Universiteit Leiden

Master Computer Science

Quantum-inspired Hackenbush

Name: Jelle Pleunes

Date: 1 August 2024

Specialisation: Foundations of Computing

1st supervisor: Dr. W.A. Kosters
2nd supervisor: Dr. E.P.L. van Nieuwenburg

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In this thesis, we study quantum-inspired variants of two-player deterministic perfect-information games (combinatorial games), in which the last player to move wins. In a quantum-inspired ruleset, a player may play multiple moves in superposition, leading to a superposition of realisations (set of game states), which can make playing optimally more difficult than in the classical ruleset. We consider different classical rulesets, and effects of their properties on their quantum-inspired variants. Then, we turn to Quantum Hackenbush (quantum-inspired Hackenbush) as a case study. Hackenbush is a canonical example of a combinatorial game, in which players take turns removing an edge of their colour from a graph, with the rule that edges that are not path-connected to the designated "ground" node disappear. We employ a combination of theoretical and programmatic approaches to analyse structural properties, such as the winnability, of commonly studied classes of restricted Hackenbush game states, when allowing for superposed moves. This work presents a varied landscape of properties that can occur in quantum-inspired variants of combinatorial games, and it constitutes a promising starting point for further analysis of Quantum Hackenbush, or other quantum-inspired combinatorial games.

Contents

1 Introduction

In this thesis, we study a quantum-inspired generalisation of the combinatorial game Hackenbush. The game of Hackenbush (sometimes also called Red-Blue Hackenbush) is played on a graph with blue and red edges[1]. A Hackenbush position can be visualised as in Figure 1.1. One node in this graph is designated to be the *ground* node, which is typically illustrated as a horizontal line on which the edges stand. The edges are connected to the ground, either directly, or possibly through a path of other edges. Edges which are directly connected to the ground are said to be *grounded*. The two players, Left and Right, take turns removing a bLue or Red edge, respectively. Edges that are no longer connected to the ground "fly away" (are removed). A player loses when they are unable to play a move. Hackenbush is a canonical example of a combinatorial game [1, 2, 3].

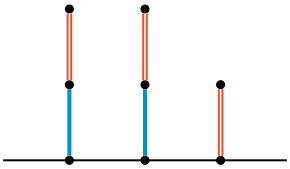


Figure 1.1: An example of a Hackenbush position.

A generalisation of Hackenbush that is also studied in literature is Red-Green-Blue Hackenbush [1, 2, 3], which introduces green edges that may be removed by either player. With these green edges, it is possible to construct positions where it is always possible for the first player to win (i.e., $\mathcal{N}$-positions, see Section 2.1). Such positions do not exist in Red-Blue Hackenbush.

We call the quantum-inspired generalisation of Hackenbush that we study Quantum Hackenbush. This generalisation introduces *superposed* moves, which means that on each turn, instead of selecting one edge to be removed, the player selects multiple edges. This leads to a *superposition* of realisations (labelled positions), each realisation corresponding to the removal of one of the selected edges. From such a superposition, a player can again play a (superposed) move. The details are explained in Section 3, where we follow the framework introduced by Dorbec and Mhalla [4] for defining quantum-inspired variants of combinatorial games, in different "flavours". An example of a small Quantum Hackenbush game tree is shown in Figure 1.2.

This thesis is structured as follows. We start with a summary of the combinatorial game theory that we use in Section 2. Then, in Section 3 we introduce quantum-inspired combinatorial games, with some discussion on how properties of certain classical rulesets impact their quantum-inspired variants, and some basic results. Sections 4 and 5 highlight our main theoretical and experimental results for different classes of restricted Hackenbush positions. Then, in Section 6, we discuss an interesting direction for future research, in finding an efficient representation of Quantum Hackenbush superposed game states. In Section 7, we discuss how quantum-inspired combinatorial games relate to "real" quantum mechanics, along with our attempt at designing a quantum circuit for playing Quantum Hackenbush, and the difficulties that we encountered. Finally, in Section 8 we discuss our conclusions, a few conjectures, and other interesting directions for further research.

