The Complexity of Hex and the Jordan Curve Theorem*

Aviv Adler[1], Constantinos Daskalakis[2], and Erik D. Demaine[3]

1 **MIT CSAIL, Cambridge, USA**
`adlera@mit.edu`
2 **MIT CSAIL, Cambridge, USA**
`costis@mit.edu`
3 **MIT CSAIL, Cambridge, USA**
`edemaine@mit.edu`

Abstract

The Jordan curve theorem and Brouwer's fixed-point theorem are fundamental problems in topology. We study their computational relationship, showing that a stylized computational version of Jordan's theorem is PPAD-complete, and therefore in a sense computationally equivalent to Brouwer's theorem. As a corollary, our computational result implies that these two theorems directly imply each other mathematically, complementing Maehara's proof that Brouwer implies Jordan [10]. We then turn to the combinatorial game of Hex which is related to Jordan's theorem, and where the existence of a winner can be used to show Brouwer's theorem [6]. We establish that determining who won an (implicitly encoded) play of Hex is PSPACE-complete by adapting a reduction (due to Goldberg [7]) from Quantified Boolean Formula (QBF). As this problem is analogous to evaluating the output of a canonical path-following algorithm for finding a Brouwer fixed point – and which is known to be PSPACE-complete [8] – we thereby establish a connection between Brouwer, Jordan and Hex higher in the complexity hierarchy.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases Jordan, Brouwer, Hex, PPAD, PSPACE

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.24

1 Introduction

The Jordan curve theorem states that a simple closed curve C in $\mathbb{R}^2$ divides the plane into two connected components S_1, S_2 [9]. In particular, any continuous curve from a point $x \in S_1$ to a point $y \in S_2$ must intersect C. There are several known proofs of this basic topological fact, including one via Brouwer's fixed point theorem by Maehara [10]. In an earlier paper, Gale explores the equivalence between Brouwer's theorem and a theorem closely related to Jordan's, pertaining to the combinatorial game Hex [6]. In Hex, a rhomboidal board is partitioned into hexagonal tiles as in Figure 1, and two players claim tiles of the board until one of the players can connect the two opposite sides of the board that belong to him. The Hex theorem states that, once all tiles on the board are claimed, at least one of the two players has won.

In this paper we study the relationship of the Jordan, Brouwer and Hex theorems from both a computational and a mathematical standpoint. In particular, our goal is to show their 'equivalence' in both domains. We say that two theorems are *mathematically equivalent* if

* This work was partially supported by the NSF (grant CCF-1551875) and ONR (grant N00014-12-1-0999).

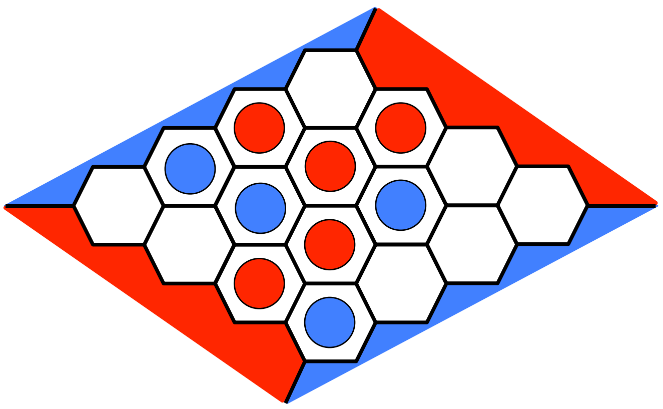

Figure 1 A sample Hex position, where Player 1 (red) has won.

(informally) they can be used to prove each other in an ‘elementary’ way, and *computationally equivalent* if the search problems with solutions guaranteed by them are complete in the same complexity class. This study is catalyzed by defining a stylized computational problem inspired by Jordan curve theorem as follows:

ZeroSurfaceCrossing

`Input:` Two circuits $F_1, F_2 : \{0,1\}^n \times \{0,1\}^n \to \{-2^m+1, \dots, 2^m-1\}$, defining two continuous surfaces $f_1, f_2 : [0,1]^2 \to \mathbb{R}$ in $\mathbb{R}^3$ via interpolation.[a]

`Output:` A point (x,y) such that $f_1(x,y) = f_2(x,y) = 0$, or a violation of the following boundary conditions:

(1) $f_1(0,y) \geq 0$ and (2) $f_1(1,y) \leq 0$ for all y;

(3) $f_2(x,0) \geq 0$ and (4) $f_2(x,1) \leq 0$ for all x.

[a] The input to each circuit is interpreted in the obvious way as specifying a point $(x,y) \in [0,1]^2$, where both x and y are integer multiples of $1/(2^n-1)$. Hence these circuits determine the values of f_1, f_2 at all such points. The values of f_1, f_2 at all other points are determined from these values via interpolation.

It is quite intuitive that, if Conditions 1–4 are satisfied, then the surfaces f_1 and f_2 should have an intersection on the zero plane. One way to show this is using the Jordan curve theorem; see Lemma 10. Moreover, given that f_1, f_2 are defined by circuits via interpolation, it is easy to see that the problem belongs to NP and thus in TFNP, the class of total problems in NP. The question is how it relates to other classes in TFNP [11]. We show the following:

▶ **Theorem 1.** ZeroSurfaceCrossing *is PPAD-complete.*

Given that Brouwer, the stylized computational problem of computing fixed points of continuous functions (defined formally in Section 3), is also PPAD-complete [11, 3, 5], Theorem 1 implies that ZeroSurfaceCrossing and Brouwer are computationally equivalent. Additionally, it helps establish the mathematical equivalence of the Brouwer and Jordan theorems. Maehera showed that Brouwer implies Jordan's theorem. Exploiting the proof of Theorem 1, we show the other direction, that Jordan implies Brouwer's theorem. Moreover, in view of Gale's proof that the Hex and Brouwer theorems are mathematically equivalent [6], our result also establishes that all three theorems are equivalent.

