So Long Sucker: Endgame Analysis

Marie Rose Jerade

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science Mathematics and Statistics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

"*Insanity is often the logic of an accurate mind over-tasked.*"

OLIVER WENDELL HOLMES SR.

Abstract

So Long Sucker is a strategy board game requiring 4 players, each with c chips of their designated color, and a board made of k empty piles. With a clear set-up come intricate rules, such as: players taking turns but not in a fixed order, agreements between some players being made and broken at any time, and a player winning the game even without any chips in hand.

One of the main points of interest in studying this game, is finding when a player has a winning strategy. The game begins with four players that get eliminated successively until the winner is left. To study winning strategies, it is of interest to look at endgame situations. We present the following game set-up: there are two players left in the game, Blue and Red, and only their respective chip colors. In this thesis, we characterize Blue's winning situations and strategies through inductive reasoning.

Dedications

You asked me if I could do it. I said that I wasn't sure but that I wanted to give it a try. I hope I did good. Thank you for believing in me.

In memory of Pieter Hofstra,
a beloved professor and mentor.

Acknowledgement

First and foremost, I would like to thank Professor Jean-Lou De Carufel for his academic guidance, stepping up when things took an unexpected turn, for knowing how to motivate me when I was feeling down, and for always being excited about the work we were doing. He's helped me in countless ways and finishing this thesis wouldn't have been possible without his support.

Thank you to the mathematics department for everything they do and caring about their students the way they do. There's a reason I'm sticking around for as long as I am.

Thank you to my parents and siblings for supporting me every step of the way even when they didn't know what it entailed to pursue a mathematics degree.

Thank you to my friends and colleagues that I've had the pleasure of meeting over the years here in Ottawa. You've made this journey all the more enjoyable and enriching. Everyone I've crossed paths with has helped me get to where I am today, and for that I am forever grateful.

Contents

Chapter 1

Introduction to Game Theory

1.1 Basics

Game theory is the study of strategic interactions between decision makers, commonly referred to as *players*, from a mathematical point of view. The publication of John Von Neumann's paper *On the Theory of Games of Strategy* [vNM44] in 1928 marked Game theory's recognition as a field of its own.

Traditional Game theory concentrates on studying situations purely from a rational and objective approach. This is known as a *normative theory*. However, in real life, people's behaviours tend to diverge from what is classified as rational thinking. This is where *Behavioural Game theory* comes into play. It studies the conditions that lead individuals to deviate from making rational decisions. This is known as a *positive theory*.

Games can be classified into various categories. For instance, certain games are known as *zero-sum games*, i.e. games of pure conflict, where the gains and losses of one player are balanced out by the gains and losses of all the other players. Games that don't fall into this category are known as *non-zero sum games*. Furthermore, a game is *sequential* if players take turns to make a move. Note that, the turns need not follow a fixed pattern. If the players make decisions at the same time, we say the game is *simultaneous*. For example, chess is a zero-sum sequential game, whereas the prisoner's dilemma is a non-zero simultaneous game. Moreover, a *cooperative* game is one where players are allowed to make enforceable deals among each other. Otherwise, the game is said to be *non-cooperative*.

We may give a formal mathematical definition of a *finite n-person game* as follows. We set a tuple (N, A, O, μ, u) [JLB09] such that:

- N is a finite set of n players, indexed by i;
- $A = (A_1, ..., A_n)$, where A_i is a finite set of actions available to player i, is called an *action profile*;

- O is a set of outcomes;
- $\mu : A \mapsto \mathcal{P}(O)$ determines the outcome as a function of the action profile; and
- $u = (u_1, \ldots, u_n)$ where $u_i : \mathcal{P}(O) \mapsto \mathbb{R}$ is a real-valued utility (or payoff) function for player i.

1.2 Utility Theory

Utility theory is the quantitative representation of the players' preferences. The set on which preferences are defined may be finite or infinite, and may or may not involve probabilities of outcomes. A *utility* refers to the ranking, on a scale specific to the game, that every individual allocates to certain outcomes or situations [Fis01]. Money, pride, reputation and social bonds can all be considered to be utilities. There are various explanations as to why players may deviate from rational thinking. These characteristics are known as *decision variables* and need to be taken into account when studying behavioural differences among players. There are many decision variables to consider but the most common ones are the following [Cam03]:

1. Repetition of teams and opponents in a game.
   One player's actions can differ greatly depending on the number of times they are playing against their opponents. If an individual is playing against others only once, then betrayal might be more beneficial than cooperation. However, if the same individuals play together on multiple occasions, then building a trustworthy reputation is important.

2. Communication of information.
   The way games and strategies are explained can have an impact on players' decisions. The level of information available to the players influence agreements. Additionally, the way this information is explained is also important. For instance, if there is an emphasis on a specific rule then players tend to pay attention more to that rule than others.

3. Information about players.
   What players know about each other can influence their decisions. For instance, a player might sympathize with another if they know they are in need of winning due to personal struggles. Similarly, a player may be wary of someone with a reputation to be distrustful.

4. Culture.
   Cultures across the globe value different traits and instil them in their people. For instance, in traditional Japanese culture, cooperation and generosity are

highly praised personality traits and common among their people. In comparison, western cultures tend to be more individualistic and adopt an "everyone for themselves" rule.

5. Age.
   As people get older, their perspectives tend to change drastically due to matured ideas. A few years difference between players may not make a difference. However, studies have shown that children tend to be more caring and willing to share than adults. They carry a sense of innocence contrary to adults who tend to be more skeptical.

1.3 Nash Equilibrium

A player's *strategy* is any of the chosen actions in a setting where the outcome depends on all the players' actions. Let S_i denote the set of available strategies for the i^{th} player and $S = S_1 \times S_2 \times \ldots \times S_n$ to be the *set of strategy profiles*. In other words, the elements of S are all possible combinations of individual strategies. A *mixed strategy* is a probability distribution on the set of available strategies. In this case, the strategies are picked with different probabilities. A pure strategy is one that does not involve randomization at all, instead a particular strategy is chosen all of the time. Note that a pure strategy is simply a mixed strategy, in which one strategy is chosen 100% of the time.

A *pay-off* is the quantitative representation of the outcome of a game that depends on the selected strategies of the players. We denote by $f_i(s)$ the pay-off of player i evaluated at strategy profile $s \in S$. Note that the pay-off of an individual player depends on the strategies of the other players as well. A *Nash Equilibrium* is a set of strategies that players act out, with the property that no player benefits from changing their strategy. We describe it as a strategy profile $s = (s_1, \ldots, s_n)$ with the property that $f_i(s) \geq f_i((s_1, \ldots, s_i', \ldots, s_n))$ for all players i, where $s_i' \in S_i$ denotes a strategy different from s_i available to player i.

Theorem 1.3.1 (Nash's Existence Theorem)**.** *Consider a non-cooperative game and suppose that for all players $i \leq n$, the set of strategies S_i is finite. Then there exists at least one mixed-strategy Nash Equilibrium of the game.*

Remark 1.3.2. *The proof of this theorem uses concepts found in real analysis and thus, goes beyond the scope of this thesis. Refer to the reference [JLB09] for an exhaustive paper on Nash Equilibrium.*

Definition 1.3.3. *Suppose we have a 2-person zero-sum game. Let player x choose strategy s_i, and player $y \neq x$ choose strategy s_{-i}. If $u_i(S)$ denotes the utility function*

*for player i on strategy profile s, the **minimax** of a game is defined as*

$$\overline{u_i} = \min_{s_{-i}} \max_{s_i} (u_i(s_i, s_{-i})).$$

In other words, the minimax is a strategy that a player uses in order to minimize their maximum possible loss during the game.

Theorem 1.3.4. *The minimax strategy of a 2-person zero-sum game is equivalent to the Nash equilibrium.*

1.4 So Long Sucker: A Classic Strategic Game

This thesis will tackle one game in particular, *So Long Sucker*. This game was developed by Mel Hausner, John Nash, Lloyd Shapley, and Martin Shubik in 1950 [HNSS64]. The game was invented with the purpose to encapsulate Martin Shubik and John Nash's personal belief that humans are inherently selfish and mischievous who are always looking out for their best interest. In fact, the game inspired the first episode of Adam Curtis's The Trap documentary, which looks at the role game theory played during the Cold War [CL07].

After introducing the rules of the game, we will look at some behavioural studies done using the game and introduce scenarios where a certain player can win the game. As we will see, it has proven to be useful in studying individual and group behaviour.

1.4.1 Setup

So Long Sucker (SLS henceforth) is a sequential and non-cooperative game. It consists of four players such that each one is designated by one of four colors (typically blue, green, red, and yellow) and starts the game with $\nu \in \mathbb{N} = \{0, 1, 2, \ldots\}$ *chips* of that color.

At the beginning, an SLS *board* consists of a fixed number k of empty *piles*. A *pile* is defined to be an ordered stack of chips. A player wins if they are the last player remaining in the game. As we will see later on, the number of chips is not correlated with winning the game. Similarly, a player could win the game without having any chips left in their possession.

There are two main versions to this game: Nash's and Hofstra's. We will introduce both.

Nash's Version

In Nash's SLS [HNSS64], each player starts with seven chips of their color and there is no restriction on the number of piles.
Each player is allocated 20 minutes in total to complete their moves. Once this time limit is up, the player is automatically eliminated from the game.

Hofstra's Version

In Hofstra's version [Hof18], players start with 5 or 6 chips each and the number of piles is set to 8. There is no time limit on the players or the game as a whole. Scoring is established with the i^{th} player to be eliminated getting a score of i with $i \in \{1, 2, 3, 4\}$.

Generalized Hofstra's Version

The thesis will concentrate on a generalization of Hofstra's version. Scoring is maintained. However, players start with a fixed number $c \in \mathbb{N}$ of chips each and the number of piles is set to $k \in \mathbb{N}$.

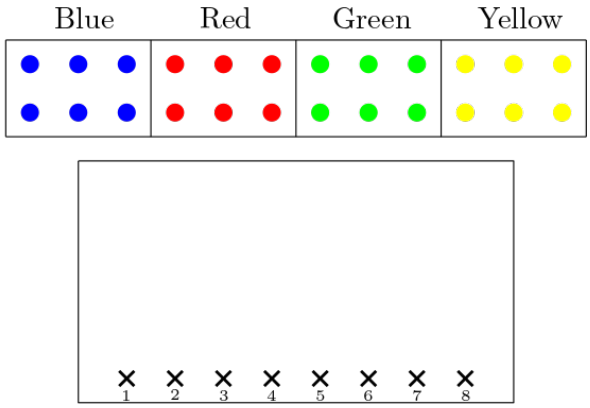

Figure 1.1: SLS Starting Setup with $k = 8$ and $c = 6$

1.4.2 Playing the Game

Definition 1.4.1. *A player's* ***move*** *consists of:*

- *placing a chip on an empty pile or on top of an existing chip on the board,*
- *picking the next player to play.*

A move is said to be valid *if the player performs the two tasks above.*
When completing a move, a player can donate chips to other players or make deals with them.

Definition 1.4.2. *The* ***played pile*** *is the pile in which the last chip was placed.*

Definition 1.4.3. *The* ***current player*** *is the player that is about to make a move.*
A player remains the current player until the next player is chosen.

Definition 1.4.4. *If a player cannot complete a valid move due to a lack of chips, with no other player willing to donate chips, then the player is* ***eliminated*** *from the game.*

Definition 1.4.5. *A player $\mathscr{X}$ may possess chips that are not of their designated color. These chips are said to be* ***prisoners*** *of player $\mathscr{X}$.*

Definition 1.4.6. *A player $\mathscr{X}$* ***discards*** *chips if they remove chips from the game entirely. A player may discard a chip if:*

- *it is a prisoner, or*
- *it is an x chip that was part of a captured pile and it is discarded during the move involving the capture.*

A discarded chip is placed in a designated area on the table called the ***deadbox*** *[HNSS64] or* ***deadzone*** *[Hof18], and are completely removed from the game.*

Definition 1.4.7. *A pile is* ***captured*** *when two chips of the same color are placed on top of each other. The chips in the captured pile go to the player associated with the color of the chip performing the capture. That player must discard at least one of the captured chips, and gets the next turn.*

The first player is chosen randomly or agreed upon among the players. From then on, the current player chooses the next player based on the following conditions:

1. If a capture takes place, then the player designated by the color making the capture has the next move. If that player is already eliminated, then the captured pile is discarded and the move goes back to the player that placed the last chip.
2. If no capture takes place, then the current player chooses the next player based on the following conditions:
   (a) The next player's color cannot be in the played pile.
   (b) If all colors are represented in the played pile, then the turn automatically goes to the player whose highest chip is the lowest among the other players' chips.
   (c) If a turn automatically goes to an eliminated player, then the turn goes to the player that gave the eliminated player their last turn.

Therefore, it is possible for the next turn to go back to the same player as long as the conditions are respected.

Example 1.4.8. *In the following scenario, it is Blue's turn to play. Blue places a yellow chip on pile 4. Since only a green chip and a yellow chip are on that pile, then Blue can give the move back to themselves.*

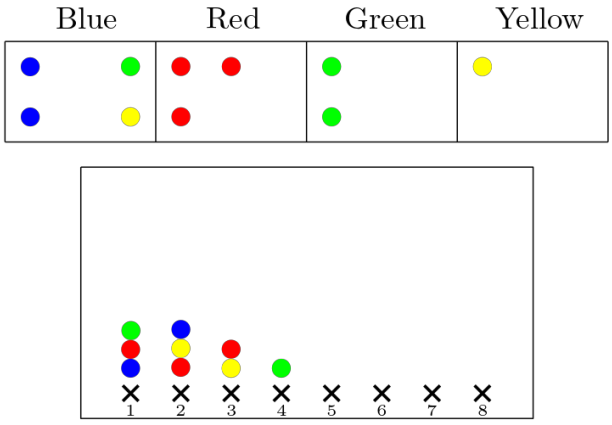

Figure 1.2: Blue's turn to play.

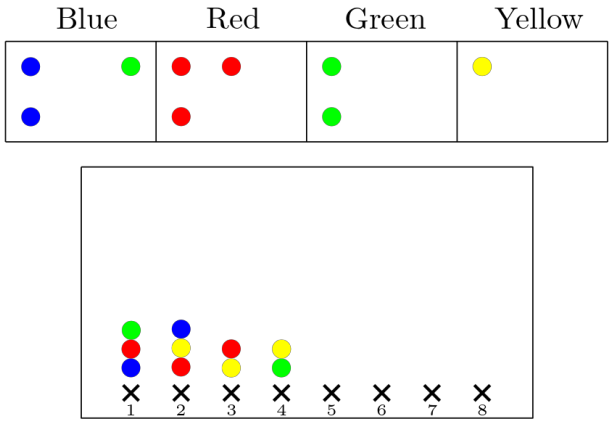

Figure 1.3: Blue places a yellow chip on pile 4.

Deal Making

When it is a player's turn to play, they may make a *non-enforceable deal* with another player. A deal must:

- be made and agreed upon in front of all players.
- remain within the current game
- be directly related to the game in play.

Although deals in SLS are not enforceable and may be broken at any time, they are necessary in order for players to move forward in the game [Shu87]. Common deals include a one-pile or two-pile capture. As we will see in the following example, the only way a capture can be guaranteed is to make a deal with another player.

Example 1.4.9. *Let us look at a situation where making a deal is necessary to guarantee a capture. In this example, it is Yellow's turn to play. Seeing that Yellow is low on chips, Red proposes to Yellow to help them capture pile 2 by placing a blue chip on an empty pile and giving the next turn to Red. As a thank you, Red would then give Yellow, the yellow chip in pile 2. Red would not have been able to capture pile 2 without Yellow's help.*

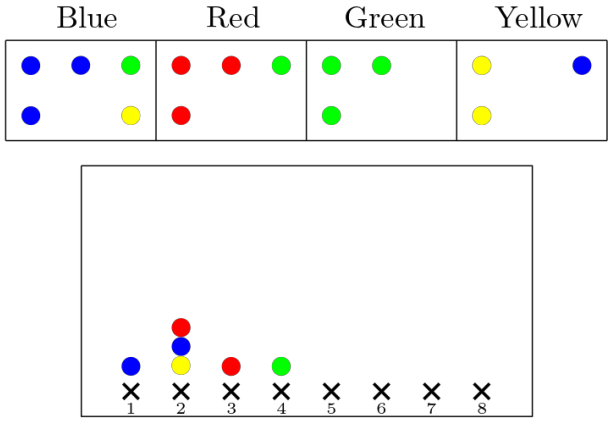

Figure 1.4: Yellow's turn to play.

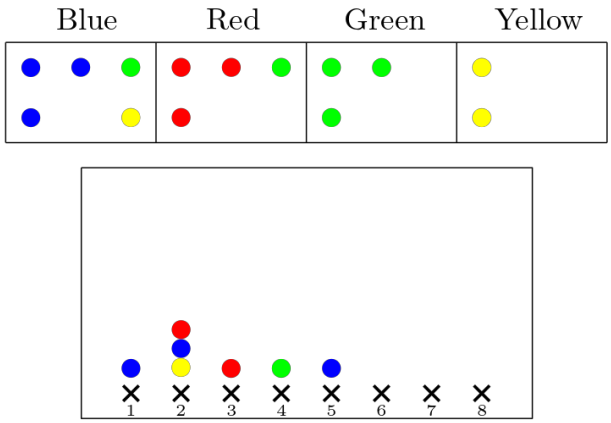

Figure 1.5: Yellow plays a blue chip on pile 5 and gives the move to Red.

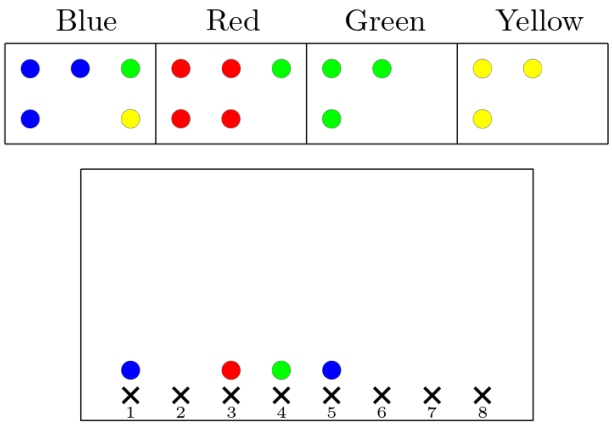

Figure 1.6: Red captures pile 2, discards the blue chip and gives the yellow chip to Yellow.

Now that we've presented all the rules to the game, we can define the game based on the definition seen earlier. SLS is a finite 4-person game described by the tuple (N, A, O, μ, u) [JLB09] such that:

- $N = \{\text{Blue, Red, Green, Yellow}\}$, such that the players are indexed by $i = 1, 2, 3, 4$ respectively;
- Action profile is $A = (A_1, A_2, A_3, A_4)$ such that $A_i = \{\text{placing a chip on board, choosing next player, discarding a chip, donating a chip, making a deal}\}$ for all i;
- $O = \{\text{pile is captured, player is eliminated, different player's turn to play}\}$;
- $\mu : A \mapsto \mathcal{P}(O)$ is the function that maps the action profile to the outcome set; and
- $u = (u_1, \ldots, u_n)$ where $u_i : \mathcal{P}(O) \mapsto \mathbb{R}$ is the real-valued utility function for player i. As seen in the previous section, the utility value is relative and depends on many factors. Although we will be discussing the behavioural aspect seen through the game, we are not interested in setting a concrete utility function.

Chapter 2

SLS in Behavioural Research

So Long Sucker is distinguished from other simpler games, such as the Ultimatum game, by the complexity of the rules and the variety of choices in moves and strategies. This brings the game closer to real life negotiations. However, we do not disregard the fact that the scenarios are indeed simplified compared to actual deals made on a daily basis, whether between acquaintances or leaders. The game's rules make it necessary, but not sufficient, to form coalitions in order to win. According to Shubik, as players get eliminated further into the game, a player will have to "double-cross" their partner to win [Shu87], [Shu02].

Broken deals can lead to deception and distrust among players. However, players who play more than once against each other will try to build a trustworthy reputation in order to make successful deals. Contrary to that, players that know they will not be playing together again might not see a need to gain others' trust and will concentrate on personal gain. The motivation for deception is stronger in these situations. Moreover, optional cooperation allows an observer to study individual and group decisions simultaneously. Generally speaking, players tend to decide quicker on their own as they do not need to consider others' interests. In comparison, group decisions have to be profitable for everyone involved in order to be agreed on, and hold everyone involved accountable. For that, group decisions take longer to be settled. However, combining the individual skills of those in the group can lead to better choices as everyone inputs a skill that may be lacking in others [Cam03].

In this section, we will highlight some of the behavioural research conducted using SLS. As a matter of fact, very little research has been published regarding this game. We will summarize three different instances that emphasize the game's potential in understanding people's behaviour.

Disclaimer: This chapter is merely a summary of previous research done by their respective authors. The thoughts and results are a rendering of what is found in the papers and poster.

2.1 Effects of Gender in a Competitive Setting

Pieter Hofstra, Silvia Bonaccio, Celine Blanchard and their team were the first to properly document participants playing the game in order to study their behaviour. The goal of this study was to observe and analyse various personality traits that may or may not have been exhibited by participants while playing the game. Here, we would like to outline some of the key results reflected in their conference presentation [LGH+15].

One particular aspect of their research concentrated on the effects of gender on decision making in a competitive and socially reliant environment . The results showed that male and female participants did not change their game play based on the other players' gender. Additionally, participants' willingness to trade with the opposite gender and the overall rank of each player did not differ significantly by gender. However, results did show that male participants were more likely to be reported as leaders and to report that gender did not have an effect on the way they played the game. Moreover, males and females were equally as likely to trust either gender.

Playing SLS places the participants in a relaxed and simpler environment than at work or during many social settings. However, this does not disregard the potential of SLS in helping researchers understand behaviour in various situations. Future research could use results found while playing the game to compare them to participants' behaviour in competitive environments such as the workplace or school.

2.2 A Clash of Cultures

In this section, we will encapsulate a paper [HTM11] that documented a social experiment that studied American and Taiwanese students' behaviour as they played Nash's version of SLS. This study took place in 2006 at Eastern Washington University. The researchers kept in mind Shubik's vision that the rules of the game are made to encourage selfish behaviour and deceit. They wanted to see how this was reflected in players of different cultures. In fact, many differences were observed between the players of both cultural backgrounds.

Based on the observations from this study, culture can have a profound influence on a player's game-play. It can have an effect on how people act and think. However, it is important to note that although the observations from this study are note-worthy, the sample size was not large enough to generalize cultural traits.

The rules of the game were given as homework one week in advance for the students to analyse. In class, the four-player groups for each game varied to allow all the students to practice the rules. We will go over the professors' observations made during play and the feedback provided by the students after playing.

Gert Jan Hofstede had established six dimensions of culture, meant to classify

the unwritten rules of a social game, such as learned attitudes, customs, and values. We will go over the six points presented in the paper [HTM11] and what was observed during play.

1. **Power Distance**
   All participants were MBA students and therefore there was no prominent power dynamics observed. However, it is worth noting that the American students were more relaxed throughout the experiment, even when it came to talking to the professors during the debriefing. The Taiwanese students took the game sessions very seriously and were more respectful to the instructor due to their perceived status difference. Additionally, when one culture was more prominent than the other, there was a shift in the game speed. When Americans dominated the game, they played faster with minimal negotiations. However, when Taiwanese dominated, they negotiated very thoroughly and took longer to make their move.

2. **Uncertainty Avoidance**
   The Taiwanese were not familiar with playing a game as part of school work and found this coursework to be stressful compared to their American peers. Moreover, American players were not wary of making deals with those that double crossed them previously. They were able to distinguish that this was only a game that had no effect on life on a greater scale. Whereas the Taiwanese students were hesitant to form an alliance with those that betrayed them.

3. **Individualism**
   Americans aimed to win as individuals, whereas Taiwanese were helping each other and did not adapt to selfish intentions. They provided each other with advice with a common goal of keeping others in the game as long as possible. They insisted on always seeking a team solution. The paper suggests that this reflects the individualist western culture versus the collectivistic Asian culture.

4. **Masculinity**
   Lack of empathy for the losing players is commonly seen as a masculine trait. American students were observed to be more competitive and goal-oriented than their Taiwanese counterpart who were more nurturing of the group as a whole. In fact, when the groups were balanced, American students learned from their Taiwanese classmates. They became more assertive and quickly formed coalitions before aggressively eliminating the opponents, showing no remorse. Comparatively, the Taiwanese students always persisted in seeking a team solution.

5. **Long-term Orientation**
   Taiwanese players were interested in long-term goals by taking less risks, engaging

in minimal double-crossing, and remaining concentrated on the game. In comparison, Americans were more interested in short-term gratification by engaging in light-hearted chitchat and taking bigger risks for smaller pay-offs. Americans did not take betrayals as seriously as Taiwanese, having no problem in making a deal with a double-crosser.

6. **Indulgence**
   Generally speaking, western culture is known to be more indulgent than east-Asian culture. This is something that was reflected in the games. The American education system and work culture tend to mix work and entertainment. Whereas Asian culture emphasizes on one's work duties and places entertainment in second place. These traits were observed as American participants engaged in small talk, whereas the Taiwanese students were heavily concentrated on the game and their conversations did not divert from the subject of the game. This mirrors Asian culture, where there is a need to follow the usual rules of appropriate behaviour, including dutifulness, and avoidance of open confrontations. Overall, the American students seemed more at ease throughout the games.

