Cooperating in Video Games? Impossible! Undecidability of Team Multiplayer Games

Michael J. Coulombe
MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge, MA 02139, USA
mcoulomb@mit.edu

Jayson Lynch
MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge, MA 02139, USA
jaysonl@mit.edu

Abstract

We show the undecidability of whether a team has a forced win in a number of well known video games including: Team Fortress 2, Super Smash Brothers: Brawl, and Mario Kart.To do so, we give a simplification of the Team Computation Game [7] and use that to give an undecidable abstract game on graphs. This graph game framework better captures the geometry and common constraints in many games and is thus a powerful tool for showing their computational complexity.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases computational complexity, undecidable, team games, imperfect information

Digital Object Identifier 10.4230/LIPIcs.FUN.2018.14

1 Introduction

Multiplayer videogames account for a large portion of the video game market and yet the additional computational complexity added by coordinating different team members has not seen much study from a theoretical standpoint. We finally bridge the gap between known theoretical models where imperfect information team games are known to be much more computationally complex and popular, commonly played video games.

In a series of papers [8–11], Reif and Peterson explored the computational complexity of games of imperfect information. One surprising result was a proof that unbounded team multiplayer games with imperfect information can be undecidable, despite having a bounded configuration space in the game itself. This work has been expanded to include formula and constraint logic games [7]; however, to the best of our knowledge, no commonly played game has been shown to be undecidable using this framework.

The computational complexity of video games has started becoming a popular topic of inquiry. Past research includes the study of classic arcade games like Pac-Man [13], classic Nintendo games such as Mario and the Legend of Zelda [1], to more modern games like Candy Crush [5], Portal [4], Angry Birds [12], and Braid [6]. However, all of these papers considered single-player, perfect information versions of the game. These are both aspects that intuitively and theoretically should make the games much more computationally challenging. This paper critically utilizes these properties to show far stronger hardness results than usually appears. We are aware of only one other video game, Braid, which has been shown to be undecidable. However, it does so by the construction of a counter machine using enemy units


9th International Conference on Fun with Algorithms (FUN 2018).
Editors: Hiro Ito, Stefano Leonardi, Linda Pagli, and Giuseppe Prencipe; Article No. 14; pp. 14:1–14:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and thus playing such a level will require unbounded computational resources. The ability for a bounded game state to be able to lead to an undecidable problem has been remarked on by others are a fascinating feature of this type of problem [7].

In addition, much of the past work on video games has focused on environmental obstacles such as toggles for moving platforms and locking doors, rather than more central mechanics of the game. An aesthetic advantage of our proofs are that they focus on player vs player interaction and use the central combat mechanics of the game as core elements in the reduction.

Organization

This paper is organized into two parts. The first half deals with abstract games and builds a framework for later reductions. In particular, Section 2 details the gadgets involved in our team multiplayer graph game. Section 3 reduces the TEAM COMPUTATION GAME to the TEAM GRAPH GAME using our simplification of the former, the TEAM DFA GAME, with further details in Appendix A. The second half, Section 4, applies this framework to show the undecidability of several popular multiplayer games.

2 Team Graph Game Components

In this section we describe the different components of our undecidability framework which will be instantiated in the TEAM GRAPH GAME which we define and show to be undecidable in Section 3. Roughly speaking, it is a multi-player game with two teams, which we will refer to as blue and red, on a graph where each team wants to get one of their players to one of the win nodes. Players take time moving from node to node and from a node other nodes may be visible, allowing the player to determine if another player is there. In addition, some nodes will allow a player to guard an edge. A player attempting to cross a guarded edge will be eliminated and no longer be able to perform any useful actions. In our reduction we want to simulate a DFA which takes input from blue and red players and changes state based on this input. The state of the DFA will be encoded in the location of one player on the blue team, called the runner, and we call the other blue team members executors. The DFA entering an accept state will correspond to the runner being on a path which leads freely to a win node. The red team will supply their inputs by guarding some of the possible paths of the executors, while the executors will provide the blue team's inputs by choosing among unguarded paths to take. Both teams' inputs will force the runner to take a certain path through the region representing the DFA transition function. This section of the paper will describe these gadgets and their function in detail and Section 3 will formalize and complete the proof.

We break this framework down into several important gadgets each given their own subsection. We require a state transition gadget to manage the state of a deterministic finite automaton. This is described in Subsection 2.3. Both teams need to set variables which are taken as input to the DFA which is done with the choice gadgets described in Subsection 2.2. We need to synchronize all of the players so that the variable choices and DFA execution all occur in the proper order. This is done with the delay gadget described in Subsection 2.1. Finally, there is an optional initializer gadget which forces players from initial locations to the pathways needed in the gadgets. This is described in Subsection 2.4. These gadgets are put together in Section 3, as shown in Figure 5.

In this paper we use the following diagram conventions. Edges and nodes in the graph potentially containing red Team players are red and use square for nodes. Edges and nodes potentially containing blue Team players except for the runner are blue with circles as nodes.

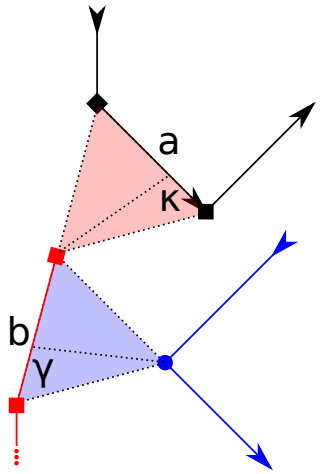

Figure 1 Gadget to delay the runner until a blue executor arrives to remove the red attacker.

Edges and nodes potentially containing the runner are black with diamonds for nodes. The graph contains both directed and undirected edges. Bold edges represent many different paths which serve similar function but are only accessed by one player. They are often accompanied by a label of how many edges are represented. Triple dots denote the continuation of a pattern, often many of the same type of edge. In contrast to bold edges, a different player will generally occupy each of these. Combat zones are pairs of nodes and edges and are denoted by a lightly colored red or blue triangle. The color dictates which team is posing a threat in the combat and always involves a node guarding an edge. If relevant, the combat zone is labeled with the length of time an enemy must spend traversing a guarded edge to be eliminated. These zones also imply visibility; however, we do not explicitly label visibility in all of our diagrams. Labeled boxes are used to refer to unrepresented gadgets, and dotted boxes are used to delineate different gadgets whose internal details are in the figure. An encircled W is a win node. Other labels and notation will hopefully be clear from context. Some of these conventions are used more liberally in the diagrams in Section 4 along side more representative pictures for the games.