▶ **Proposition 2.** *The Jordan, Brouwer and Hex theorems are mathematically equivalent.*

1.1 Jordanian Action Inside PPAD

A close variant of the problem defined above is the curve crossing problem defined as follows.

CrossingCurves
`Input:` Two circuits $F_1, F_2 : \{0,1\}^n \to \{0, 2^{-m}, 2 \cdot 2^{-m}, \ldots, 1\}^2$, defining two continuous curves $f_1, f_2 : [0,1] \to [0,1]^2$ in $[0,1]^2$ via interpolation.[a]

`Output:` A pair $t_1, t_2 \in [0,1]$ such that $f_1(t_1) = f_2(t_2)$, or a violation of the following boundary conditions:

(1) $f_1(0) = (0,0)$; (2) $f_1(1) = (1,1)$; (3) $f_2(0) = (0,1)$ and (4) $f_2(1) = (1,0)$.

[a] The input to each circuit is interpreted in the obvious way as specifying an integer multiple of $1/(2^n - 1)$, so that F_i defines the location of curve f_i for all inputs in $\{0, 1/(2^n - 1), \ldots, 1\}$. The location of the curve at any $t \in [j/(2^n - 1), (j+1)/(2^n - 1)]$, where $j \in \{0, \ldots, 2^n - 1\}$, is determined by linearly interpolating between the locations of the curve at $j/(2^n - 1)$ and $(j+1)/(2^n - 1)$.

Again it is quite intuitive that, unless Conditions (1)–(4) in the definition of the problem fail, the curves defined by any input to CrossingCurves should cross. Indeed, this can be proven via the Jordan curve theorem, and because of Proposition 2 via Brouwer's fixed point theorem as well. We provide a direct proof using Brouwer's fixed point theorem, which implies as a corollary that the problem lies within PPAD.

▶ **Theorem 3.** CrossingCurves *is in PPAD.*

Under monotonicity conditions on at least one of the two curves, it is easy to see that CrossingCurves is in P. For example, suppose that at least one of the two curves f_i satisfies that, for all $0 \le t < t' \le 1$, $f_i(t)_1 < f_i(t')_1$, where $f_i(t)_1$ represents the first coordinate of $f_i(t)$ and similarly for $f_i(t')_1$. Under this condition, CrossingCurves can be easily solved via binary search. Similar conditions can be defined with respect to the second coordinate. When neither curve satisfies such a monotonicity condition with respect to neither coordinate, we do not see how to construct a polynomial time algorithm. At the same time, we do not see how an instance of CrossingCurves can encode the several paths and cycles that may co-exist in an instance of EndOfTheLine, the canonical PPAD-complete problem—see Section 2. (In comparison, the intersections of the surfaces of ZeroSurfaceCrossing with the zero-plane may comprise several paths and cycles, which allow encoding EndOfTheLine instances.) We leave pinning down the precise complexity of CrossingCurves for future work, expecting that the complexity classes defined in [4] may be useful in this classification.

1.2 Jordanian Action Over PPAD

While so far all action has taken place inside TFNP, we also explore how the three theorems, Jordan, Brouwer and Hex, are related higher in the complexity hierarchy. It was recently established that several algorithms for computing Brouwer fixed points and Nash equilibria are in fact capable of solving all of PSPACE [8]. For example, given an instance $\mathcal{I}$ of some problem in PSPACE, one can construct a 2-player game $\mathcal{G}$ such that the Nash equilibrium output by the Lemke-Howson algorithm provides a solution to $\mathcal{I}$ as a byproduct. Similar facts are known for homotopy methods.

We are thus interested in whether computational problems relating to Jordan and Hex also have the power of solving PSPACE. We propose the problem WhoWonHex, asking to determine whether player 1 is the winner of a Hex play. An instance of the problem comprises a circuit that takes as input the binary description of a cell in the Hex board and outputs the name of the player, 1 or 2, who claimed it during the play. We provide a formal description of Hex in Section 2.3, and define WhoWonHex in Section 5.2, establishing the following:

▶ **Theorem 4.** WhoWonHex *is PSPACE-complete.*

The proof of Theorem 4 can be obtained fairly easily using recent work of Goldberg [7]. There is a canonical method to determine who is the winner in a play of Hex by performing a walk on the Hex board. The walk starts at one of the corners of the Hex board and performs pivoting steps depending on which player has claimed the cells neighboring the current location of the walk, until another corner of the board is reached, which always happens due to topological reasons. What corner is reached determines which player won. This pivoting algorithm is quite reminiscent to the canonical algorithm for solving instances of 2-dimensional Sperner. An instance of this problem provides a succinct description of the coloring of the vertices of a square lattice using 3 colors and asks to identify a trichromatic triangle or a violation of certain boundary conditions by the coloring. The problem is PPAD-complete [11, 3, 2] and Goldberg recently established that identifying the tri-chromatic triangle reachable by the standard pivoting algorithm for this problem is PSPACE-complete. Theorem 4 is proven by making an analogy between the pivoting algorithms that solve Sperner and WhoWonHex. The precise details are a bit more intricate than this intuition, as we have to exploit the structure of the Sperner instances constructed in Goldberg's proof.

It is worth pointing out that the problem WhoWonHex that we study is very different than the typical computational problem studied in combinatorial game theory, namely determining given a configuration of the board whether some player has a winning strategy. Our problem is instead to determine who is the winner, once the play is completed.

