no other sink components with two or more states. Perhaps the main technical contribution of this paper is that *BGP safety is PSPACE-complete* (Theorem 3.2), which resolves the complexity of this important question in networking. The reduction is involved and novel (and has to be quite different from that of Theorem 2.2).

Incidentally, on an aesthetic level, it is amusing to note that the observed BGP oscillations among a few ASes are known to often go on for hours or days. Exponentially long computation confined in a limited space is the hallmark of PSPACE-completeness, so this is a hint that the formal worst-case intractability proven here may occasionally be echoed in actual observed behavior.

Returning to our quest for appropriate equilibrium concepts, are sink equilibria and other variants of steady states in Nash dynamics natural and compelling solution concepts? The main shortcoming of such concepts is that they postulate complete lack of strategic behavior by the players, beyond a desire for local improvement. A game will have many sink components, and the payoffs will differ wildly from one component to the other. It would make perfect sense for a player to be strategic in the beginning (transient phase) of the game, sacrificing short-term gains so as to land on her favorite sink. How can we model such strategic behavior?

Allowing full strategic behavior brings us to the time-honored and well-studied — if somewhat inconclusively so — area of *repeated games* [26, 24, 20], historically the first domain of interaction between complexity and game theory. The basic problem with this is that repeated games allow and predict strategic behavior that can be extremely sophisticated ([24]; see [10] for an insightful discussion of the kinds of strategic behavior one should expect on the Internet). In the interest of exploring the effect of limited strategic behavior in Nash dynamics, we consider *unit recall games*. In such games, players move simultaneously, and each player's next strategy choice depends only on the current state (strategy choices by everybody). A player's strategy is thus a finite automaton whose

states are the player's own strategies, and whose transitions are labeled by the strategy combinations of everybody else. Once everyone adopts such a strategy, the Nash dynamics becomes deterministic and ends up in a cycle; the payoff is the time-average payoff of the states on the cycle.

Does the unit recall version of every game (that is, the game in which the available strategies are all possible unit recall automata) have a pure Nash equilibrium? The answer, at least for bimatrix games, is "almost": we show (Theorem 4.2) that a random bimatrix game has such an equilibrium with probability approaching 1 as the number of strategies grows. Furthermore, w.h.p., such an equilibrium is easy to find in polynomial time.

However, somewhat surprisingly, these equilibria aren't completely universal: even the extremely simple *matching pennies game* has no such equilibrium (Proposition 4.1, proven by a computer analysis of the 32×32 game). Then, how hard is it to tell whether a game has a unit-recall equilibrium in the worst case? The problem is, a priori, in the complexity class $\Sigma_2 P$ (it asks whether there exists a combination of automata such that all defection-automata are unfavorable). We conjecture that it is in P. *And we are half-way there*: we can prove, by a reduction to the problem of finding in a graph the cycle with the smallest average edge length, that the best-response problem for such games is in P, and therefore the overall problem is in NP.

Another cause for optimism is the behavior of a relaxation of the unit recall equilibrium obtained from the following observation: Going from one strategy of a unit recall game to another requires changing a large number of transitions in the automaton/strategy; what if we allow changing *just one transition* at a time?

This yields a novel kind of equilibrium that we call *componentwise equilibrium*, a generalization of pure Nash.[1] Thinking of the strategies/automata as vectors of transitions, we can define an equilibrium concept by only allowing defections that change a single component. We show that *every multi-player game has a componentwise unit recall equilibrium (CURE).* That is, in any game there are finite automata with the strategies of each player as state space such that no player can increase her long-term expected payoff by changing one transition (Theorem 4.5) In fact, we can find such an equilibrium in polynomial time; the correlated equilibrium [23] is the only other solution concept we know that is guaranteed to exist and easy to find. In view of our stated goal of identifying equilibrium concepts that are natural, tractable, and guaranteed to exist, these results suggest that the CURE concept may merit further theoretical attention, and that its more appealing subset, the unit-recall equilibrium, may also be tractable.

2 The Complexity of Sink Equilibria

In a *normal form game* G we have $n \geq 2$ players $1, 2, \ldots, n$; for each player i we have a finite set of strategies S_i. We denote by $\mathcal{S}$ the *set of pure strategy profiles*, or *states*, the Cartesian product $S_1 \times S_2 \times \cdots \times S_n$; by $\mathcal{S}_{-i}$ we denote the product of all strategy sets except S_i. Finally, for each player i we have a *utility function* u^i mapping S to the integers.

The *Nash dynamics* of a game is a directed graph with $\mathcal{S}$ as its set of vertices, and an edge (s, s') if s and s' agree on all components except for one, say the ith, and $u^i(s') > u^i(s)$. The *strict Nash dynamics* is a subgraph of the Nash dynamics, where an edge s, s' signifies, in addition, that i's strategy in s' is i's *best response* to s, that is, one of the strategies in S_i which, if substituted for

[1] It is also an instance of a more general, and very promising, idea that extends the pure Nash equilibrium: Suppose that the strategies of each player are equipped with a cost matrix, and the player defects from a strategy to another only if the payoff difference is bigger than the cost; this is an idea also treated in [3]. In the case of componentwise equilibria, the cost is zero if the two strategies differ in one component, and infinity otherwise.

the ith component of s, yields the largest possible utility. (We shall henceforth specify strict or not strict Nash dynamics only in cases that do not apply to both kinds.) A *pure Nash equilibrium* of a game is a state that is a *sink* of the Nash dynamics.

Consider now the strongly connected components of the Nash dynamics, and among those, one that has no outgoing edges. Such a component is called a *sink equilibrium* of the game. The Nash dynamics of a graph define a Markov chain, by assigning equal probabilities to all outgoing edges. The following are then computational problems[2] of interest:

- BEST (WORST) SINK: Given a game, what is the highest (or lowest) expected utility for a given player in the steady-state distribution of a sink equilibrium?
- IN A SINK: Does a given state s belong to any sink equilibrium?