The feedback sessions pushed the students to think about their observations. In fact, both culture groups were bothered by the others' way of playing. Taiwanese felt more prepared as they had taken the game as seriously as a graded homework. However, the Americans learned quicker from their mistakes and were frustrated that the Taiwanese did not play faster.

Overall, the behavioural differences were remarkable. The authors of the paper note that this is to be expected from participants from two countries that differ greatly on many cultural aspects. However, the players' performances are not only influenced by culture. For instance, the level of prior exposure to games in an educational setting, and the fact that the Americans were in their home country, could explain how comfortable they felt. To assess the importance of such contextual factors, it is of interest to conduct similar studies in Taiwan.

Conceptually, the game is simple and straightforward, with clear rules, roles, and incentives. However, as one starts to play, they realize that there is a lot of flexibility on how one should play. To Shubik [Shu05] , this combination produced intriguing results, provoked basic questions from the students, and allowed for rich social behaviours by the participants. Overall, the game is meant to reflect the artificial social world with rules and regulations, as the participants involved influence the outcome of the game.

2.3 Bad Bargaining and Betrayal

In the paper *So Long Suckers: Bargaining and Betrayal in Breaking Bad* [GP17], the authors make a parallel comparison of the SLS rules with scenes from the American

TV show *Breaking Bad*. As we have seen so far, SLS is an entertaining way of portraying strategic interactions and practicing negotiation skills. In this section, we will look at common features between the board-game and the television series. It is important to note before discussing the show, that this section may contain spoilers to those that have not seen the show. The show follows Walter White, a high school chemistry teacher that is diagnosed with cancer. Desperate, Walter partners with Jesse Pinkman, an amateur drug-dealer and one of his former students, to produce and sell meth in order to save money for cancer treatment and for his family once he passes.

As we know, the number of players in SLS is typically 4, but could be more. Guerra-Pujol [GP17] sets the 4 main characters from the show to be: Walter White, Jesse Pinkman, Gustavo Fring, a business man and drug lord; and Mike Ehrmantraut, a private investigator of Gustavo. In SLS, all players start with the same number of chips that are visible at all times to everyone. This feature is a bit far from reality, where people do not usually start off with the same skills or opportunities. For instance, Walter White has the skills of a chemist, skills that Jesse does not have. Jesse is however street smart and has connections that Walter does not have. Gus is a well established man with a lot of money and influence, and Mike is willing to do the dirty work of others. Additionally, in Breaking Bad, as in real life, people do not usually want to expose all of their skills as this makes the players appear weak and vulnerable. Another difference that is observed, is that in Breaking Bad, new players keep appearing throughout the course of the show. Whereas in SLS, the players in the game do not change from the initial set-up.

The placing of chips on an already present pile can illustrate a person's attempt to initiate contact with someone. Whereas starting a new pile may reflect a player's wish to maintain distance and awaiting other player's actions. In Breaking Bad, Walter approaches Jesse first with a proposal to start a drug business together. Walter initiates contact, and awaits Jesse's response. As Jesse responds positively, an alliance is made. This brings us into the deal-making aspect. In SLS, deals are unenforceable, double-crosses are common, and they are made in front of everyone. Comparatively, deals are made in secrecy in Breaking Bad. Walter double-crosses Jesse, Gus and Mike throughout the show. The only way he managed to get away with these betrayals is with the help of another player at every instance. These betrayals would not have succeeded if the deals were made in front of everyone. While players in SLS bargain with chips, players in Breaking Bad bargain with information. For example, in the last season of the show, Walter knows the location of Jesse's girlfriend and Jesse knows where the stash of money is buried. They both try to use these bits of information to their advantage when bargaining.

The order of play in SLS gives the player a bit of control in choosing who plays next with some predetermined restrictions. In Breaking Bad, as the meth trade is an illegal trade, the players find themselves in a lawless world. All players can do what

they want, but understand there can be serious consequences. These consequences however, are not legally recognized consequences and thus not predetermined. For instance, Walter decides to enter the methamphetamine business out of his own freewill and to suffer the consequences.

In SLS, there are instances of chips being captured and killed. If a player captures some chips, they have to kill some chips and they have a choice in what to kill. This is reflected in the show when two DEA officers are captured and subsequently killed, or when Jesse is captured and becomes Uncle Jack's (a new drug dealer that appears later on in the show) prisoner. Earlier in the show, Gus had betrayed all the players involved and could no longer be trusted. That is when all the players teamed up to have him killed.

The winner of SLS is not the one with the most chips, but the one that remains standing till the end. This feature is perfectly reflected in the show. In the last episode of the series, Mike and Gus are dead, and Jesse is Uncle Jack's prisoner. Walter uses his last resources, or "bargaining chips", ultimately sacrificing himself to save Jesse. Jesse wins the game of Breaking Bad despite having gotten weak while a prisoner, and having no money or mental strength to continue.

Chapter 3

SLS: The Program

3.1 Background

When playing SLS, the subtlety of the rules makes it hard to keep track of everything, such as remembering who gave a player the move last or if the rules are being respected. Recall that behavioural game theory revolves entirely around people's behaviour, and what makes them diverge from theoretical and rational play.

For that matter, we implemented a code that allows users to play the game (Appendix A.5). Users have the option of starting with an empty board and a default number of chips, or they can choose their own set-up. Additionally, a user can use this feature to manually transcribe a game as it is being played. It tracks all the steps of a game and verifies that each move made is valid.

Moreover, the code has the option of implementing the minimax principle to any 2-player situation (Appendix A.6). It takes as input a table set-up, the status of the only two remaining players in the game, and then outputs which of the two remaining players has a winning strategy.

In both cases, the program implements Hofstra's version of the game. Recall that this implies that the number of rows is fixed to 8, that the players' turns are not timed, and that scoring is established. One minor difference in which the program differs entirely from the game, is that players are only asked if they'd like to donate or discard chips right before it is the next player's turn to play. This does not take into consideration the instances when a capture or elimination take place. This variation does not change strategies or outcomes of a move. It was solely done to simplify the coding process.

Having the program on hand allows researchers to concentrate on the players' behaviour without worrying about whether or not the rules are being followed. Additionally, the program facilitates the possibility of testing new rules, such as enforcing deals. It can be used as a tool to analyse players' game-play in order to detect patterns and individual strategies.

3.2 Coding in Java

We would like to give a brief overview of the program and how it is structured. In order to do that, we need to go over some crucial programming terms. In computer science, a *variable* refers to a location in which data can be stored, whereas an *object* is an entity described using a specific set of variables. Moreover, a *method* refers to a block of code that performs a specific task every time it is referenced in a program. When objects and methods that share common ground in terms of the data stored in them are grouped together, they are referred to as a *class*. Classes are categories, and objects are items within each category.

The code of the program is written in Java, an object-oriented programming language. This is a type of software design that models characteristics of abstract and real entities using classes and objects. The coding of new classes were needed to properly implement the program. The various classes allow the code to be well-structured, straightforward, and easier to follow. In this section we will introduce the main classes `Table`, `Player`, and `Move`; along with their most important methods.

Class Table (Appendix A.1)

- `Table()` method initializes an *SLS board* represented by an array with 8 empty elements, representing the piles.
- method `sort(int[] t)` sorts the elements of Table t by length.

Class Player (Appendix A.2)

- `Player(String color, int b, int r, int g, int y, int score, String before)` method initializes *players* of the game, that are represented by object Player. Each of these objects stores the color identifying the player, number of chips Player has of each color, the players' score and the player that gave them the last move.
- `capture(String captured_row)` method is called whenever a pile is captured by a player. The method stores all chips in captured pile with the player making the capture. Later, the player is prompted to discard one or more chips from the captured pile.
- `discard(String discarded_chip, int num)` method is called to discard `num` chips of color `discarded_chip`.
- `eliminated()` method checks if a player is eliminated or not.
- `donate(Player other, String color, int chip_number)` method is called when a player wants to donate `chip_number` chips of color `color` to player `other`.

- method `nextPlayer(int captureturn, String played_row, LinkedList<String> remaining_players, Player[] players, String chipPlayed, String currentPlayer)` returns string containing all eligible players for next turn. It looks at the following information to return eligible next players: if a capture took place, which players are still in the game and which colors are available in the played pile.

Class Move (Appendix A.3)

- `Move(int playerCurrent, int row, String chipPlayed, int capture, boolean elimination, int playerNext, String discardingChips, String discardingCap, Table tableStart, Table tableEnd, Player[] playersStart, Player[] playersEnd)` method creates an object `Move` that stores all the information needed to describe a move in the game: who the current player is, the played pile, the chip played, whether or not a capture took place, if a player was eliminated, eligible next players, chips being discarded, state of the board and players at the beginning and end of a move.
- `discardCount(String d)` returns a string dictating all possible chips to be discarded by a player during a move.

Chapter 4

2 Players, 2 Colours

Throughout this chapter, we will study the Generalized Hofstra version of SLS with two players and their respective colors.

4.1 Properties

Definition 4.1.1. *A chip of color x is said to be the **top chip** of pile π if it is the last played chip on π. A pile with a chip of color x on top is said to be an x-**pile**.*

Proposition 4.1.2. *Let $\mathscr{X} \in \{\mathscr{B}, \mathscr{R}\}$ and let x be $\mathscr{X}$'s color. If $\mathscr{X}$ places a $y \neq x$ chip on an empty pile or on an x-pile, then $\mathscr{X}$ has the next turn.*

Proof. Suppose $\mathscr{X}$ places a y chip on an empty pile. Then, by default $\mathscr{X}$ is the only player remaining in the game whose chip color is not in the played pile. Thus, $\mathscr{X}$ gets to play again.
Similarly, if $\mathscr{X}$ places a y chip on an x-pile, then based on the rules of the game, the next turn goes to the player whose chip color is furthest down in the pile. That player is $\mathscr{X}$. □

Proposition 4.1.3. *Let $\mathscr{X}, \mathscr{Y} \in \{\mathscr{B}, \mathscr{R}\}$ such that $\mathscr{X} \neq \mathscr{Y}$. Let x and y be their respective chips. If $\mathscr{X}$ places a y chip on a y-pile, then $\mathscr{Y}$ gains at least one chip of color y and has the next move.*

Proof. Suppose $\mathscr{X}$ places a y chip on a y-pile. Based on the rules of the game, this results in a capture for player $\mathscr{Y}$. $\mathscr{Y}$ has to discard at least one chip from the pile and can keep the remaining chips. Since there are at least 2 y chips in the captured pile, it follows that $\mathscr{Y}$ gains at least one y chip and has the next move. □

Proposition 4.1.4. *Let $\mathscr{X} \in \{\mathscr{B}, \mathscr{R}\}$ and let x be $\mathscr{X}$'s color. If $\mathscr{X}$ places an x chip on an empty pile or on a y-pile, then $\mathscr{Y}$ has the next turn.*

Proof. Suppose $\mathscr{X}$ places an x chip on an empty pile. Then, the next move cannot go to $\mathscr{X}$ based on the rules of the game. Since $\mathscr{Y}$ is the only other player remaining in the game, then the next turn goes to $\mathscr{Y}$.
Similarly, if $\mathscr{X}$ places an x chip on a y-pile, then based on the rules of the game, the next turn goes to the player whose chip color is furthest down in the pile. That player is $\mathscr{Y}$. □

Proposition 4.1.5. *Suppose we have an SLS board with 2 players $\mathscr{X}$ and $\mathscr{Y}$, and their respective colors x and y. Then all the non-empty piles have alternating x and y chips.*

Proof. Suppose there exists a pile on the board that has 2 consecutive chips of the same color x. Based on the rules of the game, two consecutive chips of the same color x result in a capture for player $\mathscr{X}$. Therefore the only situation when there are 2 consecutive chips of the same color would be as a capture takes place. Additionally, since there are only two colors in total in play, all non-empty piles must alternate chips between those two colors. □

Definition 4.1.6. *A **singleton** is a pile of length 1. We denote a singleton with an x chip as an x-singleton.*

Proposition 4.1.7. *Suppose player $\mathscr{X}$ captures an x-singleton. Then $\mathscr{X}$ can guarantee that the number of x chips in their possession does not change.*

Proof. Suppose $\mathscr{X}$ captures an x-singleton. They do so by placing an x chip in their possession on top of the x-singleton. Based on the rules of the game, $\mathscr{X}$ has to discard at least one chip from the captured pile. Thus, $\mathscr{X}$ will have to discard one of the two x chips retrieved from the pile. This brings back the number of x chips in $\mathscr{X}$'s hand to the same as it was before making the capture. □

Definition 4.1.8. *We define the **length** of a pile π, denoted by $|\pi| \in \mathbb{N}$, to be the number of chips contained in that pile.*
Similarly, we denote by $|\pi|_b$ the number of blue chips in pile π and by $|\pi|_r$, the number of red chips in pile π.

Definition 4.1.9. *A pile π of length $|\pi|$ can be written as $(x_1, x_2, \ldots, x_{|\pi|})$ to designate the $|\pi|$ chips in the pile with x_1 being the furthest one down and $x_{|\pi|}$ being the top chip. It is referred to as an $(x_1, x_2, \ldots, x_{|\pi|})$-pile.*
If a chip x is placed on top of a pile π, then this new pile is referred to as an (π, x)-pile.

4.2 Rules of the Game

We recall that throughout this thesis, we have $\mathbb{N} = \{0, 1, 2, \ldots\}$.
Consider an SLS board consisting of k empty piles, denoted by $\pi_1, \ldots, \pi_k$. Since there

are only 2 players in this set-up, they are each other's opponent. The players will be looking out for their best interest. In this chapter, we will show how a player can guarantee winning without the need for donations. As a result, donations will not be taken into account.
The rules of the game are the following:

1. First player $\mathscr{B}$ has m_b blue chips and m_r red chips. Second player $\mathscr{R}$ has n_b blue chips and n_r red chips.
2. All chips are always visible throughout the game.
3. A player's move consists of placing a chip on the board. However, the consequences of a move vary.
4. If $\mathscr{B}$ plays an r chip on an empty pile or a b-pile, then $\mathscr{B}$ plays again (by Proposition 4.1.2).
5. If $\mathscr{B}$ plays a b chip on an empty pile or an r-pile, then $\mathscr{R}$ gets the next move (by Proposition 4.1.4).
6. If $\mathscr{R}$ plays a b chip on an empty pile or an r-pile, then $\mathscr{R}$ plays again (by Proposition 4.1.2).
7. If $\mathscr{R}$ plays an r chip on an empty pile or a b-pile, then $\mathscr{B}$ gets the next move (by Proposition 4.1.4).
8. If a player plays a b chip on a b-pile, then $\mathscr{B}$ captures the pile, discards at least one chip from the pile, and gets the next move.
9. If a player plays an r chip on an r-pile, then $\mathscr{R}$ captures the pile, discards at least one chip from the pile, and gets the next move.
10. A player may discard a prisoner at any moment.
11. A player loses when it is their turn to play and they are out of chips.

Remark 4.2.1. *Rule 4 and Rule 6 emphasize moves that players can make and play again. In these situations, the order in which the chips are placed does not affect the outcome of the game.* $\mathscr{X}$ *can start off their turn by doing any combination of the following moves:*

$$\begin{cases} \textit{placing } t \textit{ } y \textit{ chips on } t \textit{ empty piles,} \\ \textit{placing } t' \textit{ } y \textit{ chips on } t' \textit{ } x\textit{-piles,} \\ \textit{or placing } s \textit{ } x \textit{ chips on } s \textit{ } x\textit{-singletons.} \end{cases}$$

As we can see, we cannot clearly define the number of empty piles that are played on, or the order in which piles are played on. Without loss of generality, we will assume that the order in which chips are placed follow a specific sequence since the order does not affect the strategies or outcome of the game.

We say a player $\mathscr{X}$ applies **Strategy** Σ if $\mathscr{X}$ does the following:

1. $\mathscr{X}$ captures all x-piles. For each pile π captured by $\mathscr{X}$, if $|\pi| \geq 2$ then $\mathscr{X}$ discards all $y \neq x$ chips in the pile. If π is an x-singleton, then $\mathscr{X}$ discards the single x chip.

2. If there are y-piles on the board, then $\mathscr{X}$ tops the y-pile of greatest length with an x chip. Otherwise, $\mathscr{X}$ places an x chip on an empty pile.

Throughout this chapter, we will prove the following statement.

Suppose we have an SLS board with only players $\mathscr{R}$ and $\mathscr{B}$ remaining in the game, and only chips r and b being used. Then, one of the players can always guarantee a win by playing Strategy Σ.

Remark 4.2.2. *We cannot know with certainty what moves the losing player will make. Throughout this chapter, we assume the losing player will keep in their hand as many chips as possible, discarding only one chip of the opposing player's color when necessary. This creates an upper bound on the number of chips in the player's possession. Intuitively, if a player loses with the most chips possible in their hands, they will certainly lose if they had less.*

4.3 Type I Board

Definition 4.3.1. *An SLS board is said to be of **type I** if the board is composed of only: zero or more singletons, and at most one r-pile ρ of length at least 2.*

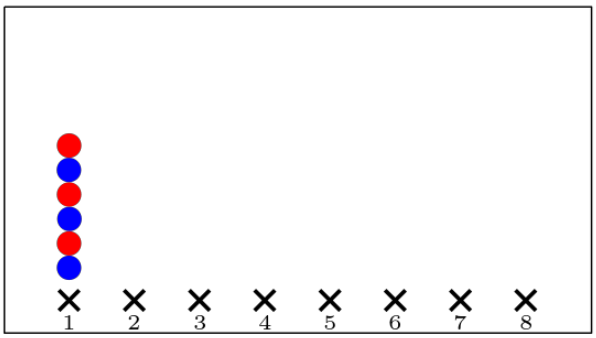

Figure 4.1: SLS Board of type I with $k = 8$

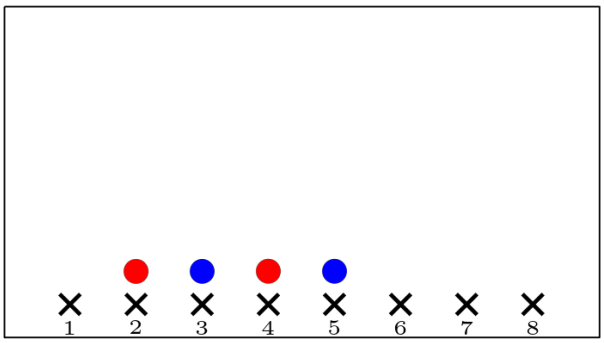

Figure 4.2: SLS Board of type I with $k = 8$

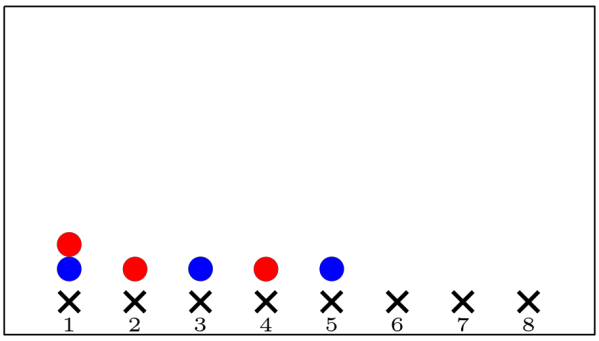

Figure 4.3: SLS Board of type I with $k = 8$

Proposition 4.3.2. *Consider an SLS board of type I. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b > n_r \geq 0$ then $\mathscr{B}$ has a winning strategy.*

Proof. Suppose we have an SLS board of type I with k piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, and κ r-pile ($\kappa \in \{0, 1\}$) of length at least 2, where $k = k_e + k_r + k_b + \kappa$. Suppose $m_b > n_r \geq 0$.

We will prove by induction on $n = n_b + n_r$ that Σ is a winning strategy for $\mathscr{B}$.

If $n_r = n_b = 0$, then $\mathscr{R}$ has no chips and $\mathscr{B}$ has at least one b chip since $m_b > n_r$ by the assumption. $\mathscr{B}$ applies Strategy Σ: captures k_b b-singletons, and places a b chip on ρ if $\kappa = 1$, on an r-singleton if $k_r > 0$, or on an empty pile otherwise. The move goes to $\mathscr{R}$ who cannot complete a turn. $\mathscr{R}$ loses and $\mathscr{B}$ wins.
Suppose $\mathscr{B}$ has a winning strategy against any player $\mathscr{R} = (i, j)$ with $0 \leq i+j \leq n-1$ and $m_b > j$. We now consider a player $\mathscr{R} = (n_b, n_r)$ with $n_b + n_r = n$ and $m_b > n_r$. Since $m_b > n_r \geq 0$, then $\mathscr{B}$ has at least one b chip. $\mathscr{B}$ applies Strategy Σ: captures k_b b-singletons, and places a b chip on ρ if $\kappa = 1$, on an r-singleton if $\kappa = 0$ and $k_r > 0$, or on an empty pile otherwise.
In all three cases, we have $\mathscr{B} = (m_b - 1, m_r)$ and it is $\mathscr{R}$'s turn to play.

$\mathscr{R}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{R}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ blue chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ blue chips on } t' \text{ } r\text{-singletons,} \\ \text{or placing } s \text{ red chips on } s \text{ } r\text{-singletons.} \end{cases}$$

- If $k_r > 0$, then the SLS board is composed of:

$$\begin{cases} k_e + k_b - t + s \text{ empty piles,} \\ k_r - t' - s - (1-\kappa) \text{ } r\text{-singletons,} \\ t \text{ } b\text{-singletons,} \\ \text{and } t'+1 \text{ } b\text{-piles } \beta_1, \ldots, \beta_{t'+1} \text{ of length at least 2.} \end{cases}$$

- If $k_r = 0$, then the SLS board is composed of:

$$\begin{cases} k_e + k_b - t - (1-\kappa) \text{ empty piles,} \\ t + (1-\kappa) \text{ } b\text{-singletons,} \\ \text{and } \kappa \text{ } b\text{-piles } \beta_\kappa \text{ of length at least 2.} \end{cases}$$

$\mathscr{R} = (n_b - t - t', n_r)$ can end their turn in 5 different ways.

1. $\mathscr{R}$ places an r chip on an empty pile. Therefore, $n_r > 0$, and there exists an empty pile. Hence, if $n_r > 0$ and $m_b > n_r$ then $m_b - 1 > 0$. The move automatically goes to $\mathscr{B} = (m_b - 1, m_r)$, who captures all b-piles, as per step 1 of Strategy Σ.

   - If $k_r > 0$, then the board is composed of $k_e + k_b + t' + s + 1$ empty piles, and $k_r - t' - s - (1-\kappa)$ r-singletons. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1 + \sum_{i=1}^{t'+1} |\beta_i|_b, m_r)$ and second player $\mathscr{R} = (n_b - t - t', n_r - 1)$. Since $(n_b - t - t') + (n_r - 1) < n_b + n_r = n$ and $m_b - 1 + \sum_{i=1}^{t'+1} |\beta_i|_b \geq m_b - 1 > n_r - 1$, then by the induction hypothesis, $\mathscr{B}$ has a winning strategy.
   - If $k_r = 0$, then the board is composed of $k_e + k_b + \kappa = k$ empty piles. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1 + |\beta_\kappa|_b + 1, m_r) = (m_b + |\beta_\kappa|_b, m_r)$ and second player $\mathscr{R} = (n_b - t - t', n_r - 1)$. Since $(n_b - t - t') + (n_r - 1) < n_b + n_r = n$ and $m_b + |\beta_\kappa|_b > m_b - 1 > n_r - 1$, by the induction hypothesis, $\mathscr{B}$ has a winning strategy.

2. $\mathscr{R}$ places an r chip on a b-singleton (assuming $n_r > 0$, $t > 0$). The move automatically goes to $\mathscr{B} = (m_b - 1, m_r)$, who captures all b-piles, as per step 1 of Strategy Σ.

- If $k_r > 0$ and $t > 0$, then the board is composed of $k_e + k_b + t' + s$ empty piles, $k_r - t' - s - (1-\kappa)$ r-singletons, and one (b,r)-pile. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1 + \sum_{i=1}^{t'+1} |\beta_i|_b, m_r)$ and second player $\mathscr{R} = (n_b - t - t', n_r - 1)$. Since $(n_b - t - t') + (n_r - 1) < n_b + n_r = n$ and $m_b - 1 + \sum_{i=1}^{t'+1} |\beta_i|_b \geq m_b - 1 > n_r - 1$, then by the induction hypothesis, $\mathscr{B}$ has a winning strategy.
- If $k_r = 0$ and $t + (1-\kappa) > 0$, then the board is composed of $k_e + k_b - t - (1-\kappa)$ empty piles, and one (b,r)-pile. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1 + \sum_{i=1}^{\kappa} |\beta_i|_b, m_r)$ and second player $\mathscr{R} = (n_b - t - t', n_r - 1)$. Since $(n_b - t - t') + (n_r - 1) < n_b + n_r = n$ and $m_b - 1 + \sum_{i=1}^{\kappa} |\beta_i|_b \geq m_b - 1 > n_r - 1$, then by the induction hypothesis, $\mathscr{B}$ has a winning strategy.

3. $\mathscr{R}$ places an r chip on a b-pile β_1 of length at least 2 (this means $\kappa = 1$). The move automatically goes to $\mathscr{B} = (m_b - 1, m_r)$, who captures all b-piles, as per step 1 of Strategy Σ.

