5-4-2017

Computability Theoretic Results for the Game of Cops and Robbers on Infinite Graphs

Rachel D. Stahl
University of Connecticut - Storrs, rachel.stahl@uconn.edu

Computability Theoretic Results for the Game of Cops and Robbers on Infinite Graphs

Rachel Stahl, Ph.D.

University of Connecticut, 2017

ABSTRACT

Several results about the game of cops and robbers on infinite graphs are analyzed from the perspective of computability theory and reverse mathematics. Computable robber-win graphs are constructed with the property that no computable robber strategy is a winning strategy, and such that for an arbitrary computable ordinal α, any winning strategy has complexity at least $0^{(\alpha)}$. Symmetrically, computable cop-win graphs are constructed with the property that no computable cop strategy is a winning strategy. However the coding methods used in the robber-win case fail here. Locally finite infinite trees and graphs are explored using tools of reverse mathematics. The Turing computability of a binary relation used to classify cop-win graphs is studied.

Computability Theoretic Results for the Game of Cops and Robbers on Infinite Graphs

Rachel Stahl

M.S. Mathematics, University of Connecticut, Storrs, CT, 2014

M.A. Secondary Mathematics Education, Columbia University Teachers College, New York, NY, 2009

B.A. Mathematics, Bard College, Annandale-on-Hudson, NY, 2008

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2017

APPROVAL PAGE

Doctor of Philosophy Dissertation

Computability Theoretic Results for the Game of Cops and Robbers on Infinite Graphs

Presented by

Rachel Stahl, B.A. Math., M.A. Math. Ed., M.S. Math.

Major Advisor ______________________
Dr. David Reed Solomon

Associate Advisor ______________________
Dr. Damir Dzhafarov

Associate Advisor ______________________
Dr. Tom Roby

University of Connecticut

2017

ACKNOWLEDGMENTS

First and foremost, I would like to acknowledge my advisor Reed Solomon. I cannot say enough about everything he has done to support me during my time at UConn. In addition to being my research advisor for the past few years, he has acted an advocate and mentor to me since I applied to UConn in 2011. As graduate director, his encouragement and accommodation is what led me to believe that UConn would be a good fit for me, and in the years since he has continued to be an unmatched role model in both research and teaching. His comments on my dissertation were invaluable, as was his guidance as I navigated the job market.

I would also like to thank the other members of my committee, Damir Dzhafarov and Tom Roby, with whom I have had the privilege of taking many classes. I have learned so much from each of them, not just in the content areas of logic and combinatorics but in the art of teaching as well. I want to thank all three of my committee members for their insight and feedback on my interview quandaries, their thoughtful letters of recommendation, and all the phone calls and hassle that came with it. I am incredibly grateful for the time they have offered me.

I want to acknowledge my officemates, in particular Rebecca and Mike (and Karin), for being there for 5 years of venting, prelims, student stories, field trips, and smelly office foods. Thank you to Jackie, Liz, and Tom for listening to me practice my job talk on vacation, and for humoring me when I want to talk about Fibonacci. Thank you to Shaun for his patience, and Mo for the same reason.

I also want to thank my family including my siblings, Omie, Becky, Jackie, and Johnny, and especially my parents. They have supported me in every imaginable way throughout this process (and my life), including but not exclusive to offering their home to my dog during my job search. I could not have made it through graduate school without their encouragement and I am incredibly grateful.

Contents

Chapter 1

Introduction

Cops and Robbers is a vertex-pursuit game played on a connected reflexive graph wherein two players, a cop and a robber, begin on a pair of vertices and alternate turns moving to adjacent vertices. The cop attempts to capture the robber, while the robber tries to evade the cop. Since the game was first studied in the late 1970's, much work has been done to study the game on finite graphs ([2]). While there are some known results about the game of cops and robbers on infinite graphs, we wish to investigate whether these theorems hold if we consider computable infinite graphs, and require that cop and robber strategies be effective.

In computability theory, we characterize and compare different sets, functions, and algebraic structures such as graphs in terms of a notion of complexity. A set or function is computable if, in short, there is some algorithm so that a computer is able to describe membership of the set or, in the case of functions, the set of ordered pairs. Similarly, a structure is computable if a computer is able to describe the domain and relations within the structure. These notions of computability are made precise using

Turing machines.

We often consider two main types of questions in the field of computability theory. First, how complicated is it to describe a given set or structure? To make this question precise, we formalize the notion of relative complexity of sets by determining whether knowledge of one set is enough information to compute another set. In particular, the Turing degree of a set gives a precise measure of its computational content.

Another common question to ask in computability theory is whether given problems in mathematics can be solved effectively. For example, we know that a ring is not a field if and only if it contains a nontrivial proper ideal. However, if we consider a computable ring which is not a field, is it necessarily the case that we are able to effectively find a nontrivial proper ideal? This idea of considering whether solutions are constructive or more complex is also tied to proof theory; it has allowed mathematicians to give a negative answer to Hilbert's tenth problem, and to show the undecidability of the word problem for finitely presented groups.

Finally, we wish to compare the proof-theoretic strength of known results. In particular, given a theorem, which standard axioms of second order arithmetic are truly necessary to prove a given result, and which axioms can we eliminate? The investigation of this question is made precise in the study of reverse mathematics.

1.1 Cops and Robbers Background

Notation. Let $G = (V, E)$ be a connected undirected graph without multiple edges, and with vertex set V and edge relation E. We often identify G with the vertex set V. In graph theory, the edge set of such a graph is considered to be a collection of two-element subsets of the vertex set, and for a pair of elements $v, w \in V$, an edge between v and w is written $\{v, w\} \in E$. However, we use the convention in logic (as seen in [7]) of writing E as a binary relation, and say that if there is an edge between v and w then $E(v, w)$ holds, or sometimes simply $E(v, w)$. Notice that since the graph is undirected, this implies that the relation is symmetric, so that $E(v, w)$ holds if and only if $E(w, v)$ holds.

Throughout, we assume that G is connected, and that G is reflexive, i.e., for each $v \in G$, we have $E(v, v)$. In diagrams, we omit drawing these reflexive edges to avoid clutter. Also, when determining the degree of a node, we will not count reflexive edges. Typically when we consider infinite graphs, we assume for the sake of studying computability theoretic questions that the vertex set is countable (though the theory works for uncountable graphs as well). Note that although we assume that G does not contain multiple edges, this assumption is to simplify notation and in fact makes no difference in game play. For $v \in G$, we define $N[v] = \{w \in G : E(v, w)\}$ to be the set of neighbors of v. Note by reflexivity that $v \in N[v]$ for all v.

Definition 1.1.1. In the game of **Cops and Robbers on Graphs**, a player C controls a single cop while a player R controls a single robber [1]. The game is played in rounds, with the cop making a *move* first followed by the robber in each round. In round 0, a move consists of the cop choosing a vertex $c_0 \in G$ to occupy, followed by the robber choosing a starting vertex $r_0 \in G$. In subsequent rounds, a move is a player's choice to occupy a neighboring vertex of that player's current vertex. Note that by the reflexivity of G, a player may "pass" on their turn by moving to the same

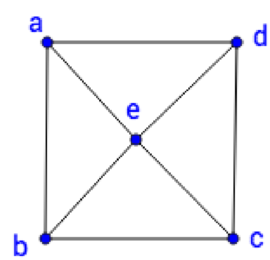
[1] While the game traditionally allows for C to control multiple cops at a time, we consider only games with a single cop and a single robber and, thus, may be inclined to call this game Cop and Robber on Graphs instead.

vertex they currently occupy.

The game ends with a win for the cop if after finitely many moves, the cop occupies the same vertex as the robber. The robber wins if he is able to evade capture indefinitely.

Informally, we call a given graph G cop-win if there exists a winning strategy for the cop, i.e. some set of rules that the cop can follow that will allow her to win no matter what the robber does. A formal definition will follow after we consider the following motivating examples.

Example 1.1.2. Let $G = \{a, b, c, d, e\}$, with edges as seen below.



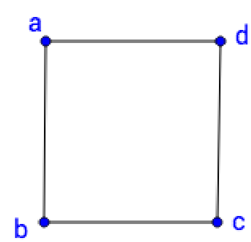
A cop C can begin the game at vertex e, which is adjacent to every other vertex in G. Thus no matter the initial starting point for the robber R, the cop will win in the next round by moving to that vertex. So G is cop-win. The key to winning in this case relies on the vertex e which is adjacent to every other vertex in the graph. Such a vertex is called *universal*, and any graph with a universal vertex is cop-win using the strategy of starting at the universal vertex.

Note that if there exists a cop-win strategy f_C for G with initial starting position c_0, then there is a cop-win strategy for G with initial starting position v for any $v \in G$.

This is because a cop starting at v can follow a strategy of first moving to vertex c_0, and then following the strategy f_C to win. Therefore, when convenient, we can fix a vertex $c_0 \in G$ and assume without loss of generality that all cop strategies start at c_0.

If the robber has a strategy that allows him to evade capture indefinitely on a graph G regardless of the cop's strategy, we say that G is robber-win.

Example 1.1.3. Let $G = \{a, b, c, d\}$, a 4-cycle as seen in the image below.



For any starting position of the cop, the robber has a strategy of starting distance 2 from the cop. On each subsequent round, no matter where the cop moves, the robber will always be able to then choose a vertex distance 2 from the cop and evade capture, making this graph robber-win. Similarly, any cycle graph on n vertices for $n > 3$ is robber-win.

These simple examples have strategies that are easily explained in words, but for more complicated graphs a rigorous definition is needed. To that end, we first define a tree (in the logical sense, rather than the graph theoretical sense). We denote $\omega = \{0, 1, 2, 3, \dots\}$ to be the set of non-negative integers, and define a tree as follows, using standard convention in logic (as seen in [7]).

Definition 1.1.4. Let S be a set. Then a **countable tree** $T \subseteq S^{<\omega}$ is a set of finite strings of elements of S which is closed under initial segments. That is, if $\sigma = \langle s_0, s_1, \ldots, s_n \rangle \in T$, then for all $i < n$, $\langle s_0, s_1, \ldots, s_i \rangle \in T$. We write $|\sigma|$ for the length of σ, and notice that by closure under initial segments, we must have an empty string of length 0 in T; we usually denote this empty string by λ. For $\sigma = \langle s_0, s_1, \ldots, s_n \rangle$, we write $\sigma(i) = s_i$ and $|\sigma| = n + 1$.

For $\sigma = \langle s_0, s_1, \ldots, s_n \rangle$, we write $\sigma * s_{n+1}$ as an abbreviation for the concatenation of σ by s_{n+1}, i.e. $\sigma * s_{n+1} = \tau$ where τ has length $n+2$, $\tau(i) = \sigma(i)$ for all $i < n+1$, and $\tau(n+1) = s_{n+1}$.

The definition of tree above is standard in logic literature, with the set S often being the set of nonnegative integers ω. This differs from the classical definition of a tree in graph theory, which is a simple connected graph without cycles, or equivalently, a simple graph such that between every pair of vertices there is a unique path. These notions, however, are equivalent in the following sense.

Given a countable tree $T \subseteq S^{<\omega}$ (in the logical sense, as defined above), we can view T as a tree G_T in the graph sense with a vertex set of the nodes of T, and an edge relation such that $E(\sigma, \tau)$ if and only if τ is an initial segment of σ with $|\tau| + 1 = |\sigma|$, or σ is an initial segment of τ with $|\sigma| + 1 = |\tau|$. It is clear that this yields a connected graph with no cycles.

On the other hand, given a tree G with countable vertex set (in the graph theoretical sense), we can enumerate the vertex set as $v_0, v_1, \ldots$. Then we can define a tree T_G in the logical sense by $\lambda \in T_G$, and $\sigma \in T_G$ if and only if $\sigma = \langle v_{i_0}, \ldots, v_{i_{n-1}} \rangle$ (where $n = |\sigma|$) is a sequence of distinct nodes with $E(v_0, v_{i_0})$ and $E(v_{i_k}, v_{i_{k+1}})$ for all $k < n - 2$. Then T_G is a tree in the logical sense, as it is closed under initial segments. The map from T_G to G defined by $\lambda \mapsto v_0$ and $\sigma \mapsto \sigma(|\sigma|-1)$ for $\sigma \neq \lambda$ is a bijection which is an isomorphism between G and G_{T_G}, i.e. between G and the tree graph version of the tree T_G.

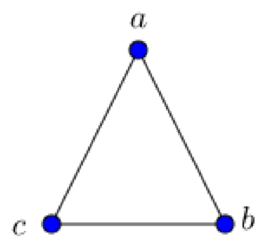
Throughout this paper, we will use the term "tree" to mean both a tree and a tree graph, and we rely on context to determine whether we refer to a tree in the logic sense or in the graph theory sense.

We define an **allowable R-play sequence** for a fixed $G = (V, E)$ to be a finite sequence σ of elements of V that describe a (perhaps partial) play of the game. In particular, we let $\sigma \in V^{<\omega}$ be such that

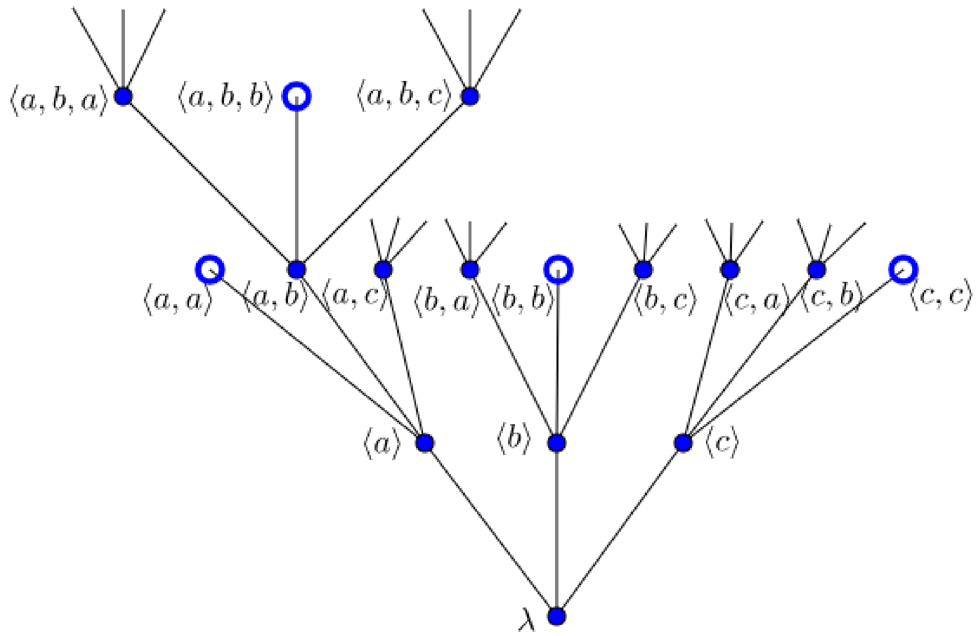
1. $|\sigma|$ is even,
2. if $|\sigma| > 0$, then $\sigma = \langle c_0, r_0, c_1, r_1, \ldots, c_n, r_n \rangle$ such that for all $i < n$, $r_{i+1} \in N[r_i]$ and $c_{i+1} \in N[c_i]$, and
3. if $\sigma(i) = \sigma(i+1)$, then $i = |\sigma| - 2$.

Note that an allowable R-play sequence is a string of vertices of G that is either the empty string (denoted λ), or describes a finite sequence of moves in a game, ending with a robber move. The third condition requires that if the cop and robber ever occupy the same vertex, the game ends and thus the string ends. We analogously define an **allowable C-play sequence** to be a string $\sigma \in V^{<\omega}$ of odd length describing a finite sequence of moves in the game, ending with a cop move. Then an **allowable play sequence** $\sigma \in V^{<\omega}$ is an allowable R- or C-play sequence. We call a play sequence σ **terminal** if $\sigma(|\sigma|-1) = \sigma(|\sigma|-2)$; that is, if the string ends in the same pair of vertices, the game is over so the string ends.

Observe that the set of allowable play sequences for $G = (V, E)$ is a tree $A_G \subseteq V^{<\omega}$, because if σ is an allowable play sequence, every initial segment of σ is as well. In fact, as long as $|V| \geq 2$, A_G is an infinite tree, since each player is allowed to remain on a vertex indefinitely. We give an example of part of the tree of allowable play sequences for a graph consisting of a 3-cycle below.



Example 1.1.5. Let $G = \{a, b, c\}$ with $E(a, b)$, $E(b, c)$ and $E(a, c)$. The diagram below is a part of the tree of allowable play sequences A_G, with hollow circles indicating terminal sequences. Note that every node that is not a terminal play sequence has extensions in the tree. This tree is in fact infinite; for example, for all n, we have the $2n$-length string $\langle a, b, a, b, \ldots, a, b \rangle$, which represents the cop and robber remaining on their initial vertices, a and b respectively, indefinitely.



We use play sequences to give a formal definition of strategies.

Definition 1.1.6. A **cop strategy** f_C is a function with domain $\{\sigma : \sigma \text{ is a non-terminal allowable } R\text{-play sequence}\}$, and such that $f_C(\langle c_0, r_0, \ldots, c_n, r_n\rangle) = c_{n+1} \in N[c_n]$. Note that since the empty string λ is a non-terminal allowable R-play sequence, this strategy includes choosing an initial starting position for the cop. **Robber strategies** are defined analogously with the set of non-terminal allowable C-play sequences as a domain.

Then f_C is a **winning cop strategy** if for any robber strategy f_R, the allowable play sequences determined by the cop following f_C and the robber following f_R eventually produces a sequence $\langle c_0, r_0, \ldots, c_n, r_n, c_{n+1}\rangle$ such that $r_n = c_{n+1}$, or a sequence $\langle c_0, r_0, \ldots, c_n, r_n\rangle$ with $c_n = r_n$. If a winning cop strategy exists for the cop, we say the graph is **cop-win**. Similarly, f_R is a **winning robber strategy** if, for every cop-strategy f_C, the play sequence generated by f_C and f_R is infinite, in which case we say the graph is **robber-win**.

Note that if f_C is a complete strategy for the cop, then f_C determines a subtree of T_G such that each non-terminal even length node has exactly one immediate successor. Similarly, if f_R is a complete robber strategy then it determines a subtree of T_G such that each non-terminal odd length node has exactly one successor.