The following result only involves depth-first search and matrix inversion:

Proposition 2.1 *Given a game in normal form, the problems* BEST (WORST) SINK *and* IN A SINK *can be solved in polynomial time*

An n-player game requires an exponential, in n, number of bits to be represented; hence, the only computationally meaningful ways of representing multi-player games must be succinct. A *graphical game* is a particularly useful succinct way of representing a game. It is defined in terms of a graph $G = ([n], E)$ whose set of vertices is the set of players. The utility function u^i is now a function mapping $\mathcal{S}_{N(i)}$ to the integers, where $\mathcal{S}_{N(i)}$ is the Cartesian product of the strategy sets of player i and of all players that are adjacent to i in G.

Our main result in this section is this:

Theorem 2.2 IN A SINK *for graphical games is PSPACE-complete.*

Proof sketch: Similarly to the BGP result below, we seek to use sink equilibria of a graphical game to capture the halting behavior of a space-bounded Turing machine.

We define an intermediate construction, a *local-dependence resetting cycle machine* (LRCM), which moves continously and unidirectionally around a circular tape, and updates tape cells based on its internal state and the state of *nearby* cells, but not the current one. This mirrors the game constraint that a player cannot decide on its new strategy *based on* its current strategy. The reduction to a graphical game then creates a graph which locally models the tape with two players per cell (a "tape cell" player and a "state" player), and $O(1)$ edges to players modeling nearby cells. We design the utilities so that the Nash dynamics, starting at the canonical starting state, have a "ripple" of activity move around the cycle until the corresponding computation halts. To guarantee that the canonical starting state of a game is in a large sink equilibrium iff the machine does *not* halt, we have the LRCM reset to its starting state after it takes enough steps to have exhausted all possible computation states and looped somewhere.

The full proof is given in an appendix. □

As an easy corollary, by taking liberties with the scale of the utility functions, we get that the problems BEST (WORST) SINK for graphical games cannot be approximated at all.

[2]More precisely, these are families of problems, parametrized by one's choice of game representation

3 The Complexity of BGP Oscillations

A *BGP system* $\mathcal{B}$ is a *directed* graph $G = (V, E)$ whose nodes are called *autonomous systems* or *ASes*, except for $v_0 \in V$, called the *target node* (we assume all traffic is directed to it). Also, for each $v \in V - \{v_0\}$ there is a function λ^v, represented as a Boolean circuit, say, mapping all simple paths between v and v_0 to the integers. λ^v also assigns an integer preference value to $\perp$, the absence of a path (in which case all paths whose utility is less than that of $\perp$ can be considered "disallowed paths").

The BGP system's *state* is a *path assignment* $\pi(u)$ mapping node u to a path from u to v_0 (or the absence thereof). Given a state, let us define $C(\pi, u) = \{(u, \pi(v))|(u, v) \in E\}$ to be the list of paths u can choose from, starting with a link to a neighbor v, and continuing with the current path for v. A path assignment π is *stable* if, for all u, $\pi(u) = \arg\max_{P \in C(\pi,u)} \lambda^u(P)$. We'll abbreviate path assignments by showing just $\nu(u)$, the next hop of the paths from u to v_0, i.e. $\pi(u) = (u, \nu(u), \nu(\nu(u)) \ldots, \nu^k(u) = v_0)$.

From a given state π, a node $a \in V$ can be *activated*, in which case a's path is updated with $\pi'(a) = \arg\max_{P \in C(\pi,a)} \lambda^a(P)$. That is, if there is a better option for a (a is "dissatisfied") then the best such option is adopted, otherwise, $\pi' = \pi$. We then say $\pi \xrightarrow{a} \pi'$. An infinite activation sequence is *fair*, ([15]) if, for all $v \in V$ and $i \in \mathbb{Z}$, there is a $j > i$ such that $a_j = v$. A system is said to be *convergent* if for all fair sequences $(a_1, a_2, \ldots)$, there exists an x such that $\pi_0 \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \ldots \xrightarrow{a_x} \pi_x$ and π_x is stable.

Our BGP model follows closely the *stable paths problem (SPP)* model of [15], see also [14]. One exception is that they provide an explicit list of ordered paths, instead of an algorithm for comparing paths. Our version is closer to the realities of both the formal BGP protocol specification, in which ASes are allowed to implement arbitrary preference functions, and real BGP implementations in routers, which somewhat limit the preference functions, but allow more freedom than just an explicit ordering of paths. Another important difference is that our model allows *ties* between paths; the real BGP actually does not (this is BGP's "Phase 2" tie-breaking as described in Sec. 9.1.2.1 of [25]). Our main lower bound proof below uses this possibility of ties. However, we can extend the result to not require ties; a sketch of this is given at the end of this section. A final difference is that the model of [15] is explicit about the asynchronous nature of the communication between the ASes by introducing message queues to the state, something that we view as an unnecessary complication that obscures the finitary, combinatorial nature of the problem.

We are interested a computational problem which we call

BGP SAFETY: given a BGP system as above, is it convergent?

Characterizing safety has been the main goal of the literature on BGP oscillations ([15, 18, 28, 21], inter alia). In [14] only a necessary condition, and another sufficient one, for safety are given, both NP-hard. We show below that the problem is PSPACE-complete.

One key observation is that *the BGP convergence problem is actually one about Nash dynamics.* Starting with a BGP system $\mathcal{B} = ((G, E), \lambda)$, consider a game $\mathcal{G}[\mathcal{B}]$ with a set $V - \{v_0\}$ of players, each with a strategy set equal to the set of outgoing edges from it. Every pure strategy profile s defines now a subgraph G_s of G with outdegree one for each player. The utility of v is then defined as $u^v(s) = \lambda^v(\pi[v, G_s])$, the value of the unique path from v to v_0 in G_s, if such a path exists, and $\lambda^v(\perp)$ otherwise. We omit the tedious proof of the following:

Proposition 3.1 *$\mathcal{B}$ is not convergent if and only if the strict Nash dynamics of $\mathcal{G}[\mathcal{B}]$ has a sink equilibrium of cardinality two or more (that is, one which is not a Nash equilibrium).*

Hence, BGP SAFETY amounts to telling whether there are non-Nash sink equilibria in the strict Nash dynamics of the novel succinct game $\mathcal{G}[\mathcal{B}]$. The situation here is much more specialized than with the graphical games proof in Section 2, and the proof of the following result is quite a bit more sophisticated (and very different) than the proof of Theorem 2.2.