2.1 Delay Gadget

The simplest gadget is the Delay Gate, as seen in Figure 1. The blue runner moves through the maze and is frequently blocked from making progress by a red player guarding a combat zone (edge) from an attack node. To progress, one of the blue executors must arrive at its own attack node which threatens the red guard, who must escape outside the combat zone (and far from its attack node) or be eliminated. As long as the red-beats-blue time $\kappa < a$ and the blue-beats-red time $\gamma < b$, the delay gadget achieves this goal.

2.2 Red Team Choice Gadget

The Red Team Choice Gadget gives the red team the ability to influence the path of a blue team player's movement. Detailed in Figure 2, a blue team member starts at node v_b and wants to exit out of v_0' or v_1', and a red team chooser at u_r (or its neighbors) will be able to force the outcome without fully preventing progress.

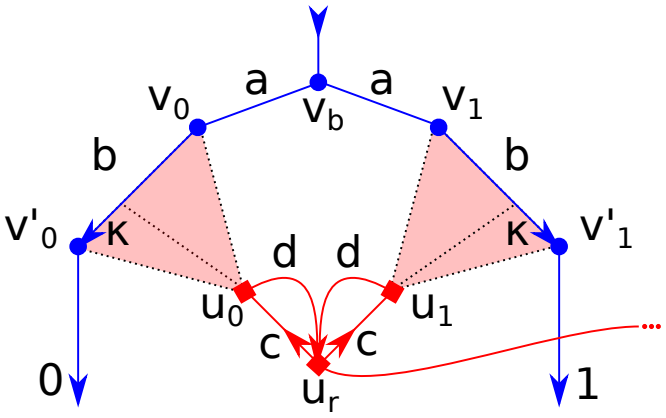

Figure 2 Gadget for a red player to force a blue player to take exit 0 or 1.

The graph is symmetric, so suppose without loss of generality that the red chooser wants the blue player to exit out of v'_1. Given their choice of where to start among the subgraph $\{u_0, u_r, u_1\}$, they can successfully block the v'_0 exit by simply waiting at u_0 and attacking if the blue player tries to traverse edge (v_0, v'_0). If $c > a + b$, no starting location of the red chooser allows them to prevent the blue player from reaching both exits: the red chooser must start at least $d = a + b - \kappa$ time units away from u_0 to block v'_0, which means starting $c + (c - d) > a + b$ away from u_1 which is too far to block v'_1 as well.

An optimal strategy for the blue player to guarantee progress is thus to immediately move towards v'_0. Either the red chooser is blocking v'_1 and the blue player will leave through the preferred exit, or red chooser is blocking v'_0 and the blue player will have time to turn around and reach v'_1 (the preferred exit) before the red chooser can reach u_1.

2.3 State Transition Gadget

Whereas the Red Team Choice Gadget is used to allow red team to influence a blue executor's path, the State Gate gadget is used to allow blue team executors to influence the blue runner's path. The "core" of a State Gate is essentially two Delay Gates sharing the same red guard who, unlike the Red Team Choice Gadget, is able to simultaneously block both exits for the blue runner. Depending on which of the two paths the blue executor is on, it will be able to safely open one of two exit paths for the blue runner.

Looking ahead to our undecidability proof for TGG, we generalize the core into a State Gate by first allowing for two independent hallways per blue executor "input" and second to allow for multiple independent hallways for the blue runner. Detailed in Figure 3, the first can be constructed using two cores (each with one hallway of each "input" type) or with one core modified such that the red guard's edges are the target of two blue executor attack nodes at once. The second generalization is simply constructed using multiple instances of the first in series along the blue executor's paths, one per required blue runner hallway.

The core works correctly as long as the red guard has visibility on the blue runner and executor and $\gamma < b < a - \kappa$. When safe, the red guard can mimic the blue runner's movement and always reach the closer attack node fast enough to block the path, but when the blue executor arrives on one side, the red guard must vacate the corresponding attack zone and can only safely block the opposite path. Thus, the blue runner strategy of repeatedly attempting to go in either direction until the red guard stops following to block will allow for guaranteed safe passage without visibility between the two blue team players. As a side note, the core