Then it is clear by analogy that a **partial cop strategy** has a domain that is a subset of all non-terminal allowable R-play sequences, and similarly for **partial robber strategies**. Given a (complete or partial) cop strategy f_C and a (complete or partial) robber strategy f_R, we can generate a string which simulates the gameplay if the cop follows f_C and the robber follow f_R as long as possible. Given f_C and f_R, we define **Play**(f_C, f_R) by defining a sequence of strings $\sigma_0 \subseteq \sigma_1 \subseteq \sigma_2 \subseteq \cdots$ such that $|\sigma_i| = i$. In particular, $\sigma_0 = \lambda$, $\sigma_1 = f_C(\lambda) = \langle v_0 \rangle$, then if σ_i has been defined, we define σ_{i+1} as follows:

Case 1: If $|\sigma_i|$ is odd:

- If f_R is not defined on σ_i, then set $\text{Play}(f_C, f_R) = \sigma_i$ and end recursion.
- If f_R is defined on σ_i, with $f_R(\sigma_i) = v_k$ for some k, then define $\sigma_{i+1} = \sigma_i * v_k$. If $\sigma_i(i) = v_k$, then define $\text{Play}(f_C, f_R) = \sigma_{i+1}$ and end recursion. Otherwise continue recursion for σ_{i+1}.

Case 2: If $|\sigma_i|$ is even:

- If f_C is not defined on σ_i, then set $\text{Play}(f_C, f_R) = \sigma_i$ and end recursion.
- If f_C is defined on σ_i, with $f_C(\sigma_i) = v_j$ for some j, then define $\sigma_{i+1} = \sigma_i * v_j$. If $\sigma_i(i) = v_j$, then define Play $(f_C, f_R) = \sigma_{i+1}$ and end recursion. Otherwise

continue recursion for σ_{i+1}.

Observe that if f_C and f_R are full strategies, then this process may define an infinite sequence, in which case we say $\mathrm{Play}(f_C, f_R) = \bigcup_{i\in\omega} \sigma_i$, and the robber playing f_R evades the cop playing f_C indefinitely.

In some cases, we may wish to consider game play from a fixed starting position. In this case, we denote this by $\mathrm{Play}(f_C, f_R, c_0)$ where we assume that $f_C(\lambda) = \langle c_0 \rangle$ and continue as above. Similarly, if we wish to fix a starting position for both the cop and the robber, we define $\mathrm{Play}(f_C, f_R, c_0, r_0)$ with the assumption that $f_C(\lambda) = \langle c_0 \rangle$ and $f_R(\langle c_0 \rangle) = \langle c_0, r_0 \rangle$.

A natural question is whether we can characterize a given graph G based on whether it is cop- or robber-win. The game of Cops and Robbers is well understood for finite graphs, and results are surveyed in *The Game of Cops and Robbers on Graphs*, by Bonato and Nowakowski ([2]). We present some illuminating examples here.

Theorem 1.1.7. *Every finite tree graph is cop-win.*

Proof. Recall that a tree graph is a connected graph with no cycles, and we have restricted the definition of graphs to consider only connected graphs. Every tree graph contains leaves, i.e., vertices of degree 1, and between any pair of vertices in a tree there is a unique path connecting them. Suppose after round 0, the cop and robber are distance n apart. The cop follows a distance-minimizing strategy of moving to the neighbor on the unique shortest path between the cop and robber. Since every maximal path in a finite tree has finite length, the robber must eventually either decrease the distance between himself and the cop by choice, or reach a leaf, at which point the distance between the players will decrease to $n-1$. Then by induction, after finitely many rounds the distance will be decreased to 0, at which point the cop has won. □

We can generalize this result for infinite trees.

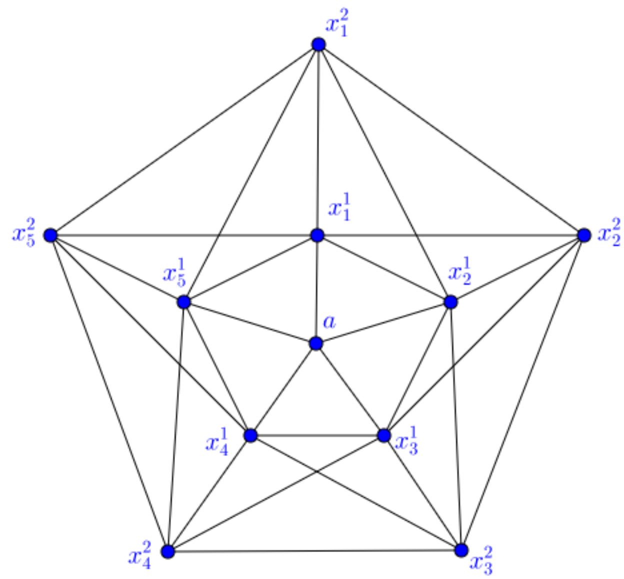
Theorem 1.1.8. *An infinite tree is cop-win if and only if it contains no infinite path.*

Proof. If G is an infinite tree with an infinite path, then the robber has a winning strategy as follows. Suppose the cop starts at c_0. The robber can choose a vertex on the infinite path that is distance at least 2 from the cop. Then by remaining on the path, the robber can evade the cop indefinitely.

On the other hand, if G is an infinite tree without an infinite path, it contains end-vertices and has the property that every maximal path has finite length. The cop can follow a distance-minimizing strategy and the robber must either let distance decrease by choice, or encounter an end vertex as in Theorem 1.1.7. By induction the distance decreases to 0 in finitely many rounds. □

In the proof above, we refer to the distance between two vertices. We take this to mean the length of the shortest path between two vertices, and we will define this more carefully in Section 3.1. Notice here that if a tree is cop-win, a distance-minimizing strategy is a winning one for the cop. However, in [3], Lehner provides an example that demonstrates that distance-minimizing strategies are not always winning strategies in cop-win graphs. Consider the following graph.

Example 1.1.9. ([3]) In the graph below, with starting positions $c_0 = x_1^2$ and $r_0 = x_3^2$, a distance-minimizing strategy will result in a loss for the cop provided the robber remains on the x_i^2 vertices.



To minimize distance, the cop must stay within the x_i^2 vertices or x_i^1 vertices, which will decrease the distance to 1. In doing so, the robber can always move to another x_j^2 vertex, increasing the distance back to 2. In order to win, the cop must move to vertex a, and on her next turn with the robber on x_i^2 move to the corresponding x_i^1. At this point, $N[x_i^2] \subseteq N[x_i^1]$, so the cop will win in the next round.

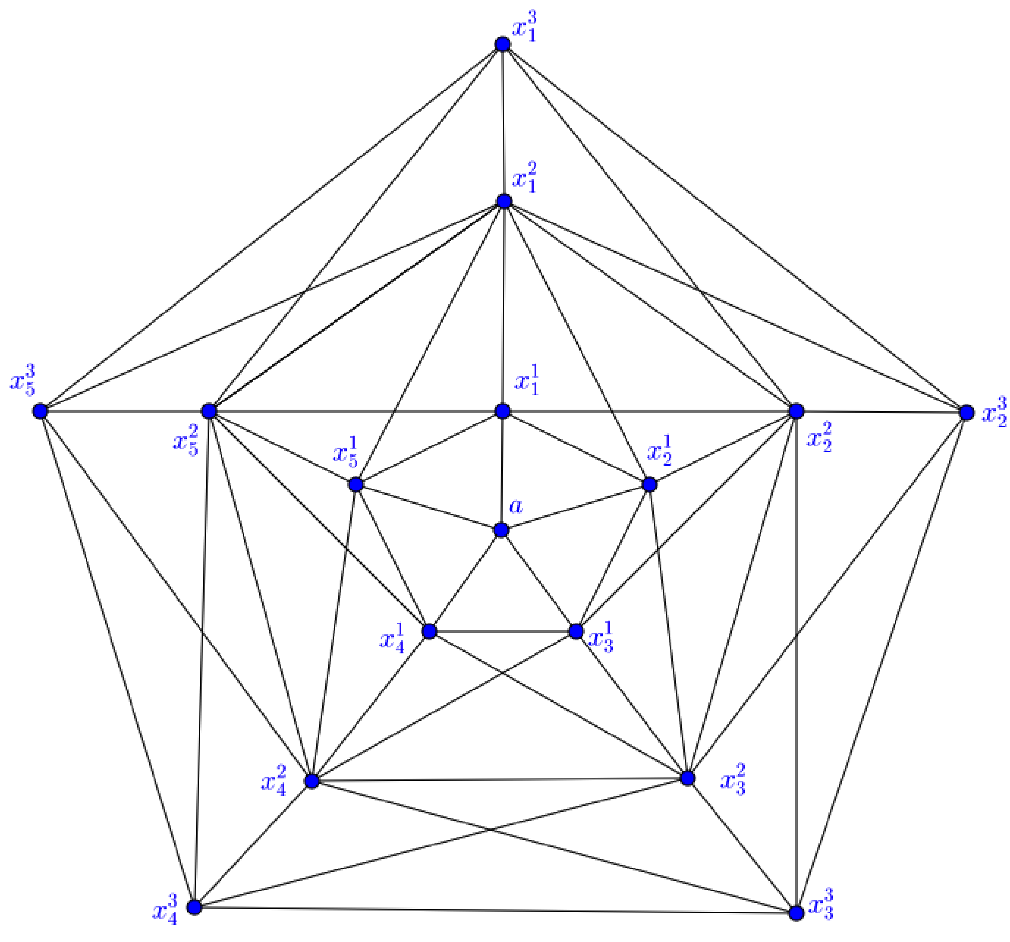
This example can be extended to show that the cop may need to move arbitrarily far away from the robber first in order to win.

Theorem 1.1.10. *For each $n \geq 3$, there is a cop-win finite graph G with fixed starting positions c_0 and r_0 that are distance 2 from each other, and a robber strategy f_R such that any winning cop strategy must increase the distance between the cop and the robber to n.*

Proof. Let $G = \{a, x_i^k : 1 \leq k \leq n, 1 \leq i \leq 5\}$. We have edge relations analogous to the example above:

- $E(a, x_i^1)$ for all $1 \leq i \leq 5$
- $E(x_i^k, x_{i+1}^k)$ and $E(x_5^k, x_1^k)$ for all $1 \leq k \leq n$ and $1 \leq i \leq 4$
- $E(x_i^k, x_i^{k+1})$ for all $1 \leq i \leq 5$ and $1 \leq k \leq n-1$
- $E(x_i^k, x_{i\pm1}^{k+1})$ and $E(x_1^k, x_5^{k+1})$ and $E(x_5^k, x_1^{k+1})$ for all $1 \leq k \leq n-1$ and $1 < i < 5$

The following image shows this graph with $n = 3$.



Fix starting position $c_0 = x_1^n$ and $r_0 = x_3^n$, and assume the robber follows a strategy of maximizing the distance between himself and the cop while staying on the

x_i^n vertices. Then a distance-minimizing strategy for the cop would result in the cop staying on either the x_i^n or x_i^{n-1} vertices, as these vertices will decrease the distance between the cop and the robber to 1. However, as long as the robber stays on vertices of the form x_i^n, the cop can only win by moving to x_j^{n-1} when the robber is on x_j^n, as $N[x_j^n] \subseteq N[x_j^{n-1}]$. From starting the starting positions $c_0 = x_1^n$ and $r_0 = x_3^n$, the robber can arrange to be at vertex $x_{(i+1) \bmod 5}^n$ whenever the cop is at x_i^k for $k \leq n$. Thus, the cop will never be able to move to x_j^{n-1} when the robber is at x_j^n unless the cop first moves to vertex a, and the distance between the cop and robber will never be less than 1.

However, if the cop increases the distance between herself and the robber by moving to vertex a, she can win the game within n rounds. Say the cop is at a and the robber is at x_i^n. Then the cop now has a strategy of moving to x_i^1. From here, if the robber moves to x_j^l, the cop moves to x_j^{a+1} when the previous position was x_i^a; in other words, the cop strategy is to keep the lower index the same as the robber's position while moving out on concentric pentagons. Then in at most n rounds, the cop will win. □

This result suggests that the winning strategies for cop-win graphs can be complex and may be worth further study from a computability theoretic perspective. A characterization for cop-win finite graphs is given using a notion of dismantlability in [2], and this idea can be used to build winning strategies on finite cop-win graphs. However, finite graphs and finite functions are less interesting in computability theory, and to further investigate the complexity of various winning strategies, we wish to understand what classes of infinite graphs are cop-win.

In [4], Nowakowski and Winkler gave a complete characterization of cop-win graphs, including infinite graphs. This characterization relies on a binary relation $\preceq$ defined on the vertices of a graph. We build up this relation by ordinal induction. First, we define

$$v \leq_0 w \Leftrightarrow v = w.$$

Now for an ordinal $\alpha > 0$, and $v, w \in G$,

$$v \leq_\alpha w \qquad \Leftrightarrow \qquad \forall x \in N[v] \exists y \in N[w] (x \leq_\beta y) \text{ for some } \beta < \alpha$$

The intuition here is that if $u \leq_\alpha v$ for some ordinal α, then the cop has some strategy to win the game if the cop occupies v while the robber occupies u.

Observe that the definition implies that if $\beta < \alpha$ and $v \leq_\beta w$, we have $v \leq_\alpha w$ so that $\leq_\beta = \{(v, w) : v \leq_\beta w\} \subseteq \{(v, w) : v \leq_\alpha w\} = \leq_\alpha$. Notice that there must be some ordinal ρ such that $\leq_\rho = \leq_{\rho+1}$, since the size of these sets are bounded above by the cardinality of $G \times G$. Then we define $\preceq = \leq_\rho$ for the least such ρ. We say that the relation $\preceq$ is trivial on the vertices of G if for all $v, w \in G$ we have $v \preceq w$. Then cop-win graphs are characterized as follows.

Theorem 1.1.11. *[4] A graph G is cop-win if and only if the relation $\preceq$ on G is trivial.*

Proof. $\Rightarrow$ Let G be a cop-win graph, and suppose by way of contradiction that there exists vertices $v, w \in G$ such that $v \not\preceq w$. Since the cop must be able to win from any starting vertex, suppose the cop begins at w while the robber begins at v, and assume $\preceq = \leq_\rho$ for least such ρ. Now the cop can choose to move to any $w' \in N[w]$. But since $v \not\leq_\rho w$, we know that there exists $v' \in N[v]$ such that for all $w' \in N[w]$, $v' \not\leq_\rho w'$, since otherwise we would have $v \leq_{\rho+1} w$, a contradiction. Thus once the cop moves to some w', the robber can move to this v' with $v' \not\preceq w'$. Then by induction, the robber is always able to survive another round and win the game. This is a contradiction so we must have $\preceq$ is trivial.

$\Leftarrow$ Suppose $\preceq = \leq_\rho$ is trivial on G. Choose an arbitrary first cop position c_0, and suppose the robber begins on r_0. As $r_0 \preceq c_0$, we have $r_0 \leq_\rho c_0$, so since $r_0 \in N[r_0]$ there exists $c_1 \in N[c_0]$ such that $r_0 \leq_{\rho_1} c_1$ with $\rho_1 < \rho$. The cop moves to this c_1. For any robber choice r_1, there will be some $c_2 \in N[c_1]$ such that $r_1 \leq_{\rho_2} c_2$ with $\rho_2 < \rho_1$, so the cop moves to this c_2.

Now suppose by induction that after n rounds we have $r_n \leq_{\rho_n} c_n$, with $\rho > \rho_1 > \cdots > \rho_n$. Once again there must exist $c_{n+1} \in N[c_n]$ such that $r_n \leq_{\rho_{n+1}} c_{n+1}$ for some ordinal $\rho_{n+1} < \rho_n$. This choice of vertices yields a decreasing sequence of ordinals. Since this sequence cannot be infinite, we conclude $\rho_k = 0$ for some k, at which point we have $r_k \leq_0 c_{k+1}$ which implies that the cop has won the game. $\square$

Observe then that if $\preceq$ is trivial on the vertices of G, we can define a cop-win strategy using $\preceq$ as follows. For a cop-win graph $G = (V, E)$, define $f_\preceq$ on non-empty R-play sequences by $f_\preceq(\langle c_0, r_0, \ldots, c_n, r_n \rangle) = c_{n+1}$ where α is the least ordinal for which $\exists\, y \in N[c_n](r_n \leq_\alpha y)$ and c_{n+1} is $\leq_{\mathbb{N}}$-least node such that $c_{n+1} \in N[c_n]$ and $r_n \leq_\alpha c_{n+1}$. Then for any node v indicating a cop starting position and any full robber strategy f_R, we have $\mathrm{Play}(f_\preceq, f_R, v)$ is a finite sequence ending in a cop win.

Notice then that if G is cop win, $f_\preceq$ is a winning cop strategy that depends only on the pair of vertices currently occupied by the cop and robber, rather than the entire current play history. This implies that on a cop-win graph, the cop has a strategy that is in some sense less complicated as it is memory-less. However, as section 4.1 will explain, this distinction makes no difference within the framework of

Turing computability.

1.2 Computability

For a more thorough background in the basic notions of computability, the reader can see [7].

Notation. Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ be a fixed enumeration of all partial computable functions on natural numbers ω. We write $\varphi_i(x) \downarrow= y$, and say φ_i converges to y on input x, if the ith partial computable function halts on input x after finitely many steps, and outputs y. On the other hand, if the ith computable function never halts on input x, we say φ_i diverges on x and write $\varphi_i(x) \uparrow$.

We also fix a canonical injective way of associating n-tuples $\sigma = (x_0, x_1, \ldots, x_{n-1})$ to natural numbers a_σ. Throughout, if φ_i is expressed as a function with an n-tuple σ as an input, this is equivalent to $\varphi_i(a_\sigma)$.

Definition 1.2.1. A set A is said to be **computable** if its characteristic function χ_A is a computable function.

A standard example of a non-computable set is the halting set, as defined below.

Example 1.2.2. Let $K = \{e \mid \varphi_e(e) \downarrow\}$, the halting set. Then K is not computable, since if it were, the function f defined by

$$f(e) = \begin{cases} \varphi_e(e) + 1 & \text{if } \varphi_e(e) \downarrow \\ 0 & \text{otherwise} \end{cases}$$

would be computable as well. However for all e we have $f \neq \varphi_e$, a contradiction.

The halting set K is an example of a **computably enumerable** (or **c.e.**) set, that is a set A such that A is the domain of some partial computable function. We define $W_e := \mathrm{dom}(\varphi_e) = \{x : \varphi_e(x) \downarrow\}$ to be the eth c.e. set. It is easy to see that a set A is c.e. if and only if A is Σ^0_1, i.e. if $x \in A \Leftrightarrow \exists\, y\, R(x, y)$ for a computable relation R.

We fix a canonical list of all Turing functionals $\Phi^A_0, \Phi^A_1, \Phi^A_2$, that is the functional related to the turing machine Φ_e with oracle A.