Theorem 3.2 BGP SAFETY *is PSPACE-complete.*

Proof sketch: We start from the following computational problem:

STRING HALTING problem: Given a function $f : \Gamma^{t-2} \mapsto \Gamma \cup \{\text{halt}\}$ for some alphabet Γ and integer $t > 2$ (assume that the function is given as a Boolean circuit), is there an initial string $T_1 \cdots T_t \in \Gamma^T$ such that the following program halts? (Indices of T are understood modulo t, and $T_{-(i,i+1)}$ means $T_{i+2}T_{i+3} \cdots T_{i-1}$.)

```
1: i ← 0
2: while f(T_{-(i,i+1)}) ≠ halt do
3:       T_i ← f(T_{-(i,i+1)})
4:       i ← i + 1
5: end while
```

To show the STRING HALTING problem PSPACE-complete, we start, as in Theorem 2.2, from the problem of telling whether a linear-bounded Turing machine will halt if started on n blanks. We modify the Turing machine (in a way reminiscent of the construction in [13]) so that (1) it has a clock that is "syntactically integrated" in its operation (meaning, it is easy to check whether a configuration contains a legitimate clock state); (2) when the clock overflows, or if the machine is about to accept, it erases the tape, zeroes the clock, and restarts; (3) if it rejects, it halts; (4) therefore, the machine cycles if and only if it accepts the empty string. The details are messy and have to be done exactly right (and are omitted here).

We conclude that it is PSPACE-complete to tell if a Turing machine (with embedded clock) will cycle. The reduction from the cycling problem to STRING HALTING is not difficult, once it is clear that f can recognize legitimate configurations (and halt in every other case).

Starting from such a function f, we use the BGP system shown in Figure 1 to simulate the operation of the program above in such a way that, at any step, no more than 2 nodes are interested in changing their path in a way that keeps the system "alive", i.e. in a state from which it might still loop (oscillate) indefinitely. The system can oscillate iff there exists a starting configuration from which f never halts.

Given an instance $I = (\Gamma, n, f)$, with $|\Gamma| = s$ of STRING HALTING, we define the BGP system $\mathcal{B}(I) = ((V, E), \{\lambda^i\}$, where $V = \{Z, A_i, B_i, C_{i,j} | i \in N = \{1, \ldots, n\}, j \in S = \{1, \ldots, s\}\}$ and $E = \{(A_i, Z), (A_i, B_i), (B_i, C_{i,j}), (C_{i,j}, A_{i-1}) | i \in N, j \in S\}$ (operations on "tape" indices are implicitly modulo n). $Z = v_0$ is the target node. Since $C_{i,j}$'s have only one outgoing link, they never update. B_i's path choices, i.e. its preference function, λ^{B_i}, implements f, while λ^{A_i} is set up to assure that updates happen in clockwise (increasing i) order.

Let $\mathcal{T}(t_x, \ldots, t_y)$ be shorthand for the path segment:

$$(C_{x,t_x}, A_{x-1}, B_{x-1}, C_{x-1,t_{x-1}}, \ldots, B_y, C_{y,t_{y-1}})$$

Note the reverse order of the tape indices. We can now define the preference functions by:

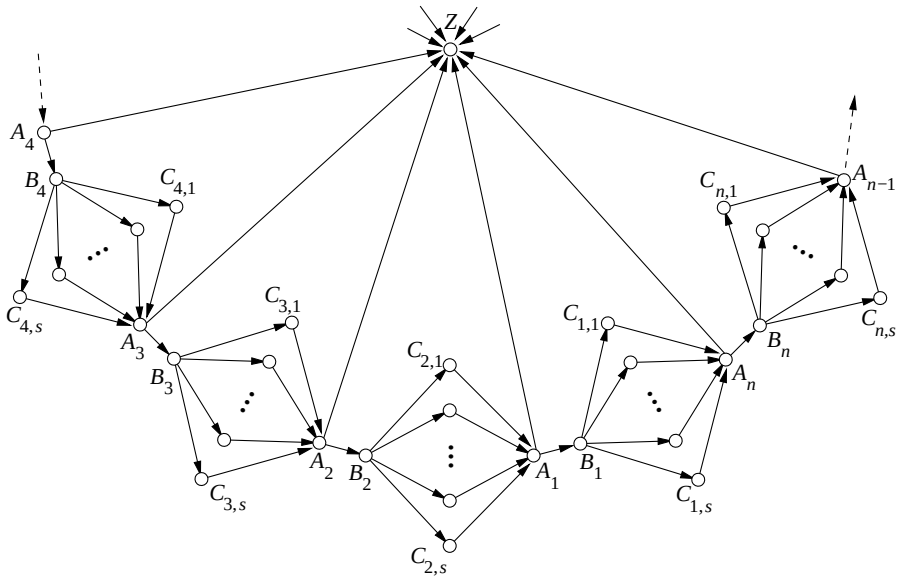

Figure 1: A BGP gadget to simulate STRING HALTING.