   - If $k_r > 0$, then the board is composed of $k_e + k_b + s + t'$ empty piles, $k_r - t' - s - (1-\kappa)$ r-singletons, and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1 + \sum_{i=2}^{t'+1} |\beta_i|_b, m_r)$ and second player $\mathscr{R} = (n_b - t - t', n_r - 1)$. Since $(n_b - t - t') + (n_r - 1) < n_b + n_r = n$ and $m_b - 1 + \sum_{i=2}^{t'+1} |\beta_i|_b \geq m_b - 1 > n_r - 1$, then by the induction hypothesis, $\mathscr{B}$ has a winning strategy.
   - If $k_r = 0$, then the board is composed of $k_e + k_b$ empty piles, and one (β_κ, r)-pile. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1, m_r)$ and second player $\mathscr{R} = (n_b - t - t', n_r - 1)$. Since $(n_b - t - t') + (n_r - 1) < n_b + n_r = n$ and $m_b - 1 > n_r - 1$, then by the induction hypothesis, $\mathscr{B}$ has a winning strategy.

4. $\mathscr{R}$ places a b chip on a b-pile (assuming $n_b - t - t' > 0$). This results in a capture for $\mathscr{B}$, who continues to capture all b-piles, as per step 1 of Strategy Σ.

   - If $k_r > 0$, then the board is composed of $k_e + k_b + s + t' + 1$ empty piles, and $k_r - t' - s - (1-\kappa)$ r-singletons. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1 + \sum_{i=1}^{t'+1} |\beta_i|_b + 1, m_r)$ and second player $\mathscr{R} = (n_b - t - t' - 1, n_r)$. Since $(n_b - t - t' - 1) + (n_r) < n_b + n_r = n$ and $m_b - 1 + \sum_{i=1}^{t'+1} |\beta_i|_b + 1 = m_b + \sum_{i=1}^{t'+1} |\beta_i|_b > n_r$, then by the induction hypothesis, $\mathscr{B}$ has a winning strategy.
   - If $k_r = 0$ and $\kappa = 1$, then the board is composed of $k_e + k_b + 1 = k$ empty piles. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1 + |\beta|_b + 1, m_r) = (m_b + |\beta|_b, m_r)$ and second player $\mathscr{R} = (n_b - t - t' - 1, n_r)$.

Since $(n_b - t - t' - 1) + (n_r) < n_b + n_r = n$ and $m_b + |\beta|_b > n_r$, then by the induction hypothesis, $\mathscr{B}$ has a winning strategy.

5. $\mathscr{R}$ runs out of chips if $n_b - t - t' = 0$ and $n_r = 0$. $\mathscr{R}$ loses by default and $\mathscr{B}$ wins.

It follows by induction that for every possible move $\mathscr{R}$ can do, $\mathscr{B}$ has winning strategy, which is Strategy Σ.

□

Proposition 4.3.3. *Consider an SLS board of type I. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $0 \leq m_b \leq n_r$ then $\mathscr{R}$ has a winning strategy.*

Proof. Suppose we have an SLS board of type I with k piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, and $\kappa \in \{0, 1\}$ r-pile of length at least 2, where $k = k_e + k_r + k_b + \kappa$. Suppose $0 \leq m_b \leq n_r$.

We will prove by induction on $m = m_b + m_r$ that Σ is a winning strategy for $\mathscr{R}$.

If $m_b = m_r = 0$, then $\mathscr{B}$ loses by default since they cannot place the first chip; and $\mathscr{R}$ wins.

Suppose $\mathscr{R} = (n_b, n_r)$ has a winning strategy against any player $\mathscr{B} = (i, j)$ with $0 \leq i + j \leq m - 1$ and $i \leq n_r$. We now consider a player $\mathscr{B} = (m_b, m_r)$ with $m_b + m_r = m$ and $m_b \leq n_r$.
As first player, $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{or placing } s \text{ blue chips on } s \text{ } b\text{-singletons.} \end{cases}$$

The SLS board is composed of:

$$\begin{cases} k_e - t + s \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - s - t' \text{ } b\text{-singletons,} \\ \text{and } t' + \kappa \text{ } r\text{-piles } \rho_1, \ldots, \rho_{t'+\kappa} \text{ of length at least 2.} \end{cases}$$

$\mathscr{B} = (m_b, m_r - t - t')$ can end their turn in 5 different ways.

1. $\mathscr{B}$ places a b chip on an empty pile (assuming $m_b > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures all r-piles, and places an r chip on a b-singleton. Now, the board is composed of $k_e + k_r + t' + s + \kappa - 1$ empty piles, $k_b - s - t'$ b-singletons, and one (b, r)-pile. The board is of type I. We have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$ and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+\kappa} |\rho_i|_r - 1)$. Since $(m_b - 1) + (m_r - t - t') < m_b + m_r = m$ and $n_r + \sum_{i=1}^{t'+\kappa} |\rho_i|_r - 1 \geq n_r - 1 \geq m_b - 1$, then by the induction hypothesis $\mathscr{R}$ has a winning strategy.

2. $\mathscr{B}$ places a b chip on an r-singleton (assuming $m_b > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures all r-piles, and places an r chip on the (r, b)-pile. Now, the board is composed of $k_e + k_r + t' + s + \kappa$ empty piles, $k_b - s - t' - 1$ b-singletons and one (r, b, r)-pile. The board is of type I. We have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$ and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+\kappa} |\rho_i|_r - 1)$. Since $(m_b - 1) + (m_r - t - t') < m_b + m_r = m$ and $n_r + \sum_{i=1}^{t'+\kappa} |\rho_i|_r - 1 \geq n_r - 1 \geq m_b - 1$, then by the induction hypothesis $\mathscr{R}$ has a winning strategy.

3. $\mathscr{B}$ places a b chip on an r-pile ρ_1 of length at least 2 (assuming $m_b > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures all r-piles, and places an r chip on the (ρ_1, b)-pile. Now, the board is composed of $k_e + k_r + t' + s + \kappa - 1$ empty piles, $k_b - s - t'$ b-singletons and one (ρ_1, b, r)-pile. The board is of type I. We have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$ and second $\mathscr{R} = (n_b, n_r + \sum_{i=2}^{t'+\kappa} |\rho_i|_r - 1)$. Since $(m_b - 1) + (m_r - t - t') < m_b + m_r = m$ and $n_r + \sum_{i=2}^{t'+\kappa} |\rho_i|_r - 1 \geq n_r - 1 \geq m_b - 1$, then by the induction hypothesis $\mathscr{R}$ has a winning strategy.

4. $\mathscr{B}$ places an r chip on an r-pile (assuming $m_r - t - t' > 0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + s + t' + \kappa$ empty piles, and $k_b - s - t'$ b-singletons. If $k_b - s - t' > 0$, then $\mathscr{R}$ places an r chip on a b-singleton. Otherwise, $\mathscr{R}$ places an r chip on an empty pile. In both cases, the board is of type I. We have first player $\mathscr{B} = (m_b, m_r - t - t' - 1)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+\kappa} |\rho_i|_r)$. Since $(m_b) + (m_r - t - t' - 1) < m_b + m_r = m$ and $n_r + \sum_{i=1}^{t'+\kappa} |\rho_i|_r \geq n_r \geq m_b$, then by the induction hypothesis $\mathscr{R}$ has a winning strategy.

5. $\mathscr{B}$ runs out of chips if $m_b = 0$ and $m_r - t - t' = 0$. $\mathscr{B}$ loses by default and $\mathscr{R}$ wins.

It follows by induction that for every possible move $\mathscr{B}$ can do, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Theorem 4.3.4. *Consider an SLS board of type I. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. We have the following:*

$$\mathscr{B} \text{ has a winning strategy} \iff m_b > n_r$$

Proof. Suppose $\mathscr{B} = (m_b, m_r)$ is the first player and $\mathscr{R} = (n_b, n_r)$ is the second player such that $m_b > n_r$. Then by Proposition 4.3.2, $\mathscr{B}$ has a winning strategy.
Now suppose $\mathscr{B} = (m_b, m_r)$ is the first player and $\mathscr{R} = (n_b, n_r)$ is the second player such that $m_b \leq n_r$. Then by Proposition 4.3.3, $\mathscr{R}$ has a winning strategy. □

4.4 Type II Board

Definition 4.4.1. *An SLS board with $k \geq 2$ piles is said to be of **type II** if the board is composed of only: zero or more singletons, one r-pile ρ of length at least 2, and one b-pile β of length at least 2.*

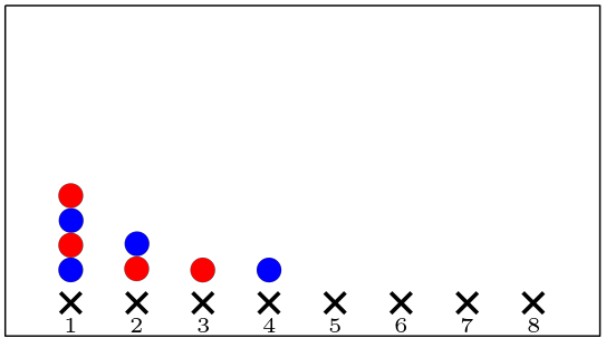

Figure 4.4: SLS Board of type II with $k = 8$

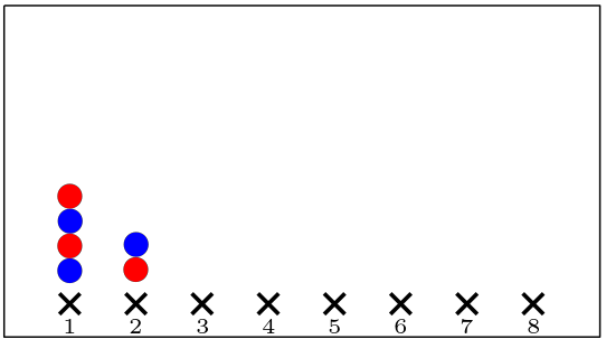

Figure 4.5: SLS Board of type II with $k = 8$

Proposition 4.4.2. *Consider an SLS board of type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b \geq 1$ and $m_b + |\beta|_b > n_r$, then $\mathscr{B}$ has a winning strategy.*

Proof. Suppose we have an SLS board of type II with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, ρ and β as described above, where $k = k_e + k_r + k_b + 2$.

Suppose $m_b \geq 1$ and $m_b + |\beta|_b > n_r$. As first player, $\mathscr{B}$ captures all b-piles, as per step 1 of Strategy Σ. The board is composed of $k_e + k_b + 1$ empty piles, k_r r-singletons and the r-pile ρ. The board is of type I. We now have first player $\mathscr{B} = (m_b + |\beta|_b, m_r)$, and second player $\mathscr{R} = (n_b, n_r)$ such that $m_b + |\beta|_b > n_r$. By Theorem 4.3.4, $\mathscr{B}$ has a winning strategy. □

Remark 4.4.3. *The cases in which second player $\mathscr{R}$ has a winning strategy are split into the multiple propositions that follow thereafter. This was done in order to facilitate the reading of the proofs.*

Proposition 4.4.4. *Consider an SLS board of type II . Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b = 0$, then $\mathscr{R}$ has a winning strategy.*

Proof. Suppose we have an SLS board of type II with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, ρ and β as described above, where $k = k_e + k_r + k_b + 2$.
Suppose $m_b = 0$. As first player, $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{or placing } t'' \in \{0, 1\} \text{ red chips on } \beta \end{cases}$$

The SLS board is now composed of:

$$\begin{cases} k_e - t \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' \text{ } b\text{-singletons,} \\ t' \text{ } (b, r)\text{-piles,} \\ 1 + t'' \text{ } r\text{-piles of length} \geq 2, \\ \text{and } 1 - t'' \text{ } b\text{-piles of length} \geq 2. \end{cases} = \begin{cases} k_e - t \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' \text{ } b\text{-singletons,} \\ t' + t'' + 1 \text{ } r\text{-piles of length at least 2,} \\ \text{and } 1 - t'' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

We rename the $t' + t'' + 1$ r-piles of length at least 2 to $\rho_1, \rho_2, \ldots, \rho_{t'+t''+1}$.
$\mathscr{B} = (0, m_r - t - t' - t'')$ can end their turn in 2 different ways.

1. $\mathscr{B}$ places an r chip on an r-pile (assuming $m_r - t - t' - t'' > 0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + t' + t'' + 1$ empty piles, $k_b - t'$ b-singletons, and $1 - t''$ b-piles of length at least 2.

   - If $1 - t'' = 1$, then $\mathscr{R}$ places an r chip on the b-piles of length at least 2.
   - Otherwise, if $k_b - t' > 0$, then $\mathscr{R}$ places an r chip on a b-singleton.
   - Otherwise, $\mathscr{R}$ places an r chip on an empty pile.

   In all three cases, the board is of type I. We have first player $\mathscr{B} = (0, m_r - t - t' - t'' - 1)$ and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r)$. Since $n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r > n_r \geq 0 = m_b$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ runs out of chips if $m_r - t - t' - t'' = 0$. $\mathscr{B}$ loses by default and $\mathscr{R}$ wins.

It follows that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ.

□

Proposition 4.4.5. *Consider an SLS board of type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player, and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b > 0$, $m_b + |\beta|_b \leq n_r$ and $\mathscr{B}$ captures β at some point during their first turn, then $\mathscr{R}$ has a winning strategy.*

Proof. Suppose we have an SLS board of type II with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, ρ and β as described above, where $k = k_e + k_r + k_b + 2$.
As first player, $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{placing } s \text{ blue chips on } s \text{ } b\text{-singletons,} \\ \text{and capturing } \beta. \end{cases}$$

The SLS board is now composed of:

$$\begin{cases} k_e - t + s + 1 \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ t' \text{ } (b,r)\text{-piles,} \\ \text{and } \rho. \end{cases} = \begin{cases} k_e - t + s + 1 \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ \text{and } t' + 1 \text{ } r\text{-piles of length at least 2.} \end{cases}$$

We rename the $t' + 1$ r-piles of length at least 2 to $\rho_1, \rho_2, \ldots, \rho_{t'+1}$.
$\mathscr{B} = (m_b + |\beta|_b, m_r + |\beta|_r - t - t' - 1)$ can end their turn in 4 different ways.

1. $\mathscr{B}$ places a b chip on an empty pile (assuming $k_e - t + s + 1 \geq 1$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures $k_r + t$ r-singletons, $t' + 1$ r-piles of length at least 2, and places an r chip on a b-singleton. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + s + t' + 1$ empty piles, $k_b - t' - s$ b-singletons and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B} = (m_b + |\beta|_b - 1, m_r + |\beta|_r - t - t' - 1)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1)$. Since $m_b + |\beta|_b \leq n_r \Rightarrow m_b + |\beta|_b - 1 \leq n_r - 1 \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ places a b chip on an r-singleton (assuming $k_r+t \geq 1$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures k_r+t-1 r-singletons, $t'+1$ r-piles of length at least 2, and places an r chip on the (r,b)-pile. The move goes back to $\mathscr{B}$. The board is composed of $k_e+k_r+s+t'+1$ empty piles, $k_b-t'-s$ b-singletons and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B}=(m_b+|\beta|_b-1, m_r+|\beta|_r-t-t'-1)$, and second player $\mathscr{R}=(n_b, n_r+\sum_{i=1}^{t'+1}|\rho_i|_r-1)$. Since $m_b+|\beta|_b \leq n_r \Rightarrow m_b+|\beta|_b-1 \leq n_r-1 \leq n_r+\sum_{i=1}^{t'+1}|\rho_i|_r-1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

3. $\mathscr{B}$ places a b chip on an r-pile ρ_j of length at least 2.
The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures k_r+t r-singletons, t' r-piles of length at least 2, and places an r chip on the (ρ_j,b)-pile. The move goes back to $\mathscr{B}$. The board is composed of $k_e+k_r+t'+s+1$ empty piles, $k_b-t'-s$ b-singletons and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B}=(m_b+|\beta|_b-1, m_r+|\beta|_r-t-t'-1)$, and second player $\mathscr{R}=(n_b, n_r+\sum_{i=1}^{t'}|\rho_i|_r-1)$. Since $m_b+|\beta|_b \leq n_r \Rightarrow m_b+|\beta|_b-1 \leq n_r-1 \leq n_r+\sum_{i=1}^{t'}|\rho_i|_r-1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

4. $\mathscr{B}$ places an r chip on an r-pile (assuming $m_r+|\beta|_r-t-t'-1>0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. Now, the board is composed of $k_e+k_r+t'+s+2$ empty piles, and $k_b-t'-s$ b-singletons.

   - If $k_b-t'-s>0$, then $\mathscr{R}$ places an r chip on a b-singleton.
   - Otherwise, $\mathscr{R}$ places an r chip on an empty pile.

   In both cases, the move goes back to $\mathscr{B}$ and the board is of type I. We now have first player $\mathscr{B}=(m_b+|\beta|_b, m_r+|\beta|_r-t-t'-2)$ and second player $\mathscr{R}=(n_b, n_r+\sum_{i=1}^{t'+1}|\rho_i|_r)$. Since $m_b+|\beta|_b \leq n_r \leq n_r+\sum_{i=1}^{t'+1}|\rho_i|_r$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

It follows that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Proposition 4.4.6. *Consider an SLS board of type II. Let $\mathscr{B}=(m_b,m_r)$ be the first player, and $\mathscr{R}=(n_b,n_r)$ be the second player. If $m_b>0$, $m_b+|\beta|_b \leq n_r$ and $\mathscr{B}$ places an r chip on β at some point during their first turn, then $\mathscr{R}$ has a winning strategy.*

Proof. Suppose we have an SLS board of type II with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, ρ and β as described above, where $k=k_e+k_r+k_b+2$.

As first player, $\mathscr{B} = (m_b, m_r)$ starts off doing any combination of the following moves (as any one of these moves allow $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{placing } s \text{ blue chips on } s \text{ } b\text{-singletons,} \\ \text{and placing one } r \text{ chip on } \beta. \end{cases}$$

The SLS board is now composed of:

$$\begin{cases} k_e - t + s \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ t' \text{ } (b,r)\text{-piles,} \\ \rho \text{ and the } (\beta, r)\text{-pile.} \end{cases} = \begin{cases} k_e - t + s \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ \text{and } t' + 2 \text{ } r\text{-piles of length at least 2.} \end{cases}$$

We rename the $t' + 2$ r-piles of length at least 2 to $\rho_1, \rho_2, \ldots, \rho_{t'+2}$.
$\mathscr{B} = (m_b, m_r - t - t' - 1)$ can end their turn in 4 different ways.

1. $\mathscr{B}$ places a b chip on an empty pile (assuming $k_e - t + s > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures $k_r + t$ r-singletons, $t' + 2$ r-piles of length at least 2 and places an r chip on a b-singleton. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + t' + s + 1$ empty piles, $k_b - s - t'$ b-singletons, and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t' - 1)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+2} |\rho_i|_r - 1)$. Since $m_b \leq m_b + |\beta|_b \leq n_r \Rightarrow m_b - 1 \leq n_r - 1 \leq n_r + \sum_{i=1}^{t'+2} |\rho_i|_r - 1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ places a b chip on an r-singleton (assuming $k_r + t > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures $k_r + t - 1$ r-singletons, $t' + 2$ r-piles of length at least 2, and places an r chip on the (r, b)-pile. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + t' + s + 1$ empty piles, $k_b - s - t'$ b-singletons, and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t' - 1)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+2} |\rho_i|_r - 1)$. Since $m_b \leq m_b + |\beta|_b \leq n_r \Rightarrow m_b - 1 \leq n_r - 1 \leq n_r + \sum_{i=1}^{t'+2} |\rho_i|_r - 1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

3. $\mathscr{B}$ places a b chip on an r-pile ρ_j of length at least 2. Without loss of generality, let us assume $\rho_j = \rho_{t'+2}$. The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures $k_r + t$ r-singletons, $t' + 1$ r-piles of length at least 2, and places an r chip on the $(\rho_{t'+2}, b)$ pile. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + t' + s + 1$ empty piles, $k_b - s - t'$ b-singletons and one r-pile of length

at least 2. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t' - 1)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1)$. Since $m_b \leq m_b + |\beta|_b \leq n_r \Rightarrow m_b - 1 \leq n_r - 1 \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

4. $\mathscr{B}$ places an r chip on an r-pile. This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + t' + s + 2$ empty piles, and $k_b - t' - s$ b-singletons.

   - If $k_b - t' - s > 0$, then $\mathscr{R}$ places an r chip on a b-singleton.
   - Otherwise, $\mathscr{R}$ places an r chip on an empty pile.

   In both cases, the move goes back to $\mathscr{B}$ and the board is of type I. We now have first player $\mathscr{B} = (m_b, m_r - t - t' - 2)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+2} |\rho_i|_r)$. Since $m_b \leq m_b + |\beta|_b \leq n_r \leq n_r + \sum_{i=1}^{t'+2} |\rho_i|_r$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

It follows that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Proposition 4.4.7. *Consider an SLS board of type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player, and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b > 0$, $m_b + |\beta|_b \leq n_r$ and $\mathscr{B}$ does not place a chip on β at any point during their first turn, then $\mathscr{R}$ has a winning strategy.*

Proof. Suppose we have an SLS board of type II with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, ρ and β as described above, where $k = k_e + k_r + k_b + 2$.
Suppose $m_b + |\beta|_b \leq n_r$ and $\mathscr{B}$ does not place a chip on β at any point during their first turn.
We will prove by induction $m = m_b + m_r$ that Σ is a winning strategy for $\mathscr{R}$.
If $m = 0$, then $\mathscr{B}$ has no chips and cannot make a move. $\mathscr{B}$ loses automatically and $\mathscr{R}$ wins.
Suppose $\mathscr{R} = (n_b, n_r)$ has a winning strategy against any player $\mathscr{B} = (x, y)$ that does not place a chip on β at any point during their turn, with $0 \leq x + y \leq m - 1$ and $x + |\beta|_b \leq n_r$.
We now consider a player $\mathscr{B} = (m_b, m_r)$ with $m_b + m_r = m$ and $m_b + |\beta|_b \leq n_r$. Since $n_r \geq m_b > 0$, then $\mathscr{R}$ has at least as many r chips as $\mathscr{B}$ has b chips.
As first player, $\mathscr{B} = (m_b, m_r)$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{and placing } s \text{ blue chips on } s \text{ } b\text{-singletons.} \end{cases}$$

The SLS board is now composed of:

$$\begin{cases} k_e - t + s \text{ empty piles}, \\ k_r + t \ r\text{-singletons}, \\ k_b - t' - s \ b\text{-singletons}, \\ t' \ (b,r)\text{-piles}, \\ \rho \text{ and } \beta. \end{cases} = \begin{cases} k_e - t + s \text{ empty piles}, \\ k_r + t \ r\text{-singletons}, \\ k_b - t' - s \ b\text{-singletons}, \\ t' + 1 \ r\text{-piles of length at least 2}, \\ \text{and } \beta. \end{cases}$$

We rename the $t'+1$ r-piles of length at least 2 to $\rho_1, \rho_2, \ldots, \rho_{t'+1}$.
$\mathscr{B} = (m_b, m_r - t - t')$ can end their turn in 4 different ways.

1. $\mathscr{B}$ places a b chip on an empty pile (assuming $k_e - t + s > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures $k_r + t$ r-singletons, $t'+1$ r-piles of length at least 2, and places an r chip on β. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + t' + s + 1$ empty piles, $k_b - t' - s$ b-singletons, and the (β, r)-pile. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1)$. Since $m_b \leq m_b + |\beta|_b \leq n_r \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r \Rightarrow m_b - 1 \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ places a b chip on an r-singleton (assuming $k_r + t > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures $k_r + t - 1$ r-singletons, $t'+1$ r-piles of length at least 2, and places an r chip on a b-pile π_j such that $|\pi_j|_b = \max\{|\beta|_b, |(r,b)|_b\}$. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + t' + s$ empty piles, $k_b - t' - s$ b-singletons, one non-singleton r-pile and one non-singleton b pile. The board is of type II. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1)$. We have that $m_b + \min\{|\beta|_b, |(r,b)|_b\} \leq m_b + \max\{|\beta|_b, |(r,b)|_b\} \leq n_r \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r$, then $(m_b - 1) + \min\{|\beta|_b, |(r,b)|_b\} \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1$. The sub-cases that arise are discussed after case 3.

3. $\mathscr{B}$ places a b chip on an r-pile ρ_i of length at least 2. The move automatically goes to $\mathscr{R}$. $\mathscr{R}$ applies Strategy Σ: captures $k_r + t$ r-singletons, t' r-piles of length at least 2, and places an r chip on a b-pile π_j such that $|\pi_j|_b = \max\{|\beta|_b, |(\rho_i, b)|_b\}$. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + t' + s$ empty piles, $k_b - t' - s$ b-singletons, one non-singleton r-pile and one non-singleton b pile. The board is of type II. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'} |\rho_i|_r - 1)$. We have that $m_b + \min\{|\beta|_b, |(r,b)|_b\} \leq m_b + \max\{|\beta|_b, |(\rho_i, b)|_b\} \leq n_r \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r$, then $(m_b - 1) + \min\{|\beta|_b, |(\rho_i, b)|_b\} \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r - 1$.

For cases 2 and 3, we have the following sub-cases.

- If $m_b - 1 = 0$, then $\mathscr{R}$ has a winning strategy by Proposition 4.4.4.
- If $m_b - 1 > 0$, and $\mathscr{B}$ captures the b-pile of length at least 2 during their move, then $\mathscr{R}$ has a winning strategy by Proposition 4.4.5.
- If $m_b - 1 > 0$, and $\mathscr{B}$ places an r chip on the b-pile of length at least 2 during their move, then $\mathscr{R}$ has a winning strategy by Proposition 4.4.6.
- If $m_b - 1 > 0$, $\mathscr{B} = (m_b - 1, m_r - t - t')$ does not place a chip on the b-pile of length at least 2 during their move, and $(m_b - 1) + (m_r - t - t') < m_b + m_r = m$, then $\mathscr{R}$ has a winning strategy by the induction hypothesis.