Definition 1.2.3. We say a set A is **Turing-reducible** to B, written $A \leq_T B$, if knowing membership of B is enough to compute membership of A. More rigorously, $A \leq_T B$ if there exists an e such that $\Phi^B_e(x) = \chi_A(x)$, that is there is a Turing functional with oracle B that is equal to the characteristic function for A. We say A is **Turing equivalent** to B, written $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$, and in this case that A and B have the same **Turing degree**.

Definition 1.2.4. (a) Given a set A, define the **jump** of A by $A' := \{e : \Phi^A_e(e) \downarrow\}$, that is, the halting set relativized to A.
(b) Let $A^{(n)}$ denote the n**th jump** of A, i.e., $A^1 = A'$ and $A^{(n+1)} = (A^{(n)})'$.

Note that for all A, $A \leq_T A'$, but $A' \not\leq_T A$. Note also that $0' = K$, the halting set.

Definition 1.2.5. A set A is **low** if $A' \equiv_T 0'$.

Observe that for any computable set A we have $A' \equiv_T 0'$ and thus computable sets are low. However there are non-computable sets which are also low.

1.3 Reverse Math

In addition to giving characterizations of sets and structures based on complexity strength, we can also compare proof-theoretic strength. In the field of Reverse Math, we use the framework of axiomatic systems in set theory to classify theorems based on the set-theoretic axioms required to prove them. Just as it can be shown in Zermelo-Fraenkel set theory that the axiom of choice is equivalent to Zorn's Lemma, in reverse math we look at axiomatic subsystems of Z_2, second order arithmetic, to find equivalences of theorems from across mathematics. We give a basic introduction here, and the reader can find more information about the field of Reverse Math in [6].

We generally work in a base system RCA_0, a weak subsystem of Z_2 which includes a finitely axiomatized fragment of Peano Arithmetic (PA^-), along with induction on Σ^0_1 formulas (that is, formulas with only one existential quantifier) and set comprehension for Δ^0_1 formulas (that is, comprehension for computable sets).

As is the case when we study known results from a computability standpoint, we will begin with a known theorem concerning the game of cops and robbers. We attempt to determine which additional axioms A are sufficient to produce a proof over this weak base system RCA_0. Then we use RCA_0 and the theorem itself to prove the axioms of A, often called a *reversal*.

Most classical theorems involving properties of natural numbers, integers, and rational numbers are provable in RCA_0, as well as several theorems from across other areas of math including the Baire Category Theorem and the Intermediate Value Theorem. This is a relatively weak axiomatic system however, in particular when dealing with non-computable sets. For example, RCA_0 can prove neither the weak nor the strong versions of König's Lemma.

Weak König's Lemma 1.3.1. Let $T \subseteq 2^{<\omega}$ be an infinite binary tree. Then T contains an infinite path.

König's Lemma 1.3.2. Let $T \subseteq \omega^{<\omega}$ be a finitely-branching, infinite tree. Then T contains an infinite path.

We can see that these theorems are not provable over RCA_0 by showing that there is a computable tree that contains no computable path (see Chapter 2). Thus, the following axiomatic system is a proper extension of RCA_0.

Definition 1.3.3. WKL_0 is a subsystem of Z_2 consisting of all the axioms of RCA_0, as well as Weak König's Lemma.

By including this one new axiom, we find that many classical theorems are provable over WKL_0, but not over RCA_0. For example, both the Heine-Borel Theorem and the Brouwer fixed point Theorem are equivalent to WKL_0 over RCA_0; that is, they are provable in WKL_0, and if we assume either theorem as an axiom in addition to RCA_0, we can prove Weak König's Lemma.

We cannot, however, prove König's Lemma 1.3.2 in WKL_0. In this case, we require a stronger set-comprehension axiom.

Definition 1.3.4. Arithmetical Comprehension ACA_0 is a subsystem of Z_2 consisting of all the axioms of RCA_0, as well as comprehension for Σ^0_1 formulas. Note that this implies comprehension of all arithmetical formulas with no quantified set variables.

While it is not immediately clear, one can show that ACA_0 is a proper extension of WKL_0, and that it is equivalent to König's Lemma. It is also equivalent to, for example, the Balzano-Weirstrass Theorem and the fact that every countable commutative ring has a maximal ideal.

There are two more subsystems of Z_2 which commonly appear in reverse mathematics.

Definition 1.3.5. Arithmetical Transfinite Recursion ATR_0 is a subsystem of Z_2 consisting of all the axioms of RCA_0, as well as the axiom schema that any arithmetical formula can be transfinitely iterated along a countable well ordering.

Definition 1.3.6. $\boldsymbol{\Pi^1_1}$**-**$\boldsymbol{\mathsf{CA}_0}$ is a subsystem of Z_2 consisting of all the axioms of RCA_0, as well as comprehension for Π^1_1 formulas.

These 5 axiomatic subsystems make up what is called "The Big Five" in reverse math, and it can be shown that they are proper extensions of each other:

$$\Pi^1_1\text{-}\mathsf{CA}_0 \Rightarrow \mathsf{ATR}_0 \Rightarrow \mathsf{ACA}_0 \Rightarrow \mathsf{WKL}_0 \Rightarrow \mathsf{RCA}_0$$

While there are theorems in mathematics that are equivalent to other fragments of Z_2, in particular many results in Ramsey Theory, most of the results in the following chapters will have equivalences to Big Five subsystems.

Chapter 2

Infinite Trees

Recall from Section 1.1 that a tree graph is robber-win if and only if it contains an infinite path. We use this result to investigate the computability of robber-win strategies on special classes of infinite trees.

2.1 Computability Results for Infinite Tree Graphs

Definition 2.1.1. A tree graph is **locally finite** if for every $v \in G$, $N[v]$ is finite.

Definition 2.1.2. A locally finite computable graph with $V = \{v_i : i \in \mathbb{N}\}$ is **highly locally finite** if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for every n, if $E(v_n, v_m)$ holds, then $m \leq f(n)$.

Note that the second condition is stronger, since for each vertex $v \in G$ it puts a computable upper bound on the indices of the neighbors of that vertex. Theorem 1.4 yields the following result.

Theorem 2.1.3. *A locally finite tree graph is cop-win if and only if it is finite.*

Proof. This result follows directly from the fact that a locally finite tree has an infinite path if and only if it is infinite. □

While this result follows directly from Theorem 1.1.8, shifting our view towards a locally finiteness property of graphs allows us to explore this result in the context of computability theory. In particular, we will see that this characterization of cop-win locally finite trees fails if we require that the cop and robber play with effective strategies on computable graphs.

Let $T \subseteq \omega^{<\omega}$ (or $T \subseteq 2^{<\omega}$) be a tree. We view T as a graph whose vertex set are the strings in T, and whose edge relation E is defined by $E(\sigma, \tau)$ holds if and only if $\sigma = \tau$ or σ is an immediate successor or predecessor of τ on T.

Lemma 2.1.4. *Let T be a tree in $\omega^{<\omega}$ (or $2^{<\omega}$) viewed as a graph. If f_R is a robber-win strategy, then f_R computes an infinite path in T.*

Proof. Let f_C be a cop strategy such that $f_C(\lambda) = \lambda$, so that the cop starts at the root of the tree T, and f_C is distance-minimizing, i.e., the cop always moves up the tree from the root toward the robber. Let f_R be a robber-win strategy.

The cop starts at $c_0 = \lambda$, and the robber starts at some node $r_0 \in T$. Let n_0 be the distance between c_0 and r_0 (which is also the length of the string r_0 in the tree). Assume that after round k, the cop is at c_k and the robber is at r_k, with $n_k := |r_k| - |c_k|$ the distance between then. At round $k+1$, the cop moves to c_{k+1} with $|c_{k+1}| = k+1$ and $c_{k+1} \subseteq r_k$. That is, the cop moves one step towards r_k on T. Then f_R does one of the following:

1. sets $r_{k+1} = r_k$, in which case $n_{k+1} = n_k - 1$

2. moves to r_{k+1} = the unique predecessor of r_k in T, in which case $n_{k+1} = n_k - 2$, or

3. moves to r_{k+1} = some successor of r_k in T.

Because f_C moves toward the robber up the tree, f_R can act as in case 1 or 2 at most (n_0-1)-many times without losing to the cop in the next round. Therefore, there is some round k such that for every round $s > k$, f_R acts as in case 3. The sequence of nodes $r_k \subseteq r_{k+1} \subseteq r_{k+2} \subseteq \cdots$ traces a path in T. This path is computable from f_C (which is computable), f_R, and the non-uniform parameter k. Thus f_R computes a path in T. □

Lemma 2.1.5. *Let T be a tree in $\omega^{<\omega}$ or $2^{<\omega}$, and let $P \subseteq T$ be an infinite path. There is a robber-win strategy f_R such that $f_R \equiv_T P$.*

Proof. We define f_R as follows. First let $f_R(c_0) = r_0$ where $|r_0| = |c_0| + 2$ and $r_0 \in P$. Then, define

$$f_R(\langle c_0, r_0, \ldots, c_n, r_n, c_{n+1}\rangle) = \begin{cases} r_n & \text{if } c_{n+1} \notin P \\ g(r_n) & \text{if } c_{n+1} \in P \end{cases}$$

where $g : P \to P$ is defined by $g(\sigma)$ = the immediate successor of σ on P. Then we claim f_R is robber-win. Note that since r_0 is on the path P and higher in the tree than c_0, we have that the robber begins at a distance of at least 2 from the cop. First assume the cop starts at $c_0 \in P$. Then each time the cop moves toward the robber on P, the robber can increase the distance between himself and the cop by 1, and evade the cop indefinitely.

Assume instead the cop begins at $c_0 \notin P$. If the cop never enters the path P, the robber can remain on the path indefinitely. If the cop does enter the path, she must enter the path at some node c_{n+1} such that $|c_{n+1}| < |r_0| + 2$, as she would need to move toward the root node at least once. Once the cop enters the path, the robber's strategy will guarantee that the distance between the cop and robber is at least 1 indefinitely.

Clearly $f_R \leq_T P$. Furthermore, f_R computes P, as a cop using distance-minimizing strategy f_C will require that the robber can only remain in his current position for finitely many rounds. So there exists k so that for all rounds $s > k$, we have $r_{k+1} \neq r_k$. Then the sequence $\lambda = r_{i_0} \subsetneq r_{i_1} \subsetneq \cdots \subsetneq r_{i_j} \subsetneq r_k \subsetneq r_{k+1} \subsetneq r_{k+2} \subsetneq \cdots$ will trace the path P in T, where the r_{i_l} sequence consists of the initial segments of r_k.

Then $f_R \equiv_T P$. □

These lemmas allow us to prove the following results.

Theorem 2.1.6. *There exists a computable infinite locally finite tree graph G such that no computable robber strategy f_R is a winning strategy for the robber. Moreover, G can be chosen so that for all v, $|N[v]| \leq 3$.*

Proof. We rely on a classic construction of a computable infinite tree $T \subseteq 2^{<\omega}$ with no computable path; such a tree satisfies that for all v, $|N[v]| \leq 3$. By Lemma 2.1.4, any robber-win strategy computes a path of T. Thus any robber-win strategy is not computable. □

Notice that in this example, the unique path in T between the cop and the robber is computable. Thus, if both players are restricted to computable strategies, the cop is able to beat the robber on this classically robber-win graph. While the robber requires a non-computable strategy to beat the cop, the following result demonstrates that there is an upper bound on the complexity of winning strategies for the robber in locally finite graphs.

Theorem 2.1.7. *For every computable infinite locally finite tree graph, there is a robber-win strategy f_R such that $f_R \leq_T 0''$.*

Proof. Let $G = (V, E)$ be a computable locally finite tree graph. Then treating v_0 as the root node, we view G as a computable finitely branching tree. This tree has an infinite path computable in $\mathbf{0}''$ by the Kreisel Basis Theorem ([8]). By Lemma 2.1.5, there is a robber-win strategy computable in $\mathbf{0}''$. $\square$

Observe that we can use these lemmas, along with an example in [8] of a computable infinite locally finite tree such that every path $P \geq_T 0'$, to show that there is a robber-win graph such that every robber-win strategy f_R is Turing equivalent to $\mathbf{0}'$. However, when we restrict these results to highly locally finite infinite computable trees, we can lower our bound on the computability of robber-win strategies.

Theorem 2.1.8. *(1) A highly locally finite tree graph is cop-win if and only if it is finite.*
(2) There exists a computable infinite highly locally finite tree graph G such that no computable robber strategy f_R is a winning strategy for the robber.

Proof. These are special cases of Theorem 2.1.3 and Lemma 2.1.4, viewing a computable infinite highly locally finite tree graph G as a computably bounded computable subtree of $\omega^{<\omega}$. $\square$

Theorem 2.1.9. *For every computable infinite highly locally finite tree graph, there is a robber-win strategy f_R that is low, i.e. such that $(f_R)' \leq_T 0'$.*

Proof. This theorem follows from the computability theoretic result that every computable infinite highly locally finite tree has a low path. Thus this path P satisfies $P' \leq_T 0'$, and P computes a strategy f_R for the robber to find the path P. Then $f_R \leq_T P$ implies $(f_R)' \leq_T 0'$. $\square$

The results in this section imply that any robber-win tree will necessarily have a robber-win strategy that is relatively low in complexity, as it will be computable from $0'$. A natural question to ask is whether there exists a robber-win graph for which any winning robber-strategy is above $0'$, and to extend even further, whether we can construct a robber-win graph such that a winning strategy is arbitrarily complex. To that end, we introduce the hyperarithmetical hierarchy.

2.2 Robber-Win Strategies of Arbitrary Complexity

In Section 1.2, we defined the jump of a set by $A' = \{e : \Phi_e^A(e) \downarrow\}$. Iterating this process beginning with the empty set gives us the arithmetical hierarchy $0 <_T 0' <_T 0^{(2)} <_T \cdots$, and a standard fact of computability theory is that a relation is Δ_n^0 if and only if it is computable relative to $0^{(n-1)}$. Similarly, a relation is Σ_n^0 or Π_n^0 if and only if it is c.e. or co-c.e. relative to $0^{(n-1)}$, respectively. The hyperarithmetical hierarchy gives us a means to transfinitely extend this arithmetical hierarchy (see [1]).

Let $\mathcal{O}$ be Kleene's set of ordinal notations. For $a \in \mathcal{O}$, we define $|a| = \alpha \in \mathbb{ON}$ as follows:

- $|1| = 0$

- $|a| = \alpha$ implies $|2^a| = \alpha + 1$
- $|3 \cdot 5^e| = \sup\{|\varphi_e(n)| : n \in \omega\}$, provided φ_e is a total computable function satisfying $\varphi_e(n) \in \mathcal{O}$ for all n, and $\varphi_e(0) <_{\mathcal{O}} \varphi_e(1) <_{\mathcal{O}} \varphi_e(2) <_{\mathcal{O}} \cdots$

Definition 2.2.1. An ordinal α that has a notation $a \in \mathcal{O}$ is said to be a **computable ordinal**. Define ω_1^{CK} to be the least non-computable ordinal.

We use Kleene's $\mathcal{O}$ to define the sets of the hyperarithmetical hierarchy by effective transfinite recursion as follows:

- $H_1 = \emptyset$
- $H_{2^a} = H_a'$
- $H_{3 \cdot 5^e} = \{\langle i, j\rangle : i \in H_{\varphi_e(j)}\}$

Note that for any finite ordinal n, there is a unique $a \in \mathcal{O}$ with $|a| = n$, and in this case we have that $0^{(n)} = H_a$. For infinite computable ordinals α, while there are multiple notations a such that $|a| = \alpha$, it can be shown that if $|a| = |b| = \alpha$ then $H_a \equiv_T H_b \equiv_T 0^{(\alpha)}$ (see [1]).

These H-sets give us benchmarks for measuring the complexity of sets that are more complicated than $0^{(n)}$ for finite n.

Theorem 2.2.2. *For each computable ordinal α, there exists a computable infinite tree $T \subseteq \omega^{<\omega}$ such that any path $P \in [T]$ computes $0^{(\alpha)}$.*

Proof. We rely on known results in computability theory to construct such a tree. Sacks shows in [5] that for each $a \in \mathcal{O}$, H_a is a Π^0_2 singleton. This is to say that for each computable ordinal with notation a, there is a Π^0_2 formula $\varphi(Y)$ such that H_a is the unique set such that $\varphi(H_a)$ holds. Thus for each $\alpha < \omega_1^{CK}$, we have that $0^{(\alpha)}$ is a Π^0_2 singleton with an associated Π^0_2 formula that we will denote ψ_α. Then there is a computable relation R_α such that $\forall n\, \exists m\, R_\alpha(n, m,\, 0^{(\alpha)})$, and such that no other set satisfies this relation. For each computable ordinal α, we will show that there exists a computable infinite tree $T \subseteq \omega^{<\omega}$ such that any path in T computes $0^{(\alpha)}$.

We can write R_α as a predicate on finite strings $R_\alpha(n, m, \sigma, \tau)$ so that

$$X = 0^{(\alpha)} \Leftrightarrow \forall n \;\; \exists m > n \;\; R_\alpha(n, m, X \upharpoonright n, X \upharpoonright m)$$

We say that a triple (m, σ, τ) is an *n-triple* if and only if the following holds:

1. $n < m$ and $\sigma \subseteq \tau$
2. $|\sigma| \geq n$ and $|\tau| \geq m$, and
3. $R_\alpha(n, m, \sigma \upharpoonright n, \tau \upharpoonright m)$ holds.

Note that since σ and τ are coded as natural numbers, the n-triple (m, σ, τ) is also coded as a natural number.

We define $T_\alpha \subseteq \omega^{<\omega}$ as follows: for $|\delta| = k$, we let $\delta \in T_\alpha$ if and only if $\delta = \langle (m_0, \sigma_0, \tau_0), \ldots, (m_{k-1}, \sigma_{k-1}, \tau_{k-1}) \rangle$ where each (m_n, σ_n, τ_n) for $n < k$ is an n-triple, and for each $n < k - 1$, $\tau_n \subseteq \sigma_{n+1}$. Since T_α is closed under initial segments, it is in fact a tree. Furthermore, since $R_\alpha(n, m, \sigma, \tau)$ is a computable relation, T_α is a computable tree.

We claim that T_α has an infinite path. For each $n \in \omega$, let $\gamma_k = (m_n, 0^{(\alpha)} \upharpoonright n, 0^{(\alpha)} \upharpoonright m_n)$ where $m_n > n$ is such that $R_\alpha(n, m_n, 0^{(\alpha)} \upharpoonright n, 0^{(\alpha)} \upharpoonright m_n)$ holds. Then for each k, $\langle \gamma_0, \gamma_1, \ldots, \gamma_{k-1} \rangle \in T_\alpha$. Thus T_α has an infinite path.