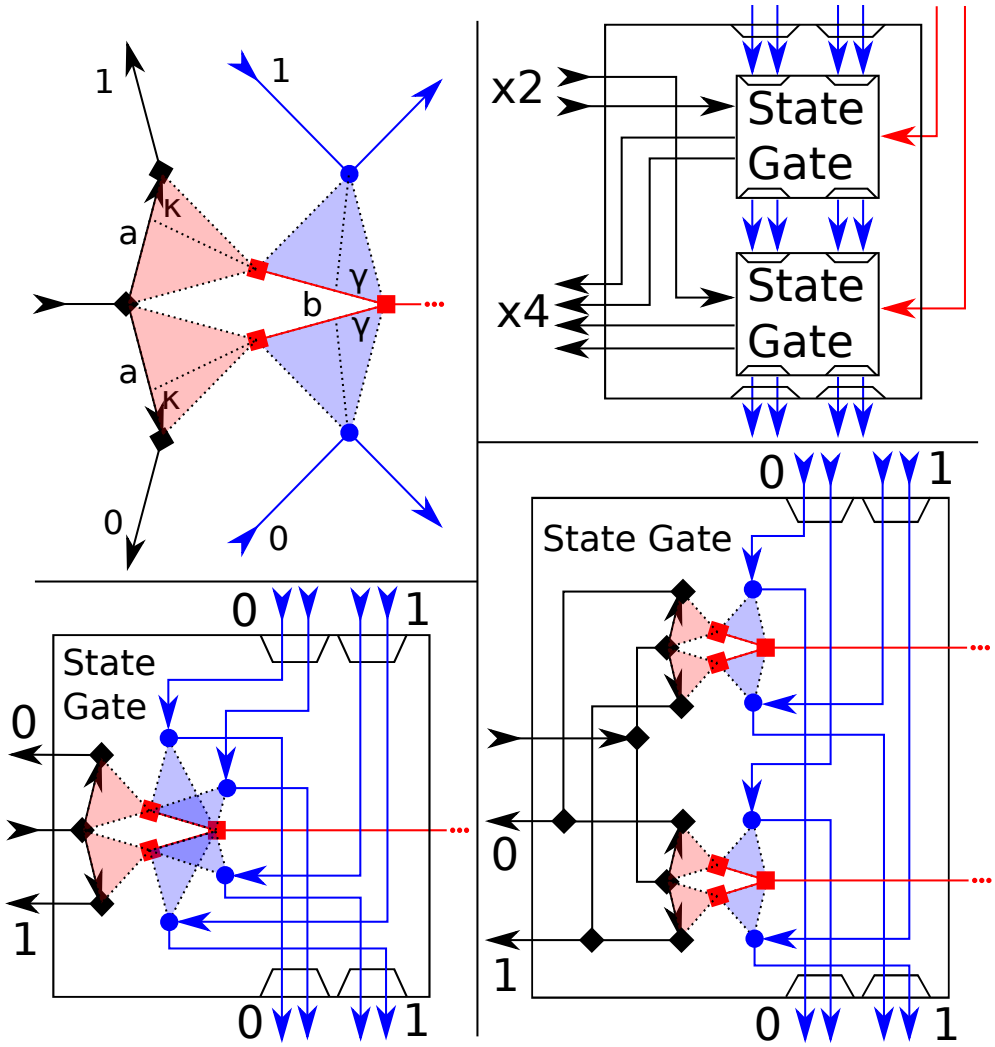

Figure 3 "State Gate" gadget schema for a blue executor to branch the blue runner. The core of player interaction (top-left) is generalized first allowing two blue paths per input (two possible constructions on bottom) then allowing multiple runner paths (top-right).

could also be implemented with two separate, unmodified Delay Gates, thus using two red guards instead of one but having no additional timing constraints.

2.4 Initialization

In many games we are modeling with TGG, all players on each team start in their team's single spawn room. In order to force the team members into separate hallways, they are coerced into guarding a set of paths, one per player (besides the runner), which all lead to the victory node w. Figure 4 shows the initializer gadget with spawn nodes s_b or s_r, where first blue must split into three hallways to block any red players from reaching w and force the red players to make progress and split up in order to block the blue runner from reaching w.

Specifically, to incentivize the blue team to fully split up, two red team "win paths" are placed and each guarded by a series of n_r blue attack zones of length $b_2 > \gamma$, so that even if the red team sends all of its players down one win path, the defending blue player could

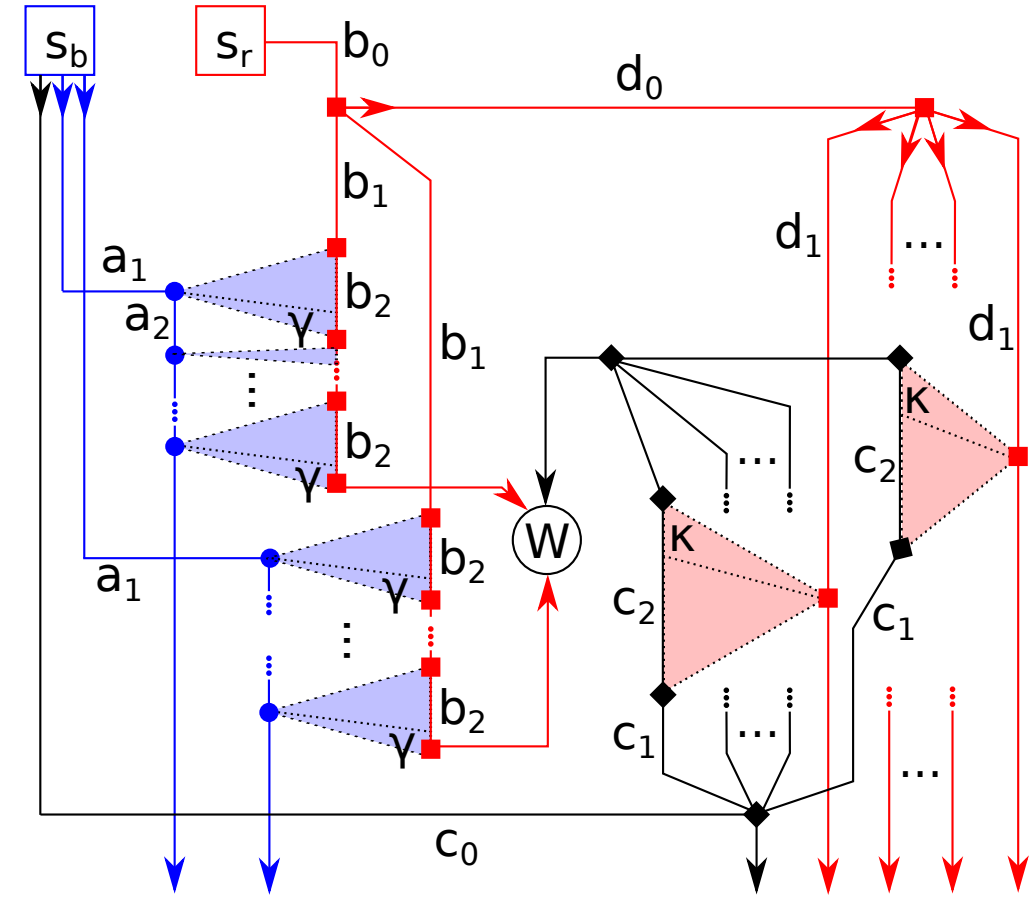

Figure 4 Initializer Gadget to separate players that must start together in team spawn rooms.

eliminate all of them by the end. If the blue team tries to send multiple players out the same hallway from s_b, they will either allow red team to win through the other win path in the initializer gadget, or have no player in the blue runner path, which is designed in our undecidability construction to be the only path to w.

If blue team does split up and guard the red team win paths, then red team must then prevent the blue runner from reaching w by going down a third path that splits into n_r branches, each responsible for guarding a different path for the blue runner. This forces the red team to separate and block every path until the blue runner gives up and exits the Initializer Gadget, at which point all other now-separated players can safely exit as well.