This thesis was written as part of the Computer Science master's programme at the Leiden Institute of Advanced Computer Science (LIACS), Leiden University. It was supervised by Walter Kosters and Evert van Nieuwenburg.

[1]To make this thesis accessible to colourblind readers, we draw bLue edges (for the Left player) as soLid lines, Red edges (for the Right player) as paRallel lines, and grEen edges (for Either player) as dottEd lines. For other coloured pieces, we use a solid fill pattern for blue pieces, and a lines pattern for red pieces.

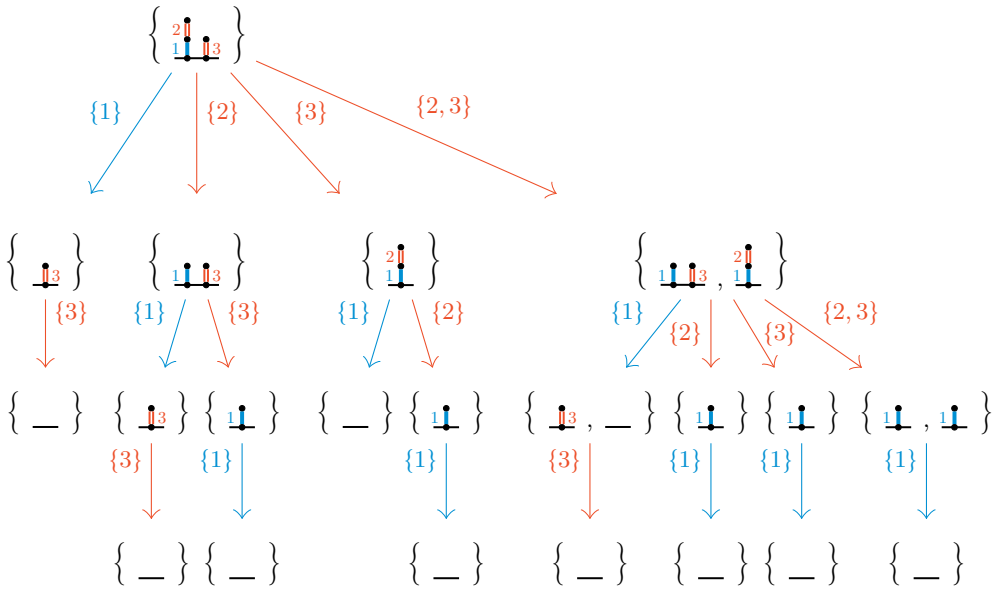


Figure 1.2: A small example Quantum Hackenbush game tree, which has superposed game states and superposed moves. A superposed game state is a set of one or more game states ("realisations"). A superposed move is a set of classical moves, each of which we attempt to apply to each realisation of the current superposed game state. If a classical move is illegal on a realisation, it does not lead to any realisation in the resulting superposed game state. Note that equivalent realisations may be left out of a set of game states, but we still show them here to emphasise how realisations are obtained from superposed moves.

2 Combinatorial games

A *combinatorial game* is a two-player deterministic game with perfect information [3]. We name the players Left and Right (or L and R in notation), and we also refer to them as the *female* player (she) and the *male* player (he), respectively (following the convention). By deterministic we mean that there are no random elements, i.e., the player will always know the result of a move exactly, before making the move. Perfect information means that both players always have complete knowledge of the game state, with no elements being "hidden" from any player.

Formally, in combinatorial game theory, a *game* is an individual position [3]. A game has *Left options* and *Right options* that can be moved to by the two respective players. Combinatorial games can be *impartial* (both players have the same options) or *partizan* (options can be specific for an individual player). For partizan games, some options may still be shared by both players. In this thesis, we are concerned with the short partizan game of Hackenbush. A *short* game is defined as a *finite loopfree* game, meaning that it has a finite number of followers and all runs are of finite length (it is not possible to loop back to an earlier position in the run) [3]. We have the following definition of short games.