Roadmap. We provide basic definitions in Section 2, recalling the Jordan curve theorem and Brouwer's fixed point theorem. We also describe the game of Hex and define the computational problems Brouwer and EndOfTheLine along with the class PPAD. In Section 3, we show that Brouwer and Jordan are equivalent, both mathematically and computationally (through the ZeroSurfaceCrossing problem). In Section 4, we discuss CurveCrossing showing that it is in PPAD. Finally, in Section 5, we show that WhoWonHex is PSPACE-complete. In Section 6 we conclude with some open problems suggested by our work.

2 Preliminaries

In this section, we will formally define all the theorems, problems, and constructs which we will analyze; in particular, we will define:

1. Brouwer's fixed-point theorem;
2. the Jordan curve theorem;
3. the game of Hex and the Hex theorem;
4. the complexity class PPAD, and the canonical computational problem EndOfTheLine that is associated with it.

Additionally, in order to formally describe the computational problems given above, we also need to define the interpolation schemes we use to transform functions over bitstrings to functions over the (continuous) unit square. We use a standard technique of using the bitstring to describe a point on a lattice; then, for inputs not on the lattice, the output is defined by interpolating from the outputs on the lattice. The formal construction is given in our full paper [1].

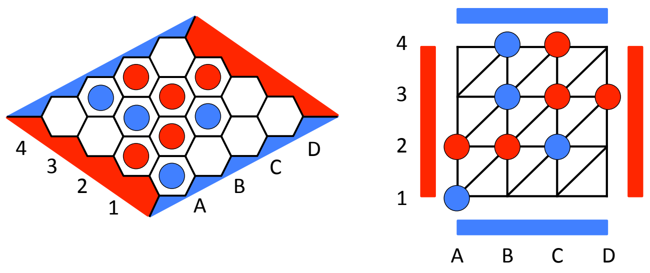

Figure 2 A Hex board and its corresponding dual-graph representation.

2.1 Brouwer's Fixed-Point Theorem and the Jordan Curve Theorem

We define here the theorems of Brouwer and Jordan. For Brouwer, we give a special case in two dimensions, involving functions from the unit square to itself. (This can be extended to the general two-dimensional case on any compact and convex set and higher dimensions; see e.g. [11, 3, 5] and their references.)

▶ **Theorem 5** (Brouwer's fixed-point theorem). *Given any continuous function $f : [0,1]^2 \to [0,1]^2$, there is a* fixed point*, i.e. some $x \in [0,1]^2$ such that $f(x) = x$.*

▶ **Theorem 6** (The Jordan curve theorem). *Any simple closed curve ϕ in $\mathbb{R}^2$ divides the space into two regions, one finite (the inside) and one infinite (the outside).*

2.2 PPAD and its Related Computational Problems

▶ **Definition 7** (The PPAD graph). Let N and P be circuits, both of which take as input an n-bit string and return an n-bit string as output. We then consider the directed graph whose vertices are n-bit strings such that there is an edge (u, v) if and only if $N(u) = v$ and $P(v) = u$.

By this definition, it is clear that each vertex has in- and out-degree of at most 1 (since any vertex u can only be preceded by $P(u)$ and succeeded by $N(u)$), and thus the graph must consist of a disjoint collection of isolated vertices, directed paths, and directed cycles. We now consider the following computational problem on this graph:

▶ **Definition 8** (EndOfTheLine). Given circuits N and P defining the PPAD graph G:
1. if the vertex 0^n is not a source vertex (with in-degree 0 and out-degree 1), return 0^n;
2. if the vertex 0^n is a source vertex, return any *other* unbalanced vertex (with in-degree and out-degree not equal).

EndOfTheLine is the canonical PPAD-complete problem. Because no directed graph can have exactly one unbalanced vertex, the existence of a solution is guaranteed. Of particular interest to us is the fact that a computational variant of Brouwer's fixed-point theorem (which we will formally describe in the next section) is also PPAD-complete.

2.3 The Game of Hex

The game of Hex is a combinatorial game, played (in its normal, two-player two-dimensional version) on a hexagonally-tiled board (such as the one shown on the left in Figure 2). Each player (player 1 represented in red, and player 2 represented in blue) starts in the possession of two opposing sides of the board; they take turns placing stones of their color on unoccupied tiles. Each has the goal of connecting their two sides with a path (or *bridge*) of stones of their color; the first to do so wins. For ease of representation, we imagine instead that the players are placing stones on the *vertices* (rather than facets) of the dual graph, which is represented on the right in Figure 2. In order to escape the restriction that the number of red and blue stones is the same (or at most differ by 1), we allow players to 'pass' (i.e. not put a stone down); there is no reason to do so if the player is trying to win, but it makes the following analysis simpler and more general.

Although seemingly just a simple combinatorial game, Hex is known to have very deep mathematical properties. First, a very elegant proof shows that some player is guaranteed to win [6], i.e. if every vertex (in the dual-graph representation) is occupied by a stone, then there must be a pair of opposing sides which are joined by a path of stones of their corresponding color. In fact, exactly one player must win (the fact that it's impossible for both players to have bridges at the same time is intuitively connected to the Jordan curve theorem). Intriguingly, the theorem that Hex must have a winner (which seems at first to be merely an interesting curiosity) can, like Sperner's Lemma, be used to actually prove Brouwer's fixed-point theorem [6].

3 Brouwer vs. Jordan in TFNP

In this section, we consider the relationship between Brouwer's fixed-point theorem and the Jordan curve theorem. In particular, we want to show that Brouwer's fixed-point theorem can be proven directly from the Jordan curve theorem, thus complementing Maehara's result that the Jordan curve theorem can be proved directly from Brouwer's fixed-point theorem [10]. We then consider a computational version of the Jordan curve theorem, which we call ZeroSurfaceCrossing; this problem is a search problem where the existence of a solution is guaranteed by the Jordan curve theorem. We then show that ZeroSurfaceCrossing is PPAD-complete. This makes it equivalent to the computational problem of finding a fixed point of a function (where the existence of a solution is guaranteed by Brouwer's fixed-point theorem), thus demonstrating a computational link between the two theorems in addition to the mathematical link.