4. $\mathscr{B}$ places an r chip on an r-pile (assuming $m_r - t - t' > 0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. $\mathscr{R}$ places an r chip on β. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + t' + s + 1$ empty piles, $k_b - t' - s$ b-singletons and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B} = (m_b, m_r - t - t' - 1)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{t'+1} |\rho_i|_r)$. Since $m_b \leq m_b + |\beta|_b \leq n_r \leq n_r + \sum_{i=1}^{t'+1} |\rho_i|_r$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

It follows by induction that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Corollary 4.4.8. *Consider an SLS board of type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player, and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b > 0$ and $m_b + |\beta|_b \leq n_r$, then $\mathscr{R}$ has a winning strategy.*

Proof. Suppose $\mathscr{B} = (m_b, m_r)$ is the first player, $\mathscr{R} = (n_b, n_r)$ is the second player, $m_b > 0$ and $m_b + |\beta|_b \leq n_r$.
As first player, $\mathscr{B}$ can choose any of the following three options during their first turn.

1. If $\mathscr{B}$ captures β, then $\mathscr{R}$ has a winning strategy by Proposition 4.4.5.
2. If $\mathscr{B}$ places an r chip on β, then $\mathscr{R}$ has a winning strategy by Proposition 4.4.6.
3. If $\mathscr{B}$ does not place a chip on β, $\mathscr{R}$ has a winning strategy by Proposition 4.4.7.

□

Theorem 4.4.9. *Consider an SLS board of type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player, and $\mathscr{R} = (n_b, n_r)$ be the second player. We have the following:*

$$\mathscr{B} \text{ has a winning strategy} \iff m_b, n_r > 0 \text{ and } m_b + |\beta|_b > n_r$$

Proof. Suppose $\mathscr{B} = (m_b, m_r)$ is the first player, and $\mathscr{R} = (n_b, n_r)$ is the second player. We separate this theorem into several cases.

1. If $m_b \geq 1$ and $m_b + |\beta|_b > n_r$, then $\mathscr{B}$ has a winning strategy by Proposition 4.4.2.

2. If $m_b = 0$, then $\mathscr{R}$ has a winning strategy by Proposition 4.4.4.

3. If $m_b \geq 1$ and $m_b + |\beta|_b \leq n_r$, then $\mathscr{R}$ has a winning strategy by Corollary 4.4.8.

This covers all possible cases and proves our theorem.

□

4.5 Generalized Type I Board

In this section, we assume that an empty sum implies there are no piles to be considered and the sum is 0 as a result.

Definition 4.5.1. *An SLS board with $k \geq 2$ piles is said to be of **generalized type I** if it contains zero or more singletons, and $l \geq 0$ r-piles of length at least 2. We denote these r-piles by $\rho_1, \rho_2, \ldots, \rho_l$.*

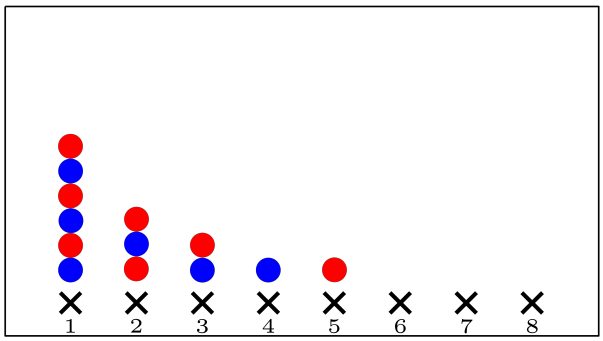

Figure 4.6: SLS Board of Generalized type I with $k = 8$

Proposition 4.5.2. *Consider an SLS board of generalized type I. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b > 0$ and $n_r = 0$, then $\mathscr{B}$ has a winning strategy.*

Proof. Suppose we have an SLS board of generalized type I with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, and $l \geq 0$ are r-piles of length at least 2, where $k = k_e + k_r + k_b + l$. Let $P(l)$ be the following statement:

"If we have an SLS board of generalized type I with l r-piles of length at least 2 such that $m_b > 0$ and $n_r = 0$, then $\mathscr{B}$ has a winning strategy."

We will prove this proposition by induction on l.
Base Case: Suppose we have an SLS board of generalized type I where $m_b > 0$ and $n_r = 0$. If $l = 0$ or $l = 1$, then the board is of type I. $\mathscr{B}$ has a winning strategy by Theorem 4.3.4.
Induction Hypothesis: Suppose $P(\lambda)$ holds for all $\lambda \in \{0, 1, \ldots, l-1\}$.
Induction Step: We now prove that $P(l)$ is also true. Suppose we have an SLS board of generalized type I where $m_b > 0$ and $n_r = 0$. As first player, $\mathscr{B}$ applies Strategy Σ. $\mathscr{B}$ captures k_b b-singletons, and places a b chip on the largest r-pile. Without loss of generality, suppose ρ_l is that pile, i.e. $|\rho_l|_r = \max\{\rho_i : i = 1, \ldots, l\}$. This SLS board is composed of

$$\begin{cases} k_e + k_b \text{ empty piles,} \\ k_r \ r\text{-singletons,} \\ l-1 \ r\text{-piles of length at least 2,} \\ \text{and one } b\text{-pile of length at least 3, namely } \rho_l. \end{cases}$$

We have $\mathscr{B} = (m_b - 1, m_r)$ and it is $\mathscr{R}$'s turn to play. $\mathscr{R}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{R}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ blue chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ blue chips on } t' \ r\text{-singletons,} \\ \text{and placing } t'' \text{ blue chips on } t'' \ r\text{-piles of length at least 2,} \end{cases}$$

The SLS board is now composed of

$$\begin{cases} k_e + k_b - t \text{ empty piles,} \\ k_r - t' \ r\text{-singletons,} \\ t \ b\text{-singletons,} \\ t' \ (r,b)\text{-piles,} \\ t'' + 1 \ b\text{-piles of length at least 3,} \\ \text{and } l - t'' - 1 \ r\text{-piles of length at least 2.} \end{cases}$$

$=$

$$\begin{cases} k_e + k_b - t \text{ empty piles,} \\ k_r - t' \ r\text{-singletons,} \\ t \ b\text{-singletons,} \\ t' + t'' + 1 \ b\text{-piles of length at least 2,} \\ \text{and } l - t'' - 1 \ r\text{-piles of length at least 2.} \end{cases}$$

Without loss of generality, the remaining r-piles of length at least 2 on the board are $\rho_1, \ldots, \rho_{l-t''-1}$. The t'' b-piles of length at least 3 that were created are $\rho_{l-t''}, \rho_{l-t''+1}, \ldots, \rho_{l-1}$. We rename the $t' + t'' + 1$ b-piles of length at least 2 to $\beta_1, \beta_2, \ldots, \beta_{t'+t''+1}$.

$\mathscr{R} = (n_b - t - t' - t'', 0)$ can end their turn in 2 different ways.

1. $\mathscr{R}$ places a b chip on a b-pile. The move automatically goes to $\mathscr{B}$, who captures all b-piles as per step 1 of Strategy Σ. The board is composed of $k_e+k_b+t'+t''+1$ empty piles, $k_r - t'$ r-singletons, and $l - t'' - 1 < l$ r-piles of length at least 2. The board is of generalized type I. We now have first player $\mathscr{B} = (m_b + \sum_{i=1}^{t'+t''+1} |\beta_i|_b, m_r)$ such that $m_b+\sum_{i=1}^{t'+t''+1} |\beta_i|_b \geq m_b > 0$. We have $l-t''-1 < l$, then $\mathscr{B}$ has a winning strategy by the induction hypothesis on l.

2. $\mathscr{R}$ runs out of chips if $n_b - t - t' - t'' = 0$ and loses automatically. $\mathscr{B}$ wins as a result.

□

Proposition 4.5.3. *Consider an SLS board of generalized type I. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If*

$$m_b, n_r > 0 \text{ and } m_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\},$$

then $\mathscr{B}$ has a winning strategy.

Proof. Suppose we have an SLS board of generalized type I with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, and $l \geq 0$ are r-piles of length at least 2, where $k = k_e + k_r + k_b + l$. Let $P(l)$ be the following statement:

"If we have an SLS board of generalized type I with l r-piles of length ≥ 2 such that $m_b, n_r > 0$ and $m_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$, then $\mathscr{B}$ has a winning strategy."

We will prove this proposition by induction on l.
Base Case: Suppose we have an SLS board of generalized type I where $m_b, n_r > 0$ and $m_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$. If $l = 0$, then the board is of type I. The sum and the maximum in the second inequality is empty. The inequalities become $m_b, n_r > 0$ and $m_b > n_r$. By Theorem 4.3.4, $\mathscr{B}$ has a winning strategy.

If $l = 1$, then the board is of type I. The inequalities become $m_b, n_r > 0$ and

$$m_b > n_r + \sum_{i=1}^{1} |\rho_i|_r - \max\{|\rho_i|_r : i = 1\} = n_r + |\rho_1|_r - |\rho_1|_r = n_r.$$

By Theorem 4.3.4, $\mathscr{B}$ has a winning strategy.
Induction Hypothesis: Suppose $P(\lambda)$ holds for all $\lambda \in \{0, 1, 2, \ldots, l-1\}$.
Induction Step: We now prove that $P(l)$ is also true. Suppose we have an SLS board of generalized type I where $m_b > 0$, $n_r > 0$ and

$$m_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}.$$

This inequality, along with the fact that $n_r > 0$, imply that $m_b \geq 2$. Let us consider what happens when $\mathscr{B}$ applies Strategy Σ. $\mathscr{B}$ captures k_b b-singletons, and places a b chip on top of the largest r-pile. Observe that $\mathscr{R}$ cannot have more than

$$\begin{aligned} &n_b + n_r + \sum_{i=1}^{l} (|\rho_i| - 1) - \max\{|\rho_i| - 1 : i = 1, \ldots, l\} - 1 \\ &= n_b + n_r + \sum_{i=1}^{l} (|\rho_i| - 1) - \max\{|\rho_i| : i = 1, \ldots, l\} \end{aligned}$$

chips at the end of their turn. This corresponds to the total number of chips in $\mathscr{R}$'s hands if $\mathscr{R}$ were to capture all r-piles on the board. Let

$$n = n_b + n_r + \sum_{i=1}^{l} (|\rho_i| - 1) - \max\{|\rho_i| : i = 1, \ldots, l\}.$$

Since there is no guarantee that $\mathscr{R}$ will play this way, the parameter n is only an upper bound on the total number of chips in their hands at the end of their turn. Nevertheless, n is a key parameter in this proof. Indeed, we will prove by induction on n that $P(l)$ is true.

Base Case (for parameter n): Consider the case where $n = 0$. Since $n_r > 0$, this means that $n_b = 0$, $n_r = 1$, and $l = 1$. Hence, $\mathscr{R}$ runs out of chips at the end of their turn and will lose during their next turn (which implies that $P(l)$ is true).

Induction Hypothesis (for parameter n): Assume that, for all $\nu \in \{0, 1, \ldots, n-1\}$, if $\mathscr{R}$ has at most ν chips in their hand at the end of their turn, then $\mathscr{B}$ has a winning strategy (which implies that $P(l)$ is true).

Induction Step (for parameter n): We now prove that $\mathscr{B}$ has a winning strategy if $\mathscr{R}$ cannot have more than n chips in their hands at the end of their turn (which implies that $P(l)$ is true).

As first player, $\mathscr{B}$ applies Strategy Σ. $\mathscr{B}$ captures k_b b-singletons, and places a b chip on the largest r-pile. Without loss of generality, suppose this r-pile is ρ_l, i.e., $|\rho_l|_r = \max\{|\rho_i|_r : i = 1, \ldots, l\}$.

The SLS board is composed of:

$$\begin{cases} k_e + k_b \text{ empty piles,} \\ k_r \text{ } r\text{-singletons,} \\ l - 1 \text{ } r\text{-piles of length at least 2,} \\ \text{and one } b\text{-pile of length at least 3, namely } \rho_l. \end{cases}$$

We have $\mathscr{B} = (m_b - 1, m_r)$ and it is $\mathscr{R}$'s turn to play. $\mathscr{R}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{R}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ blue chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ blue chips on } t' \text{ } r\text{-singletons,} \\ \text{placing } t'' \text{ blue chips on } t'' \text{ } r\text{-piles of length at least 2,} \\ \text{placing } s \text{ red chips on } s \text{ } r\text{-singletons,} \\ \text{and placing } s' \text{ red chips on } s' \text{ } r\text{-piles of length at least 2.} \end{cases}$$

The SLS board is now composed of

$$\begin{cases} k_e + k_b - t + s + s' \text{ empty piles,} \\ k_r - t' - s \text{ } r\text{-singletons,} \\ t \text{ } b\text{-singletons,} \\ t' \text{ } (r,b)\text{-piles,} \\ t'' + 1 \text{ } b\text{-piles of length at least 3,} \\ \text{and } l - t'' - s' - 1 \text{ } r\text{-piles of length at least 2.} \end{cases}$$

$=$

$$\begin{cases} k_e + k_b - t + s + s' \text{ empty piles,} \\ k_r - t' - s \text{ } r\text{-singletons,} \\ t \text{ } b\text{-singletons,} \\ t' + t'' + 1 \text{ } b\text{-piles of length at least 2,} \\ \text{and } l - t'' - s' - 1 \text{ } r\text{-piles of length at least 2.} \end{cases}$$

Without loss of generality, the s' r-piles that were captured are $\rho_1, \ldots, \rho_{s'}$ and the remaining r-piles of length at least 2 on the board are $\rho_{s'+1}, \ldots, \rho_{l-t''-1}$. The t'' b-piles of length at least 3 that were created are $\rho_{l-t''}, \rho_{l-t''+1}, \rho_{l-t''+2}, \ldots, \rho_{l-1}$. We rename the $t' + t'' + 1$ b-piles of length at least 2 to $\beta_1, \beta_2, \ldots, \beta_{t'+t''+1}$.

$$\mathscr{R} = \left(n_b - t - t' - t'' + \sum_{i=1}^{s'} \left(|\rho_i|_b - 1 \right), n_r + \sum_{i=1}^{s'} |\rho_i|_r \right)$$

can end their turn in 4 different ways.

1. $\mathscr{R}$ places an r chip on an empty pile (assuming $k_e + k_b - t + s + s' > 0$). The move automatically goes to $\mathscr{B}$, who captures all b-piles as per step 1 of Strategy Σ. The board is composed of $k_e + k_b + t' + t'' + s + s'$ empty piles, $k_r - t' - s + 1$ r-singletons, and $l - t'' - s' - 1 < l$ r-piles of length at least 2. The board is of generalized type I. We now have first player $\mathscr{B}$ and second player $\mathscr{R}$ such that

$$\begin{cases} \mathscr{B} = \left(m_b - 1 + \sum_{i=1}^{t'+t''+1} |\beta_i|_b, m_r \right), \\ \mathscr{R} = \left(n_b - t - t' - t'' + \sum_{i=1}^{s'} \left(|\rho_i|_b - 1 \right), n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1 \right). \end{cases}$$

If $n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1 = 0$, then $\mathscr{B}$ has a winning strategy by Proposition 4.5.2. Otherwise, we can apply the induction hypothesis on parameter l provided that the two inequalities are satisfied. For the first one, we have

$$m_b - 1 + \sum_{i=1}^{t'+t''+1} |\beta_i|_b \geq m_b - 1 + \sum_{i=1}^{1} |\beta_i|_b \geq m_b - 1 + 1 = m_b > 0.$$

We now prove the second one.

$$\begin{aligned}
& m_b - 1 + \sum_{i=1}^{t'+t''+1} |\beta_i|_b \\
&> m_b - 1 \\
&> n_r - 1 + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\
&= n_r - 1 + \sum_{i=1}^{l-1} |\rho_i|_r \\
&\geq n_r - 1 + \sum_{i=1}^{l-t''-1} |\rho_i|_r - \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-1\} \\
&= n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1 + \sum_{i=s'+1}^{l-t''-1} |\rho_i|_r - \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-1\}.
\end{aligned}$$

By the induction hypothesis on parameter l, $\mathscr{B}$ has a winning strategy.

2. $\mathscr{R}$ places an r chip on a b-singleton (assuming $t > 0$). The move automatically goes to $\mathscr{B}$, who captures all b-piles as per step 1 of Strategy Σ. The board is composed of $k_e + k_b + t' + t'' + s + s'$ empty piles, $k_r - t' - s$ r-singletons, and $l - t'' - s'$ r-piles of length at least 2. The board is of generalized type I. We now have first player $\mathscr{B}$ and second player $\mathscr{R}$ such that

$$\begin{cases} \mathscr{B} = \left(m_b - 1 + \sum_{i=1}^{t'+t''+1} |\beta_i|_b, m_r\right), \\ \mathscr{R} = \left(n_b - t - t' - t'' + \sum_{i=1}^{s'} \left(|\rho_i|_b - 1\right), n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1\right). \end{cases}$$

If $n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1 = 0$, then $\mathscr{B}$ has a winning strategy by Proposition 4.5.2. Otherwise, if $l - s' - t'' < l$, we can apply the induction hypothesis on parameter l provided that the two inequalities are satisfied. For the first one, we have

$$m_b - 1 + \sum_{i=1}^{t'+t''+1} |\beta_i|_b \geq m_b - 1 + \sum_{i=1}^{1} |\beta_i|_b \geq m_b - 1 + 1 = m_b > 0.$$

We now prove the second one.

$$\begin{aligned}
& m_b - 1 + \sum_{i=1}^{t'+t''+1} |\beta_i|_b \\
&\geq m_b \\
&> n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\
&= n_r + \sum_{i=1}^{l-1} |\rho_i|_r \\
&\geq n_r + \sum_{i=1}^{l-t''-1} |\rho_i|_r - \max\{|(b,r)|_r, \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-s'\}\} \\
&= n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1 \\
&\quad + \sum_{i=s'+1}^{l-t''-1} |\rho_i|_r + |(b,r)|_r - \max\{|(b,r)|_r, \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-s'\}\}.
\end{aligned}$$

By the induction hypothesis on parameter l, $\mathscr{B}$ has a winning strategy. Otherwise, we have $s' = t'' = 0$ and we can apply the induction hypothesis on parameter n provided that n has decreased. Since $s' = t'' = 0$, then

$$\begin{cases} \mathscr{B} = \left(m_b - 1 + \sum_{i=1}^{t'+1} |\beta_i|_b, m_r\right), \\ \mathscr{R} = (n_b - t - t', n_r - 1). \end{cases}$$

If $n_r - 1 = 0$, then $\mathscr{B}$ has a winning strategy by Proposition 4.5.2.
If $n_r - 1 \neq 0$, then the upper bound on the number of chips in $\mathscr{R}$'s hands at the end of the next turn is

$$
\begin{aligned}
&(n_b - t - t') + (n_r - 1) + \sum_{i=1}^{l-1}(|\rho_i| - 1) + (|(b,r)| - 1) \\
&\quad - \max\{|(b,r)|_r, \max\{|\rho_i|_r : i = 1, \ldots, l-1\}\} \\
&< n_b + (n_r - 1) + \sum_{i=1}^{l-1}(|\rho_i| - 1) + (|(b,r)| - 1) \\
&\quad - \max\{|(b,r)|_r, \max\{|\rho_i|_r : i = 1, \ldots, l-1\}\} && \text{(since } t > 0\text{)} \\
&= n_b + n_r + \sum_{i=1}^{l-1}(|\rho_i| - 1) - \max\{|\rho_i| : i = 1, \ldots, l-1\} \\
&= n_b + n_r + \sum_{i=1}^{l-1}(|\rho_i| - 1) - \max\{|\rho_i| : i = 1, \ldots, l-1\} \\
&\quad + (|\rho_l| - 1) - (|\rho_l| - 1) \\
&= n_b + n_r + \sum_{i=1}^{l}(|\rho_i| - 1) - \max\{|\rho_i| : i = 1, \ldots, l-1\} - |\rho_l| + 1 \\
&= n_b + n_r + \sum_{i=1}^{l}(|\rho_i| - 1) - \max\{|\rho_i| : i = 1, \ldots, l-1\} \\
&\quad - \max\{|\rho_i| : i = 1, \ldots, l\} + 1 \\
&\leq n_b + n_r + \sum_{i=1}^{l}(|\rho_i| - 1) - \max\{|\rho_i| : i = 1, \ldots, l\}. && \text{(since } l > 1\text{)}
\end{aligned}
$$

By the induction hypothesis on parameter n, $\mathscr{B}$ has a winning strategy.

3. $\mathscr{R}$ places an r chip on a b-pile of length at least 2. Without loss of generality, assume this b-pile is β_1. The move automatically goes to $\mathscr{B}$, who captures all b-piles as per step 1 of Strategy Σ. The board is composed of $k_e+k_b+t'+t''+s+s'$ empty piles, $k_r - t' - s$ r-singletons, and $l - t'' - s'$ r-piles of length at least 2. The board is of generalized type I. We now have first player $\mathscr{B}$ and second player $\mathscr{R}$ such that

$$
\begin{cases}
\mathscr{B} = \left(m_b - 1 + \sum_{i=2}^{t'+t''+1} |\beta_i|_b, m_r\right), \\
\mathscr{R} = \left(n_b - t - t' - t'' + \sum_{i=1}^{s'} (|\rho_i|_b - 1), n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1\right).
\end{cases}
$$

If $n_r + \sum_{i=1}^{s'} |\rho_i|_r - 1 = 0$, then $\mathscr{B}$ has a winning strategy by Proposition 4.5.2. Otherwise, if $l - s' - t'' < l$, we can apply the induction hypothesis on parameter l provided that the two inequalities are satisfied. For the first one, we have

$$m_b - 1 + \sum_{i=1}^{t'+t''+1} |\beta_i|_b \geq m_b - 1 \geq 2 - 1 = 1 > 0.$$

We now prove the second one.

$$\begin{aligned}
& m_b - 1 + \sum_{i=2}^{t'+t''+1} |\beta_i|_b \\
\geq\ & m_b - 1 \\
>\ & n_r - 1 + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\
=\ & n_r - 1 + \sum_{i=1}^{l-1} |\rho_i|_r \\
\geq\ & n_r - 1 + \sum_{i=1}^{l-t''-1} |\rho_i|_r \\
\geq\ & n_r - 1 + \sum_{i=1}^{l-t''-1} |\rho_i|_r + |(\beta_1, r)|_r \\
& - \max\{|(\beta_1, r)|_r, \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-1\}\} \\
=\ & n_r - 1 + \sum_{i=1}^{s'} |\rho_i|_r + \sum_{i=s'+1}^{l-t''} |\rho_i|_r + |(\beta_1, r)|_r \\
& - \max\{|(\beta_1, r)|_r, \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-1\}\}.
\end{aligned}$$

By the induction hypothesis on parameter l, $\mathscr{B}$ has a winning strategy.
Otherwise, we have $s' = t'' = 0$ and we can apply the induction hypothesis on parameter n provided that n has decreased. Since $s' = t'' = 0$, then

$$\begin{cases} \mathscr{B} = \left(m_b - 1 + \sum_{i=2}^{t'+1} |\beta_i|_b, m_r\right), \\ \mathscr{R} = (n_b - t - t', n_r - 1). \end{cases}$$

If $n_r - 1 = 0$, then $\mathscr{B}$ has a winning strategy by Proposition 4.5.2.
If $n_r - 1 \neq 0$, then the upper bound on the number of chips in $\mathscr{R}$'s hands at the end of the next turn is

$$
\begin{aligned}
&(n_b - t - t') + (n_r - 1) + \sum_{i=1}^{l-1} (|\rho_i| - 1) + (|(\beta_1, r)| - 1) \\
&\quad - \max\{|(\beta_1, r)|_r, \max\{|\rho_i|_r : i = 1, \ldots, l-1\}\} \\
&\leq n_b + (n_r - 1) + \sum_{i=1}^{l-1} (|\rho_i| - 1) + (|(\beta_1, r)| - 1) \\
&\quad - \max\{|(\beta_1, r)|_r, \max\{|\rho_i|_r : i = 1, \ldots, l-1\}\} \\
&\leq n_b + (n_r - 1) + \sum_{i=1}^{l-1} (|\rho_i| - 1) - 1 \\
&= n_b + (n_r - 1) + \sum_{i=1}^{l-1} (|\rho_i| - 1) + (|\rho_l| - 1) - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\
&= n_b + (n_r - 1) + \sum_{i=1}^{l} (|\rho_i| - 1) - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\
&< n_b + n_r + \sum_{i=1}^{l} (|\rho_i| - 1) - \max\{|\rho_i|_r : i = 1, \ldots, l\}.
\end{aligned}
$$

By the induction hypothesis on parameter n, $\mathscr{B}$ has a winning strategy.

4. $\mathscr{R}$ places a b chip on a b-pile, which gets captured by $\mathscr{B}$. The move automatically goes to $\mathscr{B}$, who captures all b-piles as per step 1 of Strategy Σ. The board is composed of $k_e + k_b + t' + t'' + s + s' + 1$ empty piles, $k_r - t' - s$ r-singletons, and $l - s' - t'' - 1$ r-piles of length at least 2. The board is of generalized type I. We now have first player $\mathscr{B}$ and second player $\mathscr{R}$ such that

$$
\begin{cases}
\mathscr{B} = (m_b + \sum_{i=1}^{t'+t''+1} |\beta_i|_b, m_r), \\
\mathscr{R} = \left(n_b - t - t' - t'' + \sum_{i=1}^{s'} (|\rho_i|_b - 1) - 1, n_r + \sum_{i=1}^{s'} |\rho_i|_r\right).
\end{cases}
$$

Since $l - s' - t'' - 1 < l$, we can apply the induction hypothesis on parameter l provided that the two inequalities are satisfied.