Definition 2.1. A *short game* is an ordered pair

$$G = \{\mathscr{G}^L \mid \mathscr{G}^R\}$$

of Left and Right options $\mathscr{G}^L$ and $\mathscr{G}^R$.

Here, $\mathscr{G}^L$ and $\mathscr{G}^R$ are sets containing "simpler" short games that can be moved to from G. We may also list the Left and Right options explicitly, written as

$$G = \{G_1^L, \ldots, G_n^L \mid G_1^R, \ldots, G_m^R\}.$$

A *follower* of G is any position that can be reached from G, either directly or through a sequence of moves, including G itself. An *immediate follower* of G is any $G' \in \mathscr{G}^L \cup \mathscr{G}^R$. Typically, a game is played by the two players alternating turns, which we also call *normal play*. A player loses if after a finite number of turns they are unable to move. The game $\{ \mid \}$, where neither player has any options, is called the *empty game*. The *birthday* of a short game $G = \{\mathscr{G}^L \mid \mathscr{G}^R\}$ is defined as 1 plus the maximum birthday of any short game in $\mathscr{G}^L \cup \mathscr{G}^R$, where the base case $\{ \mid \}$ has a birthday of 0 [2]. A game's birthday is also the height of its game tree. The set containing all short games G forms a partially ordered Abelian group $\mathbb{G}$ with group operation $+$, inverse operation $-$, and identity element 0 [3]. In Section 2.2, we will define the binary operation $+$ and the unary operation $-$ on games, and we will prove that $\mathbb{G}$ is partially ordered. We start by defining the set of short games.

Definition 2.2. Let $\{ \mid \} = 0$ be the empty game, and

$$\mathbb{G}_0 = \{0\}$$

the set containing only the empty game. Then for $n \geq 0$ we recursively define

$$\mathbb{G}_{n+1} = \left\{ \{\mathscr{G}^L \mid \mathscr{G}^R\} \colon \mathscr{G}^L, \mathscr{G}^R \subseteq \mathbb{G}_n \right\},$$

which finally gives us

$$\mathbb{G} = \bigcup_{n \geq 0} \mathbb{G}_n.$$

This definition makes clear the finite and loopfree nature of short games.

2.1 Outcome classes

A fundamental computational problem in (algorithmic) combinatorial game theory is to determine the outcome of a game as falling into one of four possible classes [5, 6]. The outcome class of a game (position) tells a player whether it is possible for them to win. The outcome classes are as follows:

- $\mathscr{L}$ are the games that can always be won by Left, regardless of which player starts.
- $\mathscr{R}$ are the games that can always be won by Right, regardless of which player starts.
- $\mathscr{P}$ are the games that can always be won by the second player.
- $\mathscr{N}$ are the games that can always be won by the first player.

Figure 2.1 shows a Red-Green-Blue Hackenbush example position for each outcome class.

(a) Outcome $\mathscr{L}$. (b) Outcome $\mathscr{R}$. (c) Outcome $\mathscr{P}$. (d) Outcome $\mathscr{N}$.

Figure 2.1: An example position for each of the four outcome classes.

The player can also determine *how* to win, by considering the outcome classes of the move options they have available. Table 2.1 recursively defines the outcome class $o(G)$ of a short game $G = \{\mathscr{G}^L \mid \mathscr{G}^R\}$. Game G has outcome $o(G) = \mathscr{N}$ if both players can force a win when they start, $o(G) = \mathscr{L}$ if only Left can force a win when starting, $o(G) = \mathscr{R}$ if only Right can force a win when starting, and $o(G) = \mathscr{P}$ if neither player can force a win when starting. The problem of determining the outcome class of a game can be modelled as a decision problem. The complexity of solving this decision problem is also referred to as the complexity of the game.