$$\lambda^{B_i}(P) \quad = \quad \begin{cases} 4 & \text{if } P = (B_i, \mathcal{T}(t_i, \ldots, t_{i+1}), A_i, Z) \wedge f(t_{i+2}, \ldots, t_{i-1}) = t_i \\ 1 & \text{if } P = (B_i, \mathcal{T}(t_i, \ldots, t_{i+1}), A_i, Z) \wedge f(t_{i+2}, \ldots, t_{i-1}) \neq t_i \\ 0 & \text{if } P = \bot \\ 2 & \text{else} \end{cases}$$

$$\lambda^{A_i}(P) \quad = \quad \begin{cases} 3 & \text{if } P = (A_i, Z) \\ 1 & \text{if } P = (A_i, B_i, C_{i,j}, A_{i-1}, Z) \\ 1 & \text{if } P = (A_i, B_i, \mathcal{T}(t_i, \ldots, t_{i+2}), A_{i+1}, Z) \wedge f(t_{i+2}, \ldots, t_{i-1}) \in \{t_{j \neq i}, \text{halt}\} \\ 4 & \text{if } P = (A_i, B_i, \mathcal{T}(t_i, \ldots, t_{i+2}), A_{i+1}, Z) \wedge f(t_{i+2}, \ldots, t_{i-1}) = t_i \\ 0 & \text{if } P = \bot \\ 2 & \text{else} \end{cases}$$

We call a state "alive" if either (a) exactly one A_i has $\nu(A_i) = Z$, or (b) only $\nu(A_{i+1})$ and $\nu(A_i)$ are Z and $\nu(B_i) = C_{i,f(T_{-(i,i+1)})}$, with f not returning "halt".

If a state is not alive, we can inductively show that no B_j will ever again have an incentive to switch (since it will never have a "clear view" to A_i) and no A_j will ever have an incentive to switch from Z to B_j, so any fair activation sequence will lead to all A_j's switching to Z, creating a stable configuration.

If a state is alive, its $\nu(B_j)$'s correspond to a configuration of f's tape, with the loop at cell i.

Inductively, after any activation sequence $(a_1, \ldots, a_x)$, we see that the system can only be in one of these types of live states as it simulates f:

1. For all $j \neq i$, $\nu(A_j) = B_j$, $\nu(B_j) = C_{i,T_j}$; $\nu(A_i) = Z$, and $\nu(B_i) = C_{i,f(T_{-(i,i+1)})}$. Only A_j for $j \notin \{i, i-1\}$ are dissatisfied. Updating A_{i+1} yields state type 3; updating any of the others creates a dead state.
2. For all $j \neq i$, $\nu(A_j) = B_j$, $\nu(B_j) = C_{i,T_j}$; $\nu(A_i) = Z$, and $\nu(B_i) \neq C_{i,f(T_{-(i,i+1)})}$. Only A_j for $j \notin \{i, i-1\}$ and B_i are dissatisfied. Updating B_i yields state type 1. Updating any of the A_j's yields a dead state.
3. For all $j \notin \{i, i+1\}$, $\nu(A_j) = B_j$, $\nu(B_j) = C_{i,T_j}$; $\nu(A_i) = \nu(A_{i+1}) = Z$, and $\nu(B_i) = C_{i,f(T_{-(i,i+1)})}$, with f not returning "halt". Only A_j, $j \neq i+1$ are dissatisfied. Updating A_i "advances" the simulation of C to the next step, and updates to state type 2, or, if the next step of C doesn't change the cell value, 1. Updating any of the other A_j's yields a dead state.

Thus, if I does not halt, the activation sequence $(B_1, A_2, A_1, B_2, A_3, \ldots)$ will make this system cycle between states 2, 1, and 3 (perhaps occasionally skipping 2 in the sequence when a type-3 state updates directly to type-1). If I does halt, a fair activation sequence will force the system to enter a dead and then reach a stable state in finite time. With the λ's clearly succinctly representable, we have that BGP SAFETY is PSPACE-complete. □

The above setup relies on B_j's maintaining their state by having their preference function remain "indifferent" when the "tape head" (the A_i-to-Z link) is elsewhere; this is the sole element above that ever requires "ties for first place" in the preference functions, which BGP disallows. We can eliminate this by introducing an extra state D_i for each B_i, with links from B_i to D_i and from D_i directly to Z. By setting B_i's preference function to 3 for (B_i, D_i, Z), and making A_i and $B_{j\neq i}$ dislike paths through D_i, we would introduce another category of dead states, with the same implications of forthcoming termination as above. We omit here the details of the proof that the reduction still works.

4 Unit Recall Equilibria

Sink equilibria restrict the strategic behavior by players to myopic local improvement. On the other hand, unrestricted strategic play on the states is a very involved subject [26]. The following is an interesting compromise between the two extremes, a minimal instance of bounded-recall games as studied by the game theory community [22, 2]:

Fix a game. A *unit recall strategy* by player i is a finite state automaton that has S_i as its state space and $\mathcal{S}_{-i}$ as its alphabet. It specifies a starting strategy, and a next strategy for every combination of plays by the other players. Notice that, assuming that the players act synchronously, a set of unit recall strategies, one for each player, defines a function from $\mathcal{S}$ to itself, as well as a start state, and thus an infinite walk from the start state ending up in a cycle. The payoff of this combination of automata is then the average utility in this cycle.

Notice that the above concepts allow us to define, for each game, a game by the same players whose strategies are unit recall strategies. The pure Nash equilibria of this game are called *unit recall equilibria.* It would be very exciting if every game had a unit recall equilibrium. Unfortunately, this is not the case. The following can be proved by exhaustion:

Proposition 4.1 *The game of matching pennies (with $n = 2$, $S_1 = S_2 = \{0, 1\}$, and $u^i(a, b) = (-1)^{i+a+b \bmod 2}$) has no unit recall equilibria.*

However, unit-recall equilibria are still *common*. Take a random 2-player $m \times m$ normal-form game. Pure Nash equilibria are known to exist with probability approaching $1 - 1/e$ as the number of strategies tends to infinity [12]. The probability of a URE existing, however, approaches 1:

Theorem 4.2 *A $m \times m$ bimatrix game with payoffs chosen independently from an arbitrary distribution, the probability that a unit recall equilibrium exists and can be found in polynomial time is at least $1 - 1/\Theta(\sqrt{m})$.*

Proof sketch: We show that a particular kind of URE exists w.h.p.: a *simple URE* in a bimatrix game with payoff matrices (R, C) is a pair of starting strategies (r, c) and a pair of a "punishment column" $c_p \neq c$ and "punishment row" $r_p \neq r$ for the row and the column player, respectively, such that $R_{k,c_p} \leq R_{r,c}$ and $C_{r_p,k} \leq C_{r,c}$ for all k. The unit-recall strategies are set so that, from (r, c), both players continue playing the same strategy, and from any other state, the row player defects to r_p to punish the column player, and vice versa. Such equilibria can be found in polynomial time by exhaustion.