Next we show that if P is an infinite path in T_α, then $P \geq_T 0^{(\alpha)}$. We write P as a sequence $\delta_0 \subseteq \delta_1 \subseteq \delta_2 \subseteq \cdots$, where $|\delta_k| = k$ and $\delta_k = \langle (m_0, \sigma_0, \tau_0), \ldots, (m_{k-1}, \sigma_{k-1}, \tau_{k-1}) \rangle$. Let $X = \bigcup_{k \in \omega} \sigma_k$. Then we have that $X \leq_T P$, and since $\sigma_0 \subseteq \tau_0 \subseteq \sigma_1 \subseteq \tau_1 \subseteq \cdots$, we have that $X = \bigcup_{k \in \omega} \tau_k$ also.

We claim that $\forall n \, \exists m \, R_\alpha(n, m, X \upharpoonright n, X \upharpoonright m)$. To show this, fix n. We have that δ_{n+1} ends with (m_n, σ_n, τ_n). Since (m_n, σ_n, τ_n) is an n-triple, $R_\alpha(n, m_n, \sigma_n \upharpoonright n, \tau_n \upharpoonright m_n)$ holds. But $\sigma_n \subseteq X$ and $\tau_n \subseteq X$, so $\sigma \upharpoonright n = X \upharpoonright n$, and $\tau \upharpoonright m_n = X \upharpoonright m_n$. Thus $R_\alpha(n, m, X \upharpoonright n, X \upharpoonright m)$ holds. Then we conclude that $X = 0^{(\alpha)}$. Thus any path P in T_α computes $0^{(\alpha)}$. $\square$

This result allows us to answer the question at the end of section 2.1 in the affirmative.

Corollary 2.2.3. *For each computable ordinal α, there exists a computable robber-win graph G such that any winning robber strategy $f_R \geq_T 0^{(\alpha)}$.*

2.3 Reverse Math Results for Infinite Trees

The statement that trees are cop-win if and only if they contain no infinite path has relatively weak proof-theoretic strength, as it is provable over the base axiomatic system RCA_0. However, just as viewing trees from a lens of local-finiteness allows for an examination of computability-theoretic strength of strategies, we can also see that these results are stronger in an axiomatic sense.

Note that in the context of reverse math, we have an alternate definition for highly locally finite graphs.

Definition 2.3.1. Over RCA_0, we say G is **highly locally finite** if there is a function f such that for all $v, w \in V$, if $E(v, w)$ then $w \leq f(v)$.

Observe that such a function f need not be computable; we need only to prove that such a function exists. Note that in the following proof, and in reverse math results to follow, we use $\mathbb{N}$ rather than ω to denote the (possibly non-standard) domain of the model of second order arithmetic.

Theorem 2.3.2. *(1) Over* RCA_0*, Theorem 2.1.3 is equivalent to* ACA_0*.*
(2) Over RCA_0*, Theorem 2.1.8 (1) is equivalent to* WKL_0*.*

Proof. (1) ($\Leftarrow$) Assuming ACA_0, we wish to show that a locally finite tree graph is cop-win if and only if it is finite. Let G be a locally finite tree. If G is finite, it is immediate that G is cop-win. Suppose instead that G is infinite with vertex set $\mathbb{N}$. We use the equivalence of ACA_0 to König's Lemma, which states that if $T \subseteq \mathbb{N}^{<\mathbb{N}}$ is infinite and finitely branching, it has a path. We view G as a finitely branching subtree of $\mathbb{N}^{<\mathbb{N}}$ by choosing some node, say $0 \in G$, to be the root of the tree. Then by König's Lemma G has an infinite path. Thus G is robber-win.

($\Rightarrow$) Assuming that a locally finite tree graph is cop-win if and only if it is finite over the base field RCA_0, we wish to prove arithmetic comprehension. It suffices to show König's Lemma. Let $T \subseteq \mathbb{N}^{<\mathbb{N}}$ be an infinite finitely branching tree. Then as an infinite and locally finite tree graph, it is robber-win. This implies that T has an infinite path, as the cop can use a distance-minimizing strategy and catch the robber unless the robber finds a path as in Lemma 2.1.4.

(2) ($\Leftarrow$) Assuming WKL_0, we wish to show that a highly locally finite tree graph is cop-win if and only if it is finite. Let G be a highly locally finite tree graph. If G is finite, then it is immediate that G is cop-win. Suppose instead G is infinite with

vertex set $\mathbb{N}$. We make use of the equivalence of WKL_0 to Bounded König's Lemma ([6]), which states that, over RCA_0, every infinite highly locally finite tree $T \subseteq \mathbb{N}^{<\mathbb{N}}$ has a path. By choosing an arbitrary vertex, say $0 \in G$, to be the root node of the tree, we can view G as an infinite subtree T of $\mathbb{N}^{<\mathbb{N}}$ with the property that there is a function $g : \mathbb{N} \to \mathbb{N}$ such that if $\tau \in T$ and $m < |\tau|$, then $\tau(m) < g(m)$. Such a function g comes from the definition of the function f in Definition 2.3.1. Since WKL_0 is equivalent to the statement that such bounded trees contain infinite paths ([6]), we conclude that G has an infinite path. Thus G is robber-win.

($\Rightarrow$) Now suppose over RCA_0 that a highly locally finite tree graph is cop-win if and only if it is finite. Let $T \subseteq 2^{<\mathbb{N}}$ be infinite. Then T is highly locally finite, and thus T is robber-win. As in the converse proof of 2.6.1, this implies that T has an infinite path. $\square$

These results follow easily from equivalences to König's Lemma and to Bounded König's Lemma because we can view tree graphs as subsets of $\mathbb{N}^{<\mathbb{N}}$. In the next chapter, however, we will see a generalization of these results to locally finite graphs, which are in some sense more interesting.

Chapter 3

Locally Finite Infinite Graphs

In the last chapter, we saw that if an infinite tree is locally finite, it has an infinite path and is therefore robber-win. In this chapter, we explore the more general case of locally finite infinite graphs.

3.1 Results for Locally Finite Infinite Graphs

Notice that in general, it is not enough to show that a graph has an infinite path in order for the graph to be classified as robber-win. In the example that follows, we see that a general graph with an infinite path need not be robber-win.

Example 3.1.1. The graph G consists of vertices $\{v_i, x_j : i \in \omega, 1 \leq j \in \omega\}$ with the following edge relations:

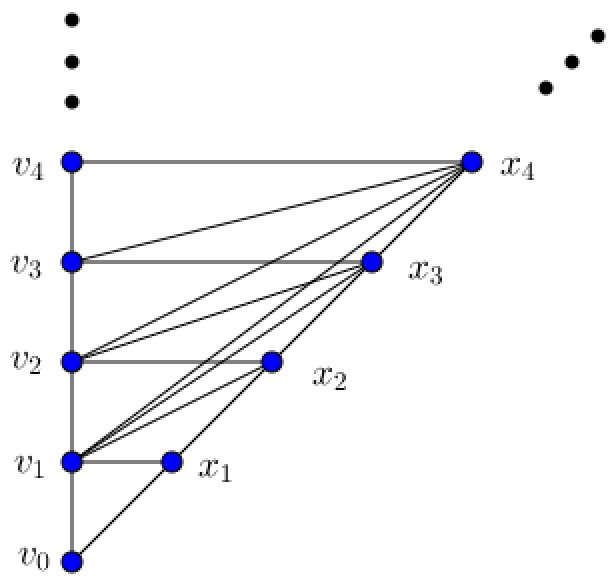
$$N[v_0] = \{v_0\} \cup \{v_1\} \cup \{x_j : 1 \leq j \in \omega\},$$

for $i > 0$

$$N[v_i] = \{v_{i-1}, v_i, v_{i+1}\} \cup \{x_j : j \geq i\},$$

and for all $j \geq 1$,

$$N[x_i] = \{v_j : j \leq i\} \cup \{x_j : 1 \leq j \in \omega\}.$$



Notice that G has an infinite path $v_0, v_1, v_2, \cdots$. However, G is cop-win. Suppose the cop begins at v_0. Then if the robber begins on x_i for any i, the cop will win in the next round. If instead the robber starts of v_i for any i, then the cop can move to x_{i+1} on her first turn. Since $N[v_i] = \{v_{i-1}, v_i, v_{i+1}\} \cup \{x_j : j \geq i\} \subseteq \{v_j : j \leq i+1\} \cup \{x_j : 1 \leq j \in \omega\} = N[x_{i+1}]$, the cop will win in the next round.

Intuitively, we can see the issue in the preceding example is the fact that the cop, in a sense, has a shortcut to get to the robber. Since v_0 has infinitely many neighbors, no matter how far along the path the robber starts, the cop has a way to reach him. Thus we might hypothesize that if a graph contains an infinite path without infinitely many "shortcuts," the robber may have a strategy to win. One way to make this idea rigorous is with the concept of local finiteness.

Definition 3.1.2. A graph is **locally finite** if for every $v \in G$, $N[v]$ is finite.

We will show that infinite locally finite graphs are robber-win through the use of a distance function on the vertices of G. Let $G = (V, E)$ be a connected graph. For $v, w \in V$, a path from v to w is a sequence of nodes $v_0, \ldots, v_n$ such that $v = v_0$, $w = v_n$, and for all $i < n$ we have $E(v_i, v_{i+1})$ holds. We say this path has length n. Then define the **distance** from v to w denoted $d(v, w) =$ least n such that there is a path of length n from v to w in G.

Observe that d is well-defined as long as G is connected. Furthermore d satisfies

- $d(v, w) = 0 \Leftrightarrow v = w$
- $d(v, w) = d(w, v)$
- $d(v, w) \leq d(v, u) + d(u, w)$ for any $v, u, w \in V$.

Theorem 3.1.3. *If G is an infinite locally finite graph, then G is robber-win.*

Proof. Let c_0 be the cop's starting vertex. Define $D_1 = \{v \in G : d(c_0, v) = 1\}$. Similarly, define $D_n = \{v \in G : d(c_0, v) = n\}$. Note that since G is connected, for every vertex $v \in G$ we have $v \in D_n$ for some n, and for all n, D_n is a finite set because G is locally finite. Furthermore, D_n is non-empty for all n, since otherwise, the largest non-empty index n, we would have the vertex set of $G = \cup_{m \leq n} D_m$ is finite.

We claim that we can find an infinite path $v_0, v_1, v_2, \cdots$ in G such that $d(v_0, v_i) = i$ for all i. To find such a path, fix c_0, and define a tree $T_G \subseteq V^{<\omega}$ as follows: let $\sigma \in T_G$ if and only if

1. for all $i < |\sigma|$, $\sigma(i) \in D_i$, and

2. for all $i < |\sigma| - 1$, $E(\sigma(i), \sigma(i+1))$ and $\sigma(i)$ is the $\leq_{\mathbb{N}}$-least element of D_i which is connected to $\sigma(i+1)$.

We claim that T_G is infinite and finitely branching. To see this, note that since each D_i is finite, T_G is clearly finitely branching. To show T_G is infinite, let $v_n \in D_n$. We show there is some $\sigma \in T_G$ such that $|\sigma| = n + 1$ and $\sigma(n) = v_n$. Define $\sigma(i)$ for $i \leq n$ by downward induction on i, maintaining that $\sigma(i) \in D_i$.

For $i = n$: Set $\sigma(n) = v_n$. Assume $\sigma(i+1)$ is defined. By construction, $\sigma(i+1) \in D_{i+1}$. Therefore there are nodes $v \in D_i$ such that $E(v, \sigma(i+1))$. Let $\sigma(i)$ be the $\leq_{\mathbb{N}}$-least such node v. This completes the definition of σ.

By construction and definition of T_G, $\sigma \in T_G$ with $|\sigma| = n + 1$. Therefore T_G is infinite. Now since T is infinite and finitely branching, it has an infinite path $v_0, v_1, v_2, \ldots$. These nodes form a path in G and satisfy $d(c_0, v_n) = n$ for all n.

We claim that this path gives the robber a winning strategy. Let the robber start on v_2. Then on the cop's first move, the cop must remain at v_0, or move to some $w_1 \in D_1$. The robber can now move to v_3. We claim that the cop can move to a vertex which is distance 2 or less from v_0; in particular, the cop cannot move to any vertex distance greater than 2 from v_0. Thus, the cop cannot win on this turn, as $d(v_0, v_3) = 3$. The robber on his turn will move to v_4. Now proceeding by induction, assume that after n rounds the cop has moved to some vertex w_n so that $d(v_0, w_n) \leq n$, and the robber has moved to v_{n+2}. On her next turn, the cop can move to some w_{n+1} that is distance at most $n + 1$ from v_0, and thus will not catch the robber in this round.

Thus the cop will never be able to occupy the same vertex as the robber and we conclude G is robber-win. $\square$

With this in mind, we can explore winning strategies for locally finite graphs from a recursion theoretic viewpoint.

3.2 Computability Results for Infinite Locally Finite Graphs

We saw in the preceding chapter that paths may be arbitrarily complex for infinite trees; in particular, for a fixed computable $\alpha \in \mathbb{O}\mathbb{N}$, there is a tree T such that any winning robber strategy computes $0^{(\alpha)}$. We see that this is not the case for locally finite infinite graphs.

Theorem 3.2.1. *For an infinite computable locally finite graph G, $0''$ can compute a robber-win strategy.*

Proof. Let $G = (V, E)$ be a computable locally finite graph. Notice that if G is computable, then the distance function $d : V^2 \to \mathbb{N}$ is computable from $0'$, because $0'$ computes the neighbor set $N[v]$ fore each $v \in V$. Thus the D_n sets are uniformly $0'$-computable for all n as well.

From G, we construct the finitely-branching infinite tree T_G as in the proof of Theorem 3.1.3. Since the construction of T_G relies on G and the distance function, we have $T_G \leq_T 0'$. Thus T_G is a $0'$-computable tree which has $0'$-computable branching, and it therefore has a path P computable from $0''$.

We claim P computes a winning strategy for G. Notice that for each i, $v_{P(i)}$ is a vertex distance i from v_0. If the cop begins at v_0, the robber can start at $v_{P(2)}$ and will be distance 2 from the cop in G. The cop must choose to move to some vertex

c_1 distance at most 1 from v_0, at which point the robber can move to $v_{P(3)}$ which is distance 3 from v_0 and thus distance at least 2 from c_1. By an inductive argument, we see that after n rounds, the cop will be distance at most n from v_0 and the robber will be able to move to $v_{P(n+2)}$ which will be distance 2 from the cop, so the robber has a winning strategy computable from $P \leq_T 0''$. $\square$

Analogous to Theorem 2.1.9, we use the construction of T in the previous proof to prove the following result.

Theorem 3.2.2. *For an infinite computable highly locally finite graph G, there is a low robber-win strategy.*

Proof. For an infinite computable highly locally finite graph $G = (V, E)$, we define T_G as in the proof of Theorem 3.1.3. We claim that since G is computable and highly locally finite, the distance function is computable and the D_n sets are uniformly computable. Thus T_G is a computable infinite finitely-branching tree. If $f : G \to \mathbb{N}$ is a function that bounds the indices of the neighbors of vertices in G, and $\sigma \in T_G$ of length n, then $\hat{f} : T_G \to \mathbb{N}$ defined by $\hat{f}(\sigma) = f(\sigma(n-1))$ places a bound on the possible successors of σ in T_G. Thus since T_G is a computable infinite highly locally finite tree, it has a low path P. An argument analogous to the proof of the last theorem shows that P computes a winning strategy for G. $\square$

Furthermore, we can generalize the reverse math results for infinite locally finite trees to infinite locally finite graphs.

3.3 Reverse Math Results for Locally Finite Graphs

Recall that over the base system of RCA_0, arithmetic comprehension is equivalent to the result that a locally finite tree is cop win if and only if it is finite. We see the following analogous result for locally finite general graphs.

Theorem 3.3.1. *The following are equivalent over* RCA_0*:*

(1) ACA_0

(2) If G is an infinite locally finite graph, G is robber-win.

Proof. $\Rightarrow$ Assume arithmetic comprehension, and let G be an infinite locally finite graph. The distance function $d : V^2 \to \mathbb{N}$ is arithmetically definable as follows:

$$d(v, w) = n \Leftrightarrow \exists\sigma(|\sigma| = n \wedge \sigma \text{ is a path from } v \text{ to } w)$$

$$\wedge\ \neg\exists\sigma(|\sigma| < n \wedge \sigma \text{ is a path from } v \text{ to } w).$$

Thus ACA_0 proves that the distance function d exists, and thus that the D_n sets exist. Then we can define a tree T_G as in the proof of Theorem 3.1.3 and we can prove in RCA_0 that this tree is infinite and finitely branching. Then by the equivalence of ACA_0 and König's Lemma, T_G has an infinite path P. This path computes a winning strategy for the robber in G.

$\Leftarrow$ In order to show arithmetic comprehension, it suffices to show König's Lemma. Let $T \subseteq \mathbb{N}^{<\mathbb{N}}$ be an infinite finitely branching tree. Then as a graph, T is infinite and locally finite, and thus is robber-win. Then T must have a path, as in the proof of Theorem 2.3.2 (1).

$\square$

Similarly, we have the following analogous result to Theorem 2.3.2 (2).

Theorem 3.3.2. *The following are equivalent over* RCA_0*:*

(1) WKL_0

(2) If G is an infinite highly locally finite graph, then it is robber win.

Proof. $\Rightarrow$ We work in WKL_0, and assume $G = (V, E)$ is an infinite highly locally finite graph. Since RCA_0 proves that the distance function d exists for a highly locally finite graph, it also proves that the D_i sets exist. As we saw in the proof of Theorem 3.2.2, the associated tree T_G is an infinite highly locally finite tree. Then by the equivalence of WKL_0 and Bounded König's Lemma, T_G has a path. This path yields a winning robber strategy.

$\Leftarrow$ Assume that any infinite highly locally finite graph G is robber-win. Let $T \subseteq 2^{<\mathbb{N}}$ be infinite. Then T is highly locally finite, and thus robber-win. This can only be true if T has a path, since otherwise a distance-minimizing strategy for the cop will be a winning one. Thus we have the result of WKL_0. $\square$

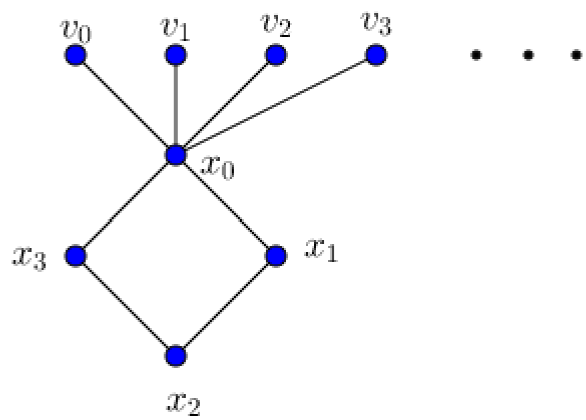
Having studied the class of locally finite graphs at length, we have only investigated robber-win graphs. We now proceed to the more general class of infinite graphs.