	$\exists G^R \in \mathscr{G}^R\colon G^R \in \mathscr{R} \cup \mathscr{P}$	$\forall G^R \in \mathscr{G}^R\colon G^R \in \mathscr{N} \cup \mathscr{L}$
$\exists G^L \in \mathscr{G}^L\colon G^L \in \mathscr{L} \cup \mathscr{P}$	$o(G) = \mathscr{N}$	$o(G) = \mathscr{L}$
$\forall G^L \in \mathscr{G}^L\colon G^L \in \mathscr{N} \cup \mathscr{R}$	$o(G) = \mathscr{R}$	$o(G) = \mathscr{P}$

Table 2.1: Recursive definition of the outcome class $o(G)$ of a short game G.

2.2 Values

In addition to being able to classify which player wins a game under ideal normal play, it is also interesting to study by "how much" one of the players is able to win. For example, in Hackenbush, a position that has exactly two blue edges (shown in Figure 2.2a) has value 2. The way to understand this is to consider how many moves each player has available. In the example, there are no red edges, so clearly Left wins. However, if Left takes one of the blue edges, then the other edge will remain. This means that Left would still be able to play another move if she was given another turn. Thus, Left wins by two turns, i.e., the position consisting of two blue edges has value 2. Analogously, a position consisting of exactly two red edges has value -2. It is also possible to construct positions in which a player wins by a "fraction of a move". An example of such a position is one consisting of a blue edge with a red edge placed on top, as shown in Figure 2.2b. Here, Left can move to the empty position (value 0), while Right can move to the position with a single blue edge (value 1). We say that Left wins by "half a move", because the position has value $\{0 \mid 1\} = \frac{1}{2}$, which is the *simplest number* between 0 and 1 (we formalise this in Theorem 2.20).

Let us now formally define how values can be assigned to short games, as has been presented in the two examples above. For this, we consider the set $\mathbb{G} = \mathscr{L} \cup \mathscr{R} \cup \mathscr{P} \cup \mathscr{N}$ which contains all short games. We define a compound $G + H$ of two short games G and H, which is called the *disjunctive sum*. This compound is again a short game, in which a player on their turn moves in *exactly one* of G or H. The disjunctive sum is defined as follows.

(a) A HACKENBUSH position with value 2. (b) A HACKENBUSH position with value $\frac{1}{2}$.

Figure 2.2: Two simple positions, with values 2 and $\frac{1}{2}$, respectively.

Definition 2.3. Given short games $G, H \in \mathbb{G}$, the disjunctive sum $G + H$ is recursively defined by

$$G + H = \left\{ \bigcup_{G^L \in \mathscr{G}^L} \{G^L + H\} \cup \bigcup_{H^L \in \mathscr{H}^L} \{G + H^L\} \;\middle|\; \bigcup_{G^R \in \mathscr{G}^R} \{G^R + H\} \cup \bigcup_{H^R \in \mathscr{H}^R} \{G + H^R\} \right\}.$$

Notice that if $H = 0$ then $G + H = G$ (and $H + G = G$, by commutativity), meaning that 0 is the identity element for the disjunctive sum operation. Although the disjunctive sum is the most commonly studied compound in literature, different compounds may be considered as well. A few examples are the *conjunctive sum*, where a player must move in *both* components, the *selective sum*, where a player must move in *one or both* component(s), or the *sequential compound*, where a player must move in *the first component, unless it has no move options; in that case in the second component* [3].

Let us now define how the *equality* of two games can be determined, by the *fundamental equivalence* [3].

Definition 2.4. Given short games $G, H \in \mathbb{G}$, we write

$$G = H \text{ if and only if } o(G + X) = o(H + X) \text{ for all } X \in \mathbb{G}.$$

Additionally, we write $G \cong H$ when G is *isomorphic* with H, meaning that each Left (Right) option of G is isomorphic with exactly one Left (Right) option of H, and vice versa. One may also observe that G and H have identical game trees in this case, which is not necessarily the case when they are merely equal. We also have that $G + 0 \cong G$ for any short game G.