For the first one, we have

$$
m_b + \sum_{i=1}^{t'+t''+1} |\beta_i|_b > m_b > 0
$$

We now prove the second one.

$$
\begin{aligned}
& m_b + \sum_{i=1}^{t'+t''+1} |\beta_i|_b \\
&> m_b \\
&> n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\
&= n_r + \sum_{i=1}^{l-1} |\rho_i|_r \\
&\geq n_r + \sum_{i=1}^{l-t''-1} |\rho_i|_r - \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-1\} \\
&= (n_r + \sum_{i=1}^{s'} |\rho_i|_r) + (\sum_{i=s'+1}^{l-t''-1} |\rho_i|_r - \max\{|\rho_i|_r : i = s'+1, \ldots, l-t''-1\})
\end{aligned}
$$

By the induction hypothesis on parameter l, $\mathscr{B}$ has a winning strategy.

This completes the proof. □

Proposition 4.5.4. *Consider an SLS board of generalized type I. Let $\mathscr{B} = (m_b, m_r)$ be the first player, and $\mathscr{R} = (n_b, n_r)$ be the second player. If*

$$m_b \leq n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$$

then $\mathscr{R}$ has a winning strategy.

Proof. Suppose we have an SLS board of generalized type I with k piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, and l are r-piles of length ≥ 2, such that $k = k_e + k_r + k_b + l$.
As first player, $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$
\begin{cases}
\text{placing } t \text{ red chips on } t \text{ empty piles,} \\
\text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\
\text{and placing } s \text{ blue chips on } s \text{ } b\text{-singletons.}
\end{cases}
$$

The SLS board is now composed of

$$\begin{cases} k_e - t + s \text{ empty piles}, \\ k_r + t \ r\text{-singletons}, \\ k_b - t' - s \ b\text{-singletons}, \\ l \ r\text{-piles of length} \geq 2, \\ \text{and } t' \ (b,r)\text{-piles}. \end{cases} = \begin{cases} k_e - t + s \text{ empty piles}, \\ k_r + t \ r\text{-singletons}, \\ k_b - t' - s \ b\text{-singletons}, \\ \text{and } l + t' \ r\text{-piles of length at least 2}. \end{cases}$$

We rename the t' (b,r)-piles to $\rho_{l+1}, \ldots, \rho_{l+t'}$. $\mathscr{B} = (m_b, m_r - t - t')$ can end their turn in 5 different ways.

1. $\mathscr{B}$ places a b chip on an empty pile (assuming $k_e - t + s > 0$ and $m_b > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R} = (n_b, n_r)$ applies Strategy Σ: captures $k_r + t$ r-singletons, $l + t'$ r-piles of length at least 2, and places an r chip on a b-singleton. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + l + t' + s - 1$ empty piles, $k_b - t' - s$ b-singletons, and one (b,r)-pile. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$ and second player $\mathscr{R}(n_b, n_r + \sum_{i=1}^{l+t'} |\rho_i|_r - 1)$. Since

$$\begin{aligned} & n_r + \sum_{i=1}^{l+t'} |\rho_i|_r - 1 \\ \geq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r - 1 \\ \geq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\ \geq\ & m_b \geq m_b - 1, \end{aligned}$$

   then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ places a b chip on an r-singleton (assuming $k_r + t > 0$ and $m_b > 0$). The move automatically goes to $\mathscr{R}$. $\mathscr{R} = (n_b, n_r)$ applies Strategy Σ: captures $k_r + t - 1$ r-singletons, $l + t'$ r-piles of length at least 2, and places an r chip on the (r,b)-pile. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + l + t' + s - 1$ empty piles, $k_b - t' - s$ b-singletons, and one (r,b,r)-pile. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{l+t'} |\rho_i|_r - 1)$. The same derivation as in case 1 shows that we can apply Theorem 4.3.4, from which $\mathscr{R}$ has a winning strategy.

3. $\mathscr{B}$ places a b chip on an r-pile of length at least 2 (assuming $l + t' > 0$ and $m_b > 0$). Without loss of generality, we suppose $\rho_{l+t'}$ is that pile. The move

automatically goes to $\mathscr{R}$. $\mathscr{R} = (n_b, n_r)$ applies Strategy Σ: captures $k_r + t$ r-singletons, $l+t'-1$ ρ_i-piles of length at least 2, and places an r chip on the (ρ_i, b)-pile. The move goes back to $\mathscr{B}$. The board is composed of $k_e + k_r + l + t' + s - 1$ empty piles, $k_b - t' - s$ b-singletons, and the (ρ_i, b, r)-pile. The board is of type I. We now have first player $\mathscr{B} = (m_b - 1, m_r - t - t')$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{l+t'-1} |\rho_i|_r - 1)$. Since $n_r + \sum_{i=1}^{l+t'-1} |\rho_i|_r - 1 \geq n_r + (\sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}) - 1 \geq m_b - 1$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

4. $\mathscr{B}$ places an r chip on an r-pile (assuming $k_r + t > 0$ or $l + t' > 0$, and $m_r - t - t' > 0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + l + t + t' + s$ empty piles, and $k_b - t' - s$ b-singletons.

   - If $k_b - t' - s > 0$, then $\mathscr{R}$ places an r chip on a b-singleton.
   - Otherwise, $\mathscr{R}$ places an r chip on an empty pile.

   In both cases, the board is of type I. We have first player $\mathscr{B} = (m_b, m_r - t - t' - 1)$, and second player $\mathscr{R} = (n_b, n_r + \sum_{i=1}^{l+t'} |\rho_i|_r)$. Since $n_r + \sum_{i=1}^{l+t'} |\rho_i|_r \geq n_r + \sum_{i=1}^{l+t'} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \geq m_b$, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

5. $\mathscr{B}$ runs out of chips if $m_b = 0$ and $m_r - t - t' = 0$. $\mathscr{B}$ loses by default and $\mathscr{R}$ wins.

It follows that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Theorem 4.5.5. . *Consider an SLS board of generalized type I. Let $\mathscr{B} = (m_b, m_r)$ be the first player, and $\mathscr{R} = (n_b, n_r)$ be the second player.*

$$\mathscr{B} \text{ has a winning strategy} \iff m_b > 0 \text{ and } m_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$$

Proof. Suppose $\mathscr{B} = (m_b, m_r)$ is the first player, and $\mathscr{R} = (n_b, n_r)$ is the second player.

- If $n_r = 0$, then $\mathscr{B}$ has a winning strategy by Proposition 4.5.2.
- If $m_b > 0$ and $m_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$, then $\mathscr{B}$ has a winning strategy by Proposition 4.5.3.
- If $m_b = 0$ or $m_b \leq n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$, then $\mathscr{R}$ has a winning strategy by Proposition 4.5.4.

This covers all possible cases and proves our theorem. □

4.6 Generalized Type II Board

In this section, we assume that if the sum is empty then this means there are no piles to be considered and the sum is 0 as a result.

Definition 4.6.1. *An SLS board is said to be of* ***generalized type II*** *if it contains zero or more singletons, $h \geq 1$ b-piles $\beta_1, \ldots, \beta_h$ of length at least 2, and exactly one r-pile of length at least 2.*

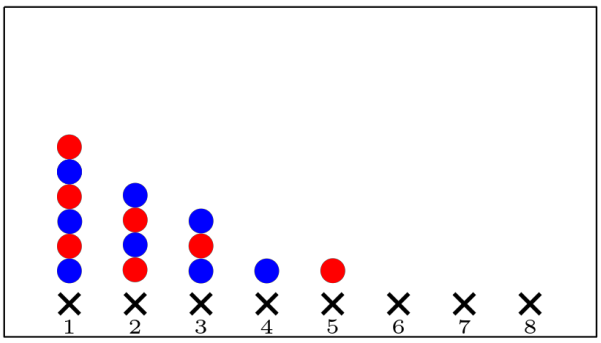

Figure 4.7: SLS Board of Generalized type II with $k = 8$

Proposition 4.6.2. *Consider an SLS board of generalized type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If*

$$m_b > 0 \text{ and } m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r$$

then $\mathscr{B}$ has a winning strategy.

Proof. Suppose we have an SLS board of generalized type II with $k \geq 2$ piles $\pi_1, \pi_2, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, h b-piles of length at least 2, and one r-pile of length at least 2, where $k = k_e + k_r + k_b + h + 1$.

Suppose $m_b > 0$ and $m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r$. As first player, $\mathscr{B}$ captures all b-piles, as per step 1 of Strategy Σ. The board is composed of $k_e + k_b + h$ empty piles, k_r r-singletons, and one r-pile of length at least 2. The board is of type I. We now have first player $\mathscr{B} = (m_b + \sum_{i=1}^{h} |\beta_i|_b, m_r)$, and second player $\mathscr{R} = (n_b, n_r)$. Since $m_b > 0$ and $m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r$ (by assumption), $\mathscr{B}$ has a winning strategy by Theorem 4.3.4. □

Proposition 4.6.3. *Consider an SLS board of generalized type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b = 0$ then $\mathscr{R}$ has a winning strategy.*

Proof. Suppose we have an SLS board of generalized type II with $k \geq 2$ piles such that k_e are empty, k_r are r-singletons, k_b are b-singletons, h b-piles of length at least 2, and one r-pile ρ of length at least 2 such that $k = k_e + k_r + k_b + h + 1$. We will prove by induction on h that Strategy Σ is a winning strategy for $\mathscr{R}$.

If $h = 1$ then the board is of type II. By Theorem 4.4.9, $\mathscr{R}$ has a winning strategy.

Suppose $\mathscr{R}$ has a winning strategy against player $\mathscr{B}$ for any board of generalized type II with j b-piles of length at least 2 such that $1 \leq j \leq h-1$ and $m_b = 0$. We now consider an SLS board of generalized type II with h b-piles of length at least 2, first player $\mathscr{B} = (0, m_r)$, and second player $\mathscr{R} = (n_b, n_r)$.

If $m_r = 0$, then $\mathscr{B}$ loses by default since they cannot make the first move; and $\mathscr{R}$ wins. Suppose $m_r \geq 1$. $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{and placing } t'' \text{ red chips on } t'' \text{ } b\text{-piles.} \end{cases}$$

The SLS board is now composed of

$$\begin{cases} k_e - t \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' \text{ } b\text{-singletons,} \\ t' \text{ } (b,r)\text{-piles,} \\ h - t'' \text{ } b\text{-piles of length at least 2,} \\ \text{and } t'' \text{ } r\text{-piles of length at least 3.} \\ \text{and 1 } r\text{-piles of length at least 2.} \end{cases} = \begin{cases} k_e - t \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' \text{ } b\text{-singletons,} \\ h - t'' \text{ } b\text{-piles of length at least 2,} \\ \text{and } t' + t'' + 1 \text{ } r\text{-piles of length at least 2.} \end{cases}$$

We denote the $t' + t'' + 1$ r-piles of length ≥ 2 by $\rho_1, \rho_2, \ldots, \rho_{t'+t''+1}$; and the $h - t''$ b-piles of length ≥ 2 by $\beta_1, \beta_2, \ldots, \beta_{h-t''}$. Without loss of generality, $\mathscr{B} = (0, m_r - t - t' - t'')$ can end their turn in 2 different ways.

1. $\mathscr{B}$ places an r chip on an r-pile (assuming $m_r - t - t' - t'' > 0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + t' + t'' + 1$ empty piles, $k_b - t'$ b-singletons, and $h - t''$ b-piles of length at least 2.

   - If $h - t'' > 0$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length. The move goes back to $\mathscr{B}$. The board is of generalized type II with $h - t'' - 1 < h$ b-piles of length ≥ 2. By the induction hypothesis, $\mathscr{R}$ has a winning strategy.
   - If $h - t'' = 0$ and $k_b - t' > 0$, then $\mathscr{R}$ places an r chip on a b-singleton. The move goes back to $\mathscr{B}$. The board is of type I. Since $m_b = 0$, $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

- If $h - t'' = 0$ and $k_b - t' = 0$, then $\mathscr{R}$ places an r chip on an empty pile. The move goes back to $\mathscr{B}$. The board is of type I. Since $m_b = 0$, $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ runs out of chips if $m_r - t - t' - t'' = 0$. $\mathscr{B}$ loses by default and $\mathscr{R}$ wins.

It follows by induction that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Proposition 4.6.4. *Consider an SLS board of generalized type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b > 0$ and*

$$m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r$$

then $\mathscr{R}$ has a winning strategy.

Proof. Suppose we have an SLS board of generalized type II with $k \geq 2$ piles $\pi_1, \ldots, \pi_k$ among which k_e are empty, k_r are r-singletons, k_b are b-singletons, $h \geq 0$ are b-piles of length at least 2 and one r-pile of length at least 2, where $k = k_e + k_r + k_b + h + 1$. We will prove this proposition by induction on h. Let $P(h)$ be the following statement:

"If we have an SLS board of generalized type II with h b-piles of length at least 2 such that $m_b > 0$ and $m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r$ then $\mathscr{R}$ has a winning strategy."

Base Case: If $h = 0$, then the board is of type I. The inequalities becomes $m_b > 0$ and $m_b \leq n_r$. By Theorem 4.3.4, $\mathscr{R}$ has a winning strategy.
If $h = 1$, then the board is of type II. The inequalities become $m_b > 0$ and $m_b + |\beta_1|_b \leq n_r$. By Theorem 4.4.9, $\mathscr{R}$ has a winning strategy.
Induction Hypothesis: Suppose $P(\eta)$ holds for all $\eta \in \{0, 1, 2, \ldots, h-1\}$.
Induction Step: We now prove that $P(h)$ is also true. Suppose we have an SLS board of generalized type II where $m_b > 0$ and

$$m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r. \tag{4.6.1}$$

This inequality, along with the fact that $m_b > 0$, imply that $n_r \geq 1$. Let us fix h and let $m = m_b + m_r + \sum_{i=1}^{h} (|\beta_i| - 1) - 1$. This corresponds to maximum possible number of chips in $\mathscr{B}$'s hands at the end of their turn, if $\mathscr{B}$ were to capture all b-piles on the board.
Since there is no guarantee that $\mathscr{B}$ will play this way, the parameter m is only an upper bound on the total number of chips in their hands. Nevertheless, m is a key parameter in this proof. Indeed, we will prove by induction on m that $P(h)$ is true.

Base Case (for parameter m): Consider the case where $m = 0$. This means that $m_b = 1$ (since $m_b > 0$ by assumption), $m_r = 0$, $\sum_{i=1}^{h}(|\beta_i| - 1) = 0$ and $h = 0$. $\mathscr{B}$ runs out of chips at the end of their first turn and will lose once the move goes back to them by Proposition 4.6.3.
Induction Hypothesis (for parameter m): Assume that, for all $\mu \in \{1, \ldots, m-1\}$, if $\mathscr{B}$ has at most μ chips in their hands at the end of their turn, then $\mathscr{R}$ has a winning strategy.
Induction Step (for parameter m): We now prove that $\mathscr{R}$ has a winning strategy if $\mathscr{B}$ cannot have more than m chips in their hands at the end of their turn.
As first player, $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{placing } t'' \text{ red chips on } t'' \text{ } b\text{-piles of length at least 2,} \\ \text{placing } s \text{ blue chips on } s \text{ } b\text{-singletons,} \\ \text{and placing } s' \text{ blue chips on } s' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

The SLS board is now composed of:

$$\begin{cases} k_e - t + s + s' \text{ empty piles,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ k_r + t \text{ } r\text{-singletons,} \\ t' \text{ } (b,r)\text{-piles,} \\ t'' + 1 \text{ } r\text{-piles of length at least 3,} \\ \text{and } h - s' - t'' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

$=$

$$\begin{cases} k_e - t + s + s' \text{ empty piles,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ k_r + t \text{ } r\text{-singletons,} \\ t' + t'' + 1 \text{ } r\text{-piles of length at least 2,} \\ \text{and } h - s' - t'' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

Without loss of generality, the s' b-piles that were captured are $\beta_1, \ldots, \beta_{s'}$ and the remaining b-piles of length at least 2 on the board are $\beta_{s'+1}, \ldots, \beta_{h-t''}$. The r-piles of length at least 3 that were created are $\beta_{h-t''+1}, \ldots, \beta_h$. We rename the $t' + t'' + 1$ r-piles of length at least 2 to $\rho_1, \ldots, \rho_{t'+t''+1}$. Player

$$\mathscr{B} = \left(m_b + \sum_{i=1}^{s'} |\beta_i|_b, m_r - t - t' - t'' + \sum_{i=1}^{s'} (|\beta_i|_r - 1) \right)$$

can end their turn in 4 different ways.

1. $\mathscr{B}$ places a b chip on an empty pile (assuming $k_e - t + s + s' > 0$). The move automatically goes to $\mathscr{R}$, who captures all r-piles as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + t' + t'' + s + s' + 1$ empty piles, $k_b - t' - s$ b-singletons and $h - t'' - s'$ b-piles of length at least 2. $\mathscr{R}$ is going to place an r chip on the board (as per step 2 of Strategy Σ). The move will then go back to $\mathscr{B}$. We will have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1, m_r - t - t' - t'' + \sum_{i=1}^{s'} (|\beta_i|_r - 1)\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r - 1\right). \end{cases}$$

The analysis of the game depends on the values of parameters m_b, s', h, and t''. If $m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1 = 0$, then $\mathscr{R}$ has a winning strategy by Proposition 4.6.3. Otherwise, we have several sub-cases to consider.

- If $h - t'' - s' - 1 > 0$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''}$ be this pile. The board is now of generalized type II. Since $0 < h - t'' - s' - 1 < h$, we can apply the induction hypothesis on parameter h, provided that the inequality is satisfied. We have

$$\begin{aligned} &(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1) + \sum_{i=s'+1}^{h-t''-1} |\beta_i|_b \\ = m_b + &\sum_{i=1}^{h-t''-1} |\beta_i|_b - 1 \\ < m_b + &\sum_{i=1}^{h} |\beta_i|_b \\ \leq n_r \quad &\text{(by 4.6.1)} \\ \leq n_r + &\sum_{i=1}^{t'+t''+1} |\rho_i|_r - 1. \end{aligned}$$

  By the induction hypothesis on parameter h, $\mathscr{R}$ has a winning strategy.

- If $h - t'' - s' = 1$, then $\mathscr{R}$ places an r chip on the only b-pile of length at least 2. The board is now of type I. Since

$$m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1 < m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r \leq n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r - 1, \quad (4.6.2)$$

  the $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

- If $h - t'' - s' = 0$, then $\mathscr{R}$ places an r chip on a b-singleton. Since we have the same inequality 4.6.2, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ places a b chip on an r-singleton (assuming $k_r + t > 0$). The move automatically goes to $\mathscr{R}$, who captures all r-piles as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + t' + t'' + s + s'$ empty piles, $k_b - t' - s$ b-singletons and $h - t'' - s' + 1$ b-piles of length at least 2. $\mathscr{R}$ is going to place an r chip on the board (as per step 2 of Strategy Σ). The move will then go back to $\mathscr{B}$. We will have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1, m_r - t - t' - t'' + \sum_{i=1}^{s'} (|\beta_i|_r - 1)\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r - 1\right). \end{cases}$$

The analysis of the game depends on the values of the parameters m_b, s', h, and t''. If $m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1 = 0$ then $\mathscr{R}$ has a winning strategy by Proposition 4.6.3. Otherwise, we have several sub-cases to consider.

- If $h - t'' - s' = h$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''} = \beta_h$ be this pile. The board is now of generalized type II. Since the number of b-piles did not decrease, we cannot apply the induction hypothesis on h. However, since $t'' = s' = 0$, we have

$$\begin{aligned} &(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1) + (m_r - t - t' - t'' + \sum_{i=1}^{s'} (|\beta_i|_r - 1)) \\ &\quad + (\sum_{i=s'+1}^{h-t''-1} (|\beta_i| - 1) + (|(r,b)| - 1) - 1) \\ &= m_b + m_r - t - t' + \sum_{i=1}^{h-1} (|\beta_i| - 1) - 1 \\ &< m_b + m_r + \sum_{i=1}^{h} (|\beta_i| - 1) - 1, \end{aligned}$$

  and we can apply the induction hypothesis on parameter m. Therefore, $\mathscr{R}$ has a winning strategy.

- If $0 < h - t'' - s' < h$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length. Without loss off generality, let $\beta_{h-t''}$ be this pile. The board is of generalized type II. Since $0 < h - t'' - s' < h$, we can apply the induction

hypothesis on parameter h provided tat the inequality is satisfied. Using inequality 4.6.1, we have

$$\begin{aligned}
&(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1) + \sum_{i=s'+1}^{h-t''-1} |\beta_i|_b + |(r,b)|_b \\
&= m_b + \sum_{i=1}^{h-t''-1} |\beta_i|_b \\
&< m_b + \sum_{i=1}^{h} |\beta_i|_b \\
&\leq n_r \\
&\leq n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r - 1.
\end{aligned}$$

By the induction hypothesis on parameter h, $\mathscr{R}$ has a winning strategy.

- If $h - t'' - s' = 0$, then $\mathscr{R}$ places an r chip on the (r,b)-pile. The board is now of type I. Since

$$m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1 < m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r \leq n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r - 1,$$

then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

3. $\mathscr{B}$ places a b chip on an r-pile of length at least 2. Without loss of generality, suppose $\rho_{t'+t''+1}$ is that pile. The move automatically goes to $\mathscr{R}$, who captures all r-piles as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + t' + t'' + s + s'$ empty piles, $k_b - t' - s$ b-singletons and $h - t'' - s' + 1$ b-piles of length at least 2. $\mathscr{R}$ is going to place an r chip on the board (as per step 2 of Strategy Σ). The move will then go back to $\mathscr{B}$. We will have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1, m_r - t - t' - t'' + \sum_{i=1}^{s'} (|\beta_i|_r - 1)\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{t'+t''} |\rho_i|_r - 1\right). \end{cases}$$

Te analysis of the game depends on the values of parameters m_b, s', h, and t''. If $m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1 = 0$, then $\mathscr{R}$ has a winning strategy by Proposition 4.6.3. Otherwise, we consider 3 sub-cases based on the number of b-piles of length at least 2.

- If $h - t'' - s' + 1 = h + 1$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''} = \beta_h$ be this pile. The board is now of generalized type II. The number of b-piles of length at least 2 remains h and we cannot apply the induction hypothesis on parameter h. However, since β_h was the b-pile of greatest length among $\beta_{s'+1}, \ldots, \beta_h$, and $(\rho_{t'+t''+1}, b)$, we have $|(\rho_{t'+t''+1}, b)| \leq |\beta_h|$. Therefore, we have

$$\begin{aligned}
&(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1) + (m_r - t - t' - t'' + \sum_{i=1}^{s'} (|\beta_i|_r - 1) \\
&\quad + \sum_{i=s'+1}^{h-t''-1} (|\beta_i| - 1) + (|(\rho_{t'+t''+1}, b)| - 1) - 1 \\
&= m_b - 1 + m_r - t - t' + \sum_{i=1}^{h-1} (|\beta_i| - 1) + (|(\rho_{t'+1}, b)| - 1) \\
&\leq m_b - 1 + m_r + \sum_{i=1}^{h-1} (|\beta_i| - 1) + (|\beta_h| - 1) - 1 \\
&< m_b + m_r + \sum_{i=1}^{h} (|\beta_i| - 1) - 1.
\end{aligned}$$

  and we can apply the induction hypothesis on parameter m. Therefore, $\mathscr{R}$ has a winning strategy.

- If $1 < h - t'' - s' + 1 \leq h$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''}$ be this pile. The board is now of generalized type II. Since $0 < h - t'' - s' < h$, we can apply the induction hypothesis on parameter h provided that the inequality is satisfied. Since $\beta_{h-t''}$ was the b-pile of greatest length among $\beta_{s'+1}, \ldots, \beta_{h-t''}$, and $(\rho_{t'+t''+1}, b)$, we have $|(\rho_{t'+t''+1}, b)|_b \leq |\beta_{h-t''}|_b$. Using

inequality 4.6.1, we have

$$\begin{aligned}
&(m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1) + (\sum_{i=s'+1}^{h-t''-1} |\beta_i|_b + |(\rho_{t'+t''+1}, b)|_b) \\
&= m_b + \sum_{i=1}^{h-t''-1} |\beta_i|_b + |(\rho_{t'+t''+1}, b)|_b - 1 \\
&< m_b + \sum_{i=1}^{h-t''-1} |\beta_i|_b + |\beta_{h-t''}|_b \\
&= m_b + \sum_{i=1}^{h} |\beta_i|_b \\
&\leq n_r \\
&\leq n_r + \sum_{i=1}^{t'+t''} |\rho_i|_r - 1.
\end{aligned}$$

By the induction hypothesis on parameter h, $\mathscr{R}$ has a winning strategy.