The constraints on the Initializer Gadget are light beyond the need for visibility so each player can learn when it is safe to stop guarding an attack zone and make progress. No information needs to be private at this point so full visibility is allowed within the gadget, although a set of hallways at the exit for the blue runner to pass within visibility range of every other player would be a sufficient signal for games being modeled by TGG with occlusion or view distance constraints. For the blue players to have time to block the red players, the attack nodes should be close enough together such that $\forall i \in [0, n_r) : a_1 + ia_2 < b_0 + b_1 + (i+1)(b_2 - \gamma)$. So that the red players have time to block the blue runner, it must be that $b_0 + d_0 + d_1 < c_0 + c_1 + c_2 - \kappa$.

3 Reductions

The TEAM COMPUTATION GAME (TCG), as defined in [3], is a game about two teams ($\exists$ and $\forall$) whose players alternate writing symbols onto certain cells of a finite-length tape of a Turing machine, which takes a fixed number of steps during each round and if it halts then the game ends and one team wins based on whether it accepts or rejects. A simplifying insight is that this Turing machine is effectively a DFA that teams are alternatively feeding input symbols into until it ends up in a final state that determines which team wins. The following modified definition will use this terminology instead for the purposes of the later reduction. Reductions establishing the equivalence of TDA with TCG and thus its undecidability can be found in Appendix A.

▶ **Definition 1.** The TEAM DFA GAME (TDG) is a two-versus-one team game. An instance of the game is a DFA $D = (\Sigma = \{0, 1\}, Q, q_0, \delta, F = F_\exists \Delta F_\forall)$. The existential team $\{\exists_1, \exists_2\}$ competes against the universal team $\{\forall\}$. The game starts with D in state q_0 and each round proceeds as follows:

1. If D's state $q \in F_\exists$ then team existential wins. If $q \in F_\forall$ then team universal wins.
2. $\forall$ learns the state q of D then inputs two bits b_1, b_2 into D.
3. $\exists_1$ learns b_1 then inputs one bit m_1 into D. $\forall$ learns m_1.
4. $\exists_2$ learns b_2 then inputs one bit m_2 into D. $\forall$ learns m_2.

We now go on to define the TEAM GRAPH GAME and show it is undecidable by a reduction from the TEAM DFA GAME.

▶ **Definition 2.** The TEAM GRAPH GAME is a team multiplayer game. Let the TGG of red team vs blue team consist of:

- Directed Graph $G = (V, E)$ with edge weights $\in \mathbb{N}$
- Designated team start nodes $s_r, s_b \in V$ and win node $w \in V$
- Directed visibility relation $S \subseteq V^2$
- (Uni)Directed attack relation $A \subseteq V^2$
- Initial number of players per team $n_r, n_b \in \mathbb{N}$

The execution of the TEAM GRAPH GAME starts with n_r red player tokens at node s_r and n_b blue player tokens at node s_b. Blue team wins if either every red token is eliminated or any blue token reaches the node w. Red team wins similarly.

The game proceeds as a sequence of time steps, or frames. Each frame, all active players simultaneously commit to their action and then all effects are triggered and handled before the frame ends. The action of a player consists of a node $n \in N[v]$ to move towards (or none to signify not moving). Once players have performed their moves, each player whose token can "see" another player's token learns of said token's position and team. Visibility zones are defined at nodes by S and on edges by union of the visibilities of the endpoints; combat zones are defined similarly.

▶ **Theorem 3.** *TDG reduces to the TEAM GRAPH GAME (TGG). Namely, $\exists h : \langle D \rangle \mapsto \langle I \rangle$ which maps instances $\langle D \rangle$ of TDG and instances $\langle I \rangle$ of TGG such that the existential team has a forced win in the TDG on D iff the blue team has a forced win in TGG on I.*

Proof. Figure 5 gives an overview of the structure of $I = h(D)$. Once the initializer gadget distributes each blue and red player into their proper hallways, each loop of the blue team in the graph simulates one round of TDG. The universal team's decisions b_1, b_2 are made (cooperatively) by the two decision-making red team members in the red choice gadgets,

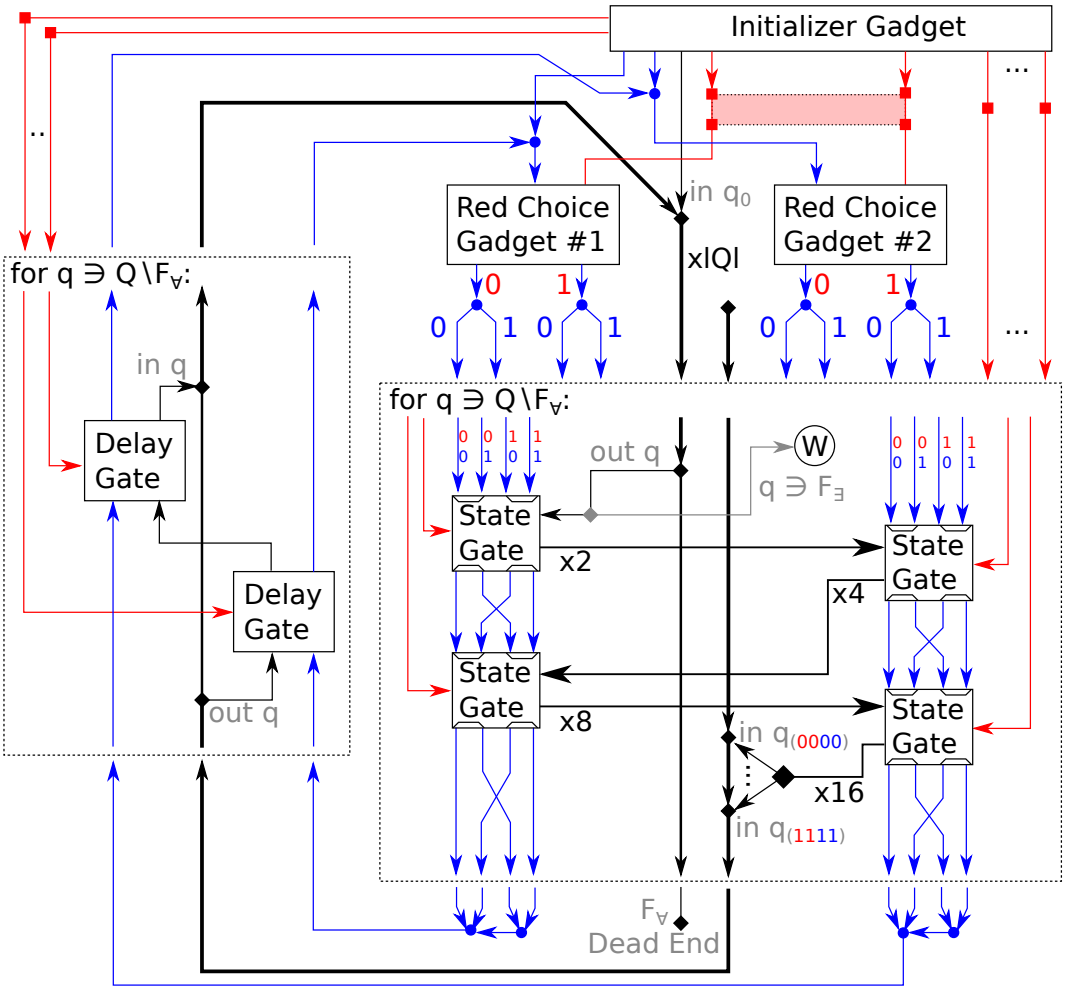

Figure 5 A diagram of how the gadgets are put together.