Set, with foresight, $f(m) = \frac{1}{4} m \log m$, and pick the top $f(m)$ entries from all of R, and the top $f(m)$ entries from all of C (break ties uniformly at random). Let I^R and I^C be 0-1 matrices which have 1's wherever those top $f(m)$ entries occur, and 0 elsewhere. For a simple URE to exist, it is sufficient to have (i) event M: there exist (r, c) such $I^R_{r,c} = I^C_{r,c} = 1$, and (ii) event P: there exist some r_p and c_p such that I^R_{k,c_p} and $I^C_{r_p,k}$ are 0 for all k.

Since the entries of R and C are chosen independently, I^R and I^C are chosen uniformly at random from the set of 0-1 $m \times m$ matrices with exactly $f(m)$ 1's. A simple combinatorial argument gives $\Pr[\overline{M}] \leq 1/m^{\frac{1}{16} \log m}$. With some algebra, we can apply the Chebyshev bound to the number of empty columns (rows) of R (C) to get $\Pr[\overline{P}] \leq 4/\sqrt{m}$, so, asymptotically, $\Pr[\text{simple URE exists}] \geq 1 - 1/\Theta(\sqrt{m})$. We expect that a tighter analysis of $\Pr[P]$ will tighten the result to $1 - 1/\text{superpoly}(m)$. □

How difficult is then the problem UNIT RECALL EQUILIBRIUM (given a game, find a unit recall equilibrium)? Though the problem is prima facie a non-trivial member of $\Sigma_2 P$, the answer is, "not quite that difficult":

Theorem 4.3 *There is a polynomial algorithm which, given an n-player game and unit recall strategies for $n - 1$ players, calculates the best response by the nth player. Therefore,* UNIT RECALL EQUILIBRIUM *is in NP.*

Proof sketch: The $n - 1$ given unit recall strategies specify a function ϕ from $\mathcal{S}$ to $\mathcal{S}_{-i}$. Define a graph with vertex set $\mathcal{S}$ and edges $\{(s, (\phi(s), a)) : s \in \mathcal{S}, a \in S_i\}$, and assign to each state s the weight $u^i(s)$. It is not hard to see that the best response is determined by the cycle in this graph that has the highest average weight, a problem solvable in polynomial time [16]. □

Conjecture 4.4 UNIT RECALL EQUILIBRIUM *is in P.*

Let us now weaken this equilibrium concept. A unit recall equilibrium requires that, for each player, there is no way to change some subset of its transitions so that an improvement results. But suppose now that we only require that no improvement result by changing *any one* of the transitions. This could be a reasonable relaxation, if one assumes that transitions in the automaton get changed slowly and sequentially, and that lower utility will prevent a player from even starting the defection process. We call this weaker notion *componentwise unit recall equilibrium*, or *CURE*.

Theorem 4.5 *Every game has a CURE, which can be found in polynomial time, given a normal-form game.*

Proof sketch: Discard any players with only one choice of strategy. Let $m = \min_i |S_i| \geq 2$ and $M = \max_i |S_i|$. We'll give the proof that works when $n \geq 7$ or $m \geq 3$.

Consider an n-player game ($n \geq 2$), and a state s. We say that s is *i-punishable* if there is another state $s^{(i)}$, differing from s in the strategy of two or more players, such that $u^i(s^{(i)}) \leq u^i(s)$. We call a state s *punishable* if it is i-punishable for all players i. The following is proven in the appendix:

Lemma 4.6 *Every game with 7 or more players, or with every player having 3 or more strategies, has a punishable state.*

Now, given a punishable state s, we shall construct n unit recall strategies, one for each player, which, we shall argue, form a CURE. We shall describe these strategies implicitly, by the function ψ from $\mathcal{S}$ to itself that they induce. This function starts at s, and it maps s to itself. For each player i, any state s' such that $s_{-i} = s'_{-i}$ is mapped to the state $s^{(i)}$ guaranteed to exist by i-punishability. All other states are mapped to themselves, and this concludes the description of the CURE.

To verify that s is a CURE, notice that, since the play is on s alone, only changes of the transition for s by a player could possibly lead to disequilibrium. However, if player i defects from s_i, the play will move to a state s' with $s_{-i} = s'_{-i}$, and then immediately to $s^{(i)}$ where it stays forever, thus failing to improve i's utility (by the assumption that $u^i(s^{(i)} \leq u^i(s)$. This completes the proof of existence.

In a normal-form game, a punishable state can be found by exhaustion.

Some geometric arguments, omitted here, extend the lemma to guarantee a punishable state for all games except 2×2, $2 \times 2 \times 2$, $2 \times 2 \times 3$, and $2 \times 2 \times 2 \times 2$ games. The finite corner cases are handled by tweaking the "punishable state" approach. □

We are not claiming that the equilibria guaranteed to exist by this result are in any sense natural. State s is supported, for each player i, by both "threats" by other players (their transitions from s' to $s^{(i)}$ and the persistence in $s^{(i)}$) and by what is essentially a commitment by player i to collaborate with the other players in punishing itself if it ever deviates from playing s. Still, besides correlated equilibria, CUREs in normal-form games are the only equilibrium concepts we know that are both tractable and universal.

5 Open Problems

We believe that sink equilibria are intractable in other general situations as well, for example in congestion games (with large, splitable flows), and in valid utility games (it has been shown in [11] that a special case thereof already produces PLS-completeness, and we expect that the whole story is much bleaker).