- If $h - t'' - s' + 1 = 1$, then $\mathscr{R}$ places an r chip on the b-pile of length at least 2. The board is now of type I. Since

$$m_b + \sum_{i=1}^{s'} |\beta_i|_b - 1 < m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r \leq n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r - 1,$$

then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

4. $\mathscr{B}$ places an r chip on an r-pile. The move automatically goes to $\mathscr{R}$, who captures all r-piles as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + t' + t'' + s + s' + 1$ empty piles, $k_b - t' - s$ b-singletons and $h - t'' - s'$ b-piles of length at least 2. $\mathscr{R}$ is going to place an r chip on the board (as per step 2 of Strategy Σ). The move will then go back to $\mathscr{B}$. We will have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=1}^{s'} |\beta_i|_b, n_r - t - t' - t'' + \sum_{i=1}^{s'} (|\beta_i|_r - 1) - 1\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r\right). \end{cases}$$

If $m_b + \sum_{i=1}^{s'} |\beta_i|_b = 0$ then $\mathscr{R}$ has a winning strategy by Proposition 4.6.3. Otherwise, we consider 3 sub-cases based on the number of b-piles of length at least 2.

- If $h - t'' - s' > 1$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''}$ be this pile. The board is now of generalized type II. Since $0 < h - t'' - s' - 1 < h$, we can apply the induction hypothesis on parameter h provided that the inequality is satisfied. Using inequality 4.6.1, we have

$$\begin{aligned} & m_b + \sum_{i=1}^{s'} |\beta_i|_b + \sum_{i=s'+1}^{h-t''-1} |\beta_i|_b \\ = \; & m_b + \sum_{i=1}^{h-t''-1} |\beta_i|_b \\ < \; & m_b + \sum_{i=1}^{h} |\beta_i|_b \\ \leq \; & n_r \\ \leq \; & n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r. \end{aligned}$$

  By the induction hypothesis on parameter h, $\mathscr{R}$ has a winning strategy.
- If $h - t'' - s' = 1$, then $\mathscr{R}$ places an r chip on the only b-pile of length at least 2. The board is now of type I. Since

$$m_b + \sum_{i=1}^{s'} |\beta_i|_b \leq m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r \leq n_r + \sum_{i=1}^{t'+t''+1} |\rho_i|_r, \tag{4.6.3}$$

  then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.
- If $h - t'' - s' = 0$, then $\mathscr{R}$ places an r chip on a b-singleton (if $k_b - t' - s > 0$) or on an empty pile (if $k_b - t' - s = 0$). The board is now of type I. Since we have the same inequality 4.6.3, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

□

Theorem 4.6.5. *Consider an SLS board of generalized type II. Let $\mathscr{B} = (m_b, m_r)$ be the first player, and $\mathscr{R} = (n_b, n_r)$ be the second player.*

$$\mathscr{B} \text{ has a winning strategy} \iff m_b > 0 \text{ and } m_b + \sum_{j=1}^{h} |\beta_i|_b > n_r$$

Proof. Suppose $\mathscr{B} = (m_b, m_r)$ is the first player, and $\mathscr{R} = (n_b, n_r)$ is the second player. We separate the proof into 3 cases.

1. If $m_b > 0$ and $m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r$, then $\mathscr{B}$ has a winning strategy by Proposition 4.6.2.
2. If $m_b = 0$, then $\mathscr{R}$ has a winning strategy by Proposition 4.6.3.
3. If $m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r$, then $\mathscr{R}$ has a winning strategy by Proposition 4.6.4.

This covers all possible cases and proves our theorem. □

4.7 General Case

In this section, we look at SLS boards composed of k piles such that k_e are empty, k_r are r-singletons, k_b are b-singletons, h are b-piles $\beta_1, \beta_2, \ldots, \beta_h$ of length at least 2, and l are r-piles $\rho_1, \rho_2, \ldots, \rho_l$ of length at least 2.

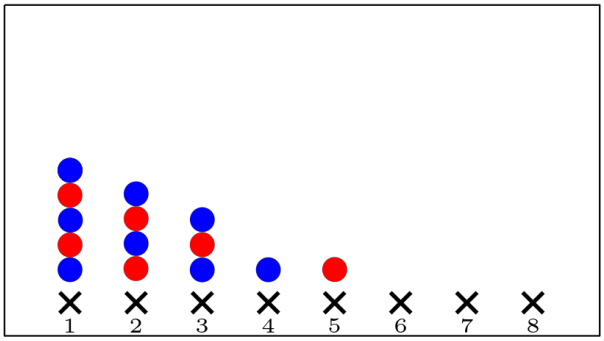

Figure 4.8: SLS Board with $k = 8$, $h = 3$, and $l = 0$

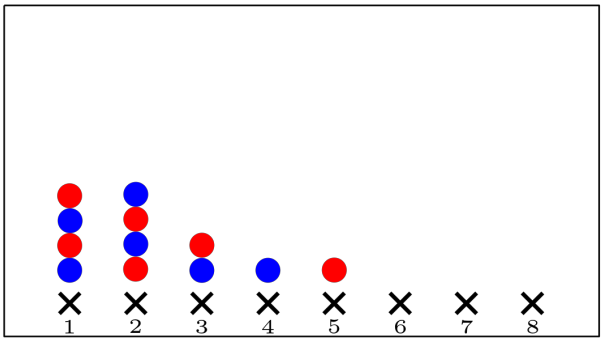

Figure 4.9: SLS Board with $k = 8$, $h = 1$, and $l = 2$

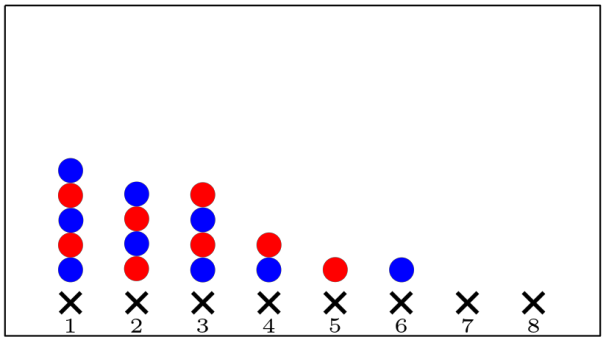

Figure 4.10: SLS Board with $k = 8$, $h = 2$, and $l = 2$

Proposition 4.7.1. *Consider an SLS board composed of k piles such that k_e are empty, k_r are r-singletons, k_b are b-singletons, h are b-piles $\beta_1, \beta_2, \ldots, \beta_h$ of length at least 2, and l are r-piles $\rho_1, \rho_2, \ldots, \rho_l$ of length at least 2. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b > 0$ and $n_r = 0$, then $\mathscr{B}$ has a winning strategy.*

Proof. As first player and since $m_b > 0$, $\mathscr{B}$ captures all b-piles, as per step 1 of Strategy Σ. The board is now composed of $k_e + k_b + h$ empty piles, k_r r-singletons, and l r-piles of length at least 2. The board is of generalized type I. We have first player $\mathscr{B} = (m_b + \sum_{i=1}^{h} |\beta_i|_b, m_r)$, and second player $\mathscr{R} = (n_b, 0)$ such that $m_b + \sum_{i=1}^{h} |\beta_i|_b \geq m_b > 0$. By Proposition 4.5.2, $\mathscr{B}$ has a winning strategy. □

Proposition 4.7.2. *Consider an SLS board composed of k piles such that k_e are empty, k_r are r-singletons, k_b are b-singletons, h are b-piles $\beta_1, \beta_2, \ldots, \beta_h$ of length at least 2, and l are r-piles $\rho_1, \rho_2, \ldots, \rho_l$ of length at least 2. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If*

$$m_b, n_r > 0 \text{ and } m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$$

then $\mathscr{B}$ has a winning strategy.

Proof. As first player and since $m_b > 0$, $\mathscr{B}$ captures all b-piles, as per step 1 of Strategy Σ. The board is now composed of $k_e + k_b + h$ empty piles, k_r r-singletons, and l r-piles of length at least 2. Then the board is of generalized type I. We have first player $\mathscr{B} = (m_b + \sum_{i=1}^{h} |\beta_i|_b, m_r)$, and second player $\mathscr{R} = (n_b, n_r)$. The inequality $m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$ holds by assumption. By Theorem 4.5.5, $\mathscr{B}$ has a winning strategy. □

Proposition 4.7.3. *Consider an SLS board composed of k piles such that k_e are empty, k_r are r-singletons, k_b are b-singletons, h are b-piles $\beta_1, \beta_2, \ldots, \beta_h$ of length*

at least 2, and l are r-piles $\rho_1, \rho_2, \ldots, \rho_l$ of length at least 2. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b = 0$ then $\mathscr{R}$ has a winning strategy.

Proof. Suppose $m_b = 0$. As first player, $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{and placing } t'' \text{ red chips on } t'' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

The SLS board is now composed of

$$\begin{cases} k_e - t \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' \text{ } b\text{-singletons,} \\ t' \text{ } (b,r)\text{-piles,} \\ l \text{ } r\text{-piles of length at least 2,} \\ t'' \text{ } r\text{-piles of length at least 3,} \\ \text{and } h - t'' \text{ } b\text{-piles of length at least 2.} \end{cases} = \begin{cases} k_e - t \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' \text{ } b\text{-singletons,} \\ l + t' + t'' \text{ } r\text{-piles of length at least 2,} \\ \text{and } h - t'' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

$\mathscr{B} = (0, m_r - t - t' - t'')$ can end their turn in 2 different ways.

1. $\mathscr{B}$ places an r chip on an r-pile (assuming $m_r - t - t' - t'' > 0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. The boad is now composed of $k_e + k_r + l + t' + t''$ empty piles, $k_b - t'$ b-singletons, and $h - t''$ b-piles of length at least 2. The analysis of the game depends on the number of b-piles of length at least 2.

   (a) If $h - t'' > 1$, then $\mathscr{R}$ places an r chip on the b-pile of greatest length (as per step 2 of Strategy Σ). The board is now of generalized type II. Since $m_b = 0$, then $\mathscr{R}$ has a winning strategy by Theorem 4.6.5.

   (b) If $h - t'' = 1$, then $\mathscr{R}$ places an r chip on the b-pile of length at least 2 (as per step 2 of strategy Σ). The board is now of type I. Since $m_b = 0$, $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

   (c) If $h - t'' = 0$, then $\mathscr{R}$ places an r chip on a b-singleton (if $k_b - t' > 0$) or on an empty pile (if $k_b - t' = 0$). The board is now of type I. Since $m_b = 0$, $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ runs out of chips if $m_r - t - t' - t'' = 0$. $\mathscr{B}$ loses by default and $\mathscr{R}$ wins.

It follows that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Proposition 4.7.4. *Consider an SLS board composed of k piles such that k_e are empty, k_r are r-singletons, k_b are b-singletons, h are b-piles $\beta_1, \beta_2, \ldots, \beta_h$ of length at least 2, and l are r-piles $\rho_1, \rho_2, \ldots, \rho_l$ of length at least 2. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. If $m_b, n_r > 0$ and*

$$m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \tag{4.7.1}$$

then $\mathscr{R}$ has a winning strategy.

Proof. If $h = 0$ then the board is of generalized type I. The inequality becomes

$$m_b + \sum_{i=1}^{h} |\beta_i|_b = m_b + 0 \leq n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}.$$

By Theorem 4.5.5, $\mathscr{R}$ has a winning strategy.

Now suppose $h \geq 1$ and $l \geq 0$. As first player, $\mathscr{B}$ starts off by doing any combination of the following moves (as any one of these moves allows $\mathscr{B}$ to keep playing):

$$\begin{cases} \text{placing } t \text{ red chips on } t \text{ empty piles,} \\ \text{placing } t' \text{ red chips on } t' \text{ } b\text{-singletons,} \\ \text{placing } t'' \text{ red chips on } t'' \text{ } b\text{-piles } \beta_i, \\ \text{placing } s \text{ blue chips on } s \text{ } b\text{-singletons,} \\ \text{and placing } s' \text{ blue chips on } s' \text{ } b\text{-piles } \beta_i. \end{cases}$$

The SLS board is now composed of

$$\begin{cases} k_e - t + s + s' \text{ empty piles,} \\ k_r + t \text{ } r\text{-singletons,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ t' \text{ } (b, r)\text{-piles,} \\ l \text{ } r\text{-piles of length at least 2,} \\ t'' \text{ } r\text{-piles of length at least 3,} \\ \text{and } h - t'' - s' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

$=$

$$\begin{cases} k_e - t + s + s' \text{ empty piles,} \\ k_b - t' - s \text{ } b\text{-singletons,} \\ k_r + t \text{ } r\text{-singletons,} \\ l + t' + t'' \text{ } r\text{-piles of length at least 2,} \\ \text{and } h - t'' - s' \text{ } b\text{-piles of length at least 2.} \end{cases}$$

Without loss of generality, the remaining $h-t''-s'$ b-piles of length at least 2 are $\beta_1, \beta_2, \ldots, \beta_{h-t''-s'}$, and then t'' r-piles of length at least 3 are $\beta_{h-t''-s'+1}, \ldots, \beta_{h-s'}$. $\mathscr{B}$ can end their turn in 4 different ways.

1. $\mathscr{B}$ places a b chip on an empty pile (assuming $k_e - t + s + s' > 0$). The move goes back to $\mathscr{R}$, who captures all r-piles as per step 1 of strategy Σ. Now, the board is composed of $k_e + k_r + l + t' + t'' + s + s' - 1$ empty piles, $k_b - t' - s + 1$ b-singletons, and $h - t'' - s'$ b-piles of length at least 2. $\mathscr{R}$ will place an r chip on the board and the move goes back to $\mathscr{B}$. We have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1, m_r - t - t' - t'' + \sum_{i=h-s'+1}^{h} (|\beta_i|_r - 1)\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'} (|\beta_i|_r + 1) - 1\right). \end{cases}$$

   The analysis of the game depends on the number of b-piles of length at least 2.

   (a) If $h - t'' - s' > 1$, then $\mathscr{R}$ placed an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''-s'}$ be this pile. The board is now of generalized type II. Since

$$\begin{aligned} & m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1 + \sum_{i=1}^{h-t''-s'-1} |\beta_i|_b \\ < & \, m_b + \sum_{i=1}^{h} |\beta_i|_b - 1 \\ \leq & \, n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} - 1 (\text{by } 4.7.1) \\ \leq & \, n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'} (|\beta_i|_r + 1) - 1, \end{aligned}$$

   then $\mathscr{R}$ has a winning strategy by Theorem 4.6.5.

   (b) If $h - t'' - s' = 1$, then $\mathscr{R}$ places an r chip on the only b-pile of length at least 2. The board is now of type I. Since

$$\begin{aligned} & m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1 \\ \leq\ & m_b + \sum_{i=1}^{h} |\beta_i|_b - 1 \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\left\{|\rho_i|_r : i = 1, \ldots, l\right\} - 1 \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'} (|\beta_i|_r + 1) - 1, \end{aligned} \tag{4.7.2}$$

then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

(c) If $h - t'' - s' = 0$, then $\mathscr{R}$ places an r chip on a b-singleton. The board is now of type I. Since we have the same inequality 4.7.2, then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

2. $\mathscr{B}$ places a b chip on an r-singleton (assuming $k_r + t > 0$). The move automatically goes to $\mathscr{R}$, who captures all r-piles as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + l + t' + t'' + s + s' - 1$ empty piles, $k_b - t' - s$ b-singletons, and $h - t'' - s' + 1$ b piles of length at least 2. $\mathscr{R}$ will place an r chip on the board and the move goes back to $\mathscr{B}$. We have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1, m_r - t - t' - t'' + \sum_{i=h-s'+1}^{h} (|\beta_i|_r - 1)\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'} (|\beta_i|_r + 1) - 1\right). \end{cases}$$

The analysis of the game depends on the number of b-piles of length at least 2.

(a) If $h - t'' - s' + 1 > 1$, then $\mathscr{R}$ placed an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''}$ be this pile. The board is now of generalized type II. Using inequality 4.7.1, we have

$$
\begin{aligned}
& m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1 + \sum_{i=1}^{h-t''-s'-1} |\beta_i|_b + |(r,b)|_b \\
&= m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b + \sum_{i=1}^{h-t''-s'-1} |\beta_i|_b \\
&\leq m_b + \sum_{i=1}^{h} |\beta_i|_b - 1 \\
&\leq n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} - 1 \\
&\leq n_r + \sum_{i=1}^{l} |\rho_i|_r - 1 \\
&\leq n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'} (|\beta_i|_r + 1) - 1.
\end{aligned}
$$

Then $\mathscr{R}$ has a winning strategy by Theorem 4.6.5.

(b) If $h - t'' - s' + 1 = 1$, then $\mathscr{R}$ places an r chip on the only b-pile of length at least 2. The board is now of type I. Using inequality 4.7.1, we have

$$
\begin{aligned}
& m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1 \\
&\leq m_b + \sum_{i=1}^{h} |\beta_i|_b - 1 \\
&\leq n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} - 1 \\
&\leq n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'} (|\beta_i|_r + 1) - 1.
\end{aligned}
$$

Then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

(c) Observe that the case $h - t'' - s' + 1 = 0$ is impossible since by definition, $t'' + s' \leq h$.

3. $\mathscr{B}$ places a b chip on an r-pile of length at least 2. Without loss of generality, let $\beta_{h-s'}$ be this pile. The move goes bacck to $\mathscr{R}$ who captures all r-piles as per step 1 of Strategy Σ. Now, the board is composed of $k_e + k_r + l + t' + t'' + s + s' - 1$

empty piles, $k_b - t' - s$ b-singletons and $h - t'' - s' + 1$ b-piles of length at least 2. $\mathscr{R}$ will place an r chip on the board and the move goes back to $\mathscr{B}$. We have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1, m_r - t - t' - t'' + \sum_{i=h-s'+1}^{h} (|\beta_i|_r - 1)\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'-1} (|\beta_i|_r + 1) - 1\right). \end{cases}$$

The analysis of the game depends on the number of b-piles of length at least 2.

(a) If $h - t'' - s' + 1 > 1$, then $\mathscr{R}$ placed an r chip on the b-pile of greatest length. Without loss of generality, this pile is $\beta_{h-s'}$. The board is now of generalized type II. Using inequality 4.7.1, we have

$$\begin{aligned} & m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1 + \sum_{i=1}^{h-t''-s'} |\beta_i|_b \\ \leq\ & m_b + \sum_{i=1}^{h} |\beta_i|_b - 1 \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} - 1 \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'-1} (|\beta_i|_r + 1) - 1. \end{aligned}$$

Then $\mathscr{R}$ has a winning strategy by Theorem 4.6.5.

(b) If $h - t'' - s' + 1 = 1$, then $\mathscr{R}$ placed an r chip on the only b-pile of length at least 2. The board is now of type I. Using inequality 4.7.1, we have

$$\begin{aligned} & m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b - 1 \\ \leq\ & m_b + \sum_{i=1}^{h} |\beta_i|_b - 1 \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} - 1 \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'-1} (|\beta_i|_r + 1) - 1. \end{aligned}$$

Then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

(c) Observe that the case $h-t''-s'+1=0$ is not possible since by definition $t''+s' \leq h$.

4. $\mathscr{B}$ places an r chip on an r-pile (assuming $k_r+l+t+t'+t''>0$). This results in a capture for $\mathscr{R}$, who continues to capture all r-piles, as per step 1 of Strategy Σ. Now, the board is composed of $k_e+k_r+l+t'+t''+s+s'$ empty piles, $k_b-t'-s$ b-singletons, and $h-t''-s'$ b-piles of length at least 2. Then $\mathscr{R}$ places an r chip on the board and the move goes back to $\mathscr{B}$. We have

$$\begin{cases} \mathscr{B} = \left(m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b, m_r - t - t' - t'' + \sum_{i=h-s'+1}^{h} (|\beta_i|_r - 1) - 1\right), \\ \mathscr{R} = \left(n_b, n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'-1} (|\beta_i|_r + 1)\right). \end{cases}$$

The analysis of the game depends on the number of b-piles of length at least 2.

(a) If $h-t''-s'>1$, then $\mathscr{R}$ placed an r chip on the b-pile of greatest length. Without loss of generality, let $\beta_{h-t''-s'}$ be this pile. The board is now of generalized type II. Using inequality 4.7.1, we have

$$\begin{aligned} & m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b + \sum_{i=1}^{h-t''-s'-1} |\beta_i|_b \\ \leq\ & m_b + \sum_{i=1}^{h} |\beta_i|_b \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'-1} (|\beta_i|_r + 1). \end{aligned}$$

Then $\mathscr{R}$ has a winning strategy by Theorem 4.6.5.

(b) If $h-t''-s'=1$: then $\mathscr{R}$ placed an r chip on the only b-pile of length at

least 2. The board is now of type I. Using inequality 4.7.1, we have

$$\begin{aligned} & m_b + \sum_{i=h-s'+1}^{h} |\beta_i|_b \\ \leq\ & m_b + \sum_{i=1}^{h} |\beta_i|_b \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \\ \leq\ & n_r + \sum_{i=1}^{l} |\rho_i|_r + t' + \sum_{i=h-t''-s'+1}^{h-s'-1} (|\beta_i|_r + 1). \end{aligned} \tag{4.7.3}$$

Then $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

(c) If $h-t''-s'=0$, then $\mathscr{R}$ placed an r chip on a b-singleton (if $k_b-t'-s>0$) or on an empty pile (if $k_b-t'-s=0$). The board is now of type I. Using inequality 4.7.1, we derive the same inequality 4.7.3 as above, and thus $\mathscr{R}$ has a winning strategy by Theorem 4.3.4.

It follows that for every possible move $\mathscr{B}$ does, $\mathscr{R}$ has a winning strategy, which is Strategy Σ. □

Theorem 4.7.5. *Consider an SLS board composed of k piles such that k_e are empty, k_r are r-singletons, k_b are b-singletons, h are b-piles $\beta_1, \beta_2, \ldots, \beta_h$ of length at least 2; and l are r-piles $\rho_1, \rho_2, \ldots, \rho_l$ of length at least 2. Let $\mathscr{B} = (m_b, m_r)$ be the first player and $\mathscr{R} = (n_b, n_r)$ be the second player. We have the following:*

$\mathscr{B}$ has a winning strategy

$$\iff m_b > 0 \text{ and } \left(m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\} \text{ or } n_r = 0\right).$$

Proof. Suppose $\mathscr{B} = (m_b, m_r)$ is the first player, and $\mathscr{R} = (n_b, n_r)$ is the second player. We separate this proof into 3 cases.

1. If $m_b > 0$ and $n_r = 0$, then $\mathscr{B}$ has a winning strategy by Proposition 4.7.1.
2. If $m_b, n_r > 0$ and $m_b + \sum_{i=1}^{h} |\beta_i|_b > n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$, then $\mathscr{B}$ has a winning strategy by Proposition 4.7.2.
3. If $m_b = 0$, then $\mathscr{R}$ has a winning strategy by Proposition 4.7.3.

4. If $m_b, n_r > 0$ and $m_b + \sum_{i=1}^{h} |\beta_i|_b \leq n_r + \sum_{i=1}^{l} |\rho_i|_r - \max\{|\rho_i|_r : i = 1, \ldots, l\}$, then $\mathscr{R}$ has a winning strategy by Proposition 4.7.4.

□

Lemma 4.7.6. *Strategy Σ is a winning strategy for players $\mathscr{R}$ and $\mathscr{B}$.*

Proof. As we have seen in this chapter so far, given any setting of a 2-player 2-color game, there is exactly one player at a time that has Strategy Σ as their winning strategy. It follows that Strategy Σ is a winning strategy for each player. □

Chapter 5

Conclusion

As seen throughout this thesis, *So Long Sucker* can be used in behavioural research as well as theoretical game theory. The game has a lot of potential in helping to understand social dynamics between players. For instance, it is useful in studying people's personality traits such as their eagerness to lead, their bargaining skills, and their principles.

Game theory has long had applications in economics. Thus, if the game was to be approached from an economical perspective, it would be useful to use players' behaviour in order to study their risk aversion. It is of interest to understand players' attempt to lower uncertainty and their perception of cost and benefit when accepting or rejecting a deal [Sam16].

The 2-players and 2-colors scenario seen throughout this thesis may be used to simulate a conflict between 2 parties. It was established that first player Blue has a winning strategy if they had more chips than their opponent, or the possibility of gaining more chips than their opponents. As the initiator of the conflict, Blue needs to be the stronger player. Whereas Red, as the responder to the conflict, needs to be only as strong as their adversary. Red only needs to be prepared to retaliate to Blue's moves in order to stay in the game and exhaust Blue's efforts in order to win. As we saw, Red needs just as many chips as Blue, or to be able to capture just as many chips.

The 2-players and 2-colors case has been fully studied. Given any board with any 2 players, where only their respective colors are being used, an observer can immediately determine which player has a winning strategy. The next step in the research would be to look at 2-players, 3-colors situation and characterize Blue's winning strategies then. How does adding an eliminated player's chips in the game affect Blue's strategies? We hypothesize that these cases can be studied in a similar matter to the 2-colors cases. The ultimate goal would be to establish which player has a winning strategy when there are only 2 players left in the game, but with any possible colors left out of the 4 colors. From then on, one would question studying

the 3-player cases. However, given the versatility of the rules, studying the 3-player cases might require different tools than the ones seen in this thesis.

Another point of interest is the Nash Equilibria of a given SLS setting. In the 2-players situation, the game pay-off comes down to win-lose. Thus, the losing player has no incentive to play any particular way as they will lose regardless of how they play, and their pay-off does not change. One variation that may be added to the game is giving value to the chips. This way, in the Nash Equilibrium case, the losing player will aim to lose as little chips as possible in order to minimize their loss.