3.1 The Zero Surface Crossing Problem

We define a problem where the existence of a solution is intended to be guaranteed by Jordan.

▶ **Definition 9** (Zero Surface Crossing). We are given continuous functions $f_1, f_2 : [0,1]^2 \to [-1,1]$ (which are therefore 2-dimensional surfaces in $\mathbb{R}^3$) satisfying the following conditions:
1. $f_1(0,y) \geq 0$ and $f_1(1,y) \leq 0$ for all y;
2. $f_2(x,0) \geq 0$ and $f_2(x,1) \leq 0$ for all x.

The goal is to find some $(x,y) \in [0,1]^2$ such that $f_1(x,y) = f_2(x,y) = 0$.

We wish to show that the Jordan curve theorem implies that such a point (x,y) exists; in order to give this proof, we consider the computational version defined in the introduction as ZeroSurfaceCrossing.

We will first show that the Jordan curve theorem implies that the computational version is guaranteed to have a valid output; we will then use basic topological principles to show that the mathematical version given in Definition 9 also must have a solution. We need the following notation:
1. let $X_0 = \{(0,y) : 0 \leq y \leq 1\}$ and $X_1 = \{(1,y) : 0 \leq y \leq 1\}$;
2. let $Y_0 = \{(x,0) : 0 \leq x \leq 1\}$ and $Y_1 = \{(x,1) : 0 \leq x \leq 1\}$.

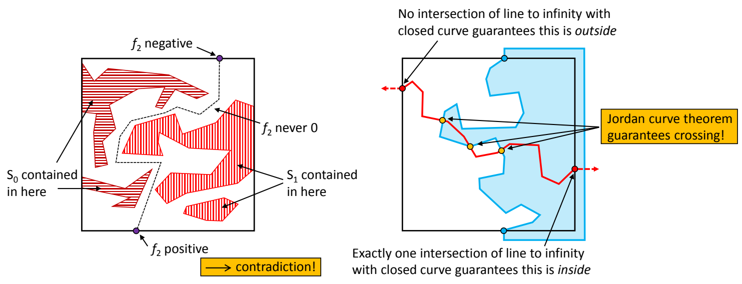
f_2 negative
f_2 never 0
S_0 contained in here
S_1 contained in here
f_2 positive
⟶ contradiction!
No intersection of line to infinity with closed curve guarantees this is *outside*
Jordan curve theorem guarantees crossing!
Exactly one intersection of line to infinity with closed curve guarantees this is *inside*

Figure 3 *Left:* illustration of proof of assertion 1; *Right:* illustration of proof of assertion 2.

In short, these sets are the four sides of the square $[0, 1]^2$. For these lemmas, we sketch the proofs; the full formal proofs are given in the full version of the paper [1].

▶ **Lemma 10** (Jordan to Computational ZeroSurfaceCrossing)**.** *The Jordan curve theorem implies that the computational version of* ZeroSurfaceCrossing *has a valid output.*

Proof Sketch. We assume that the boundary conditions hold; we thus want to show the existence of a point $(x, y) \in [0, 1]^2$ such that $f_1(x, y) = f_2(x, y) = 0$. We will show the following:

1. there exists a path from X_0 to X_1 such that at every point on the path, f_2 has value 0; by symmetry, there then must also be a path from Y_0 to Y_1 such that at every point, f_1 has value 0.
2. the Jordan curve theorem then implies that these two paths must cross, giving a point $(x, y) \in [0, 1]^2$ such that $f_1(x, y) = f_2(x, y) = 0$.

The proofs are depicted in Figure 3 (assertion 1 on the left, assertion 2 on the right). We remark that assertion 1 only holds because of the computational setting, where the function is defined by a circuit and the interpolation procedure (see full proof for details). ◀

We can now use this to tackle the problem of showing the existence of a solution to the mathematical version as given in Definition 9.

▶ **Proposition 11** (Jordan to Non-Computational Zero Surface Crossing)**.** *The Jordan curve theorem implies that the mathematical version of Zero Surface Crossing always has a solution.*

Proof Sketch. We use the following general strategy: we note that only the set of points where f_1 is 0 and the set of points where f_2 is 0 matters. Thus, we can use instead two functions f_1^* and f_2^* which are 0 at the same points, but which are Lipschitz. We can then show that for any $\epsilon > 0$, if we approximate f_1^* and f_2^* with circuits F_1, F_2 with sufficiently many bits, any point which is a zero of the circuits must be within ϵ of 0 for f_1^*, f_2^*. We then take a sequence of such approximate zeroes as $\epsilon \to 0$; by compactness of $[0, 1]^2$, this sequence must have at least one limit point, which we can then show is a shared zero of f_1^*, f_2^* and hence of f_1 and f_2. ◀

Finally, we can use this result to prove Brouwer as a consequence of Jordan.

▶ **Theorem 12** (Jordan implies Brouwer)**.** *Brouwer's fixed-point theorem can be shown as a direct consequence of the Jordan curve theorem.*

Proof. Given any mapping $g : [0,1]^2 \to [0,1]^2$, we wish to show the existence of a fixed point (x, y) such that $g(x, y) = (x, y)$. Let g_x and g_y be the x- and y-components of g respectively.