and the existential team's decisions m_1, m_2 are made (independently of each-other) by the decision-making blue team members directly after exiting the red choice gadgets. The blue runner's location corresponds directly to the state of the DFA, and their teammates open paths inside state gates which allows the runner to implement the DFA transition function δ.

Each state $q \in Q \setminus F_\forall$ of the DFA has an "arena" with two sides: the right side with a series of four state gates of increasing arity and a left side with a series of two Delay Gates. When the blue runner enters the right side of the arena for q before the first state gate, the DFA is in state q. If $q \in F_\exists$ then there will also be a hallway here leading directly to the win node. The four state gates encode the tree of states reachable from q in up to 4 transitions, outputting the runner in one of 16 hallways each corresponding to a state $q' = \mathsf{foldl}(\delta, q, [b_1, b_2, m_1, m_2])$ and leading to the left side of the arena for q'. Once the runner passes through the Delay Gates, they enter the right side of the arena for q'. Lastly, if $q \in F_\forall$, then all hallways entering its arena lead to a dead-end.

As we showed in Section 2.4, each team has a course of action which will prevent any players on the other team from reaching the Win node. Further, this puts every player on a path whose only way forward is out of the initializer gadget. At that point there is no incentive to stay in the initializer gadget and we may as well assume they continue into the rest of the map.

$\Longrightarrow$ Suppose the existential team has a forced win in TDG on D. This means that there are optimal strategy functions $\mathsf{s}_i : ([b_{i,1}, b_{i,2}, ..., b_{i,j-1}], [m_{i,1}, ..., m_{i,j-1}], b_{i,j}) \mapsto m_{i,j}$ which produce a win-preserving move for $\exists_i$ in round j given $\forall$'s move and what they learned in the past $j-1$ turns.
For decision-making blue player i, on the j^{th} time they pass through red choice gate i, let $b_{i,j} = 0$ if exiting the A side else $b_{i,j} = 1$ if exiting the B, let $m_{i,j} = \mathsf{s}_i([b_{i,1}, ..., b_{i,j-1}], [m_{i,1}, ..., m_{i,j-1}], b_{i,j})$, then at the upcoming branch take path $m_{i,j}$. The blue runner should follow the hallways and wait until combat zones are safe before passing through, and the decision-making blue team members should open combat zones long enough for the runner to pass through safely and to defeat the red team member there if necessary. By the structure of the graph, the path of the runner will lead to a $q \in F_\exists$ no matter what choices red team makes in the red choice gadgets, and every attack zone along the way will be opened up for the blue runner by their teammates, thus blue team has a forced win in TGG on I.

$\Longleftarrow$ Now suppose blue team has a forced win in TGG on I. Since only the blue runner can reach win node (outside the initializer gadget), any winning execution entails a path through the graph that the runner took which starts by entering the right side of the q_0 arena, passes through n arena right sides and left sides (as described earlier), and ends at the entrance of the right side of an arena for some $q_n \in F_\exists$.
In order for the runner to pass through the combat zones in the gates along the path, the decision-making blue teammates must have dealt with the attacking red team members. Since blue team has a forced win, they still have a forced win even if red team attackers always leave their attack zone before the decision-making blue team member has a chance to defeat them, thus that strategy forces the blue runner at the entrance of the right side of an arena to take a path through the state gates determined by the red and blue teams' choices at the start of the loop.
This implies the existence of functions $\mathsf{s}_i : ([b_{i,1}, b_{i,2}, ..., b_{i,j-1}], [m_{i,1}, ..., m_{i,j-1}], b_{i,j}) \mapsto m_{i,j}$ which produce a win-preserving branch for decision-making blue team member i to take on the loop j after exiting red choice gate i from exit $b_{i,j}$ and what they learned in the past $j-1$ loops. By the structure of the graph, s_i is also an optimal strategy function for $\exists_i$ in TDG on D, thus the existential team has a forced win. ◀

▶ **Corollary 4.** *The TEAM GRAPH GAME is undecidable.*

Proof. If TEAM GRAPH GAME were decidable, then TDG would be decidable using h from Theorem 3 to get a homomorphic instance, but since TDG is undecidable by Corollary 8, TEAM GRAPH GAME cannot be either. ◀

4 Applications

We now show how to apply the TEAM GRAPH game to generalized versions of several popular video games. In particular we will show that it is undecidable to determine whether a team can force a win in the following games: Team Fortress 2, Mario Kart, and Super Smash Bros. Brawl. For all of these games we generalize the map size and number of players able to participate in a single game. In addition, we assume that players on the same team have no way of communicating with each other beyond their actions in the game. This means players are not co-located, there is no screen-sharing, and any sort of team or global chat is disabled.

The following are the essential components needed in the game to fit the TGG framework. 1) The game needs a 3D map or crossover gadgets in 2D because the TGG graph used in

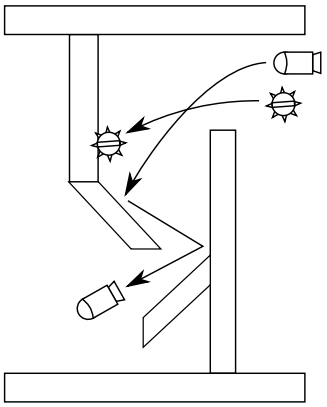

Figure 6 Grenade-only Attack Gadget (vertical 2D slice)

our reduction is non-planar. 2) One-way Doors. 3) Visibility zones such that we can have two players communicate their location without being able to reach each others path, and ways of blocking visibility so communication can only occur in specific regions. 4) Combat zones which allow the attacker a guaranteed strategy to eliminate or disable the defender and which has no path between the attacker and defender. 5) A win condition that can be activated by one player in a limited location.