Appendix A

SLS: The Code

A.1 class Table


```
1  import java.io.*;
2  import java.util.LinkedList;
3
4  public class Table {
5
6    LinkedList<String>[] rows;
7
8    // constructor
9    public Table (int n){
10     rows = new LinkedList[n];
11     for (int i = 0; i<n ; i++){
12       rows[i] = new LinkedList<String>();}}
13
14   // display the table
15   public void display(){
16     for (int i = 0; i<8; i++){
17       System.out.print("Row #"+i+": ");
```

```
18          for (int j = 0; j<this.rows[i].size();j++){
19            System.out.print(this.rows[i].get(j));}
20        System.out.println();}}
21
22    // replace this Table with other
23    public static Table replace(Table other){
24      Table rep = new Table(8);
25      for (int i = 0 ; i < 8 ; i++){
26        for (int j = 0 ; j < other.rows[i].size() ; j++){
27          rep.rows[i].addLast(other.rows[i].get(j));}}
28      return rep;}
29
30    // sort this table by length
31    public void sort(int[] t) {
32
33      for (int i = 0; i < 7; i++) {
34
35        int max_idx = i;
36        for (int j = i+1; j < 8; j++){
37          if ( this.rows[j].size() > this.rows[max_idx].size() ){
38            max_idx = j;}}
39
40        if (i != max_idx){
41          t[i] = max_idx;
42          t[max_idx] = i;
43          LinkedList<String> temp = new LinkedList<String>();
44          for (int k = 0 ; k<this.rows[i].size() ; k++){
45            temp.addLast(this.rows[i].get(k));}
46          this.rows[i].clear();
47          for (int k = 0 ; k<this.rows[max_idx].size() ; k++){
48            this.rows[i].addLast(this.rows[max_idx].get(k));}
49          this.rows[max_idx].clear();
50          for (int k = 0 ; k<temp.size() ; k++){
51            this.rows[max_idx].addLast(temp.get(k));}}}}
52  }
```

A.2 class Player

```
1  import java.io.InputStream;
2  import java.util.Scanner;
3  import java.util.LinkedList;
4
5  public class Player {
6
7
8    private int score, blue, red, green, yellow;
9    private String color, before;
10
11   // constructor
12   public Player( String color, int b, int r, int g, int y, int score, String before){
13     this.color = color;
14     this.blue = b;
15     this.red = r;
16     this.green = g;
17     this.yellow = y;
18     this.score = score;
19     this.before = before;}
20
21   // getters
22   public String getPlayer(){
23     return this.color;}
24
25   public int getChip(int chip_color){
26     if (chip_color == 0){ return this.blue;}
27     else if (chip_color == 1) {return this.red;}
28     else if (chip_color == 2) {return this.green;}
29     else if (chip_color == 3) {return this.yellow;}
30     return -1;}
31
32   public int getTotal(){
33     return this.blue + this.red + this.green + this.yellow;}
```

```
34
35   public int getScore(){
36     return this.score;}
37
38   public String getBefore(){
39     return this.before;}
40
41   public int getPrisoners(){
42     if (this.getPlayer().equals("b")){
43       return this.red + this.green + this.yellow;}
44     else if (this.getPlayer().equals("r")){
45       return this.blue + this.green + this.yellow;}
46     else if (this.getPlayer().equals("g")){
47       return this.blue + this.red + this.yellow;}
48     else if (this.getPlayer().equals("y")){
49       return this.blue + this.red + this.green;}
50     return -1;}
51
52   public String getDiscard(){
53     String discard = "";
54     if (this.getPrisoners() != 0){
55       for (int i = 0; i<4; i++){
56         if (this.convertInt(this.color) != i){
57           for (int j = 0; j< this.getChip(i); j++){
58             discard = discard.concat(this.convertStr(i));}}}
59     }
60     return discard;}
61
62   //setters
63   public void setColor(String c){
64     this.color = c;}
65
66   public void setChip (int chip_color, int chip_num){
67     if (chip_color == 0){this.blue = this.blue + chip_num;}
68     else if (chip_color == 1){this.red = this.red + chip_num;}
69     else if (chip_color == 2){this.green = this.green + chip_num;}
```

```
70      else if (chip_color == 3){this.yellow = this.yellow + chip_num;}}
71
72    public void setScore(int i){
73      this.score = i;}
74
75    public void setBefore(String b){
76      this.before = b;}
77
78    // convert color to number
79    public int convertInt(String s){
80      if (s.equals("b")){return 0;}
81      else if (s.equals("r")){return 1;}
82      else if (s.equals("g")){return 2;}
83      else {return 3;}}
84
85    // convert number to color
86    public String convertStr(int i){
87      if (i == 0) {return "b";}
88      else if (i == 1) {return "r";}
89      else if (i == 2) {return "g";}
90      else {return "y";}}
91
92    // replace this Player with other
93    public static Player replace(Player other){
94      Player rep = new Player("", 0, 0, 0, 0, 0, "");
95      rep.color = other.color;
96      rep.blue = other.blue;
97      rep.red = other.red;
98      rep.green = other.green;
99      rep.yellow = other.yellow;
100     rep.score = other.score;
101     rep.before = other.before;
102     return rep;}
103
104   // this Player captures row i
105   public void capture(String captured_row){
```

```
106     int c = 0;
107     for (int i = 0; i < captured_row.length(); ++i){
108       c = this.convertInt(Character.toString(captured_row.charAt(i)));
109       this.setChip(c,1);
110     }
111   }
112
113   // discards chip from this Player pile
114   public void discard(String discarded_chip, int num){
115     this.setChip(this.convertInt(discarded_chip) , -num);}
116
117   // returns true if this Player is eliminated
118   public boolean eliminated () {
119     return (this.getScore() != 0);}
120
121   // donate chip_number amount of color chips from Player this to other
122   // eligibility for donation verified in main code
123   public void donate(Player other, String color, int chip_number){
124     other.setChip(this.convertInt(color) , chip_number);
125     this.setChip(this.convertInt(color) , -chip_number);}
126
127   // returns string of next possible players
128   public String nextPlayer(int captureturn, String played_row, LinkedList<String> remaining_players,
129                            Player[] players, String chipPlayed, String currentPlayer){
130
131     String possibleNP = "";
132     String refrain = "";
133
134     if (captureturn == 0){
135
136       // creating played_row_sub
137       String played_row_sub="";
138       for (int m = 0 ; m<played_row.length() ; m++){
139         if (remaining_players.contains(Character.toString(played_row.charAt(m)))){
140           played_row_sub = played_row_sub.concat(Character.toString(played_row.charAt(m)));}}
141
```

```
142       // if not all players are in played_row
143       for(int i = 0; i < remaining_players.size() ; i++) {
144         if (played_row_sub.contains(remaining_players.get(i))==false){
145           possibleNP = possibleNP.concat(remaining_players.get(i));}}
146
147       // if all players are represented in played_row
148       if (possibleNP.isEmpty()==true){
149         for (int j = played_row_sub.length() - 1 ; j>=0 ; j--){
150           if (refrain.indexOf(played_row_sub.charAt(j)) == -1){
151             refrain = refrain.concat(Character.toString(played_row_sub.charAt(j)));}}
152         possibleNP = possibleNP.concat(Character.toString(refrain.charAt(refrain.length()-1)));}
153
154     } // end- if (capturereturn == 0)
155
156     else if (capturereturn ==1 &&
157              players[this.convertInt(chipPlayed)].eliminated()==false){
158       possibleNP = possibleNP.concat(chipPlayed);}
159
160     else if (capturereturn == 1 &&
161              players[this.convertInt(chipPlayed)].eliminated()==true){
162       possibleNP = possibleNP.concat(currentPlayer);}
163
164     return possibleNP;}
165
166   // display all information about this player
167   public void display(){
168     System.out.println("Player "+this.color+":("+this.blue+" Blue," +this.red+" Red,"
169                        +this.green+" Green," +this.yellow+" Yellow). Score: "+this.score);}
170
171 }
```

A.3 class Move

```
1  public class Move {
2
3    private int row;                  // row being played
4    private int playerCurrent;        // player making the move
5    private String chipPlayed;        // chip being played
6    private int playerNext;           // player who gets next move
7    private int capture;              // 0 = capture; 1 = no capture
8    private boolean elimination;      // 0 = player eliminated
9    private String discardChips;      // in-hand chips to discard
10   private String discardCap;        // captured chips to discard
11   private Table tableStart;
12   private Table tableEnd;
13   private Player[] playersStart;
14   private Player[] playersEnd;
15
16
17   // constructor
18   public Move(int playerCurrent, int row, String chipPlayed,
19               int capture, boolean elimination, int playerNext,
20               String discardChips, String discardCap,
21               Table tableStart, Table tableEnd,
22               Player[] playersStart, Player[] playersEnd){
23
24     this.playerCurrent = playerCurrent;
25     this.row = row;
26     this.chipPlayed = chipPlayed;
27     this.capture = capture;
28     this.elimination = elimination;
29     this.playerNext = playerNext;
30     this.discardChips = discardChips;
31     this.discardCap = discardCap;
32     this.tableStart = tableStart;
33     this.tableEnd = tableEnd;
```

```
34      this.playersStart = playersStart;
35      this.playersEnd = playersEnd;}
36
37    // getters
38    public int getPlayer(){
39      return this.playerCurrent;}
40
41    public int getRow(){
42      return this.row;}
43
44    public String getChipPlayed(){
45      return this.chipPlayed;}
46
47    public int getCapture(){
48      return this.capture;}
49
50    public boolean getElimination(){
51      return this.elimination;}
52
53    public int getPlayerNext(){
54      return this.playerNext;}
55
56    public String getDiscardingChips(){
57      return this.discardChips;}
58
59    public String getDiscardingCap(){
60      return this.discardCap;}
61
62    public Table getTableStart(){
63      return this.tableStart;}
64
65    public Table getTableEnd(){
66      return this.tableEnd;}
67
68    public Player getPlayersStart(int i){
69      return this.playersStart[i];}
```

```
70
71    public Player getPlayersEnd(int i){
72      return this.playersEnd[i];}
73
74    // returns highest score at beginning of move
75    public int getMaxScore(){
76      int max = 0;
77      for (int i = 0; i<4; i++){
78        if ( this.getPlayersEnd(i).getScore() > max){
79          max = this.getPlayersEnd(i).getScore();}
80      }
81      return max;}
82
83    // returns string of remaining players at beginning of move
84    public String getRemPlayers(){
85      String remPlayers = "";
86      for (int i = 0; i<4; i++){
87        if ( this.getPlayersStart(i).getScore() == 0){
88          remPlayers = remPlayers + this.getPlayersEnd(i).getPlayer();}}
89      return remPlayers;}
90
91    // setters
92    public void setPlayer(int p){
93      this.playerCurrent = p;}
94
95    public void setRow(int r){
96      this.row = r;}
97
98    public void setChipPlayed(String c){
99      this.chipPlayed = c;}
100
101   public void setCapture(int c){
102     this.capture = c;}
103
104   public void setElimination(boolean e){
105     this.elimination = e;}
```

```
106
107   public void setPlayerNext(int p){
108     this.playerNext = p;}
109
110   public void setDiscardingChips(String d){
111     this.discardChips = d;}
112
113   public void setDiscardingCap (String d){
114     this.discardCap = d;}
115
116   public void setTableStart(Table t1){
117     this.tableStart = t1;}
118
119   public void setTableEnd(Table t2){
120     this.tableEnd = t2;}
121
122   public void setPlayersStart(int i, Player p1){
123     this.playersStart[i] = p1;}
124
125   public void setPlayersEnd(int i, Player p2){
126     this.playersEnd[i] = p2;}
127
128   public void replace(Move other){
129     this.playerCurrent = other.playerCurrent;
130     this.row = other.row;
131     this.chipPlayed = other.chipPlayed;
132     this.capture = other.capture;
133     this.elimination = other.elimination;
134     this.playerNext = other.playerNext;
135     this.discardChips = other.discardChips;
136     this.discardCap = other.discardCap;
137     this.tableStart = Table.replace(other.tableStart);
138     this.tableEnd = Table.replace(other.tableEnd);
139     for (int i = 0 ; i < 4 ; i++){
140       this.playersStart[i] = Player.replace(other.playersStart[i]);
141       this.playersEnd[i] = Player.replace(other.playersEnd[i]);}}
```

```
142
143   public static int[] discardCount(String d){
144     int[] discard_cap_count = new int[4];
145     for (int i = 0; i < d.length(); i++){
146       if (d.charAt(i) == 'b'){
147         discard_cap_count[0] = discard_cap_count[0] +1;}
148       else if (d.charAt(i) == 'r'){
149         discard_cap_count[1] = discard_cap_count[1] +1;}
150       else if (d.charAt(i) == 'g'){
151         discard_cap_count[2] = discard_cap_count[2] +1;}
152       else if (d.charAt(i)== 'y'){
153         discard_cap_count[3] = discard_cap_count[3] +1;}}
154     return discard_cap_count;}
155
156   // display information about this move
157   public void display(){
158     System.out.println("Current Player: " +this.playerCurrent);
159     System.out.println("Row #" +this.row);
160     System.out.println("Chip placed: " +this.chipPlayed);
161     System.out.println("There is a capture: " +this.capture);
162     System.out.println("Next Player: " +this.playerNext);
163     System.out.println("General chips to discard " +this.discardChips);
164     System.out.println("Capture chips to discard " +this.discardCap);
165     System.out.println("Table Start");
166     this.tableStart.display();
167     System.out.println("Table End");
168     this.tableEnd.display();
169     System.out.println("Players Start");
170     for (int i = 0 ; i < 4 ; i++){
171       this.playersStart[i].display();}
172     System.out.println("Players End");
173     for (int i = 0 ; i < 4 ; i++){
174       this.playersEnd[i].display();}}
175
176 }
```

A.4 class minimax

```
1  import java.io.*;
2  import java.io.InputStream;
3  import java.util.Scanner;
4  import java.util.LinkedList;
5  import java.util.*;
6  import java.util.Arrays;
7  import java.util.Collections;
8
9  public class minimax {
10
11   // return LinkedList of all possible next moves
12   public static LinkedList<Move> getAllMoves(Table table, Player[] players, int currentPlayerInt){
13
14     LinkedList<String> remaining_players = new LinkedList<String>();
15     int score = 0;
16     LinkedList<String> played_sub = new LinkedList<String>();
17
18     if (players[0].getScore()==0){
19       remaining_players.addLast("b");}
20     else if (players[0].getScore() > score) {
21       score = players[0].getScore();}
22     if (players[1].getScore()==0){
23       remaining_players.addLast("r");}
24     else if (players[1].getScore() > score){
25       score = players[1].getScore();}
26     if (players[2].getScore()==0){
27       remaining_players.addLast("g");}
28     else if (players[2].getScore() > score){
29       score = players[2].getScore();}
30     if (players[3].getScore()==0){
31       remaining_players.addLast("y");}
32     else if (players[3].getScore() > score){
33       score = players[3].getScore();}
```

```
34
35
36
37      Table minimal_table = Table.replace(table);
38      int[] pointer = new int[]{0, 1, 2, 3, 4, 5, 6, 7};
39      minimal_table.sort(pointer);
40
41      int min_idx = 0;
42      while (minimal_table.rows[min_idx].isEmpty() == false && min_idx<7){
43        min_idx = min_idx+1;}
44
45      Player temp = new Player("", 0, 0, 0, 0, 0, "");
46      String currentPlayer = temp.convertStr(currentPlayerInt);
47      Player[] playersStart = new Player[] {temp,temp,temp,temp};
48      Player[] playersEnd = new Player[] {temp,temp,temp,temp};
49
50      for (int u = 0; u<4; u++){
51        playersStart[u] = Player.replace(players[u]);
52        playersEnd[u] = Player.replace(players[u]);}
53
54      // list to store all Submoves
55      LinkedList<Move> allSubMoves = new LinkedList<Move>();
56      int temp1 = currentPlayerInt;
57      int temp2 = 0;
58      Table tableStart = Table.replace(minimal_table);
59      Table tableEnd = Table.replace(minimal_table);
60
61      Move move = new Move(temp1, 0, "", 0, false, temp2, "", "", tableStart, tableEnd,
62                           playersStart, playersEnd);
63
64      if (playersEnd[temp1].getTotal() > 0){
65        allSubMoves.addLast(move);}
66
67      else if (playersEnd[temp1].getTotal() == 0){
68        move.setElimination(true);
69        LinkedList<String> remaining_players_temp =
```

```
70          new LinkedList<String>();
71        for (int l = 0 ; l<remaining_players.size() ; l++){
72          remaining_players_temp.addLast(remaining_players.get(l));}
73        remaining_players_temp.remove(currentPlayer);
74        LinkedList<String> played_sub_temp = new LinkedList<String>();
75        for (int l = 0 ; l<played_sub.size() ; l++){
76          if (remaining_players_temp.contains(played_sub.get(l))){
77            played_sub_temp.addLast(played_sub.get(l));}}
78        move.setPlayerNext( temp.convertInt(temp.nextPlayer(0, "", remaining_players_temp, playersEnd,
79                                                            "", currentPlayer)) );
80
81        int s = Math.abs(remaining_players_temp.size()-4);
82        move.getPlayersEnd(temp1).setScore(s);
83        players[temp1].setScore(s);
84        allSubMoves.addLast(move);
85
86        ////////////////////add moves with donations ////////////////
87
88        if (remaining_players_temp.size() > 1){
89          for (int m = 0; m<remaining_players_temp.size(); m++){
90            int temp3 = temp.convertInt( remaining_players_temp.get(m) );
91            if (move.getPlayersEnd(temp3).getPrisoners() > 0){
92              for (int n = 0; n<4; n++){
93                if (temp3 !=  n && move.getPlayersEnd(temp3).getChip(n)>0){
94                  playersEnd = new Player[] {temp,temp,temp,temp};
95
96                  for (int u = 0; u<4; u++){
97                    playersEnd[u] = Player.replace(players[u]);}
98                  Move move_sub = new Move(temp1, 0, "", 0, false, temp2, "", "", tableStart, tableEnd,
99                                           playersStart, playersEnd);
100                 move_sub.getPlayersEnd(temp3).donate(move_sub.getPlayersEnd(temp1),
101                                                      temp.convertStr(n),1);
102                 allSubMoves.addLast(move_sub);}
103             }
104           }
105         }
```

```
106         }
107       }
108
109       // list to store all moves
110       LinkedList<Move> allMoves = new LinkedList<Move>();
111
112       for (int i = 0; i<min_idx+1 ; i++){
113
114         for (int z = 0; z < allSubMoves.size(); z++){
115
116           if (allSubMoves.get(z).getPlayersEnd(temp1).getTotal() == 0 && i==0){
117             allMoves.addLast(allSubMoves.get(z));}
118
119           else if (allSubMoves.get(z).getPlayersEnd(temp1).getTotal() > 0){
120
121             for (int j = 0; j<4 ; j++){
122
123               if ( allSubMoves.get(z).getPlayersEnd(temp1).getChip(j) > 0 ){
124                 tableStart = new Table(8);
125                 tableEnd = new Table(8);
126                 playersEnd = new Player[] {temp,temp,temp,temp};
127
128                 for (int u = 0; u<4; u++){
129                   playersEnd[u] = Player.replace(players[u]);}
130                 move = new Move(temp1, i, "", 0, false, temp2, "", "",
131                                 tableStart, tableEnd, playersStart, playersEnd);
132                 move.replace(allSubMoves.get(z));
133                 move.setChipPlayed(temp.convertStr(j));
134                 move.getPlayersEnd(temp.convertInt(move.getPlayersEnd(temp1).
135                                                    getPlayer())).setChip(j,-1);
136                 move.getTableEnd().rows[i].addLast(temp.convertStr(j));
137                 String played_row = "";
138                 for (int k = 0 ; k < move.getTableEnd().rows[i].size() ; k++){
139                   played_row += move.getTableEnd().rows[i].get(k);}
140                 if ( move.getTableEnd().rows[i].size() > 1
141                       && move.getTableEnd().rows[i].
```

```
142                 get(move.getTableEnd().rows[i].size()-2).
143                 equals(move.getTableEnd().rows[i].get(move.getTableEnd().rows[i].size()-1))
144                         ==true){
145             move.setCapture(1);
146               // reinitiates the row to an empty row
147             while(move.getTableEnd().rows[i].isEmpty() == false){
148               move.getTableEnd().rows[i].removeLast() ;}
149             String possibleNP = temp.nextPlayer( move.getCapture(), played_row,
150                                                  remaining_players, players, move.getChipPlayed(),
151                                                  move.getPlayersEnd(temp1).getPlayer());
152             move.setPlayerNext(temp.convertInt(possibleNP) ); // assumption: only 1 possibleNP
153             String discard_cap = "";
154             discard_cap = played_row;
155             move.getPlayersEnd(j).capture(played_row);
156             int[] discard_cap_count = new int[4];
157             discard_cap_count = move.discardCount(discard_cap);
158
159             for (int a = 0; a<=discard_cap_count[0]; a++){
160               for (int b = 0; b<=discard_cap_count[1]; b++){
161                 for (int c = 0; c<=discard_cap_count[2]; c++){
162                   for (int d = 0; d<=discard_cap_count[3]; d++){
163                     String discard_cap_sub = "";
164                     discard_cap_sub = "bbbbbbbb".substring(0, a) + "rrrrrrrr".substring(0, b)
165                       + "gggggggg".substring(0, c) + "yyyyyyyy".substring(0, d);
166
167                     if (remaining_players.size() == 2 && discard_cap_sub.length() == 1){ //
168                       playersEnd = new Player[] {temp,temp,temp,temp};
169
170                       for (int u = 0; u<4; u++){
171                         playersEnd[u] = Player.replace(players[u]);}
172                       Move move_sub = new Move(temp1, i, "", 0, false, temp2, "", "",
173                                                tableStart, tableEnd, playersStart, playersEnd);
174                       move_sub.replace(move);
175                       move_sub.setDiscardingCap(discard_cap_sub);
176                       move_sub.getPlayersEnd(j).discard("b",a);
177                       move_sub.getPlayersEnd(j).discard("r",b);
```

```
178                         move_sub.getPlayersEnd(j).discard("g",c);
179                         move_sub.getPlayersEnd(j).discard("y",d);
180                         allMoves.addLast(move_sub);
181                       }
182                       else if(remaining_players.size() > 2 && discard_cap_sub.length() > 0){
183                         playersEnd = new Player[] {temp,temp,temp,temp};
184
185                         for (int u = 0; u<4; u++){
186                           playersEnd[u] = Player.replace(players[u]);}
187                         Move move_sub = new Move(temp1, i, "", 0, false, temp2, "", "",
188                                                  tableStart, tableEnd, playersStart, playersEnd);
189                         move_sub.replace(move);
190                         move_sub.setDiscardingCap(discard_cap_sub);
191                         move_sub.getPlayersEnd(j).discard("b",a);
192                         move_sub.getPlayersEnd(j).discard("r",b);
193                         move_sub.getPlayersEnd(j).discard("g",c);
194                         move_sub.getPlayersEnd(j).discard("y",d);
195                         allMoves.addLast(move_sub);
196                       }
197                     }}}}
198           } // if (capture = 1)
199           else{
200             String possibleNP = temp.nextPlayer( move.getCapture(), played_row,
201                                                  remaining_players, players,
202                                                  move.getChipPlayed(),
203                                                  move.getPlayersEnd(temp1).getPlayer());
204             for (int x = 0; x<possibleNP.length(); x++){
205               Move move_sub = new Move(temp1, i, "", 0, false, temp2, "", "", tableStart,
206                                        tableEnd, playersStart, playersEnd);
207               move_sub.replace(move);
208               move_sub.setPlayerNext(temp.convertInt(Character.toString(possibleNP.charAt(x))));
209               allMoves.addLast(move_sub);
210             }
211           }
212         }
213       }
```

```
214           } // end-if (move.getPlayer().getTotal() > 0)
215         }
216       }
217
218
219       LinkedList<Move> allMoves2 = new LinkedList<Move>();
220
221       // add discarding chips
222       for (int y = 0; y < allMoves.size(); y++){
223         // create list to store all moves
224         tableStart = Table.replace(minimal_table);
225         tableEnd = new Table(8);
226         playersEnd = new Player[] {temp,temp,temp,temp};
227         for (int u = 0; u<4; u++){
228           playersEnd[u] = Player.replace(players[u]);}
229         move = new Move(temp1, 0, "", 0, false, temp2, "", "", tableStart, tableEnd, playersStart,
230                         playersEnd);
231         move.replace(allMoves.get(y));
232
233         if(move.getPlayersEnd(temp.convertInt(move.getPlayersEnd(temp1).getPlayer())).getPrisoners()>0
234               && remaining_players.size()>2){
235           String discard =
236             move.getPlayersEnd(temp.convertInt(move.getPlayersEnd(temp1).getPlayer())).getDiscard();
237           int[] discard_count = move.discardCount(discard);
238           discard_count = new int[4];
239           for (int e = 0; e < discard.length(); e++){
240             if (discard.charAt(e) == 'b'){
241               discard_count[0] = discard_count[0] +1;}
242             else if (discard.charAt(e) == 'r'){
243               discard_count[1] = discard_count[1] +1;}
244             else if (discard.charAt(e) == 'g'){
245               discard_count[2] = discard_count[2] +1;}
246             else if (discard.charAt(e)== 'y'){
247               discard_count[3] = discard_count[3] +1;}
248           }
249           int limit_discard = 1;
```

```
250         for (int a = 0; a<=Math.min(discard_count[0],limit_discard); a++){
251           for (int b = 0; b<=Math.min(discard_count[1],limit_discard); b++){
252             for (int c = 0; c<=Math.min(discard_count[2],limit_discard); c++){
253               for (int d = 0; d<=Math.min(discard_count[3],limit_discard); d++){
254                 String discard_sub = "";
255                 tableStart = Table.replace(minimal_table);
256                 tableEnd = new Table(8);
257                 playersEnd = new Player[] {temp,temp,temp,temp};
258                 for (int u = 0; u<4; u++){
259                   playersEnd[u] = Player.replace(players[u]);}
260                 Move move_sub = new Move(temp1, 0, "", 0, false, temp2, "", "", tableStart,
261                                          tableEnd, playersStart, playersEnd);
262                 move_sub.replace(move);
263                 discard_sub = "bbbbbbbb".substring(0, a) + "rrrrrrrr".substring(0, b)
264                   + "gggggggg".substring(0, c) + "yyyyyyyy".substring(0, d);
265                 move_sub.setDiscardingChips(discard_sub);
266                 move_sub.getPlayersEnd(temp.convertInt(move.getPlayersEnd(temp1).getPlayer()))
267                   .discard("b",a);
268                 move_sub.getPlayersEnd(temp.convertInt(move.getPlayersEnd(temp1).getPlayer()))
269                   .discard("r",b);
270                 move_sub.getPlayersEnd(temp.convertInt(move.getPlayersEnd(temp1).getPlayer()))
271                   .discard("g",c);
272                 move_sub.getPlayersEnd(temp.convertInt(move.getPlayersEnd(temp1).getPlayer()))
273                   .discard("y",d);
274                 allMoves2.addLast(move_sub);}}}}
275       }
276       else {
277         allMoves2.addLast(move);}
278     }
279       return allMoves2;
280   }
281
282   public static int minimax(Move m){
283
284     if (m.getMaxScore() == 3){
285       if (m.getPlayerNext() == 1){
```

```
286           return -1;}
287         else {
288           return +1;}
289       }
290
291       else {
292
293         LinkedList<Move> allMoves = new LinkedList<Move>();
294         Player[] players = new Player[] {m.getPlayersEnd(0), m.getPlayersEnd(1), m.getPlayersEnd(2),
295           m.getPlayersEnd(3)};
296         allMoves = (LinkedList) minimax.getAllMoves(m.getTableEnd(), players,
297                                                     m.getPlayerNext()).clone();
298         Integer[] s = new Integer[allMoves.size()];
299         for (int i = 0; i<allMoves.size(); i++){
300           s[i] = minimax(allMoves.get(i));
301         }
302         if (m.getPlayerNext() == 1){
303           return Collections.min(Arrays.asList(s));}
304         else{
305           return Collections.max(Arrays.asList(s));}
306       }
307     }
308 }
```