We then define the functions $f_1, f_2 : [0,1]^2 \to [-1,1]$ as follows: $f_1(x, y) = g_x(x, y) - x$ and $f_2(x, y) = g_y(x, y) - y$. First, we note that the outputs of f_1, f_2 indeed must fall in $[-1, 1]$; this is because g_x and g_y have outputs in $[0, 1]$ and $-x$ and $-y$ each range through $[-1, 0]$. We also note that f_1, f_2 satisfy the boundary conditions given in Definition 9, since $f_1(0, y) = g_x(0, y) \geq 0$ and $f_1(1, y) = g_x(1, y) - 1 \leq 0$ for all y, and similarly $f_2(x, 0) = g_y(x, 0) \geq 0$ and $f_2(x, 1) = g_y(x, 1) - 1 \leq 0$ for all x. Thus, we can apply Proposition 11 to show that there is some (x, y) such that $f_1(x, y) = f_2(x, y) = 0$.

But this implies that $g_x(x, y) = f_1(x, y) + x = x$ and $g_y(x, y) = f_2(x, y) + y = y$, i.e. that $g(x, y) = (x, y)$, thus proving Brouwer's fixed-point theorem as a consequence of Zero Surface Crossing and hence as a consequence of the Jordan curve theorem. ◀

This, along with Maehara's result on showing Jordan from Brouwer [10] and Gale's result on the equivalence of Brouwer and the Hex theorem [6], thus concludes the proof of Proposition 2 (that Brouwer, Jordan, and Hex are all mathematically equivalent theorems). We also remark that reversing the above reduction yields an alternative proof that Brouwer's fixed-point theorem can be used to prove the Jordan curve theorem; however, as this is a known result, we will not show this in detail.

3.2 Computational Equivalence of Brouwer and Jordan

We now want to show that ZeroSurfaceCrossing is PPAD-complete. To do this, we first define the computational version of Brouwer, which is well-known to be PPAD-complete [11]:

Brouwer
`Input:` A circuit $G : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$, defining a mapping $g : [0,1]^2 \to [0,1]^2$ via interpolation (as described in Appendix A).

`Output:` a point (x, y) such that $g(x, y) = (x, y)$ (i.e. a fixed point of g).

▶ **Lemma 13.** ZeroSurfaceCrossing *is PPAD-hard.*

Proof. We prove this by showing that Brouwer can be reduced to it, using the same reduction as in the proof of Theorem 12. In particular, we note that this reduction requires no fudging with outputs, as if (x, y) is a lattice point (where x and y can each be expressed by an n-bit string) then $f_1(x, y) = g_x(x, y) - x$ can be expressed by an $(n+1)$-bit string, since the output $g_x(x, y)$ is an n-bit string as well; the same obviously holds for $f_2(x, y) = g_y(x, y) - y$. We then note that since all lattice points behave well under the transformation, and the interpolation given in Appendix A is linear in both cases, the reduction requires no further steps; a solution to the derived ZeroSurfaceCrossing instance is immediately a fixed point of the original Brouwer instance. ◀

▶ **Lemma 14.** ZeroSurfaceCrossing *is in PPAD.*

Proof. We show this by reducing ZeroSurfaceCrossing to Brouwer. Without loss of generality, we assume that f_1 and f_2 take two n-bit strings as input and output two $(n+1)$-bit strings (we add the extra bit to account for the fact that they can have negative output, so as to keep the interval size constant). We can now define $g : [0,1]^2 \to [0,1]^2$ as follows; this is essentially the above reduction, reversed and with outputs truncated to be within $[0, 1]$:

$$g_x(x, y) = \max\big[\min[x + f_1(x, y), 1], 0\big], \text{ and } g_y(x, y) = \max\big[\min[y + f_2(x, y), 1], 0\big].$$

As before, since the reduction holds exactly at lattice points, it holds exactly everywhere else as well (by how the interpolation works). Furthermore, it trivially has a fixed point at any (x, y) such that $f_1(x, y) = f_2(x, y) = 0$. We now wish to show that any *additional* fixed points can only be a result of f_1 or f_2 violating a boundary condition (which we recally is an acceptable output to ZeroSurfaceCrossing).

The only way g could have another fixed point is if the capping of g_x or g_y to be between 0 and 1 held the displacement to 0 when otherwise it would have been nonzero. This happens only if the input is already on the boundary (otherwise the cap cannot completely remove a nonzero displacement in any direction); hence, it can only happen if one of the following four events happens: (a) $f_1(0, y) < 0$; (b) $f_1(1, y) > 0$; (c) $f_2(x, 0) < 0$; or (d) $f_2(x, 1) > 0$, which all represent violations of the boundary conditions set by ZeroSurfaceCrossing. Hence, any fixed point of g corresponds to a solution to ZeroSurfaceCrossing, which means that ZeroSurfaceCrossing can be reduced to Brouwer, and so it is in PPAD. ◀

Lemmas 13 and 14 thus imply Theorem 1 (ZeroSurfaceCrossing is PPAD-complete).

4 Crossing Curves

We now discuss the CrossingCurves problem; in particular, we show that it is in PPAD.

Proof of Theorem 3. We recall that the CrossingCurves problem involves two curves $f_1, f_2 : [0, 1] \to [0, 1]^2$ such that $f_1(0) = (0, 0)$, $f_1(1) = (1, 1)$, $f_2(0) = (0, 1)$ and $f_2(1) = (1, 0)$; these curves are formally defined by circuits which map n-bit strings to points on a discrete lattice in $[0, 1]^2$ (with the continuous curves defined by interpolation over these circuits). We denote f_1^x, f_1^y, f_2^x, and f_2^y as the functions describing f_1 and f_2's outputs in the dimensions x and y. We note that although $f_1^x(0) = f_2^x(0) = 0$ and $f_1^x(1) = f_2^x(1) = 1$, f_1^x and f_2^x are not necessarily monotonically increasing; the two curves can snake back and forth. Nevertheless, the Jordan curve theorem does guarantee that there will be a crossing point, i.e. a pair of *times* (if we interpret the input of f_1 and f_2 to be a time between 0 and 1) t_1, t_2 such that $f_1(t_1) = f_2(t_2)$. The task is to find the crossing point.