Chapter 4

Cop-Win Strategies for Infinite Graphs

By studying the general class of infinite (non-locally finite) graphs, we are able to see examples of infinite cop-win graphs. Notice that since Theorem 3.1.3 is not a biconditional statement, it is not the case that every infinite, non-locally finite graph is cop-win, as we see in the example below.

Example 4.0.3. In the graph G below, $N[x_0] = \{x_0, x_1, x_3\} \cup \{v_i : i \in \omega\}$. While G is not locally finite, it is also robber win, as the robber can move opposite the cop in the 4-cycle indefinitely.



However, we do know that every infinite cop-win graph is not locally finite. Recall that by Theorem 1.1.11 from [4] we have a characterization of cop-win graphs. This is a natural result to investigate from a recursion theoretic perspective, and we will show there is a graph G such that the relation $\preceq$ is trivial but such that no computable cop-strategy is a winning one. We will also see that, unlike the case of winning robber strategies, it is difficult to code non-computable information into cop strategies in cop-win graphs.

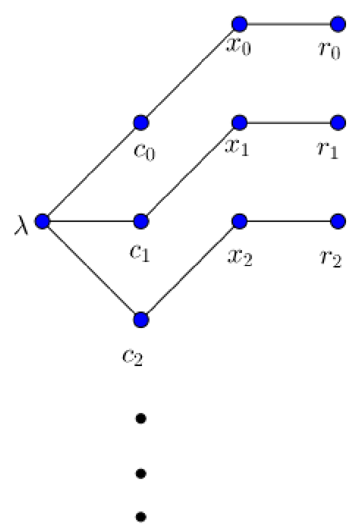
4.1 Computability Results for Cop-Win Infinite Graphs

Recall that a graph G is cop-win if and only if the relation $\preceq$, defined as in Section 1.1, is trivial on the vertices of G; i.e., for all $v, w \in G$, $v \preceq w$. We saw in Theorem 2.1.6 that this characterization fails for computable graphs if we require strategies to be computable, since there is a robber-win graph, in particular an infinite binary tree, with no computable path and thus no computable robber-win strategy. We now see a parallel result for cop-win trees using a common tool in computability theory of diagonalizing against all possible strategies.

Theorem 4.1.1. *There exists a computable cop-win graph G such that no winning cop-strategy is computable.*

Proof. We build G computably in stages in order to diagonalize against every possible computable strategy φ_e.

Stage 0: We define G_0 to be an infinitely branching tree with countably many paths of length three branching from a root λ.



We let G_s denote the graph at stage s and for a node $x \in G_s$, we let $N_s[x]$ denote the set of neighbors of x in G_s.

At any subsequent stage $s+1>0$ with $e<s$ we may or may not modify the path including vertices c_e, x_e, and r_e in order to diagonalize against φ_e. As noted in Chapter 1, we can assume without loss of generality that φ_e starts with initial cop position λ.

Stage s: For each $e<s$, we act to diagonalize against φ_e as follows:

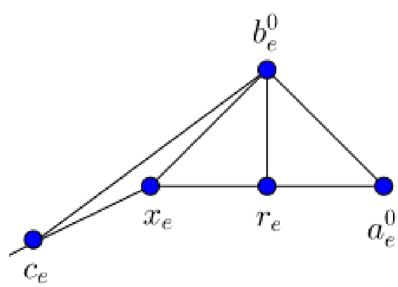
Initial Module: We consider φ_e acting on the initial cop position λ and initial robber position r_e. We want to check if φ_e describes a cop stratgy which eventually moves the cop to x_e while the robber remains at r_e. Formally, we check whether there is a sequence of stages $s_0 \leq s_1 \leq \cdots \leq s_k < s$ and a sequence of nodes $w_0 \leq w_1 \leq \cdots \leq w_k$ such that

$$\varphi_{e,s_0}(\langle \lambda, r_e \rangle) \downarrow = w_0,$$

and for $i < k$,

$$\varphi_{e,s_i}(\langle \lambda, r_e, w_0, r_e, w_1, r_e, \cdots, w_i, r_e \rangle) \downarrow = w_{i+1}$$

and $E(\lambda, w_0)$, and $E(w_i, w_{i+1})$ for $i < k$, and $w_k = x_e$. If not, we take no action for φ_e at this stage. If so, then we say $\varphi_{e,s}$ has executed a sequence of cop moves ending in x_e while the robber remains fixed at r_e. In this case, we add in vertices a_e^0 and b_e^0 as seen below.



Note that $N_s[a_e^0] = \{a_e^0, b_e^0, r_e\}$ and $N_s[b_e^0] = \{b_e^0, a_e^0, c_e, x_e, r_e\}$. These are the only new vertices we add on this particular path at this stage. In this case, the robber may move to node a_e^0 and remain distance 2 from the current cop position.

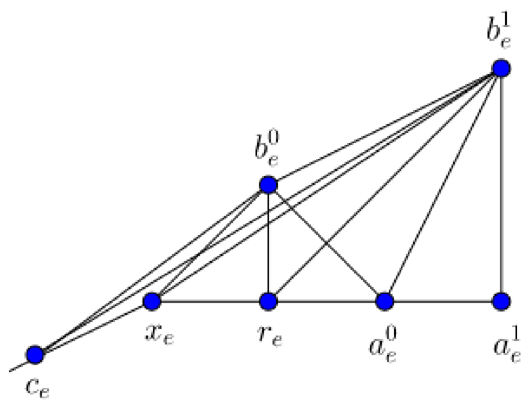
Induction Module: Assume we have added neighbors $b_e^0, b_e^1, \ldots, b_e^i$ to r_e as well as auxiliary nodes $a_e^0, a_e^1, \ldots, a_e^i$. When this module starts, in order to beat φ_e, the robber is currently at a_e^i and the cop is distance 2 from a_e^i. We check if $\varphi_{e,s}$ executes a sequence of cop moves ending in a_e^{i-1} or b_e^{i-1}, the only nodes currently at distance 1 from a_e^i, while the robber remains fixed at a_e^i as in the Initial Module. If not, then we do nothing for φ_e at this stage. If so, then we include extra vertices a_e^{i+1} and b_e^{i+1} with neighbor sets as follows:

$$N_s[a_e^{i+1}] = \{a_e^{i+1}, a_e^i, b_e^{i+1}\}$$

and

$$N_s[b_e^{i+1}] = \{x_e, r_e, c_e\} \cup \{b_e^j : j \leq i+1\} \cup \{a_e^j : j \leq i+1\}.$$

As an example, if $i = 0$, we would include a_e^1 and b_e^1 as follows.



Note that the robber can move to a_e^{i+1} to remain at distance 2 from the cop.

End construction.

We claim that $G = \bigcup_{s\in\omega} G_s$ is a cop-win graph with the property that no computable cop strategy is a winning strategy.

First we show that G is cop-win. Assume the cop begins on λ. For each initial path c_e, x_e, r_e, we either added finitely many a_e^i and b_e^i vertices, or infinitely many. First suppose the robber begins on some vertex in a path indexed e with only finitely many a_e^i vertices. Let i be the largest index such that $a_e^i \in G$. The cop moves to c_e.

Now regardless of the robber's move, the cop can move on her next turn to b_e^i. Now since $N[y] \subseteq N[b_e^i]$ for all $y \in \{c_e, x_e, r_e, b_e^j, a_e^j : j \leq i\}$, the cop will win on her next turn.

If on the other hand the robber begins on some vertex in a path indexed by e with infinitely many a_e^i vertices, the cop first moves to c_e. If the robber then occupies either x_e or any b_e^j vertex, he will lose in the next round as c_e is adjacent to x_e and every b_e^j. If instead the robber occupies a_e^j for some j, then we claim that the cop should move to b_e^{j+1} to win in the next round. This is because, since there are infinitely many a_e^i and b_e^i vertices in G, then we have

$$\begin{aligned} N[a_e^j] &= \{a_e^j, a_e^{j-1}, a_e^{j+1}\} \cup \{b_e^k : k \geq j\} \\ &\subseteq \{x_e, r_e, c_e\} \cup \{b_e^i : i \in \omega\} \cup \{a_e^i : i \leq j+1\} \\ &= N[b_e^{j+1}]. \end{aligned}$$

Note that if the robber had chosen to occupy r_e, then the cop could move to b_e^0 to win in the next round.

Thus G is cop-win. However, we claim that no computable function will be a winning strategy for the cop. To prove this, suppose by way of contradiction that the cop had some computable winning strategy f. Then $f = \varphi_e$ for some e. As before, this path c_e, x_e, r_e has either finitely many a_e^i vertices, or infinitely many.

In the first case, suppose i is the largest index such that $a_e^i \in G$. Since we never added any more a_e^{i+1} vertices to G, we know that for every stage s, φ_e^s did not move the cop within distance 1 of a_e^i. Therefore, by staying on a_e^i, the robber has a winning strategy to beat φ_e.

In the second case, suppose the cop is on c_e and the robber on r_e. We know that when following φ_e, the cop must eventually move to x_e since we only include a_e^0 if $\varphi_{e,s}$ executed a sequence of cop moves ending in x_e while the robber remains fixed at r_e. Then the robber can move to a_e^1, and remain there until the cop moves to either r_e or b_e^0. We know the cop must do that, as we would only add in vertices a_e^1 and b_e^1 after the cop moves to r_e or b_e^0 while the robber remains fixed at a_e^0.

Now we claim that by induction, the robber will always have a way to evade capture for another round from the cop using φ_e. Suppose after some rounds the robber has just moved to a_e^{i+1} at stage s. Because we added a_e^{i+1} and moved the robber to this node at stage s, the cop is currently at a_e^{i-1} or b_e^i and the nodes in the current part of the graph dedicated to e are $\{x_e, r_e, b_e^j, a_e^j : j < i+1\}$. The cop may move around these nodes (or nodes dedicated to other strategies), but must eventually move to a_e^i or b_e^{i+1} (because by assumption we eventually add a_e^{i+2}) before moving to a_e^{i+1} (since a_e^i and b_e^{i+1} are the only current nodes within distance 1 of a_e^{i+1}). At this point, we add a_e^{i+2} and b_e^{i+2} and move robber to a_e^{i+2} to increase the distance from cop to robber back to 2. Thus the cop will not win before the robber can move to a_e^{i+2} and by induction the robber can evade capture against φ_e indefinitely. $\square$

This result is the cop-win analogue of Theorem 2.1.6, as it shows the existence of a cop-win graph with no computable cop-win strategy. We further showed that for any computable ordinal α, there is a robber-win graph with the property that any winning strategy computes $0^{(\alpha)}$, which indicates that it is possible to code complex information into a robber-win strategy. Thus a natural follow-up question is to investigate whether there are cop-win graphs such that any winning cop strategy is arbitrarily complex. However, the following result suggests that this is not the case, by showing that for a cop-win graph and some fixed non-computable set, there is a cop strategy which can win against countably many robber strategies, and which does not compute A. This suggests that it is difficult to code any non-computable information into a cop-win strategy, and illustrates a lack of symmetry in the complexity of winning strategies for cop-win graphs when compared with winning strategies for robber-win graphs.

Theorem 4.1.2. *Suppose G is an infinite cop-win graph, and A is a fixed non-computable set. If $\{f_i : i \in \omega\}$ is a countable set of robber strategies, then there is a cop strategy f_C which beats each f_i strategy, and such that $f_C \not\geq_T A$.*

Proof. We index the vertex set of G by $\{v_0, v_1, \dots\}$. Recall that an allowable R-play sequence for G was defined to be a finite sequence of vertices $\langle c_0, r_0, c_1, r_1, \dots, r_n\rangle$ such that $c_{i+1} \in N[c_i]$ and $r_{i+1} \in N[r_i]$ for $0 < i < n$ which describes a finite sequence of moves in the game, ending in a robber move. Analogously an allowable C-play sequence ends with a cop move.

We build a tree $T \subseteq G^{<\omega}$ which describes all possible moves of the game. Assume the cop starts at v_0. Then we say a non-empty string $\sigma \in T$ if and only if the following conditions hold:

1. $\sigma(0) = v_0$
2. $\forall\, i < |\sigma| - 2\ (\sigma(i + 2) \in N[\sigma(i)])$
3. if $\sigma(i) = \sigma(i + 1)$ then $|\sigma| = i + 1$

Note that this tree consists of all allowable R- and C-play sequences that begin with v_0. In particular, if a path on T is finite, this implies that in some play sequence the cop has won the game.

In order to find a cop-win strategy to beat each f_i, we wish to build a subtree F of T which describes a full strategy for the cop, and such that if a robber follows f_i, the result is a finite path $\sigma \in F$ with $\sigma = \langle v_0, x_1, \ldots, x_k, x_{k+1}\rangle$ where $f_i(\langle v_0\rangle) = x_1$, and $f_i(\langle v_0, f_i(\langle v_0\rangle), x_2\rangle) = x_3$, and in general, x_{2j+1} is the output of f_i on the input $\langle v_0, x_1, \ldots, x_{2j}\rangle$, and ending in $x_k = x_{k+1}$.

To find such an F, we will define a sequence of subtrees $F_{-1} \subseteq F_0 \subseteq F_1 \subseteq \cdots$ such that $F = \bigcup_{i \geq -1} F_i$ to satisfy requirement R_e and P_e for all e, with priority order $R_0 < P_0 < R_1 < P_1 < \ldots$, with requirements defined as follows:

$$R_e : \Phi_e^F \neq A$$

and

$$P_e : F \text{ yields a cop strategy that beats } f_e.$$

Note that in order to satisfy R_e at stage $2e$, we will choose F_{2e} so that F_{2e} forces that $\Phi_e^{F_{2e}}$ is partial for all cop strategies F extending F_{2e}, or there is an x such that $\Phi_e^{F_{2e}}(x) \downarrow \neq A(x)$.

In order to build these subtrees, we define our forcing conditions to be finite approximations of a cop-win strategy as follows. A finite subtree $F \subseteq T$ is a forcing condition if

- For each $\sigma \in F$ with $|\sigma|$ even, if $\sigma * v_k \in F$, then for every $j < k$ such that $\sigma * v_j \in T$ we also have $\sigma * v_j \in F$.
- For each $\sigma \in F$ with $|\sigma|$ odd, there is exactly one v_k such that $\sigma * v_k \in F$.

The first bullet point will ensure that F takes into account every possible choice of move for a robber. The second will ensure that F gives a well-defined cop strategy in the end, as it gives only one possible move for the cop at any stage of the game. Note that since G is computable, the set of forcing conditions is also computable.

For a forcing condition F, we assume without loss of generality that if $\Phi^F(x)\downarrow$, then

1. if F queries the oracle about an odd length string and $\sigma \notin F$, then $\sigma \notin T$, and

2. if F queries the oracle about an even length string σ and $\sigma \notin F$, then either $\sigma \notin T$, or σ has the form $\tau * v_i$ and for some $v_j \neq v_i$ we have $\tau * v_j \in F$.

We claim that these two conditions will require that no extension of F could contain σ. If some extension F' of F did contain σ, then $\sigma \in T$ so we would be in case 2. However, then we have for odd length $\tau \in F$, we have both $\tau * v_i \in F'$ and $\tau * v_j \in F'$, a contradiction since strings of length $2k+1$ in a forcing condition have exactly one extension of length $2k+2$.

We now construct our forcing conditions. Let $F_{-1} = \{\langle v_0 \rangle\}$.

To satisfy R_e **for** $\mathbf{e \geq 0}$: Assume F_{2e-1} has been defined to be a forcing condition. We will define F_{2e} to be a forcing condition extending F_{2e-1} in order to satisfy R_e.

Case 1: If there is any x such that for every forcing condition F^* extending F_{2e-1} we have $\Phi_e^{F^*}(x)\uparrow$, then define $F'_{2e} := F_{2e-1}$. In this case, R_e is satisfied, as $F = \cup F_e$ will be an extension of F_{2e-1} and thus will not compute $A(x)$.

Case 2: If there is some x such that for some forcing condition F^* extending F_{2e-1}, we have $\Phi_e^{F^*}(x)\downarrow \neq A(x)$, then define $F'_{2e} := F^*$. In this case, R_e will be satisfied, as $F = \cup F_e$ will be an extension of F^*, and thus $\Phi_e^{F^*}(x) = \Phi_e^F(x) \neq A(x)$ so F does not compute A.

We claim that we must be in one of these two cases; otherwise, A would in fact be computable. If neither case 1 nor case 2 held, we would have

1. for every x, there is some forcing condition F^* extending F_{2e-1} such that $\Phi_e^{F^*}(x)\downarrow$, and

2. for every x, there is no forcing condition F^* extending F_{2e-1} such that $\Phi_e^{F^*}(x) \downarrow\neq A(x)$.

But if this is the case, then A is in fact computable by the following algorithm: search for a forcing condition F^* extending F_{2e-1} such that $\Phi_e^{F^*}(x) \downarrow$. We know from (1) that this search will terminate at some finite stage. When it does, we know by (2) that for this F^* we have $\Phi_e^{F^*}(x) \downarrow= A(x)$. Since A is not computable, this is a contradiction and thus either case 1 or case 2 must hold.

Now having defined F'_{2e} to satisfy R_e, we define F_{2e} to be a forcing condition extending F'_{2e} in order to ensure in the end that F is a full cop strategy. To that end, for every $\sigma \in F_{2e-1}$ of even length which does not yet indicate a cop win, we check to see if the least-indexed $2e$-many successors $\sigma' \in T$ of σ are in F'_{2e}. If so we do nothing. If not, we add in any missing successors to F_{2e}. Then, for each σ', we choose exactly one successor $\sigma'' \in T$ of σ' to include in F_{2e}. Notice that this will still be a forcing condition, and since it extends F'_{2e}, it will also satisfy R_e.

To satisfy P_e for $e \geq 0$: Assume F_{2e} is a forcing condition. We will define F_{2e+1} to be a forcing condition extending F_{2e} that satisfies P_e.

Recall the Play function defined in Section 1.1. Since F_{2e} is a finite partial strategy for the cop, and f_e is a full robber strategy, we have $\sigma :=$Play(F_{2e}, f_e, v_0) is a finite string representing game play when the cop follows F_{2e} as long as possible and the robber follows f_e. Thus $\sigma \in T$. Let $n = |\sigma|$.