The sink equilibria concept remains one important direction to explore in our search for equilibria. It would be interesting to consider strategic enhancements of this concept, besides the unit recall idea offered here, in line with the realities of the Internet: Asynchronous players with different reaction speeds whose move depends not on the state, but on a rough (and perhaps delayed...) estimate of their utility.

Finally, can our complexity result on BGP oscillations be extended to the case in which the utilities are given explicitly as a preference order of paths? We cannot see how to prove this, but we expect that this problem is, indeed, also PSPACE-complete.

Acknowledgements

We'd like to thank Vijay Ramachandran and Robert Kleinberg for stimulating discussions.

References

[1] R. J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, March 1974.

[2] R. J. Aumann. Survey of repeated games. In *Collected papers*, pages 411–438. MIT Press, 2000.

[3] E. Ben-Sasson, A. Kalai, and E. Kalai. An approach to bounded rationality. In *Proceedings of NIPS 2006*.

[4] X. Chen and X. Deng. Settling the complexity of 2-player Nash equilibrium. In *Proceedings of IEEE FOCS 2006*, pages 261–270.

[5] X. Chen, X. Deng, and S.-H. Teng. Computing Nash equilibria: Approximation and smoothed complexity. In *Proceedings of IEEE FOCS 2006*, pages 603–612.

[6] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of ACM STOC 2006*, pages 71–78.

[7] C. Daskalakis, A. Mehta, and C. Papadimitriou. A note on approximate Nash equilibria. In *Proceedings of WINE 2006*.

[8] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *Proceedings of ACM STOC 2004*, pages 604–612.

[9] T. Feder, H. Nazerzadeh, and A. Saberi. Breaking the factor of two in approximate Nash equilibria. Manuscript.

[10] E. J. Friedman and S. Shenker. Learning and implementation on the Internet. Working paper 1998-21, Rutgers Univ., Dept. of Economics, 1998. Available from http://www-snde.rutgers.edu/Rutgers/wp/rutgers-wplist.html.

[11] M. Goemans, V. Mirrokni, and A. Vetta. Sink equilibria and convergence. In *Proceedings of IEEE FOCS 2005*, pages 142–151.

[12] K. Goldberg, A. J. Goldman, and M. Newman. The probability of an equilibrium point. *J. Res. Nat. Bur. Standards Sect.*, 72B:93–101, 1968.

[13] P. Gopalan, P. Kolaitis, E. Maneva, and C. H. Papadimitriou. The connectivity of boolean satisfiability: Computational and structural dichotomies. In *Proceedings of ICALP 2006*, pages 346–357.

[14] T. Griffin, F. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243, 2002.

[15] T. G. Griffin and G. Wilfong. An analysis of BGP convergence properties. In *Proceedings of ACM SIGCOMM 1997*, pages 277–288.

[16] R. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[17] S. Kontogiannis, P. Panagopoulou, and P. Spirakis. Polynomial algorithms for approximating Nash equilibria of bimatrix games. In *Proceedings of WINE 2006*.

[18] C. Labovitz, G. R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Trans. on Networking*, 6:515–528, 1998.

[19] R. J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proceedings of ACM EC 2003*, pages 36–41.

[20] M. L. Littman and P. Stone. A polynomial-time Nash equilibrium algorithm for repeated games. In *Proceedings of ACM EC 2003*, pages 48–54.

[21] D. McPherson, V. Gill, D. Walton, and A. Retana. BGP persistent route oscillation condition. In *Proceedings of 50th IETF*, March 2001.

[22] A. Neyman. Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. *Economics Letters*, 19(3):227–229, 1985.

[23] C. H. Papadimitriou. Computing correlated equilibria in multi-player games. In *Proceedings of ACM STOC 2005*, pages 49–56.

[24] C. H. Papadimitriou and M. Yannakakis. On complexity as bounded rationality. In *Proceedings of ACM STOC 1994*, pages 726–733.

[25] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard), 1995.

[26] S. Sorin. Cooperation through repetition. In S. Hart and A. M. Colell, editors, *Cooperation: Game Theoretic Approaches*, pages 169–198. Springer, 1997.

[27] J. W. Stewart. *BGP4: Inter-Domain Routing on the Internet*. Addison-Wesley, 1998.

[28] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. Technical Report 96-631, USC ISI, 1996.

A Proof of Theorem 2.2

Formally, let an LRCM be a tuple of a state space, a designated starting state, a tape alphabet, the length of its circular tape, and a transition function mapping the machine state and the contents of tape cells $i-2$, $i-1$, and $i+2$, where i is the current tape cell, to the new state and the new content of the current cell, or a "halt" command: $C = (Q, q_0 \in Q, \Gamma, t, f : Q \times \Gamma^3 \to (Q \times \Gamma) \cup \{\text{halt}\})$ (we'll use (f_q, f_T) for components of f's output). We require LRCMs to obey two additional semantic constraints:

1. (Resetting constraint) When started on a blank tape, it must either halt or return to the same configuration (state q_0 and a blank tape) in finite time.

2. (Self-loop-free constraint) There may not be any self-loops in the finite-state control, i.e. whenever $f(q, \ldots) = (q', \ldots)$, $q \neq q'$.

C is said to halt on starting tape state $(T_0, \ldots, T_t) \in \Gamma^t$ if the following loop halts (using $T_{N(i)}$ as shorthand for $(T_{i-2}, T_{i-1}, T_{i+2})$):

```
1: i ← 0
2: while f(q, T_N(i)) ≠ halt do
3:       (q, T_i) ← f(q, T_N(i))
4:       i ← i + 1
5: end while
```

Lemma A.1 *It is PSPACE-complete to tell whether an LRCM C will halt on a starting tape state.*

Proof:

We start with the generic PSPACE-complete problem, the space-bounded halting problem for TMs. We first reduce an instance of this problem (a Turing machine M, an input x, and a tape bound t), to the halting problem for a TM variant $M' = (Q', \Sigma', \Gamma', \delta', q'_0, q'_a, q'_r)$, with a circular tape of length t' which, independently of its own input, simulates M on x with space bound t and halts iff M accepts. Additionally, we use $t \log |Q|$ extra cells of the tape to store a step counter that counts up to the total number of possible configurations of M, and, when the counter overflows, have the machine enter a special state which forces the entire tape, including the counter, to be reset to blank, moves the tape head to the starting position, and sets the control to q_0.