Let us also define the negative of a short game.

Definition 2.5. The negative of a short game G is recursively defined by

$$-G = \left\{ \bigcup_{G^R \in \mathscr{G}^R} \{-G^R\} \;\middle|\; \bigcup_{G^L \in \mathscr{G}^L} \{-G^L\} \right\}.$$

This corresponds to recursively swapping the Left and Right options, giving us the inverse of the original game. Thus, we also have that $-(-G) \cong G$.

Now we can define the difference between two short games.

Definition 2.6. The difference $G - H$ between short games G and H is defined as

$$G - H = G + (-H).$$

By considering the outcome class of a difference game $G - H$ we will be able to compare any G and H. For this we define a partial order on the outcome classes as shown in Figure 2.3.

Using this partial order on outcome classes, a partial order on the short games $\mathbb{G}$ can be defined. We start with the condition that is necessary and sufficient for $G \geq H$.

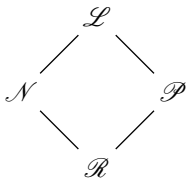


Figure 2.3: A partial order on the outcome classes in terms of desirability for player Left [3].

Definition 2.7. Given short games $G, H \in \mathbb{G}$, we write

$$G \geq H \text{ if and only if } o(G + X) \geq o(H + X) \text{ for all } X \in \mathbb{G}.$$

This means that in any sum of short games, the component G is always at least as desirable as H for Left. Replacing H by G in any sum cannot result in an outcome class that is worse for Left.

Theorem 2.8. The comparison relation $\geq$ induces a partial order on $\mathbb{G}$.

Proof. Given short games $G, H, J \in \mathbb{G}$ we have that $\geq$ respects the following properties.

(a) Reflexivity, since $G \geq G$.

(b) Anti-symmetry, because if $G \geq H$ and $H \geq G$ then $G = H$.

(c) Transitivity, because if $G \geq H$ and $H \geq J$ then $G \geq J$.

□

Theorem 2.9. Given short games $G, H, J \in \mathbb{G}$ such that $G \geq H$, it holds that $G + J \geq H + J$.

Proof. Given $G \geq H$ and using that the disjunctive sum operation is associative, we have that

$$o((G + J) + X) = o(G + (J + X)) \geq o(H + (J + X)) = o((H + J) + X) \text{ for all } X \in \mathbb{G}.$$

□

Using Definition 2.6 and Theorem 2.9, the following result can be obtained.

Theorem 2.10. Two short games $G, H \in \mathbb{G}$ can be compared based on the comparison of their difference to 0, as given by

$$G \geq H \iff G - H \geq H - H \iff G - H \geq 0.$$

Now also note that $G \geq 0 \iff o(G) \geq \mathscr{P}$, since $G = 0$ for all $G \in \mathscr{P}$. Extending this partial order relationship between the value 0 and the outcome class $\mathscr{P}$, we get the following correspondence.

Definition 2.11. The partial order correspondence between the outcome classes and the short games $G \in \mathbb{G}$ is given by

$$\begin{aligned} o(G) = \mathscr{L} &\iff G > 0, \\ o(G) = \mathscr{R} &\iff G < 0, \\ o(G) = \mathscr{P} &\iff G = 0, \\ o(G) = \mathscr{N} &\iff G \parallel 0. \end{aligned}$$

Here, $>$ and $<$ are defined based on $\geq$ and $\leq$ in the way that one would expect. The notation $G \parallel H$ means that G is incomparable with H, i.e., neither $G \geq H$ nor $G \leq H$ holds.