4.1 Team Fortress 2 and many other team FPS games

Like many others of its kind, Team Fortress 2 is a first person shooter with 3D environments (1), one-way doorways (2), clear unbreakable glass/fences and opaque walls (3) made out of polygons, grenades and sniper rifles (4), and a capture point where one team can win by standing on it (5). These features allow TF2 and others to directly simulate TGG, leading to their undecidability. Note: only the base TF2 game with default loadouts are considered.

The nodes and edges of the graph are generally represented as hallways made of opaque walls connecting at intersections, possibly lengthened or bent-out-of-shape to enforce a required minimum traversal time. Visibility is limited by the first-person view, and visibility zones are constructed by making walls out of glass that gives a line-of-sight between desired locations and possibly additional walls to block view elsewhere.

The combat zones are constructed based on which team the attacker is on. A blue team member attacking a red team member will be faced with a room with a wall that only Demomen grenades can be shot over and succeed at damaging the defender. Figure **??** shows how to construct a hole which only physics-enabled grenades can tumble through and sticky bombs and other weapons cannot penetrate. A red team member attacking a blue team member will be faced with a small hole in the wall at Sniper-eye-level which gives a long-distance view of the defender's head such that only a Sniper's sniper rifle can kill the defender before they can pass through the attack zone at optimal speed.

In order to further enforce desired class choices, the red and blue teams are incentivized to choose the Sniper and Demoman classes (respectively) by the map design. The blue team spawn room is separated by a deadly chasm that can only be crossed using the Demoman's unique ability to sticky bomb jump long distances through the air without touching a surface (as a Soldier requires). Health pack pick-ups and distance-based fall damage may be used to force the health of players down so one sniper shot or grenade explosion will defeat any opponent.

By playing in a king-of-the-hill match with unlimited-time and with text and voice chat disabled, this map structure will exactly simulate TGG.

4.2 Super Smash Brothers

Super Smash Brothers is a popular Nintendo fighting game series. Out of the series' five releases, the most recent three (Super Smash Bros. Brawl, Super Smash Bros. for 3DS, and Super Smash Bros. for Wii U, henceforth referred to as Brawl, SSB4 3DS, and SSB4 Wii U, respectively) share a number of gameplay elements which we will shortly show result in undecidability.

We consider a generalized Super Smash Bros. game, where an arbitrary number of players on red or blue team control fighters (who are followed by the players' personal, local cameras, as in SSB4 3DS Smash Run mode) which fight on a stage (a bounded 2D plane with gravity, solid polygonal ground, and other obstacles) in Stamina mode (where each player starts with a given number of hit points and dies when they are depleted). Fighters are selected among a set of characters, each with unique traits, and can walk, run, jump off the ground and jump in the air finitely-many times before landing, and fight using aerial and ground attacks (which may create hitboxes which damage and knockback other characters, may move the attacker, and may provide defense), and defensive maneuvers such as shielding (a bubble around character which blocks attacks at the expense of temporary shrinkage), air and ground dodging (temporary invincibility at the cost of short vulnerability before and afterwards) and rolling (a ground dodge with fixed motion left or right). Due to close-quarters, we also consider obtrusive stage background music such that all character sound effects are drowned-out.

▶ **Theorem 5.** *In generalized Super Smash Bros. match between two teams of Pikachus on some stage, it is undecidable whether Player 1's team has a forced win.*

Proof. Reducing from TGG constrained to graphs constructed from DFA as in Theorem 3, we consider only the character Pikachu due to its unique Thunder attack that temporarily spawns a damaging cloud and lightning strike at a fixed position above Pikachu, even if there are obstacles in between. Instead of 3D hallways, our construction of the stage simulating the graph only needs to bound 2D areas with strings of solid blocks (as in Brawl's and SSB4 Wii U's stage builder) that are thin enough in certain areas for Thunder to attack other characters through ceilings. We also use thin floors, which allow for jumping upwards through but do not allow for falling through, to construct one-way doors.

The most striking problem for this 2D fighting game is the need for a crossover gadget. We make use of the barrel cannon stage obstacle, as seen in the Kongo Jungle stage from the first Super Smash Bros. as well as all future titles in some form, which captures a player upon contact and, when activated by the player inside, launches them along a fixed path without the player having aerial control until the end. Notably, we consider the original design of the cannon where a launched player does not hurt others via collision. By using two barrels and two one-way floors, a section of the stage as in Figure 7 can allow for crossovers without player interaction, although it does provide visibility. Because the constrained TGG graphs can be embedded in the plane where all edge crossings are either outside of the main loop before the simulation begins, same-player crossings, or between players who are allowed to know where the other's token is located, visibility does not transmit information that is useful for making red or blue team "choices."

As mentioned, attack zones are built around Pikachu's Thunder attack, which unconditionally creates a hitbox at a fixed distance high above the character. For attack zones that guard the traversal of an edge, the idea is to force the defending Pikachu to predictably position itself in a vulnerable state above the attacker, so that the attacking Pikachu can always hit them with Thunder if traversal is attempted. In Delay Gates, such as in Figure 8, where the red attacker of the blue runner is under attack themselves, the blue attacker is

able to Thunder the only location at which the red attacker can use Thunder to hit the blue runner, so as to open the path safely. The Red Team Choice Gadget can be implemented in Brawl similarly to the Delay Gate, and the State Gates directly out of Delay Gates, thus the given TGG graph is fully representable.

When the blue runner reaches the win node, they can themselves open a path for the other blue Pikachus and all go into a new series of pathways that lead underneath every red team player so they can work together to eliminate them all, as properly-timed Thunders by multiple players can break shields and hit for longer than dodge invincibility. and end the match with a blue victory. This path-opening can be a Delay Gate or even compactly implemented using Brawl's Falling Block object, which is a solid obstacle that temporarily falls and disappears after a player (the blue runner, in this case) stands on it, reappearing at its original position after a short period of time. ◀

4.3 Mario Kart

In an earlier paper, two player, perfect information Mario Kart was shown to be PSPACE-complete [2]. It also did not consider the commonly enjoyed Battle game type. Here we show that a generalized version of Mario Kart in team Balloon Battle mode is undecidable by a reduction from TGG.