If $\sigma(n-1) = \sigma(n)$, then F_{2e} is already defined enough to win against f_e, and thus P_e will be satisfied. In this case, we define $F'_{2e+1} = F_{2e}$. Otherwise, $|\sigma|$ is even, since the cop plays last in F_{2e}, and then f_e, which is a full robber strategy, makes one more robber move.

Let τ be the longest initial segment of $\sigma = \text{Play}(F_{2e}, f_e, v_0)$ such that $\tau \in F_{2e}$. Then the length of τ is odd and the last bit of τ represents a cop move to some vertex x_k. Define $\tau' = \text{Play}(f_{\preceq}, f_e, x_k)$ where $f_{\preceq}$ is as winning cop strategy as described at the end of Section 1.1. Since $f_{\preceq}$ is a cop-win strategy and f_e is a full robber strategy, τ' is guaranteed to be a finite string that ends in a win for the cop, say $\tau' = \langle x_k, y_1, y_2, \ldots, y_j, y_j \rangle$. Then we add the string $\tau * \langle y_1, y_2, \ldots, y_j, y_j \rangle$ to F'_{2e+1}.

In order to make F'_{2e+1} a forcing condition, we further require that if we add $\sigma * x_{k+1}$ to F'_{2e+1}, and there is some x_j whose index in G is less than x_{k+1}'s index in G such that $\sigma * x_j \in T$, we also include $\sigma * x_j \in F_{2e+1}$, as well as exactly one successor, say $\sigma * x_j x_{j+1}$ for some x_{j+1}.

Similarly, for any odd string $\sigma * x_{k+1} x_{k+2} \cdots x_{k+2i+1}$ we include in $F'_{2e+1} \setminus F_{2e}$, we also include $\sigma * x_{k+1} x_{k+2} \cdots x_{k+2i} x_l$ for any x_l who has a smaller index in G and such that $\sigma * x_{k+1} x_{k+2} \cdots x_{k+2i} x_l \in T$. Then we choose an appropriate successor in order to satisfy the second forcing requirement. This will yield an F'_{2e+1} that is a forcing condition and satisfies P_e.

Now finally we will again extend F'_{2e+1} to a forcing condition F_{2e+1} in order to ensure F yields a full strategy, as we did for the even stages. For every $\sigma \in F_{2e}$ of even length which is not yet cop win, we check to see if the first $(2e+1)$-indexed successors σ' of σ are in F'_{2e+1}. If so, we do nothing. If not, then we add in any missing successors to F_{2e+1}. Then, for each σ', we choose exactly one successor $\sigma'' \in T_G$ of σ' to include in F_{2e}. Notice that this will still be a forcing condition, and since it extends F'_{2e+1}, it will also satisfy P_e, since we will have defined the strategy enough for the cop to beat f_e.

Now we define $F = \bigcup_i F_i$. We claim that F defines a full cop strategy f_C which does not compute A, and which beats any robber strategy φ_i. By our forcing conditions, every leaf node in F ends at a cop's position. It is a full strategy: once the cop chooses v_0 to begin the game, the robber can choose any vertex v_i; we know that $\langle v_0, v_i\rangle \in F_i \subseteq F$; if $v_0 \neq v_i$, then there is a unique successor $\langle v_0, v_i, x_1\rangle \in F_i \subseteq F$ also, by the second forcing requirement. Thus $c(v_0, v_i) = x_1$. Following this argument, any possible robber move will be included in some F_s by our first forcing requirement, and since there will be either a unique successor for every string of even length, or no successors (indicating a leaf node ending with the cop's move), we can use F to determine the cop's next move.

Now since F satisfies requirement R_e for all e, we know that $F \not\geq_T A$. Then since f_C is computable from F, it follows that $f_C \not\geq_T A$.

Finally if f_e is a robber strategy, F satisfies requirement P_e, thus F contains a string which codes a cop's moves against a robber using f_e that ends in a win for the cop. Then f_C will defeat a robber playing this strategy. □

Through a standard technique in computability theory, we can easily extend the proof of the previous theorem in order to diagonalize against countably many non-computable sets $\{A_i : i \in \omega\}$, which yields the following corollary.

Corollary 4.1.3. *For a countable set $\{A_i\}$ of non-computable sets, and a countable set of robber strategies $\{f_j\}$, if G is an infinite cop-win graph, there is a strategy f_C for the cop such that f_C defeats a robber playing f_j for all j, and $f_C \not\geq_T A_i$ for all i.*

The theorems in this section show that although we can build a cop-win graph with no computable winning cop strategy, it is difficult to code non-computable information into winning cop strategies in general. However, if we relax our definition of what it means for a graph to be cop-win, we may be able to code more information into winning strategies.

We call a graph G **cop-win from $(\mathbf{c_0}, \mathbf{r_0})$** if there is a strategy f_C that wins against every possible robber strategy when we require the cop begins on c_0 and the robber begins on r_0; in other words, G is cop-win from (c_0, r_0) if $r_0 \preceq c_0$. In this scenario, we are able to code non-computable information into a winning cop strategy in the form of separating sets, a common object of study in computability theory.

4.2 Separating Sets

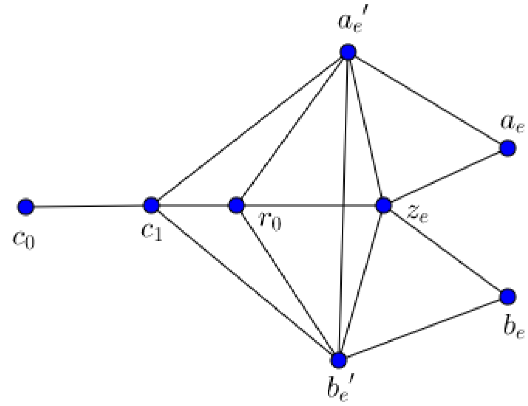
Definition 4.2.1. Let A and B be disjoint subsets of ω. Then $D \subseteq \omega$ is a **separating set** for A and B if $A \subseteq D$ and $B \cap D = \emptyset$. If no computable separating set exists for a pair A and B, we call the pair **recursively inseparable**.

The existence of a separating sets for a pair of set A and B is a rich field of study in computability. It is easy to show that there exist disjoint c.e. sets A and B such that there is no computable separating set D; i.e., A and B are recursively inseparable. However, given any pair of disjoint c.e. sets A and B, we can build a graph which is cop-win from a specified starting position (c_0, r_0), for which any winning strategy from (c_0, r_0) computes a separating set.

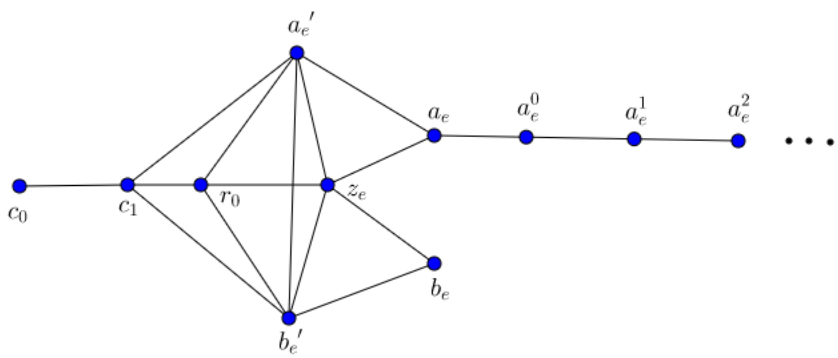
Theorem 4.2.2. *For a fixed pair of disjoint c.e. sets A, B, there exists a computable graph G such that is cop-win from (c_0, r_0), and any cop-win strategy from this starting position computes a separating set D.*

Proof. We construct the computable graph G as follows. At stage 0, define G_0 with finite path c_0, c_1, r_0. Additionally, r_0 has countably many neighbors $\{z_e : e \in \omega\}$, $\{a'_e : e \in \omega\}$, and $\{b'_e : e \in \omega\}$. We further have $E(c_1, a'_e)$ and $E(c_1, b'_e)$ for all e, and $E(a'_e, b'_e)$ for all e. Finally for each e we have a_e with neighbors $\{a_e, z_e, a'_e\}$ and b_e

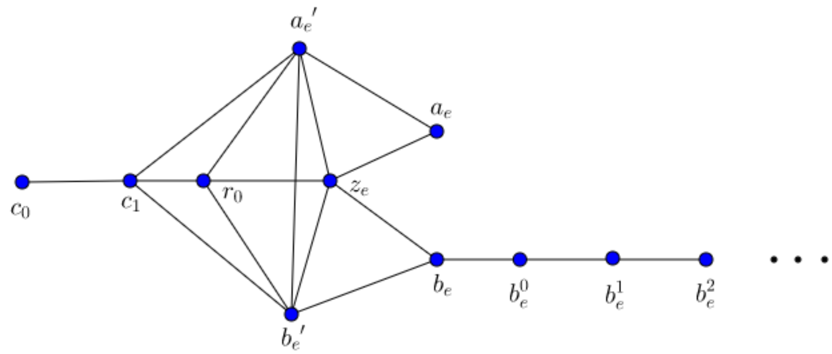
with neighbors $\{b_e, z_e, b'_e\}$. See the graph below for one of infinitely many sections of the graph.



At each subsequent stage $s > 0$, for each $e < s$ we see if $e \in A_e^s \setminus A_e^{s-1}$. If so, we include the following infinite path from a_e. After this stage, we do not add any further vertices to this part of the graph.



If on the other hand we see $e \in B_e^s \setminus B_e^{s-1}$, we include the following infinite path from b_e. After this stage, we do not add any further vertices to this part of the graph.



Note that since A and B are disjoint, we will never have infinite paths from both a_e and b_e. It is also possible that neither a_e nor b_e will have infinite paths, if $e \notin A \cup B$.

We let $G = \bigcup_{s \in \omega} G_s$, and observe that G is a computable graph.

We claim that the cop has a winning strategy from starting positions (c_0, r_0). First the cop can move to c_1. If the robber remains at r_0, or moves to c_1, a_e', or b_e' for any e, the cop will win in the next round. So assume the robber moves to z_e for some e. The cop can then move to either a_e' or b_e', depending on whether there is an infinite path $\{a_e^i\}$ or $\{b_e^i\}$. In the first case, the cop should move to a_e^i. Then the robber must move to b_e or b_e' to evade capture in the next round. But the cop can then win within two rounds by moving to r_0 and z_e if necessary.

If there is no infinite path $\{a_e^i\}$, the cop can win by moving to b_e^i and using an analogous strategy. Thus the graph is cop-win from (c_0, r_0).

Now let f_C be any winning cop strategy from (c_0, r_0), and define the set

$$D := \{e : f_C(\langle c_0, r_0, c_1, z_e \rangle) = a_e'\}.$$

Then D is clearly computable from f_C. We claim that D is a separating set for A and B; that is, $A \subseteq D$ and $B \cap D = \emptyset$.

First we show that $A \subseteq D$. If $e \in A$, we have an infinite path $\{a_e^i\}$. If $f_C(\langle c_0, r_0, c_1, z_e\rangle) \neq a'_e$, then on his next turn the robber can move to a_e. Since the cop will still be distance at least 2 from a_e, the robber reaches the infinite path and will win. Thus in order for f_C to be a winning strategy, we must have $e \in D$.

Now we show $B \cap D = \emptyset$. If not, say $e \in B \cap D$, then since $e \in B$ we have the infinite path $\{b_e^i\}$. But if $f_C(\langle c_0, r_0, c_1, z_e\rangle) = a'_e$, then the robber will move to b_e on his next turn. Once again, the cop is still distance 2 from b_e, so the robber can reach the path before the cop and win. Thus in order for f_C to be a winning strategy, we cannot have $e \in B \cap D$. So D is in fact a separating set, and any winning strategy from (c_0, r_0) computes D, a separating set for A and B. $\square$

Notice that Theorem 4.2.2 suggests the existence of a graph G which is cop-win from (c_0, r_0), with no computable winning strategy for the cop from (c_0, r_0). Given c.e. disjoint recursively inseparable sets A and B, we build a graph G as in the proof of Theorem 4.2.2. If f_C is a computable winning cop strategy from (c_0, r_0), this implies a computable separating set, a contradiction.

Recall that a graph G cop-win from (c_0, r_0) implies that $r_0 \preceq c_0$. Then this computability result for computing a separating set suggests to us that the we can code non-computable information into this relation $\preceq$. In the following chapter, we investigate computability theoretic and reverse math properties of $\preceq$, and $\leq_\alpha$ in general, at length.

Chapter 5

Properties of the Binary Relation $\preceq$

Recall from Chapter 1 that Nowakowski and Winkler give a characterization of infinite cop-win graphs in general which relies on a relation $\preceq$ on pairs of vertices. This relation is defined recursively on ordinals, with $v_1 \leq_0 v_2$ if $v_1 = v_2$, and $v_1 \leq_\alpha v_2$ if for every neighbor w_1 of v_1 there is a neighbor w_2 of v_2 such that $w_1 \leq_\beta w_2$ for some $\beta < \alpha$. The intuition behind this relation is that $v_1 \leq_\alpha v_2$ if, when the robber occupies v_1 and the cop occupies v_2, the cop is able to win the game in at most α-many rounds. Since $\beta < \alpha$ implies $\leq_\beta \subseteq \leq_\alpha$ as sets of pairs of vertices, we have that for some ordinal ρ these relations will stabilize, with $\leq_\rho = \leq_{\rho+1}$. Then $\preceq$ is defined to be this $\leq_\rho$, and Theoreom 1.1.11 shows G is cop win if and only for every $v, w \in G$ $v \preceq w$.

At the end of the last chapter, we saw that it is possible to construct graphs in a way that codes non-computable information into $\preceq$. We will extend this idea to the relations $\leq_\alpha$ in this chapter, as well as explore reverse math results for $\preceq$. Furthermore, we will investigate the least ρ such that $\preceq = \leq_\rho$ and see that we can construct graphs such that $\preceq$ stabilizes at the level arbitrarily large computable ordinals ρ.

5.1 Computability results for $\leq_\alpha$

Theorem 1.1.11 provides a complete characterization of cop-win infinite graphs, and a natural question to ask from a computability theoretic standpoint is how complicated it is to determine whether a (computable) graph is cop-win using this criteria. That is, we wish to consider the computability theoretic properties of the sets $\leq_\alpha$ for computable infinite graphs $G = (V, E)$. The answers to these questions can depend on graph-theoretic properties. In particular, we see that the set $\leq_0$ is a computable set for any graph G, as the vertex set V is computable and $v_1 \leq_0 v_2$ if and only if $v_1 = v_2$. However, $\leq_1$ is only computable for some classes of graphs.

Consider the case of computable highly locally finite graphs. Since the set $\{(v, w) \in V \times V : v \leq_1 w\}$ is equal to the set of $\{(v, w) \in V \times V : N[v] \subseteq N[w]\}$, we conclude that for highly computable graphs G, the set $\leq_1$ is computable. However, we also have $v \leq_1 w$ if and only if $\forall x(E(v, x) \to E(w, x))$, which is a Π^0_1 statement. Thus when G is not highly computable, $\leq_1$ may not be computable, as the result below shows.

Theorem 5.1.1. *There is a computable graph G such that $\{(v, w) : v \leq_1 w\}$ is non-computable, and in fact is Π^0_1-complete.*

Proof. Let $R(e, i)$ be a computable relation. We will build a computable graph G with distinguished computable sets of nodes $\{x^e : e \in \omega\}$ and $\{y^e : e \in \omega\}$ which satisfies the following property for each $e \in \omega$:

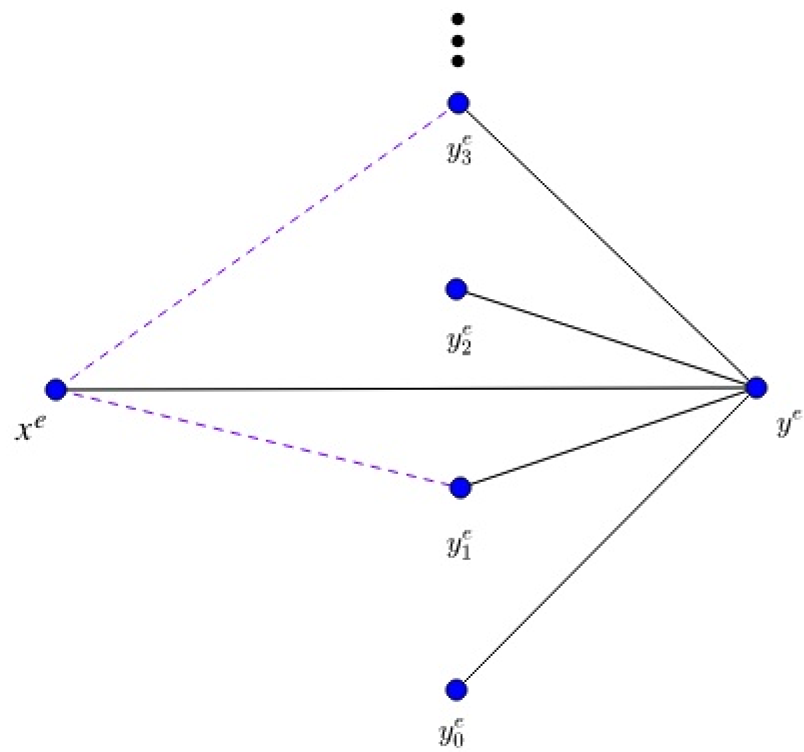
$$y^e \leq_1 x^e \Leftrightarrow \forall i(R(i,e)).$$

Observe that if this property holds, then for any computable relation $R(e,i)$ there is a computable graph for which the set $\{e : \forall i R(i,e)\}$ is 1-reducible to the set $\{(v,w) : v \leq_1 w\}$. Letting $R(e,i)$ be such that $\{e : \forall i R(e,i)\}$ is Π^0_1-complete will prove the theorem.

We construct $G = (V,E)$ with vertex set $V = \{\lambda\} \cup \{x^e : e \in \omega\} \cup \{y^e : e \in \omega\} \cup \{y^e_i : i \in \omega\}$. The edge relation is the reflexive and symmetric closure of the following conditions:

- For $e \in \omega$, $E(\lambda, x^e)$ and $E(x^e, y^e)$ hold
- For $e, i \in \omega$, $E(y^e, y^e_i)$ holds
- For $e, i \in \omega$, $E(x^e, y^e_i)$ holds $\Leftrightarrow R(e,i)$ holds.