To map M' to a LRCM, we:

- simulate bidirectional movement by making the CM do an "idle round" around the tape after each step of M'
- store the direction of tape movement in the state during the idle round
- remove the transition function's dependence on the current tape cell by storing a copy of cell i's contents in cell $i+2$
- double the state space by adding a "flip-flop" component which flips at each step to ensure the absence of control self-loops

Formally, let $\Gamma'' = \Gamma' \times \Gamma' \times \{0,1\} \times \{0,1\}$ and $Q'' = Q' \times \{\pm 1\} \times \{\text{active}, \text{idle}\} \times \{0,1\}$. Use $T_i = (T_i^t, T_i^c, T_i^l, T_i^{cl})$ for the "this cell", "copy of cell $i-2$", "last-change bit", and "copy of $i-2$'s last-change bit" components of T_i; and $q = (q^q, q^d, q^i, q^f)$ for "state", "direction of movement", "idle indicator", and "flip-flop" components of the state q.

Then, let f be the following:

---

if $q^i = \text{active}$ **then**
  $(\tilde{q}^q, \tilde{T}^t, \tilde{q}^d) \leftarrow \delta'_t(q^q, T^c_{i+2})$ // "action step"
  if $\tilde{q}^q \in \{q'_a, q'_r\}$ **then**
    return halt
  end if
  $\tilde{T}^l \leftarrow 1$; $\tilde{q}^i = \text{idle}$
else
  $\tilde{T}^l \leftarrow 0$; $\tilde{T}^t \leftarrow T^c_{i+2}$; $\tilde{q}^{q,d} \leftarrow q^{q,d}$ // "idle step"
  if $T^l_{i+2} = 1 \wedge q^d = -1$ or $T^{cl}_{i+2} = 1 \wedge q^d = +1$ **then**
    $\tilde{q}^i = \text{active}$ // prepare to write in next cell
  end if
end if
$\tilde{T}^{c,cl} \leftarrow T^{t,l}_{i-2}$; $\tilde{q}^f \leftarrow \neg q^f$
return $(\tilde{q}, \tilde{T})$

As above, we claim that if we start LRCM $C = (Q'', (q'_0, \text{active}, +1, 0), \Gamma'', t' \geq 3, f)$ on tape $T_i = (\text{␣}, 0, \text{␣}, 0)$ $\forall i$, it will halt iff M' halts, and otherwise will eventually return to state q_0 and the starting tape state, with each step of M' getting simulated by an action step of C.

An induction on the steps of C shows that, at the beginning of any step of C at cell i:

- $T^{c,cl}_{k+2} = T^{t,l}_k$ for at least all k other than $i-1$ and $i-2$.
- C's state is one of the following 4 configurations:
  1. Exactly 1 last-change bit T^l_k is on, $q^i = \text{idle}$. If $k = i+1$ or $k = i$, q^d may only be $+1$. Corresponds to M' having just written to tape cell k, switched the state to q^q, and about to move in direction q^d
  2. Only T^l_{i+1} is on, $q^i = \text{active}$, $q^d = -1$. Same M' state as in 1 above.
  3. Only T^l_{i-1} and T^l_i are on, $q_i = \text{idle}$. Corresponds to M' having just written to $i-1$, switched to state q^q, about to move by q^d (having moved by -1 on the previous step).
  4. No last-change bit T^l_k is on, $q_i = \text{active}$, $q_d = +1$. Corresponds to M' having just written to $i-1$, switched to state q^q, and about to move by $+1$.
- Each step of C which starts with $q^i = \text{active}$ changes the state q^q and the sequence $\{T^t_j\}$ to match the state and tape contents of the next step of M''s computation, hence causing C to halt iff M' halts.

Allowing a succinct representation of f as above, the reduction is clearly polynomial, hence yielding PSPACE-completeness. □

Our reduction from an LRCM $C = (Q, q_0, \Gamma, t, f)$ and starting tape T will create a game $\mathcal{G}$ which, for each tape cell i, has a "cell" player C_i and a "state" player S_i, with the utilities of both

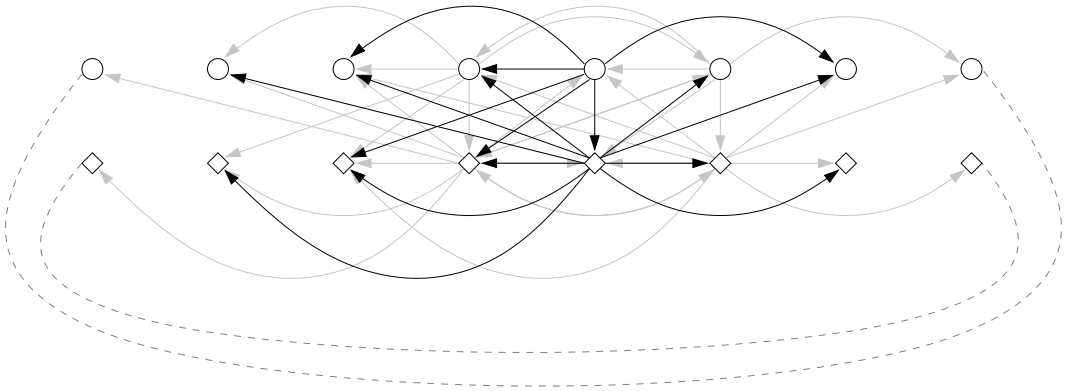

Figure 2: A segment of a graphical game gadget to simulate an LRCM.

depending on several nearby C_j and S_j players. We will arrange $\mathcal{G}$ so that its Nash dynamics follow the computation of C, with no players ever given an incentive to deviate from "canonical" game states, i.e. those that directly correspond to a configuration of C. Thus, the game state corresponding to the starting state of the LRCM will be in a sink equilibrium if and only if the LRCM does not halt.