We do this by defining the function $g : [0, 1]^2 \to \mathbb{R}^2$ (where its x and y components are denoted g^x and g^y respectively) such that

$$g^x(t_1, t_2) = t_1 - f_1^x(t_1) + f_2^x(t_2) \text{ and } g^y(t_1, t_2) = t_2 - f_1^y(t_1) + f_2^y(t_2)\,.$$

Clearly this function is continuous, and (t_1, t_2) is a fixed point if and only if $f_1^x(t_1) = f_2^x(t_2)$ and $f_1^y(t_1) = f_2^y(t_2)$, i.e. if and only if $f_1(t_1) = f_2(t_2)$. This is already very close to showing what we need to show; the only trouble is that an application of g might end up at a point outside of $[0, 1]^2$, breaking our use of Brouwer. We thus define $\hat{g}$ to be g, but with upper and caps to its values at 1 and 0 respectively. Formally:

$$\hat{g}^x(t_1, t_2) = \max\left[\min[g^x(t_1, t_2), 1], 0\right] \text{ and } \hat{g}^y(t_1, t_2) = \max\left[\min[g^y(t_1, t_2), 1], 0\right].$$

We now have a function which does not leave $[0, 1]^2$ and has a fixed point at any t_1, t_2 such that $f_1(t_1) = f_2(t_2)$. The only trouble is we need to make sure we did not create any new fixed points by this capping method;

A new fixed point can only happen if one of the following four events happens:

1. $-f_1^x(t_1) + f_2^x(t_2) < 0$ when $t_1 = 0$;
2. $-f_1^x(t_1) + f_2^x(t_2) > 0$ when $t_1 = 1$;
3. $-f_1^y(t_1) + f_2^y(t_2) < 0$ when $t_2 = 0$;
4. $-f_1^y(t_1) + f_2^y(t_2) > 0$ when $t_2 = 1$.

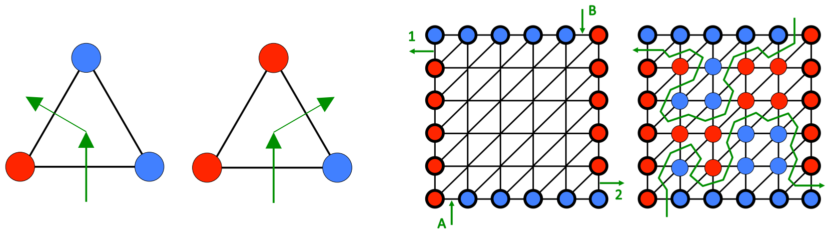

Figure 4 *Left:* proceeding on a Hex walk through a cell; each step is forced. *Right:* an empty board (with player-owned edges) has two entries, A and B, and two exits 1 and 2. If the walk entering at A exits through 2 (as shown at far right), then player 1 is the winner; if it exits through 1, then player 2 is the winner.

But we note that since $f_1^x(0) = 0$ and $f_1^x(1) = 1$ (and both f_1 and f_2 are constrained to be in $[0,1]$ in both components) that the first two cannot happen; furthermore, since $f_2^y(0) = 1$ and $f_2^y(1) = 0$, the second two cannot happen. Thus, a fixed point still occurs at (t_1, t_2) if *and only if* $f_1(t_1) = f_2(t_2)$, as we wanted. We can thus apply the Brouwer fixed-point algorithm to $\hat{g} : [0,1]^2 \to [0,1]^2$ and read off a solution to the crossing-curves problem. Since finding a Brouwer fixed-point is in PPAD [11], CrossingCurves is also in PPAD. ◀

5 Hex, Brouwer, and Jordan in PSPACE

We have already established that the Hex, Brouwer, and Jordan theorems are mathematically equivalent. We have also identified two problems motivated by Jordan's curve theorem that lie in PPAD. One of these is in fact PPAD-complete and thereby computationally equivalent to Brouwer. It is natural to ask whether a computational version of the Hex theorem is also related to PPAD. The link between PPAD and the Hex theorem is more immediate and striking than that between the Jordan curve theorem and PPAD. In particular, Gale's proof that Hex always has a winner [6] is strikingly similar to the proof of Sperner's lemma, another well-known topological fact giving rise to the PPAD-complete problem called Sperner. Gale's proof generates a PPAD type graph, as defined in Section 2.2. In the next sections, we briefly summarize Gale's results and discuss natural questions arising from it in relation to PPAD and, as it turns out, PSPACE.

5.1 Hex and PPAD

As with all PPAD-type problems, in Hex the proof of existence of a 'bridge' (a winning sequence for one of the two players) in a filled board comes with a pivoting algorithm to find it; this algorithm was described by Gale, who (remarkably) used it to show that Brouwer's fixed-point theorem and the Hex theorem are mathematically equivalent. The pivoting algorithm is briefly sketched in Figure 4 on the dual-graph representation of the Hex board (in which the stones are placed on vertices, rather than faces, of the graph). In brief, there are two places on the edge of the board where one can enter with red on their left and blue on their right, as shown in the figure; suppose one starts at A. Then, one walks over the faces of the dual graph (or, equivalently, on the vertices of the original board), keeping red on their left and blue on their right, until they exit the board at either 1 or 2. Exiting at 2 implies a victory for player 1 (red), and exiting at 1 implies a loss. This pivoting algorithm is especially reminiscent of the Sperner's Lemma pivoting algorithm [11], and induces a (directed) PPAD graph with nodes corresponding to the faces of the dual-graph board.