Thus for each e, we have the following subgraph in which a dashed line between x^e and y^e_i indicates an edge if and only if $R(e,i)$:



We see then that for each e, $N[y^e] = \{y^e\} \cup \{x^e\} \cup \{y_i^e : i \in \omega\}$ and $N[x^e] = \{\lambda\} \cup \{y^e\} \cup \{x^e\} \cup \{y_i^e : R(e,i)\}$.

Since by definition $y^e \leq_1 x^e$ if and only if $N[y^e] \subseteq N[x^e]$, it follows immediately that $y^e \leq_1 x^e \Leftrightarrow \forall i \; R(e,i)$. □

Building from here, we see that $v \leq_2 w$ if and only if

$$\forall x \exists y (E(x,v) \to (E(y,w) \wedge x \leq_1 y)),$$

which is a Π_3^0 statement. We can extend this to an analogous result for Π_3^0-complete sets.

Theorem 5.1.2. *There is a computable graph G such that $\{(v,w) : v \leq_2 w\}$ is Π_3^0-complete.*

Proof. Let $R(i, j, k, e)$ be a computable relation. We will build G computably with distinguished nodes $\{x^e : e \in \omega\}$ and $\{y^e : e \in \omega\}$ such that for all $e \in \omega$

$$y^e \leq_2 x^e \Leftrightarrow \forall i \exists j \forall k (R(i, j, k, e)).$$

Observe that if this property holds, then considering a relation $R(i, j, k, e)$ such that $\{e : \forall i \ \exists j \ \forall k \ R(i, j, k, e)\}$ is Π^0_3-complete will prove the theorem, as in Theorem 5.1.1.

To build such a $G = (V, E)$, we begin with the following vertex set V which includes vertices:

- λ
- x^e and y^e for all $e \in \omega$
- y^e_i for all $e, i \in \omega$
- $y^e_{i,j}$ and $x^e_{i,j}$ for all $e, i, j \in \omega$

Then we define the edge relation E to be the reflexive, symmetric closure of the following conditions:

1. For each $e \in \omega$,

    - $E(\lambda, x^e)$
    - $E(x^e, y^e)$
    - $E(y^e, y^e_i)$ for all $i \in \omega$
    - $E(x^e, x^e_{i,j})$ for all $i, j \in \omega$
    - $E(y^e, x^e_{i,j})$ for all $i, j \in \omega$

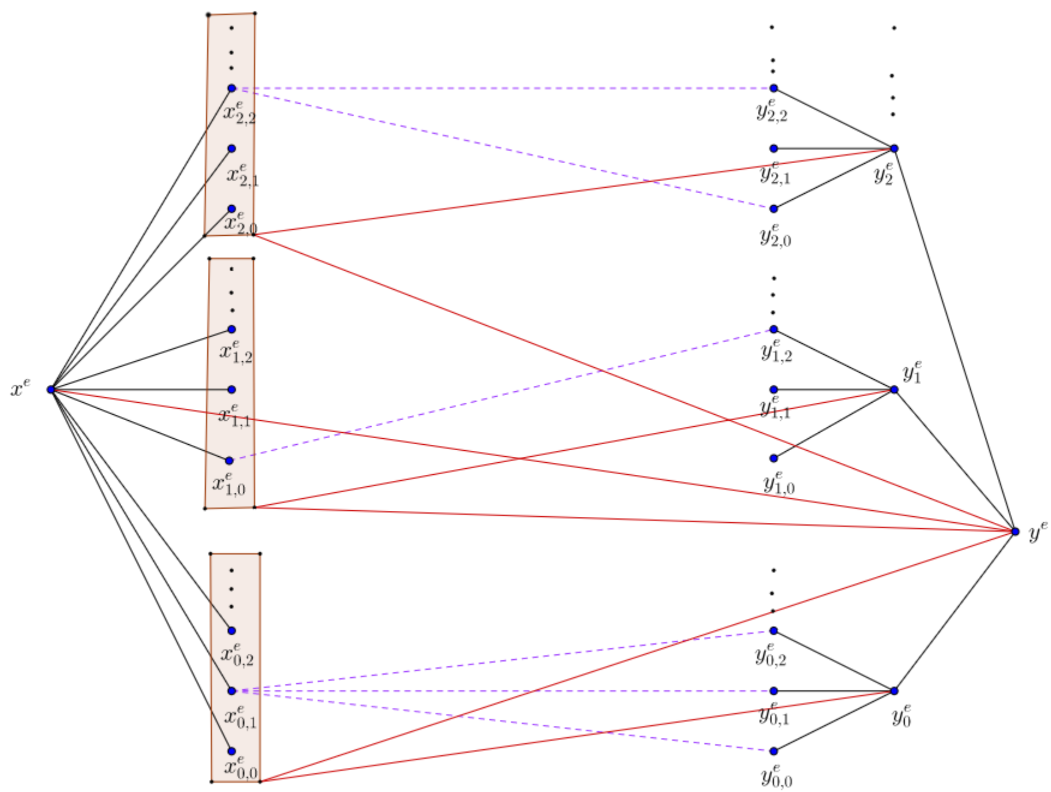
2. For each $e, i \in \omega$,

   - $E(y^e, y^e_{i,j})$ for all $j \in \omega$
   - $E(y^e_i, x^e_{i,j})$ for all $j \in \omega$
   - $E(x^e_{i,j}, x^e_{i,k})$ for all $j, k \in \omega$

3. For each $e, i, j, k \in \omega$,

   - $E(x^e_{i,j}, y^e_{i,k})$ if and only if $R(i, j, k, e)$

Then for each e, we will have the following subgraph:



Note that the dashed lines indicate that $x^e_{i,j}$ is adjacent to $y^e_{i,k}$ if and only if $R(i, j, k, e)$. The countable sets of vertices within a box represent a complete subgraph; that is, for all $e, i, j, k \in \omega$ we have $E(x^e_{i,j}, x^e_{i,k})$. Furthermore, an edge connecting a vertex to a box represents edges from that vertex to each of the countably many vertices within the box. We can summarize these edge relations by describing the neighbors of each vertex type as follows:

$$N[x^e] = \{\lambda, x^e, y^e\} \cup \{x^e_{i,j} : i, j \in \omega\}$$

$$N[y^e] = \{y^e, x^e\} \cup \{x^e_{i,j} : i, j \in \omega\} \cup \{y^e_i : i \in \omega\}$$

$$N[x^e_{i,j}] = \{x^e, y^e\} \cup \{x^e_{i,k} : k \in \omega\} \cup \{y^e_i\} \cup \{y^e_{i,k} : R(i, j, k, e)\}$$

$$N[y^e_i] = \{y^e, y^e_i\} \cup \{y^e_{i,k} : k \in \omega\} \cup \{x^e_{i,j} : j \in \omega\}$$

$$N[y^e_{i,k}] = \{y^e_i, y^e_{i,k}\} \cup \{x^e_{i,j} : R(i, j, k, e)\}.$$

We will show that $y^e \leq_2 x^e$ if and only if $\forall i\ \exists j\ \forall k\ (R(i, j, k, e))$. Observe that for fixed $e, i, j \in \omega$, $y^e_i \leq_1 x^e_{i,j}$ if and only if $N[y^e_i] \subseteq N[x^e_{i,j}]$ by definition, which is true if and only if $\forall k\ R(i, j, k, e)$.

Fix $e \in \omega$. Note that $y^e \leq_2 x^e$ if and only if for every neighbor a of y^e there exists a neighbor b of x^e such that $a \leq_1 b$. We claim that this is true if and only if, for each i, we have there is some neighbor b of x^e such that $y^e_i \leq_1 b$. The reason for this is that the other neighbors of y^e are y^e, x^e and $x^e_{i,k}$ for $i, k \in \omega$, and these are also elements of $N[x^e]$.

Now for neighbor $a = y^e_i$ of y^e, the only possible candidate for such a neighbor b of x^e with $a \leq_1 b$ is $b = x^e_{m,j}$ for some m and j. By definition of the edge relation, the only index m for which this is possible is $m = i$, and so we have $y^e \leq_2 x^e$ if and only if for all i there exists j such that $y^e_i \leq_1 x^e_{i,j}$. Thus, $y^e \leq_2 x^e \Leftrightarrow \forall i\ \exists j\ \forall k\ (R(i, j, k, e))$.

This proves the theorem. $\square$

Given the fact that we can express $v \leq_2 w$ as a Π^0_3 statement, it follows by induction on n that $v \leq_n w$ can be expressed as a Π^0_{2n-1} statement. While we might expect to be able to show for each n the existence of a graph G with the property that $\{(v,w) : v \leq_n w\}$ is a Π^0_{2n-1}-complete set, this is still currently an open question.

We may also consider a result about the complexity of the index set $\{e : G_e$ is a cop-win graph$\}$; that is, the set of indices e such that the partial computable function φ_e describes the edge-relation of a cop-win graph with vertex set $\mathbb{N}$.

Theorem 5.1.3. *The set $\{e : G_e$ is a cop-win graph$\}$ is Π^1_1-complete.*

Proof. We rely upon the fact that there is a computable sequence $\{T_e : e \in \omega\}$ of trees $T_e \subseteq \omega^{<\omega}$ such that $\{e : T_e$ has an infinite path$\}$ is a Σ^1_1-complete set ([6]).

Since trees with no infinite paths are cop-win graphs, it follows that the index set $\{e : G_e$ is a cop-win graph $\}$ must be at least Π^1_1. We can show that the index set is no more than Π^1_1 by noting that

$$\begin{aligned}
G_e \text{ is a cop-win graph} \quad &\Leftrightarrow \quad \preceq \text{ is trivial for } G \\
&\Leftrightarrow \quad \text{for every relation } R(x,y) \text{ such that} \\
&\qquad (\forall w R(w,w) \text{ holds) and} \\
&\qquad \big(\forall w \forall z \big[(\forall x \in N[w] \exists y \in N[z](R(x,y))) \to R(w,z)\big]\big), \\
&\qquad \text{we have } \forall x \forall y R(x,y).
\end{aligned}$$

To prove this equivalence, let $G = (V,E)$. We first assume for all $u, v \in V$ that $u \preceq v$, and assume that α is least such that $\preceq = \leq_\alpha$. Let R be such that $\forall w R(w,w)$, and for all w and z, if $\forall x \in N[w] \exists y \in N[z]$ such that $R(x,y)$, we have $R(w,z)$. We wish to show that for all $x, y \in V$, $R(x,y)$.

We claim that for any $\beta \leq \alpha$ and any $u, v \in G$, if $u \leq_\beta v$ then $R(u,v)$. We proceed by induction on β.

For $\beta = 0$, $u \leq_0 v$ implies $u = v$ so $R(u, v)$.

Assume for all $x, y \in G$ that $x \leq_\beta y$ implies $R(x, y)$. Suppose $u \leq_{\beta+1} v$. Then for all $x \in N[u]$ there exists $y \in N[v]$ such that $x \leq_\beta y$, which implies by induction $R(x, y)$. Then by our second assumption for R we conclude $R(u, v)$.

Finally assume for any γ less than a limit β, $x \leq_\gamma y$ implies $R(x, y)$. Assume $u \leq_\beta v$. Then $\forall x \in N[u]$ there exists $y \in N[v]$ such that $u \leq_\gamma v$ for some $\gamma < \beta$. So by induction $R(x, y)$. Then by our second assumption for $\leq$ we have $R(u, v)$. Then since for every $u, v \in V$ there is some $\beta \leq \alpha$ such that $u \leq_\beta v$, we have $R(u, v)$ for every $u, v \in V$.

To prove the other direction, assume that every binary relation R satisfying $\forall w R(w, w)$ and $\forall w \forall z (\forall x \in N[w] \exists y \in N[z] R(x, y) \rightarrow R(w, z))$ also satisfies $\forall u \forall v$ $R(u, v)$. The relation $\preceq$ satisfies $\forall w (w \preceq w)$ since $\forall w (w \leq_0 w)$. Furthermore, assume $\forall x \in N[w]\ \exists y \in N[z]\ (x \preceq y)$, and we show $w \preceq z$. Let $\preceq = \leq_\rho$ and fix $\alpha \leq \rho$ such that $\forall x \in N[w]\ \exists y \in N[z]\ (x \leq_\alpha y)$. By definition, $w \leq_{\alpha+1} z$ and we have $w \preceq z$. Since $\preceq$ satisfies both specified conditions on R, we conclude $\forall u \forall v\ (u \preceq v)$, so $\preceq$ is trivial.

Now we have shown that the statement that $\preceq$ is trivial can be expressed as a Π^1_1 statement. Thus $\{e : G_e \text{ is cop-win}\}$ is at most Π^1_1 and we conclude that it is Π^1_1-complete. $\square$

This result yields the following corollary.

Corollary 5.1.4. *The set $\{e : G_e$ is a robber-win graph$\}$ is Σ^1_1-complete.*

We have seen that the relation $\preceq$ allows us to fully characterize cop-win graphs and that this relation is not necessarily computable. However, if we restrict to computable locally finite graphs, we will see that we can put bounds on the complexity of $\preceq$. To prove this, we first give the following lemma regarding the stabilizing point for $\preceq$ in locally finite graphs.

Lemma 5.1.5. *If G is a locally finite graph, then the least α such that $\leq_\alpha = \leq_{\alpha+1}$ is at most ω.*

Proof. We will show that for every $x, y \in G$ such that $x \preceq y$, we have $x \leq_n y$ for some $n < \omega$. If this were not the case, then there is some pair $x, y \in G$ with $x \leq_\omega y$ but for all $n < \omega$ it is not the case that $x \leq_n y$.

Since $x \leq_\omega y$, then for all neighbors $x_i \in N[x]$ for $1 \leq i \leq n$ for some n, there is some $y_i \in N[y]$ for $1 \leq i \leq m$ such that $x_i \leq_{k_i} y_j$ for some $k_i < \omega$. But since x has only finitely many neighbors, there is some maximum k_i. Then we must have $x \leq_{k_i+1} y$, a contradiction.

Thus if $x \leq_{\omega+1} y$, then for all $x_i \in N[x]$ there exists $y_j \in N[y]$ such that $x_i \leq_\omega y_j$. But we have seen that this means that $x_i \leq_k y_j$ for some k. Thus $x \leq_\omega y$, and so $\leq_\omega = \leq_{\omega+1}$ and so $\preceq = \leq_\alpha$ for some $\alpha \leq \omega$. $\square$

Theorem 5.1.6. *There exists a computable highly locally finite graph such for which $\preceq\ \geq_T 0'$. Furthermore, if G is a computable locally finite graph, then $0'' \geq_T\ \preceq$.*

Proof. In this proof we use the definition of $\preceq$ as $\preceq = \leq_\alpha$ for the least α such that $\leq_\alpha = \leq_{\alpha+1}$. By Lemma 5.1.5, we know that $\alpha \leq \omega$.

We first assume that for every (highly) locally finite graph G, the set $\preceq = \{(a, b) : a \preceq b\}$ exists, and show the existence of computable highly locally finite G for which $\preceq\ \geq_T 0'$. Let G be the graph defined as follows: at stage 0, let G_0 consists of an infinite path $v_0, v_1, v_2, \ldots$ with $E(v_i, v_{i+1})$ for all i. Additionally, we add a single leaf $x_{i,0}$ adjacent to each v_i. Now at stage $s > 0$, for each $e < s$, we see if $\varphi_{e,s}(e)\downarrow$. If so, we do nothing. If $\varphi_{e,s}(e)\uparrow$, we extend the path $v_e, x_{e,0}, \ldots, x_{e,s-1}$ by adding in a vertex $x_{e,s}$ and edge $E(x_{e,s-1}, x_{e,s})$. Define this new graph to be G_s.

Note that $G = \cup_{s\in\omega} G_s$. The result will be a graph G such that if $e \in \mathbf{0}'$, the path $x_{e,i}$ will be finite, while if $e \notin \mathbf{0}'$, the path will be infinite. It is clear that G is locally finite. Thus we have that $\preceq = \{(a, b) : a \preceq b\}$. We claim that for all e, $(x_{e,0}, v_e) \in \preceq$ if and only if $e \in \mathbf{0}'$. If $x_{e,0} \preceq v_e$, then we know that if the cop occupies v_e, the robber occupies $x_{e,0}$, and it is the robber's turn, the cop will win in finitely many rounds. This is true if and only if the path $v_e, x_{e,0}, \ldots$ is finite. That is, there must be some stage s such that we stop extending the path. Then for this s we have $\varphi_{e,s}(e)\downarrow$. Thus $e \in \mathbf{0}'$. If $x_{e,0} \not\preceq v_e$, then the robber is able to evade capture indefinitely. This will occur only if there is an infinite path $v_e, x_{e,0}, x_{e,1}, \ldots$. Thus there is no stage s such that $\varphi_{e,s}(e)\downarrow$ so $e \notin \mathbf{0}'$. Thus the set $\preceq \cap \{(x_{e,0}, v_e) : e \in \omega\} = 0'$.

Now we show that if G is computable and locally finite graph, then $0'' \geq_T \preceq$.

Suppose first that T is a (highly) locally finite tree, and $x, y \in V$. We will show that $\mathbf{0}''$ can determine whether $x \preceq y$. We can consider T to be a finitely branching subset of $\omega^{<\omega}$ with root node y. Suppose we have $x = v_0, v_1 \cdots, v_k = y$ is the unique path from x to y in T. Let $i = \lceil k/2 \rceil - 1$. Then we claim that $x \preceq y$ if and only if the tree above v_i is finite.

Note that if the robber occupies x and the cop occupies y, if the cop utilizes a distance-minimizing strategy, the robber can only win by reaching a path before the cop does. The robber can only reach x_i before the cop reaches him, and so the robber can win if and only if the tree above x_i is infinite. Since $\mathbf{0}''$ can determine whether the tree above x_i is finite or infinite, we have that $\mathbf{0}'' \geq_T \{(x, y) : x \preceq y\}$.

Now suppose G is a (highly) locally finite graph. For a given $c_0, r_0 \in G$ we wish to determine whether $r_0 \preceq c_0$ using $\mathbf{0}''$. Define a finitely-branching tree $T \subseteq \omega^{<\omega}$ with root node λ_{c_0,r_0} which simulates possible game plays from these initial positions. The strings of length 1 are exactly the (finitely many) neighbors of r_0, which we will denote r_1^0, r_1^1, $r_1^2, \ldots, r_1^n$. Now the strings of length 2 will be all nodes $\sigma * c_1^i \succeq \sigma \in T$ such that c_1^i is a neighbor of c_0. We stop extending any string if the final two bits of the string represent the same vertex in G; that is, if the cop and robber occupy the same vertex. In the interest of making sure that leaves have even length, if we have a leaf of odd length $\langle r_1^{i_1}, c_1^{j_1}, \ldots, r_k^{i_k}, c_k^{j_k}, r_{k+1}^{i_{k+1}} \rangle$ where $r_{k+1}^{i_{k+1}} = c_k^{j_k}$, then we have exactly one extension by adding $c_{k+1}^{j_{k+1}} = c_k^{j_k}$. Note that in the end T will be an infinite, finitely-branching tree as G is locally finite. Any leaf of G will have even length and will represent a play history that results in a cop win.