Mario Kart takes place in a 3D environment where each player has a personal third-person camera view of their character; when playing online or on local wireless, players cannot see other players' screens. In Balloon Battle, the players are placed in an enclosed, obstacle-filled Battle Course with a small number of balloons that pop when the player is damaged, eliminating the player if none remain. By searching the course for item boxes (in fixed, reusable spawn locations), players can get items from a given distribution to damage other players and avoid attacks against themselves. There is a blue team and a red team, and if one team is completely eliminated, the other team wins.

▶ **Theorem 6.** *In generalized Mario Kart Balloon Battle with the Bob-ombs Only item distribution, it is undecidable whether or not the blue team has a forced win.*

Proof. We reduce from TGG constrained to graphs constructed from DFA as in Theorem 3, which involves building a Battle Course that simulates the graph. Mario Kart courses are polygonal 3D environments with a finite maximum movement speed, one-way jumps, clear glass, and opaque walls, so the primary complexity is describing the attack zones and how to win.

A player using a Bob-omb item causes a Bob-omb to be thrown from the character's kart in an arc. It can bounce off walls and will explode into a large, temporary, damaging sphere on contact with another player or after a short time interval. One common obstacle in Mario Kart is the Thwomp, which are large spike-covered boxes which can move along fixed paths.

To construct an attack zone where the attacker is preventing the defending character from traversing an edge, said edge is a short, thin hallway with exits guarded by Twomps that alternate moving up and down between the ceiling and ground such that at least one is always on the ground blocking the path and the space between is smaller than the diameter of a Bob-omb explosion. The attacking character is spawned in a raised hallway with an item box and an uncrossable pit such that a Bob-omb can be thrown by the attacker and create an explosion to eliminate any player between the Thwomps but no Bob-omb can be thrown back high enough to reach the attacker. In an attack zone where the defender is itself an attacker in a dead-end hallway, there need only be one Thwomp guarding the single exit

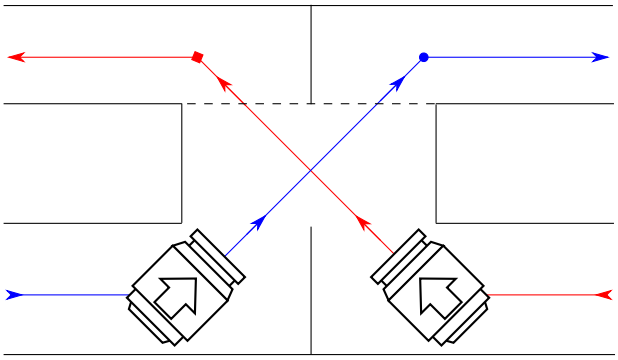

Figure 7 Super Smash Bros Crossover Gadget using Barrel Cannons

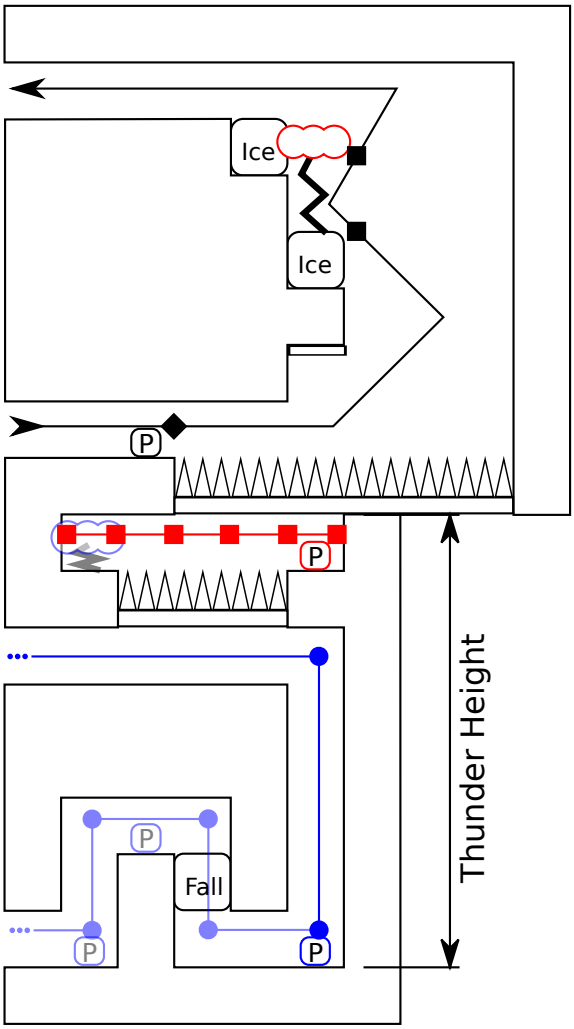

Figure 8 Delay Gate constructed using Brawl's Custom Stage Builder parts. A single player's screen is approximately 5 blocks tall, so the blue executor can never see the runner. Each "P" is an example location of a Pikachu, "Ice" is a block with no edge to hang onto, and "Fall" represents a Falling Block. Shaded blue figures are only relevant during the blue victory phase. Example Thunder clouds and associated lightning strikes are also shown.

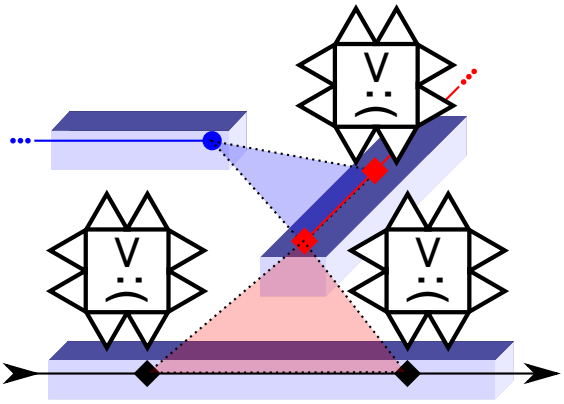

Figure 9 Mario Kart Delay Gadget's 3D Layout with Thwomps (opaque walls not shown).

and trapping the defender for a period of time such that the attacker in an even-more-raised hallway could safely throw down a Bob-omb to eliminate them. Figure 9 gives an overview of this construction.