Even though Gale's proof induces a PPAD graph, there is no natural analogue of the EndOfTheLine problem for Hex. This is because the PPAD graph induced by Gale's argument only has four unbalanced vertices, corresponding to the entries A and B and the exits 1 and 2. The natural question is which pairs of unbalanced vertices are connected, which is equivalent to asking *"who won?"* after a game of Hex has filled the board. This corresponds to finding the *specific* unbalanced vertex of the PPAD graph which is connected to the starting one, corresponding to entry A. Of course, this would be polynomial-time solvable if the board were polynomially-sized, so we will assume an *exponentially-large* board where a polynomially-sized circuit tells us who claimed any particular tile.

5.2 WhoWonHex

WhoWonHex

Input: a Hex board, such that it takes n bits to uniquely specify a tile (the canonical way to do this is via the dual graph, by having $n/2$ bits dedicated to specifying the position of the tile in each dimension); a circuit C which takes an n-bit string (i.e. a tile) and outputs either 0 ('player 1 does not occupy this tile') or 1 ('player 1 does occupy this tile').

Output: 0 if there does not exist a bridge of tiles occupied by player 1 from the left facet to the right facet ('player 1 has not won'), and 1 if there does exist such a bridge ('player 1 has won'). Formally, with the canonical tile-specifying scheme described above, a 'bridge' is defined as a sequence of tiles $v_1, v_2, \ldots, v_m$ such that: (a) v_i is adjacent to v_{i+1} for all $i = 1, 2, \ldots, m-1$; (b) $C(v_i) = 1$ for all i; (c) the first $n/2$ bits of v_1 are all 0 (it's adjacent to the left facet), and the first $n/2$ bits of v_m are all 1 (it's adjacent to the right facet).

We want to show that WhoWonHex is PSPACE-complete (Theorem 4). It is tempting to try to prove this hardness via a reduction from the so-called OtherEndOfThisLine problem,[1] which is PSPACE-hard [8], by simulating paths in a given PPAD graph via "Gale paths." However, Gale paths do not cross, and the usual embedding of a PPAD graph into a 2-dimensional plane without crossings results in a drastic change of the graph topology (unbalanced vertices remain unbalanced vertices, but which unbalanced vertices are connected to each other changes) [2, 7]. Instead, we will obtain our result as a direct consequence of Goldberg's proof that it is PSPACE-hard to identify the solution output by a path-following algorithm for Sperner [7].

Goldberg's proof proceeds directly from the well-known PSPACE-complete problem of evaluating a *quantified Boolean formula* (QBF). Since our proof is an application of Goldberg's proof, we only explain how to make the gadgets required to modify his proof for our purposes.

Proof of Theorem 4. As described above, we will be adapting Goldberg's proof that identifying the solution chosen by the standard pivoting algorithm for Sperner is PSPACE-hard. (The inclusion into PSPACE is easy to establish; the details are given in the full version of our paper [1])). Given a QBF instance Goldberg defines horizontal and vertical wires in an exponentially large square lattice in such a way that the existence of a connected path of

[1] This problem is: given a PPAD graph via circuits P, N, identify the unbalanced vertex connected to 0^n, if 0^n is itself unbalanced.

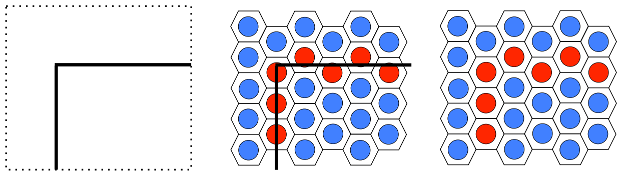

Figure 5 *Left:* A wire occupying part of the space. *Middle:* The wire, superimposed on the Hex board, with the appropriate tiles turned to red to carry the wire. *Right:* The wire, represented only as a sequence of red Hex tiles.

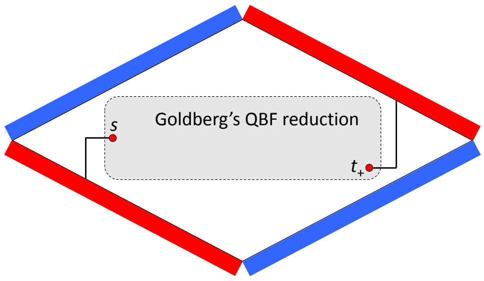

Figure 6 Embedding Goldberg's rectangular reduction into a Hex board, so that player 1 wins if and only if s is connected to t_+.

wires from a special point s to a special point t_+ (which themselves always lie on wires but may not be connected through wires) is equivalent to the QBF instance evaluating to 1. He also provides an efficient algorithm to determine if a wire passes through a specific point, and then simulates wires with paths in the PPAD graph of a SPERNER instance.

Our result will be based on a 'wire' gadget that is based on Hex rather than Sperner – which is simply a connected path of red tiles 'insulated' from other red tiles by blue tiles (in fact, all tiles not on a 'wire' gadget are defined to be occupied by blue stones, i.e. player 2). For simplicity, we use only wires whose segments are perfectly vertical or horizontal; how a wire translates into tiles is shown in Figure 5.

Usefully the special points s and t_+ in Goldberg's construction lie on the left and right boundary of the square lattice respectively. This allows us to embed Goldberg's construction into a Hex board as shown in Figure 6. In particular:

- we allocate a part of the board where we replicate Goldberg's construction using our wire gadget, as described above given a QBF instance;
- we add wires connecting s to the bottom-left edge of the board;
- we add wires connecting t_+ to the up-right edge of the board.

Since s and t_+ lie on wires they are red tiles.