We now define a function f used to mark certain strings in T. Define $f(*, s)$: $T_{2n} \times \omega \to \{0, 1\}$, where T_{2n} denotes the strings $\sigma \in T$ of even length, as follows:

$$f(\sigma, 0) = \begin{cases} 1 & \sigma \text{ is a leaf} \\ 0 & \text{otherwise} \end{cases}$$

Then we define $f(\sigma, s)$ by

$$f(\sigma, s+1) = \begin{cases} 1 & f(\sigma, s) = 1 \text{ or } \forall \sigma * n \in T \exists m \\ & (\sigma * n * m \in T \wedge f(\sigma * n * m, s) = 1) \\ 0 & \text{otherwise} \end{cases}$$

We will prove by induction that for each $\sigma = r_1^{i_1} c_1^{j_1} \cdots r_k^{i_k} c_k^{j_k} \in T$, $f(\sigma, s) = 1$ if and only if $r_k^{i_k} \leq_s c_k^{j_k}$. Notice that $f(\sigma, 0) = 1$ if and only if the last two bits of σ

correspond to the same vertices in G, i.e., $r_k^{i_k} = c_k^{j_k}$. Suppose now by induction that for all σ, $f(\sigma, s) = 1$ if and only if $r_k^{i_k} \leq_s c_k^{j_k}$. If $f(\sigma, s+1) = 1$, then either $f(\sigma, s) = 1$, which would imply either $r_k^{i_k} \leq_s c_k^{j_k}$ by induction, or $\forall \sigma * n \in T \exists m(\sigma * n * m \in T \wedge f(\sigma * n * m, s) = 1)$, that is for every neighbor r_{k+1}^t of $r_k^{i_k}$ there exists a neighbor c_{k+1}^r of $c_k^{j_k}$ such that $f(\sigma * r_{k+1}^t * c_{k+1}^r, s) = 1$ and thus $r_{k+1}^t \leq_s c_{k+1}^r$. This is true if and only if $r_k^{i_k} \leq_{s+1} c_k^{j_k}$.

By Lemma 5.1.5, $\preceq$ stabilizes by ω, so for all $u, v \in G$, $u \preceq v$ if and only if there is some $s < \omega$ such that $u \leq_s v$. Thus $r_k^{i_k} \preceq c_k^{j_k}$ if and only if $f(\sigma, s) = 1$ for some s.

Then we claim $\lim_s f(\sigma, s) \leq_T 0''$. Note that $f(\sigma, 0) \leq_T 0'$, since $f(\sigma, 0) = 1$ if and only if σ has no extension in T. Now since $f(\sigma, 1) = 1$ if and only if $f(\sigma, 0) = 1 \vee \forall \sigma * m \in T \; \exists \; \sigma * m * n \in T \; (f(\sigma * m * n, 0) = 1)$, we have that $f(\sigma, 1) \leq_T 0'$, as $0'$ can compute the neighbor set of each $v \in G$ and thus we have only bounded quantifiers. By induction, for each s, we have $f(\sigma, s) \leq_T 0'$.

Now since $f(\sigma, s) \leq_T 0'$ for all s, and because the limit over s exists by Lemma 5.1.5, the limit lemma relative to $\mathbf{0}'$ allows us to conclude that $\lim_s f(\sigma, s) \leq_T 0''$.

Finally, notice that $r_0 \preceq c_0$ if and only if the root node λ of this tree satisfies $f(\lambda) = 1$. This is because $f(\lambda) = 1$ if and only if for every neighbor r' of r_0 there exists a neighbor c' of c_0 such that $f(\lambda * r' * c') = 1$. This implies $\forall r' \in N[r_0] \; \exists c' \in N[c] \; (r' \preceq c')$. Thus $r_0 \preceq c_0$.

Thus for any pair of vertices $u, v \in G$ for computable locally finite G, we are able to determine from $\mathbf{0}''$ whether $u \preceq v$. $\square$

5.2 Rank Functions and the Binary Relation $\leq_\alpha$

Definition 5.2.1. If α is the least ordinal such that $\leq_\alpha = \leq_{\alpha+1} = \preceq$, we say that $\preceq$ **stabilizes** at α.

Given the intuition that in a cop-win graph if $\preceq$ stabilizes at a finite n, the cop will win in at most n rounds, we may wish to investigate possible ordinals at which $\preceq$ can stabilize. In fact, for any finite n it is easy to construct graphs such that $\preceq$ stabilizes at n. For infinite ordinals, however, it is less obvious. One way to study this is to consider rank functions for well-founded trees.

Definition 5.2.2. A tree $T \subseteq \omega^{<\omega}$ is **well-founded** if it has no infinite path.

Note that we can take any cop-win infinite tree graph T, designate some vertex λ as the root, and view the tree graph as a well-founded tree. In this context we visualize the root λ as the top of the tree and view the tree as growing downward. We call this well-founded tree T_λ, and define the rank function as follows.

Definition 5.2.3. In a well-founded tree T_λ, we say **u is below v** in T_λ if v is on the unique path from u to λ. We say u is **immediately below** v if $u \neq v$, u is below v, and $u \in N[v]$. A vertex u is a leaf of T_λ if it has nothing below it. Then for each leaf u, we define $\mathbf{rank}_\lambda\mathbf{(u)} = \mathbf{r}_\lambda\mathbf{(u)} = 0$, and in general, $\mathbf{rank}_\lambda\mathbf{(v)} = \mathbf{r}_\lambda\mathbf{(v)} = \sup\{ \text{rank}_\lambda(u) + 1 : u \text{ is below } v \text{ in } T_\lambda\}$.

Notice then that if u is below v in T_λ, then v has greater rank than u and so λ is the element of T_λ with greatest rank.

Since our well-founded trees are countable, if $r_\lambda(u) = \alpha \in \mathbb{ON}$, then $\alpha < \omega_1$. That is, the rank of each element must be a countable ordinal. For computable trees, if rank$(u) = \alpha$, then we have $\alpha < \omega_1^{CK}$; that is, each element of a well-founded tree has computable ordinal rank. We will show a connection between the rank function and the relation $\leq_\alpha$.

Theorem 5.2.4. *Let $x \in N[y]$ such that x is strictly below y in T_λ.*
(a) If $r_\lambda(x)$ is finite, then $x \leq_{r_\lambda(x)+1} y$ and for all $\beta \leq r_\lambda(x)$, $x \not\leq_\beta y$.
(b) If $r_\lambda(x)$ is infinite, then $x \leq_{r_\lambda(x)} y$ and for all $\beta < r_\lambda(x)$, $x \not\leq_\beta y$.

Proof. First, we prove (a) by induction on $r_\lambda(x)$.
Base Case: Assume $r_\lambda(x) = 0$. In this case, x is a leaf, so $N[x] = \{x, y\}$. Since $N[x] \subseteq N[y]$ we have $x \leq_1 y$. Since x is strictly below y, we have $x \neq y$ so $x \not\leq_0 y$.
Inductive Case: Assume $r_\lambda(x) = n + 1$. Note that for each w immediately below x, we have $r_\lambda(w) \leq n$ and thus by induction we have $w \leq_{n+1} x$. This gives $x \leq_{n+2} y$ as required.

To prove (b), we split into limit and successor steps, with the base case being $r_\lambda(x) = \omega$.
Base Case: Assume $r_\lambda(x) = \omega$. Let $w_0, w_1, \dots$ be the nodes immediately below x and assume $r_\lambda(w_i) = n_i$, with $\sup\{n_i\} = \omega$. Then we claim $x \leq_\omega y$, since for each $w_i \in N[x]$, we have $x \in N[y]$ with $w_i \leq_{n_i+1} x$ as a result of part (a). Since $n_i + 1 < \omega$ for all i, this yields the result.

It is clear in this case that $x \not\leq_n y$ for any n, by $w_i \in N[x]$ such that $r_\lambda(w_i) = n_i > n$.
Successor Case: Let $r_\lambda(x) = \alpha + 1$ for $\alpha \geq \omega$. Let $w_0, w_1, \dots$ be nodes immediately below x and assume without loss of generality that $r_\lambda(w_0) = \alpha$. We claim that $x \leq_{\alpha+1} y$, since for each $w_i \in N[x]$, we have either $w_i \leq_{n+1} x$ (if $r_\lambda(w_i) = n < \omega$), or $w_i \leq_{\beta_i} x$ (if $r_\lambda(x) = \beta_i \geq \omega$). In either case, $w_i \leq_\alpha x$ as required. Thus $x \leq_{\alpha+1} y$.

Furthermore we show $x \not\leq_\alpha y$. Choose $w_i \in N[x]$ strictly below x. By the induction hypothesis, $w_0 \leq_\alpha x$ and $w_0 \not\leq_\beta x$ for $\beta < \alpha$. Therefore, we cannot have $x \leq_\alpha y$, since this would force $w_0 \leq_\beta x$ for some $\beta < \alpha$.

Limit Case: Let $r_\lambda(x) = \alpha > \omega$, a limit ordinal. Again we let $w_0, w_1, \ldots$ be the nodes immediately below x and assume $r_\lambda(w_i) = \beta_i$, with $\sup\{\beta_i\} = \alpha$. By induction, we have either $w_i \leq_{\beta_i+1} x$ or $w_i \leq_{\beta_i} x$ for all i, depending on whether β_i is finite or infinite. In any case, since $\beta_i < \beta_i + 1 < \alpha$ for all i, this implies $x \leq_\alpha y$.

For $\beta < \alpha$, we do not have $x \leq_\beta y$ since taking w_i such that $\beta < \beta_i < \alpha$ gives us $w_i \leq_{\beta_i} x$. This proves the theorem. $\square$

This result now allows us to show that for a well founded tree T_λ, every element of the tree is at most $\leq_{r_\lambda(\lambda)} \lambda$. This, together with the fact that in a computable well-founded tree, all elements have computable ordinal rank, will allow us to conclude that in a computable well-founded tree, $\preceq$ must stabilize at some computable ordinal.

Theorem 5.2.5. *In T_λ, for every $x \in T$, $x \leq_{r_\lambda(\lambda)} \lambda$.*

Proof. We proceed by induction on $r_\lambda(\lambda)$.

Base case: $r_\lambda(\lambda) = 0$. In this case, λ is a leaf so $T_\lambda = \{\lambda\}$ has only one node. Therefore, if $x \in T$ we have $x = \lambda$ and so $x \leq_0 \lambda$.

Induction case: Assume $r_\lambda(\lambda) > 0$.

Subcase 1: Suppose $x = \lambda$. Then $x \leq_0 \lambda$ so $x \leq_{r_\lambda(\lambda)} \lambda$.

Subcase 2: Suppose x is immediately below λ. By Theorem 5.2.4, $x \leq_{r_\lambda(x)+1} \lambda$. But $r_\lambda(x) + 1 \leq r_\lambda(\lambda)$. Thus $x \leq_{r_\lambda(\lambda)} \lambda$.

Subcase 3: Suppose $x \neq \lambda$ and x is not immediately below λ. Let λ' be the node immediately below λ on T_λ which is on the unique path from λ to x. Let $T_{\lambda'}$ be the tree with root λ' consisting of λ' and the nodes below λ' in T_λ.

By definition, $r_{\lambda'}(\lambda') < r_\lambda(\lambda)$ and $N[x] \subseteq T_{\lambda'}$. By induction hypothesis, for all $w \in N[x]$ we have $w \leq_{r_{\lambda'}(\lambda')} \lambda'$. Therefore, we have

$$\forall w \in N[x] \exists y \in N[\lambda](w \leq_{r_{\lambda'}(\lambda')} y)$$

by choosing $y = \lambda'$. This means $x \leq_{r_{\lambda'}(\lambda')+1} \lambda$. But $r_{\lambda'}(\lambda') + 1 \leq r_\lambda(\lambda)$, so $x \leq_{r_\lambda(\lambda)} \lambda$. □

Note that to be precise, when we write $w \leq_{r_{\lambda'}(\lambda')} \lambda'$, we are calculating within the game on $T_{\lambda'}$ and not the game on T_λ. But, on a tree, the optimal cop strategy is to move on the unique path towards the robber. Therefore, by following this strategy, the robber cannot exit $T_{\lambda'}$ without meeting the cop (who starts at λ'), so $w \leq_{r_{\lambda'}(\lambda')} \lambda'$ in $T_{\lambda'}$ implies $w \leq_{r_{\lambda'}(\lambda')} \lambda'$ in T_λ for each w below λ'.

Lemma 5.2.6. *In T_λ, if $r_\lambda(\lambda) = \alpha$ is a limit ordinal, then for all $\delta < \alpha$, there is a node x immediately below λ such that $x \not\leq_\delta \lambda$.*

Proof. Suppose this property fails. Fix $\delta < \alpha$ such that $x \leq_\delta \lambda$ for all x immediately below λ. By Theorem 5.2.4, we know that for all $\beta < r_\lambda(x)$, $x \not\leq_\beta \lambda$. Therefore, we must have $\delta \geq \sup\{r_\lambda(x) : x \text{ is immediately below } \lambda\}$. However, this implies $r_\lambda(\lambda) \leq \delta + 1 < \alpha$ because α is a limit. This is a contradiciton. □

The preceding results allow us to make the following assertion about the level at which $\preceq$ stabilizes.

Theorem 5.2.7. *If T is a well-founded tree, then $\preceq$ stabilizes by $r_\lambda(\lambda)+\omega$. Furthermore, if $r_\lambda(\lambda)$ is a limit ordinal, this is the best possible bound.*

Proof. Fix $x, y \in T$. We will show that $x \leq_{r_\lambda(\lambda)+\omega} y$.

Consider a robber at x and a cop at y. The cop takes n moves to reach λ, for some $n \in \omega$. No matter where the robber moves during these n rounds, say to x', we have $x' \leq_{r_\lambda(\lambda)} \lambda$ by Theorem 5.2.5.

To see that $r_\lambda(\lambda)+\omega$ is the best possible bound when $r_\lambda(\lambda)$ is a limit ordinal, suppose the robber starts at λ and the cop starts at some y such that $d(y, \lambda) = n+1$.

Claim: If x is immediately below λ, then $\lambda \leq_{r_\lambda(\lambda)} x$ but for all $\beta < r_\lambda(\lambda)$, $\lambda \not\leq_\beta x$.

To see that $\lambda \leq_{r_\lambda(\lambda)} x$, we check

$$\forall w \in N[\lambda] \exists u \in N[x] \exists \beta < r_\lambda(\lambda)(w \leq_\beta u).$$

For any $w \in N[\lambda]$, choose $u = \lambda$ and we know $w \leq_{r_\lambda(w)+1} \lambda$. Since $r_\lambda(w)+1 < r_\lambda(\lambda)$ as λ is a limit ordinal, we are done.

To see the second part of the claim, fix $\beta < r_\lambda(\lambda)$. By Lemma 5.2.6, we can choose w such that $w \not\leq_\beta \lambda$, which suffices to prove our claim.

Now we use this claim to show $r_\lambda(\lambda)+\omega$ is the optimal bound. Let x be immediately below λ on the path to y. In n rounds, the cop moves to x and the robber stays at λ. At this point, the robber begins to move. Since this is the optimal strategy for the cop on a tree, this implies that for any α, if $\lambda \leq_{\alpha+n} y$, then $\lambda \leq_\alpha x$. Therefore, $\lambda \leq_{r_\lambda(\lambda)+n} y$ is the best bound. It follows that $\preceq$ cannot stabilize before $\sup\{r_\lambda(\lambda)+n : n \in \omega\} = r_\lambda(\lambda)+\omega$. $\square$

This theorem yields the following results given the fact that for limit ordinals, we have optimal bounds.

Theorem 5.2.8. *If T is a computable well-founded tree, then $\preceq$ stabilizes at some ordinal $\alpha < \omega_1^{CK}$.*

Proof. For a computable well-founded tree T, $r_\lambda(\lambda) < \omega_1^{CK}$, and so $r_\lambda(\lambda) + \omega < \omega_1^{CK}$ because computable ordinals are closed under addition. By Theorem 5.2.7, $\preceq$ stabilizes before $r_\lambda(\lambda) + \omega < \omega_1^{CK}$. □

On the other hand, we also know that for each computable ordinal α, there is a computable well-founded tree with root λ such that $\text{rank}_\lambda(\lambda) = \alpha$. Then we have the following result for $\leq_\alpha$

Theorem 5.2.9. *Let $\alpha \in \mathbb{ON}$ be a computable ordinal. Then there exists a tree T such that $\preceq$ stabilizes at some $\beta \geq \alpha$.*

Proof. First suppose α is a limit ordinal, and let T_λ be a well-founded tree such that $\text{rank}_\lambda(\lambda) = \alpha$. Then by Theorem 5.2.7, $\preceq$ does not stabilize until $\alpha + \omega > \alpha$.

If α is not a limit, we can consider $\alpha + \omega$, which is a limit. Then if $r_\lambda(\lambda) = \alpha + \omega$, we know that $\preceq$ stabilizes at $\alpha + \omega + \omega$. □

This tells us that there exist computable trees such that $\preceq$ stabilizes at an arbitrarily large computable ordinal level.

Bibliography

[1] Chris J. Ash and Julia Knight, *Computable structures and the hyperarithmetical hierarchy*, vol. 144, Newnes, 2000.

[2] Anthony Bonato and Richard J. Nowakowski, *The game of cops and robbers on graphs*, vol. 61, American Mathematical Society Providence, 2011.

[3] Florian Lehner, *Pursuit evasion on infinite graphs*, Theoretical Computer Science **655** (2016), 30–40.

[4] Richard Nowakowski and Peter Winkler, *Vertex-to-vertex pursuit in a graph*, Discrete Mathematics **43** (1983), no. 2-3, 235–239.

[5] Gerald E. Sacks, *Higher recursion theory*, Springer Publishing Company, Incorporated, 2010.

[6] Stephen George Simpson, *Subsystems of second order arithmetic*, vol. 1, Cambridge University Press, 2009.

[7] Robert I. Soare, *Recursively enumerable sets and degrees: A study of computable functions and computably generated sets*, Springer Science & Business Media, 1999.

[8] ———, *Turing computability: Theory and applications*, Springer, 2016.