A.5 class Game

```
1  // APPENDIX E
2
3  import java.io.InputStream;
4  import java.util.Scanner;
5  import java.util.LinkedList;
6  import java.util.*;
7
8  class Game {
9
10   // Generic method to get subarray of a non-primitive array
11   // between specified indices
12   public static<T> T[] subArray(T[] array, int beg, int end) {
13     return Arrays.copyOfRange(array, beg, end + 1);}
14
15
16   public static void main(String[] args){
17
18     int score = 0;            // score of the game (first player out gets 1, last player out gets 4)
19     int turn = 0;             // turn of the game being played
20     int captureturn = 0;      // if no capture == 0; else == 1
21     String playAgain = "0";
22     String currentPlayer = "";
23     String played_row = "";
24     String chipPlayed = "";
25     String checkElim = "";
26     Scanner scanner;
27
28     // played[i]: player who played turn i
29     LinkedList<String> played = new LinkedList<String>();
30
31     // played: same as player[i] minus eliminated_players
32     LinkedList<String> played_sub = new LinkedList<String>();
33
```

```
34      // possibleNextPlayers[i]: list of eligible players for turn i+1
35      // assuming <= 100 turns are played
36      LinkedList<String>[] possibleNextPlayers;
37      possibleNextPlayers = new LinkedList[100];
38      for (int i = 0; i<100; i++){
39        possibleNextPlayers[i] = new LinkedList<String>();}
40
41      // trackTables[i]: state of table at beginning of turn i
42      LinkedList<Table> trackTables = new LinkedList<Table>();
43
44      // trackPlayers[i]: state of players at beginning of turn i
45      // after eliminating, placing, capturing, discarding, donating
46      LinkedList<Player>[] trackPlayers;
47      trackPlayers = new LinkedList[100];
48      for (int i = 0; i<100 ; i++){
49        trackPlayers[i] = new LinkedList<Player>();}
50
51      // initializing the 4 players
52      // b == 0; r == 1; g == 2; y == 3
53      Player blue = new Player("b", 0, 0, 0, 0, 0, "r");
54      Player red = new Player("r", 1, 1, 0, 0, 0, "y");
55      Player green = new Player("g", 0, 1, 1, 0, 0,"b");
56      Player yellow = new Player("y", 0, 0, 0, 1, 0,"g");
57      Player[] players = new Player[] {blue,red,green,yellow};
58
59      // players not yet eliminated
60      LinkedList<String> remaining_players = new LinkedList<String>();
61      remaining_players.addLast("b");
62      remaining_players.addLast("r");
63      remaining_players.addLast("g");
64      remaining_players.addLast("y");
65
66      // initializing and displaying table
67      Table table = new Table(8);
68      table.display();
69
```

```
70      // displaying player information
71      for (int i = 0 ; i<4 ; i++){
72        System.out.print(players[i].getPlayer()+": ");
73        players[i].display();}
74
75      // inserting chips into the initial table
76      String input = "yes";
77      int j = 0;
78
79      ///////////////// START: SETTING THE TABLE ////////////////
80      while ( input.equals("yes") && j<8 ){
81
82        scanner = new Scanner(System.in);
83        System.out.print("Do you want to insert chips in row " + j + "? Answer yes or no.");
84        input = scanner.nextLine();
85
86        if (input.equals("yes")){
87          boolean valid = false;
88          String inputRow = new String();
89
90          // checks if string is valid
91          while (valid == false){
92            scanner = new Scanner(System.in);
93            System.out.print("How would you like the row to look?");
94            inputRow = scanner.nextLine();
95            valid = true; // assumes input is true and then checks for error
96            if (inputRow.contains("bb") || inputRow.contains("rr") || inputRow.contains("gg") ||
97             inputRow.contains("yy")){
98              valid = false;
99              System.out.print("The input is not a valid row. Please try again.");}
100           else{
101             for (int i = 0; i< inputRow.length(); i++){
102               if (inputRow.charAt(i) != 'b' && inputRow.charAt(i) != 'r' &&
103                inputRow.charAt(i) != 'g' && inputRow.charAt(i) != 'y'){
104                 valid = false;
105                 System.out.print("The input is not a valid row. Please try again.");}}}
```

```
106           }
107
108           for (int i = 0; i < inputRow.length(); i++){
109             table.rows[j].addLast(Character.toString(inputRow.charAt(i)));}
110         } // end if-statement
111
112         j++;} // end while loop
113 ///////////////// END: SETTING THE TABLE ////////////////
114
115       while (playAgain.equals("0") && score<4) { // while-loop to play multiple turns
116
117         System.out.println("This is turn #"+turn);
118         // track state of players at beginning of turn
119         Player blue2 = Player.replace(blue);
120         Player red2 = Player.replace(red);
121         Player green2 = Player.replace(green);
122         Player yellow2 = Player.replace(yellow);
123
124         trackPlayers[turn].addLast(blue2);
125         trackPlayers[turn].addLast(red2);
126         trackPlayers[turn].addLast(green2);
127         trackPlayers[turn].addLast(yellow2);
128
129         // track state of table at beginning of turn
130         Table table2 = Table.replace(table);
131         trackTables.addLast(table2);
132         table.display();
133
134         captureturn = 0;
135
136         // choosing player for turn = 0
137         if (turn == 0){
138           for (j = 0; j<4; j++){
139             if (players[j].getScore() == 0){
140               possibleNextPlayers[turn].addLast(blue.convertStr(j));}
141           }}
```

```
142
143         System.out.print("The possible next players are: ");
144         for (j = 0; j<possibleNextPlayers[turn].size(); j++){
145           if (remaining_players.contains(possibleNextPlayers[turn].get(j))){
146             System.out.print(possibleNextPlayers[turn].get(j));}}
147         System.out.println();
148
149         int flagValidPlayer = 1;
150         while (flagValidPlayer!=0){
151           scanner = new Scanner(System.in);
152           System.out.println("Who do you want to play? Answer: b; r; g; y");
153           currentPlayer = scanner.nextLine();
154           if (possibleNextPlayers[turn].indexOf(currentPlayer) != -1 &&
155            (players[blue.convertInt(currentPlayer)].eliminated() == false) ){
156             flagValidPlayer=0;
157             if (turn > 0) {players[blue.convertInt(currentPlayer)].setBefore(played.get(turn-1));}
158           }
159         }
160
161         int currentPlayerInt = blue.convertInt(currentPlayer);
162         System.out.println( "It is "+ currentPlayer +"'s turn to play.");
163         played.addLast( players[currentPlayerInt].getPlayer() );
164         played_sub.addLast( players[currentPlayerInt].getPlayer() );
165
166 ////////////////// START: CHECK ELIMINATION ////////////////
167         checkElim = "";
168         // checks if player should be eliminated
169         if ((players[currentPlayerInt].getTotal()== 0) && (players[currentPlayerInt].getScore()==0)){
170           System.out.println(players[currentPlayerInt].getPlayer()
171           +" is eliminated unless a donation take place.");
172
173           scanner = new Scanner(System.in);
174           System.out.println("Would any players like to donate chips? Answer: b, r, y, g, none.");
175           String ansDonatePlayer = scanner.nextLine();
176
177           if (ansDonatePlayer.compareTo("none")!=0){
```

```
178             scanner = new Scanner(System.in);
179             System.out.println("Which color would you like to donate?");
180             String ansDonateColor = scanner.nextLine();
181
182             scanner = new Scanner(System.in);
183             System.out.println("How many would you like to donate?");
184             int ansDonateNum = Integer.parseInt(scanner.nextLine());
185
186             players[blue.convertInt(ansDonatePlayer)].donate(players[currentPlayerInt]
187             ,ansDonateColor,ansDonateNum);}
188
189           else {
190
191             players[currentPlayerInt].setScore(++score);
192             System.out.println("Player is eliminated. Their score is "
193             +players[currentPlayerInt].getScore());
194             remaining_players.remove(currentPlayer);
195
196             // remove eliminated player from played_sub
197             while (played_sub.contains(players[currentPlayerInt].getPlayer()) == true){
198               played_sub.remove(players[currentPlayerInt].getPlayer());}
199
200             if (played_sub.size() > 0 && score < 4) {
201               checkElim = played_sub.get(played_sub.size()-1);}
202
203             else if ( played_sub.size()==0 && score<4 ){
204               checkElim = players[currentPlayerInt].getBefore();
205               possibleNextPlayers[turn+1].addLast(checkElim);}
206             System.out.println("Next turn goes to " + checkElim);
207           }
208         } // end if-statement
209 ////////////////  END: CHECK ELIMINATION ////////////////
210         //
211         // Turn continues if player is not eliminated
212         boolean flagValidChip = false;
213         chipPlayed = "";
```

```
214
215       if (checkElim.equals("") && score<4){
216         while ( flagValidChip == false ) {
217
218           scanner = new Scanner(System.in);
219           System.out.print("Chip color to play: b = blue ; r = red ; g = green ; y = yellow");
220           chipPlayed = scanner.nextLine();
221           flagValidChip = true;
222
223           if ( players[currentPlayerInt].getChip(blue.convertInt(chipPlayed))>0){
224             players[currentPlayerInt].setChip(blue.convertInt(chipPlayed) ,
225             players[currentPlayerInt].getChip(blue.convertInt(chipPlayed))-1);}
226           else {
227             System.out.println("No chips of that color. Please choose another color.");
228             flagValidChip = false; }
229
230         } // end while loop
231
232         scanner = new Scanner(System.in);
233         System.out.print("Row to place the chip:");
234         String rowPlayed = scanner.nextLine();
235         table.rows[Integer.parseInt(rowPlayed)].addLast(chipPlayed);
236
237         System.out.println(chipPlayed+" chip is added to row "+rowPlayed);
238
239         played_row = "";
240         for (int i = 0 ; i < table.rows[Integer.parseInt(rowPlayed)].size() ; i++){
241           played_row += table.rows[Integer.parseInt(rowPlayed)].get(i);}
242
243
244         // if-statement for capturing
245
246         if (table.rows[Integer.parseInt(rowPlayed)].size()>1 &&
247             table.rows[Integer.parseInt(rowPlayed)]
248             .get(table.rows[Integer.parseInt(rowPlayed)].size()-2)
249             .equals(table.rows[Integer.parseInt(rowPlayed)]
```

```
250             .get(table.rows[Integer.parseInt(rowPlayed)].size()-1))){
251
252           captureturn = 1;
253           System.out.println("A capture will take place.");
254           if (players[blue.convertInt(chipPlayed)].eliminated() == true ) {
255             System.out.println("Player " + chipPlayed
256             + " is eliminated. The row will be discarded and the move goes back to "
257             +currentPlayer);}
258           else{
259             int b = 0 ;
260             while (b < played_row.length() ) {
261               String a = Character.toString(played_row.charAt(b));
262               if ( players[blue.convertInt(a)].eliminated() == true ){
263                 System.out.println("Chip "+ a + " will be discarded because Player " + a
264                 + " is eliminated.");
265                   played_row = played_row.replace(a,"");}
266               b = ++b ;
267             }
268             System.out.println("Player " + chipPlayed
269             + " takes all the remaining chips, discards one and the next move goes back to them.");}
270           if (players[blue.convertInt(chipPlayed)].eliminated() == false){
271             players[blue.convertInt(chipPlayed)].capture(played_row);
272
273             // discards chip
274             int index = -1;
275             String chipDiscard = "";
276
277             while (index == -1){
278               scanner = new Scanner(System.in);
279               System.out.print("Which chip color do you want to discard?
280 ---------------b = blue ; r = red ; g = green ; y = yellow");
281               chipDiscard = scanner.nextLine();
282               index = played_row.indexOf(chipDiscard);}
283
284             players[blue.convertInt(chipPlayed)].discard(chipDiscard,1);}
285
```

```
286           // reinitiates the row to an empty row
287           while(table.rows[Integer.parseInt(rowPlayed)].isEmpty() == false){
288             table.rows[Integer.parseInt(rowPlayed)].removeLast() ;}} // end capturing if-statement
289
290         // finding possible next players
291         String nxtp = players[blue.convertInt(chipPlayed)].nextPlayer(capturereturn, played_row,
292         remaining_players, players, chipPlayed, currentPlayer);
293         for (int i = 0; i<nxtp.length(); i++){
294           possibleNextPlayers[turn+1].addLast(Character.toString(nxtp.charAt(i)));}
295
296         // donating chip
297         // same for donate as for discarding
298         boolean flagDonate = false;
299
300         while (flagDonate == false && players[blue.convertInt(chipPlayed)].getPrisoners()>0) {
301
302           String ansDonate = new String();
303           String colorDonate = new String();
304
305           scanner = new Scanner(System.in);
306           System.out.print("Would you like to donate a chip to a player? Answer: yes or no.");
307           ansDonate = scanner.nextLine();
308
309           if (ansDonate.equals("yes")){
310
311             scanner = new Scanner(System.in);
312             System.out.print("Who would you like to donate the chip to? Answer: b, r, y, g.");
313             String chipDiscard = scanner.nextLine();
314
315             scanner = new Scanner(System.in);
316             System.out.print("Which color would you like to donate?");
317             colorDonate = scanner.nextLine();
318
319             scanner = new Scanner(System.in);
320             System.out.print("How many would you like to donate?");
321             String numbDonate = scanner.nextLine();
```

```
322             players[blue.convertInt(chipPlayed)]
323             .donate(players[blue.convertInt(chipDiscard)],colorDonate,Integer.parseInt(numbDonate));}
324
325           else{
326             flagDonate=true;}
327         } // end while loop
328
329         // discarding chip
330
331         boolean flagDiscard = false;
332
333         while (flagDiscard == false && players[blue.convertInt(chipPlayed)].getPrisoners()>0) {
334
335           // if prisoners > 0: will ask
336           // then only ask about possible chips to discard + give numbers
337           // keep asking till input is correct
338
339           scanner = new Scanner(System.in);
340           System.out.print("Would you like to discard a chip? Answer: yes or no.");
341           String ansDiscard = scanner.nextLine();
342           if (ansDiscard.equals("yes")){
343
344             scanner = new Scanner(System.in);
345             System.out.print("Which chip would you like to discard? Answer: b, r, y, g.");
346             String chipDiscard = scanner.nextLine();
347
348             scanner = new Scanner(System.in);
349             System.out.print("How many chips would you like to discard?");
350             String numbDiscard = scanner.nextLine();
351
352             if (chipPlayed.equals(chipDiscard) == false){
353               players[blue.convertInt(chipPlayed)]
354               .discard(chipDiscard, Integer.parseInt(numbDiscard));}
355           }
356           else { flagDiscard = true;}
357         } // end while loop
```

```
358
359         } // end if ( checkElim.equals("") && score<4 )
360
361
362         // displays the table in the end and information about players
363         table.display();
364         System.out.print("Blue: ");
365         blue.display();
366         System.out.print("Red: ");
367         red.display();
368         System.out.print("Green: ");
369         green.display();
370         System.out.print("Yellow: ");
371         yellow.display();
372
373
374         for (int i = 0 ; i<4 ; i++){
375           if (players[i].getScore() == 4){
376           System.out.println(players[i].getPlayer() + " wins!");
377           }
378         }
379
380         scanner = new Scanner(System.in);
381         System.out.println("Do you want to play another turn or stop the game?
382 0 = next turn ; 1 = quit ");
383         playAgain = scanner.nextLine();
384         if (playAgain.equals("0")){;
385           turn = turn + 1;}
386         else if (playAgain.equals("1")){
387           System.out.println("The End.");}
388
389
390
391
392       } // end while-loop of game
393
```

```
394     // add state of Players and Table at end of game
395     Player blue2 = Player.replace(blue);
396     Player red2 = Player.replace(red);
397     Player green2 = Player.replace(green);
398     Player yellow2 = Player.replace(yellow);
399
400     // track state of players at end of turn
401     trackPlayers[turn].addLast(blue2);
402     trackPlayers[turn].addLast(red2);
403     trackPlayers[turn].addLast(green2);
404     trackPlayers[turn].addLast(yellow2);
405
406     Table table2 = Table.replace(table);
407     trackTables.addLast(table2);
408
409     // end game analysis
410     scanner = new Scanner(System.in);
411     System.out.println("Would you like to look back on the turns? Answer yes or no");
412     String ans1 = scanner.nextLine();
413
414     if (ans1.equals("yes")){
415       scanner = new Scanner(System.in);
416       System.out.println("What would you like to look back on ? 1 = table ; 2 = players");
417       String ans2 = scanner.nextLine();
418       if (ans2.equals("1")){
419         scanner = new Scanner(System.in);
420         System.out.println("At which turn would you like to see the table?");
421         ans2 = scanner.nextLine();
422         trackTables.get(Integer.parseInt(ans2)).display();}
423       else if (ans2.equals("2")){
424         scanner = new Scanner(System.in);
425         System.out.println("After which turn would you like to see the players' chips?");
426         ans2 = scanner.nextLine();
427         for (int i = 0 ; i < trackPlayers[Integer.parseInt(ans2)].size() ; i++){
428           trackPlayers[Integer.parseInt(ans2)].get(i).display();}
429       }
```

```
430         }
431     }
432 }
```

A.6 class Analysis

```
1  import java.io.InputStream;
2  import java.util.*;
3  import java.util.LinkedList;
4  import java.util.Scanner;
5  import java.util.Arrays;
6  import java.util.Collections;
7  import java.time.format.DateTimeFormatter;
8  import java.time.LocalTime;
9
10 class Game {
11
12   public static void main(String[] args){
13
14     /*1*/ Player blue = new Player("b", 1, 1, 0, 0, 0, "r");
15     /*2*/ Player red = new Player("r", 1, 1, 0, 0, 0, "y");
16     /*3*/ Player green = new Player("g", 0, 0, 0, 0, 1,"b");
17     /*4*/ Player yellow = new Player("y", 0, 0, 0, 0, 2,"g");
18     boolean flag = true;
19     int row = 0;
20     int score = Math.max(Math.max(blue.getScore(), red.getScore()), Math.max(green.getScore(),
21                                                                              yellow.getScore()));
22     Scanner scanner = new Scanner(System.in);
23     String input = "yes";
24     String currentPlayer = "";
25     LinkedList<Move> allMoves = new LinkedList<Move>();
26     LinkedList<String> possibleNP = new LinkedList<String>();
27     Player[] players = new Player[] {blue,red,green,yellow};
28     Table table = new Table(8);
29
30     // set up the table
31     while ( input.isEmpty() == false && row<8 ){
32
33       flag = false;
```

```
34
35        while (flag == false){
36
37          System.out.println("Input Table Row " + row + ": ");
38          input = scanner.nextLine();
39
40          flag = true;
41          if (input.contains("bb") || input.contains("rr") || input.contains("gg") ||
42              input.contains("yy")){
43            flag = false;}
44          else{
45            for (int i = 0; i< input.length(); i++){
46              if (input.charAt(i) != 'b' && input.charAt(i) != 'r' &&
47                  input.charAt(i) != 'g' && input.charAt(i) != 'y'){
48                flag = false;}}}
49        }
50
51        for (int i = 0; i < input.length(); i++){
52          table.rows[row].addLast(Character.toString(input.charAt(i)));}
53
54        row++;} // end while loop
55
56      // display set up
57      for (int i = 0 ; i<4 ; i++){
58        players[i].display();}
59
60      table.display();
61
62      if (blue.getScore()==0){
63        possibleNP.addLast("b");}
64      if (red.getScore()==0){
65        possibleNP.addLast("r");}
66      if (green.getScore()==0){
67        possibleNP.addLast("g");}
68      if (yellow.getScore()==0){
69        possibleNP.addLast("y");}
```

```
70
71      while (flag != false){
72        System.out.println("The␣possible␣next␣players␣are:␣" + possibleNP.toString());
73        System.out.println("Who␣plays␣next?␣Answer:␣b,␣r,␣g␣or␣y");
74        input = scanner.nextLine();
75        if (possibleNP.indexOf(input) != -1){
76          flag = false;}
77      }
78
79      int currentPlInt = blue.convertInt(input);
80      allMoves = (LinkedList) minimax.getAllMoves(table, players, currentPlInt).clone();
81
82      Integer[] results = new Integer[allMoves.size()];
83
84      LocalTime t1 = LocalTime.now();
85      System.out.println(t1);
86
87      for (int i = 0; i < allMoves.size(); i++){
88        results[i] = minimax.minimax(allMoves.get(i));
89        System.out.println(i + "␣" + results[i]);}
90
91      System.out.println();
92
93      if (currentPlInt == 1){
94        System.out.println(Collections.min(Arrays.asList(results)));}
95      else{
96        System.out.println(Collections.max(Arrays.asList(results)));}
97
98      LocalTime t2 = LocalTime.now();
99      System.out.println(t2);
100   }
101 }
```

Bibliography

[Cam03] Colin Camerer, *Behavioral Game Theory: Experiments in Strategic Interaction*, The Roundtable Series in Behavioral Economics, Princeton University Press, 2003.

[CL07] Adam Curtis and Stephen Lambert, *The Trap: What Happened to our Dream of Freedom*, British Broadcasting Corporation, 2007.

[Fis01] Peter Fishburn, *Utility and Subjective Probability: Contemporary Theories*, International Encyclopedia of the Social and Behavioral Sciences (Neil J. Smelser and Paul B. Baltes, eds.), Pergamon, Oxford, 2001, pp. 16113–16121.

[GP17] Enrique Guerra-Pujol, *So Long Suckers: Bargaining and Betrayal in Breaking Bad*, Journal of Strategic Contracting and Negotiation **3** (2017), no. 4, 234–248.

[HNSS64] Melvin Hausner, John Nash, Lloyd Shapley, and Martin Shubik, *So Long Sucker - A Four-Person Game*, Game Theory and Related Approaches to Social Behavior: Selections (1964).

[Hof18] Pieter Hofstra, *SLS - Illustrated Rules*, Private Communication, 2018.

[HTM11] Gert Jan Hofstede and Elizabeth Tipton Murff, *Repurposing an Old Game for an International World*, Simulation and Gaming **43** (2011), no. 1, 34–50.

[JLB09] Albert Xin Jiang and Kevin Leyton-Brown, *A Tutorial on the Proof of the Existence of Nash Equilibria*, University of British Columbia Technical Report **14** (2009), 2007–25.

[LGH+15] Drake Levere, Annie Girard, Jennifer Ho, Celine Blanchard, Silvia Bonnacio, Pieter Hofstra, and Stuart Hammond, *The Effect of Gender on Decision Making in a Competitive, Socially Reliant Environment: Investigating the Influence of Gender in a New Laboratory Paradigm*, 2015.

[Sam16] Larry Samuelson, *Game Theory in Economics and Beyond*, Journal of Economic Perspectives **30** (2016), no. 4, 107–30.

[Shu87] Martin Shubik, *What Is an Application and When Is Theory a Waste of Time?*, Management Science **33** (1987), no. 12, 1511–1522.

[Shu02] _____, *Chapter 62 Game Theory and Experimental Gaming*, Handbook of Game Theory with Economic Applications **00** (2002), no. 00, 2327–2351.

[Shu05] _____, *A Double Auction Market: Teaching, Experiment, and Theory*, Simulation & Gaming **36** (2005), no. 2, 166–182.

[vNM44] John von Neumann and Oskar Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, N.J., 1944. MR 0011937