Since we do not have any wires not defined by Goldberg's construction (other than those from s and t_+ to the boundary of the Hex board shown in Figure 6), there is a bridge connecting the red edges of the Hex board if and only if s is connected to t_+ by wires inside Goldberg's construction. This is PSPACE-hard to determine, hence it is PSPACE-hard to determine if Player 1 wins. ◀

We remark that this proof had to be derived from Goldberg's *construction* itself, and not his result in a black-box manner. This is because the construction guarantees that the end-points s and t_+ in the proof above lie on the boundaries of the construction (and thus can be wired to the bottom-left and upper-right edges of the Hex board).

In contrast, general SPERNER instances may very well have the solution reachable through the standard path-following algorithm residing in the interior of the construction (thereby making it hard to translate into Hex, since the question in Hex is whether there is a bridge of red stones joining the left and right edges of the board).

6 Conclusion

In this paper, we explored the links between Brouwer's fixed-point theorem, the Jordan curve theorem, and the game of Hex. We showed that Brouwer and Jordan are mathematically and computationally equivalent, complementing Maehara's result that Jordan is a consequence of Brouwer. Combined with Gale's result that Brouwer's fixed-point theorem is mathematically equivalent to the seemingly innocuous Hex theorem (that a completed game of Hex must have a winner), our result implies that all three theorems, Brouwer, Jordan and Hex, are mathematically equivalent. Within PPAD, we defined two computational problems, CURVECROSSING and ZEROSURFACECROSSING, which always have solutions by dint of the Jordan curve theorem. We show that both lie in PPAD, and the second is also PPAD-complete. Finally, we relate the Hex theorem to results in the literature pertaining to the complexity of standard pivoting algorithms for ENDOFTHELINE, BROUWER and SPERNER. It has been shown that identifying the solution computed by standard pivoting algorithms for these problems is PSPACE-complete, and we show that the problem WHOWONHEX, of determining who is the winner in a play of Hex, is also PSPACE-complete. We thereby establish computational relations among Brouwer, Hex and Jordan both within PPAD and at the level of PSPACE. The main problem left open by our work is the complexity of CURVECROSSING. Is it PPAD-complete? We discuss structural properties of instances of CURVECROSSING that make us believe that the problem could lie lower in TFNP. It would be interesting to identify a potential function argument guaranteeing a solution to this problem, thereby placing it in the intersection of PLS and PPAD, and potentially one of the classes defined in [4].

References

1 Aviv Adler, Constantinos Daskalakis, and Erik Demaine. The Complexity of Hex and the Jordan Curve Theorem. *Arxiv*, 2016.
2 Xi Chen and Xiaotie Deng. On the Complexity of 2D Discrete Fixed Point Problem. In *the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2006.
3 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. In *the 38th Annual ACM Symposium on Theory of Computing (STOC)*, 2006.
4 Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.
5 Kousha Etessami and Mihalis Yannakakis. On the Complexity of Nash Equilibria and Other Fixed Points (Extended Abstract). In *the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.
6 David Gale. The game of Hex and the Brouwer fixed-point theorem. *American Mathematical Monthly*, pages 818–827, 1979.
7 Paul Goldberg. The Complexity of the Path-following Solutions of Two-dimensional Sperner/Brouwer Functions. *arXiv*, 2015.
8 Paul W Goldberg, Christos H Papadimitriou, and Rahul Savani. The Complexity of the Homotopy Method, Equilibrium Selection, and Lemke-Howson Solutions. *ACM Transactions on Economics and Computation*, 1(2):9, 2013.

9 Camille Jordan. *Cours d'analyse de l'École polytechnique*, volume 1. Gauthier-Villars et fils, 1893.

10 Ryuji Maehara. The Jordan curve theorem via the Brouwer fixed point theorem. *American Mathematical Monthly*, pages 641–643, 1984.

11 Christos H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

A Interpolations of Functions on the Unit Square

In order to develop computational problems relating to our theorems of interest, we need to give a formal definition of how circuits can represent functions on the unit square. We will consider two types of functions we wish to represent: $f : [0,1]^2 \to [-1,1]$, and $g : [0,1]^2 \to [0,1]^2$. They are represented by the following circuits:

1. f is represented by a circuit F which takes two n-bit strings as input (representing a point in $[0,1]^2$) and returns an m-bit string (representing the value of the function);
2. g is represented by a circuit G which takes two n-bit strings as input and returns two new n-bit strings.

These circuits directly define the values of their respective functions at lattice points in $[0,1]^2$, namely points whose coordinates are integer multiples of $1/(2^n - 1)$; for the values of f and g at points which are not on this lattice, we use the triangulations depicted in Figure 7. For f, we take the output at lattice points (which is directly given by F) and use them to generate a 'mesh' which defines f at points not on the lattice; for g, we make a similar construction as shown at center and right in the figure.

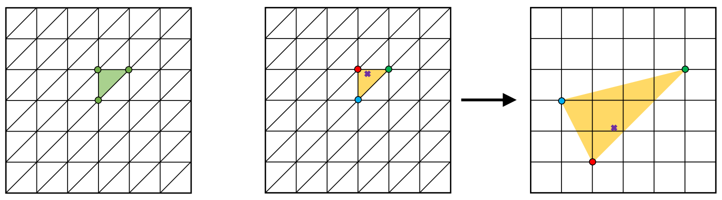

Figure 7 *Left:* the value of f at the three marked lattice points uniquely determines a plane; for input points in the triangle shared by the three, the output of f is consistent with this plane. *Center and Right:* the marked lattice points at right are the outputs of g at the marked lattice points at center; inputs in the triangle shared by them are mapped to the corresponding point in the triangle shared by their outputs (for example, the purple 'x' at center is mapped to the purple 'x' at right).