The dependencies in the game follow the pattern in Figure 2. C_i players are shown as circles and S_i — as diamonds. The black links show the dependencies for a one pair of C_i and S_i nodes. The same pattern of links repeats for all t pairs of players (C_j, S_j), which are arranged in a closed loop.

The strategy set for each C_i is just Γ; the strategy set for S_i is $Q \cup \{\emptyset\}$. The utility function for S_i depends on the strategies of C_{i-3}, C_{i-2}, C_{i-1}, C_{i+1}, C_{i+2}, S_{i-3}, S_{i-2}, S_{i-1}, S_{i+1}, S_{i+2}, and itself; and that for C_i — on C_{i-2}, C_{i-1}, C_{i+2}, S_{i-2}, S_{i-1}, S_i, and itself. We set the utility function for S_i according to the rules in Table 1.

C_{i-3}	C_{i-2}	C_{i-1}	C_{i+1}	C_{i+2}	S_{i-3}	S_{i-2}	S_{i-1}	S_{i+1}	S_{i+2}	S_i	u^{S_i}
*	*	*	*	*	$\emptyset$	$\emptyset$	q_1	q_2	$\emptyset$	q_2 else	1 0
V	W	$f_T(q_1, V, W, Y)$	Y	*	$\emptyset$	q_1	$q_2 \neq q_1$	$\emptyset$	$\emptyset$	q_2 else	1 0
*	W	X	*	Z	$\emptyset$	$\emptyset$	q_1	$\emptyset$	$\emptyset$	$f_q(q_1, W, X, Z) \neq \text{halt}$ else	1 0
*	*	*	*	*	$\emptyset$	$\emptyset$	$\emptyset$	q_1	$\emptyset$	*	0
else										$\emptyset$ else	1 0

Table 1: The utility function for S_i. Asterisks indicate don't-cares, q_1 and q_2 are arbitrary elements of Q (i.e. not $\emptyset$), and V, W, X, Y, Z are arbitrary elements of Γ.

For C_i, we set $u^{C_i}(C_{i-2}, C_{i-1}, C_i, C_{i+2}, S_{i-2}, S_{i-1}, S_i)$ to 1 iff $S_{i-2} = \emptyset$, $\emptyset \neq S_i \neq S_{i-1} \neq \emptyset$, and $C_i = f_T(S_{i-1}, C_{i-2}, C_{i-1}, C_{i+2})$. For any other configuration of strategies, we set it to 0.

We start the game $\mathcal{G}$ with the tape players C_i playing the contents of the C's starting tape, S_0 and S_{-1} playing q_0, and all the other S_i's playing $\emptyset$. This puts the game state in "phase" (iv) as

phase	$C_{\dots}$	C_{i-3}	C_{i-2}	C_{i-1}	C_i	C_{i+1}	C_{i+2}	C_{i+3}	$C_{\dots}$
	$S_{\dots}$	S_{i-3}	S_{i-2}	S_{i-1}	S_i	S_{i+1}	S_{i+2}	S_{i+3}	$S_{\dots}$
(i)	$\cdots$	A	B	C	$\boxed{\text{D}}$	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	a	b	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
(ii)	$\cdots$	A	B	C	H	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	a	b	$\boxed{\emptyset}$	$\emptyset$	$\emptyset$	$\emptyset$
(iii)	$\cdots$	A	B	C	H	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	$\boxed{\text{a}}$	b	b	$\emptyset$	$\emptyset$	$\emptyset$
(iv)	$\cdots$	A	B	C	H	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	b	$\boxed{\text{b}}$	$\emptyset$	$\emptyset$	$\emptyset$
(i)	$\cdots$	A	B	C	H	$\boxed{\text{E}}$	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	b	c	$\emptyset$	$\emptyset$	$\emptyset$

Table 2: Canonical states of $\mathcal{G}$. Capital letters are tape symbols in Γ, lower-case letters are states in Q. $H = f_T(a, B, C, E)$, $c = f_q(b, C, H, F)$

shown in Table 2. By induction, it can be seen that, given that starting configuration:

- at each step of the Nash dynamics, the game state is in one of the four phases listed in the table,
- only the player whose strategy is marked by a square may have an incentive to deviate, and
- the game will cycle through states in phases (i)-(iv), simulating the computation of C, unless and until C halts, at which point the game will be in a pure Nash equilibrium.

Due to the resetting constraint on C, if C does not halt, $\mathcal{G}$ must return to its starting state after a finite number of steps. Since no other players have an incentive to deviate at any step, this cycle in $\mathcal{G}$'s state space forms a sink equilibrium. Conversely, if C does halt, there is a path from the starting configuration to the pure Nash equilibrium corresponding to the halted computation of C, thus assuring that the starting state of $\mathcal{G}$ is not part of a sink equilibrium.

Since the degree of $\mathcal{G}$'s graph is 10, we thus have that IN A SINK is PSPACE-complete.

B Proof of Lemma 4.6

The proof proceeds by a counting argument. For each player i, pick any s from the set $\arg\min_s u_i(s)$ to be $s^{(i)}$. For each such state, there are $\sum_i |S_i| - n + 1$ states that differ from it in at most one player's strategy. Via the union bound, the number of punishable states must be at least:

$$\prod_i |S_i| - n\Big(\sum_i |S_i| - n + 1\Big) > \prod_i |S_i| - n\sum_i |S_i| \geq Mm^{n-1} - n^2 M$$

Since $M > 0$, this is non-negative (guaranteeing at least one punishable state) whenever $n \geq 2$ and $m \geq 7$ and whenever $n \geq 3$ and $m \geq 3$. For $n = 2$ and $m = 3$, just set $s_i \notin \{s_i^{(1)}, s_i^{(2)}\}$.