When the Mario Kart character simulating blue runner is supposed to reach the win node, they are first able to open a path for their blue teammates (normally blocked by a red attack zone) to join them into a set of hallways above the rest of the course which lead to attack zones spanning each red team character's small region of the graph. With plentiful item boxes, the blue team characters can thus trap and eliminate each red team member using coordinated Bob-omb threats and throws, winning them the game. ◀

5 Conclusion and Open Problems

Our TEAM GRAPH GAME framework has proven useful in showing the undecidability of more natural team multi-player games, as shown in our application to various video games. We currently wonder how far this framework can go. Can we capture other popular genera of video-games with teams such as MMORPGs like World of Warcraft and Guild Wars, real time strategy games like Starcraft or Age of Empires, MoBAs like DotA and Heroes of the Storm, or others? Each of these has their own challenges in adapting to our framework, but given our success with Super Smash Brothers which was a 2D game that lacked vision blockers and a location based victory condition, we believe a lot can be done with a little work. We also pose the question of whether this framework can be used to understand the complexity of any real world multi-agent coordination scenarios.

There are also a number of interesting questions about imperfect information team multi-player games, many of which would be very useful in allowing broader application of this framework. First, can TGG be adapted to use only a constant number of players on each team? The TCG needs only three; however, we find it useful to assign different players to many of our gadgets, leading to a linear scaling. Is the TCG or TGG still undecidable if we allow a limited amount of communication between players on the same team? For example, players may be allowed to pairwise communicate or broadcast a constant number of bits per round. At what point does this problem become equivalent to a two player game of imperfect information? Finally, is there a way to adapt these abstract games to describe semi-cooperative games and are these still undecidable. For example, instead of having fixed teams and asking for a forced win, we might define optimal play to involve trying to maximize an individual player's probability of winning and ask whether a certain player has

a strategy which wins with some fixed probability. If players have some chance of winning by working together but zero chance of winning otherwise, we might be able to force players to simulate teams in such a game.

References

1. Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Viglietta. Classic Nintendo games are (NP-)hard. In *Proceedings of the 7th International Conference on Fun with Algorithms (FUN 2014)*, Lipari Island, Italy, July 1–3 2014.
2. Jeffrey Bosboom, Erik D Demaine, Adam Hesterberg, Jayson Lynch, and Erik Waingarten. Mario kart is hard. In *Japanese Conference on Discrete and Computational Geometry and Graphs*, pages 49–59. Springer, 2015.
3. Erik D Demaine and Robert A Hearn. Constraint logic: A uniform framework for modeling computation as games. In *Computational Complexity, 2008. CCC'08. 23rd Annual IEEE Conference on*, pages 149–162. IEEE, 2008.
4. Erik D Demaine, Joshua Lockhart, and Jayson Lynch. The computational complexity of portal and other 3d video games. *arXiv preprint arXiv:1611.10319*, 2016.
5. Luciano Guala, Stefano Leucci, and Emanuele Natale. Bejeweled, candy crush and other match-three games are (np-) hard. In *IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.
6. Linus Hamilton. Braid is undecidable. *arXiv preprint arXiv:1412.0784*, 2014.
7. Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. A. K. Peters, Ltd., Natick, MA, USA, 2009.
8. G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers and Mathematics with Applications*, 41(7):957–992, 2001. doi:10.1016/S0898-1221(00)00333-3.
9. Gary L Peterson and John H Reif. Multiple-person alternation. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 348–363. IEEE, 1979.
10. John H Reif. Universal games of incomplete information. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 288–308. ACM, 1979.
11. John H Reif. The complexity of two-player games of incomplete information. *Journal of computer and system sciences*, 29(2):274–301, 1984.
12. Matthew Stephenson, Jochen Renz, and Xiaoyu Ge. The computational complexity of angry birds and similar physics-simulation games. 2017.
13. Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54(4):595–621, 2014.

A TEAM DFA GAME is Undecidable

▶ **Lemma 7.** *TDG is reducible from and to TCG. Namely,* $\exists f : \langle D \rangle \to \langle I \rangle$ *and* $\exists g : \langle I \rangle \to \langle D \rangle$ *which map between instances* $\langle D \rangle$ *of TDG and instances* $\langle I \rangle$ *of TCG which both preserve the predicate of whether or not the existential team has a forced win.*

Proof.

$\Longleftarrow$ Consider an instance $I = \langle S, O, k, \Gamma \supset O \cup \{\mathsf{A}, \mathsf{B}\}\rangle$ of the TCG.

The TDG on the corresponding DFA D will directly simulate the TCG on I. The state space $Q(D)$ is the configurations of S as well as additional counters for input tracking. The first $\forall$ turn runs S without input from the existential team, thus $q_0(D)$ is the result of immediately applying δ_S k times (or until termination) from its initial configuration. After that, both games check for termination in the same way (accept states of S are win

states of existential team, reject for universal), then begin writing to S's tape or feeding bits into D. The only significant difference is that the existential moves O must be input to D in binary over $2\lfloor \log_2 |O| \rfloor$ rounds where the universal player's moves are ignored by D. The transition function δ_D simply writes the appropriate bits of the moves from $\forall, \exists_1, \exists_2$ onto the tape of the current configuration, and once everything is input then it updates the configuration by applying δ_S k times (or until termination).

$\Longrightarrow$ Consider an instance D of the TEAM DFA COMPUTATION GAME.

The TCG on the corresponding instance $I = \langle S, O, k, \Gamma \rangle$ will similarly be a direct simulation of the TDG. Using $k = 6$ and $\Gamma = O = \{0, 1\}$, the tape of S is just the cells for each input bit b_1, b_2, m_1, m_2 plus unused space at the end. Its state space simply augments $Q(D)$ with input reading states. The first k steps, S will be in $q_0(D)$ and move nowhere, but each following time S is simulated for k steps, starting at tape position 0, S will read each bit, applying δ_D to update its DFA state for each read (unless it has entered a final state), then just return to position 0.

At the start, TCG runs S for k steps, which does nothing. The termination check for each game is the same, as before, then each player will input their move onto the appropriate cell of the tape (in the same order in both games) then run S again, which will simulate the same inputs being given to D and updating its state. ◀

▶ **Corollary 8.** *The TEAM DFA GAME is undecidable.*

Proof. If TDG were decidable, then TCG would be decidable using f from Theorem 7 to get a homomorphic instance, but since TCG is undecidable [3], TDG cannot be either. ◀