The ability to compare short games $G, H \in \mathbb{G}$ by considering $o(G - H)$ can be used to rewrite games into a simpler form. If we apply such a simplification to a short game $G \in \mathbb{G}$, we obtain a $G' = G$ equivalent to the original game, but for which the game tree has a smaller number of nodes. Below we will consider two kinds of simplifications; namely *domination* and *reversibility*. By repeatedly applying these simplifications until they cannot be applied anymore (the order does not matter), we obtain a unique "simplest" form $K = G$ of our original short game G, which is also called the *canonical form* of G [3]. Canonical forms of short games are unique up to isomorphism, meaning that if $G = H$ for two short games $G, H \in \mathbb{G}$ which are both in canonical form, then $G \cong H$.

Let us define how a short game can be simplified by domination.

Definition 2.12. Given a short game $G = \{\mathscr{G}^L \mid \mathscr{G}^R\} \in \mathbb{G}$, an option is said to be *dominated* in one of the following cases.

- A Left option $G^L \in \mathscr{G}^L$ is dominated if there exists some $G^{L'} \in \mathscr{G}^L, G^{L'} \ncong G^L$ such that $G^{L'} \geq G^L$.
- A Right option $G^R \in \mathscr{G}^R$ is dominated if there exists some $G^{R'} \in \mathscr{G}^R, G^{R'} \ncong G^R$ such that $G^{R'} \leq G^R$.

Theorem 2.13. If a short game G has a Left or Right dominated option and we obtain G' by removing this option, then $G' = G$.

More intuitively, we can say that outcome classes and values are not affected by the removal of suboptimal options, since these properties are based on the assumption of optimal play.

The second way to simplify a short game is through reversibility.

Definition 2.14. Given a short game $G = \{\mathscr{G}^L \mid \mathscr{G}^R\} \in \mathbb{G}$, an option is said to be *reversible* in one of the following cases.

- A Left option $G^L = \{\mathscr{G}^{LL} \mid \mathscr{G}^{LR}\} \in \mathscr{G}^L$ is reversible if there exists some $G^{LR} \in \mathscr{G}^{LR}$ such that $G^{LR} \leq G$. It is also said that G^L is *reversible through* G^{LR}.
- A Right option $G^R = \{\mathscr{G}^{RL} \mid \mathscr{G}^{RR}\} \in \mathscr{G}^R$ is reversible if there exists some $G^{RL} \in \mathscr{G}^{RL}$ such that $G^{RL} \geq G$. It is also said that G^R is *reversible through* G^{RL}.

Theorem 2.15. If a short game G has a Left option G^L that is reversible through $G^{LR} = \{\mathscr{G}^{LRL} \mid \mathscr{G}^{LRR}\}$, then we can obtain $G' = G$ by

$$G' = \{\mathscr{G}^{LRL} \cup \mathscr{G}^L \backslash \{G^L\} \mid \mathscr{G}^R\}.$$

The set $\mathscr{G}^{LRL}$ is also called the *replacement set* [2]. The symmetrical analogue of Theorem 2.15 can be used in the case of a reversible Right option G^R of G. Intuitively, if for a Right option G^{LR} of a Left option G^L it holds that $G^{LR} \leq G$, then that means that player Right can immediately counteract any advantage that Left may have obtained if she moved to G^L, by him moving to G^{LR}. Right will of course always do this, so from position G Left may as well immediately consider the options in $\mathscr{G}^{LRL}$ instead of the option G^L. Left already knows what Right's response will be if she moves to G^L.

Let us now finally discuss how a number value can be assigned to a short game G by considering its canonical form $K = G$. As a start, we have already defined that $\{ \mid \} = 0$. We have also observed that the game 0 acts as the identity element in disjunctive sums, just as the integer 0 is the identity element in sums of rationals (or reals). We also know that if Left has two free moves from K, i.e., $K = 2$, then she has the option to move to a position with one free move, while Right has no options (since K is in canonical form), meaning that $2 = \{1 \mid \}$. In general, integer games in canonical form can be defined as follows.

Definition 2.16. Given an integer $n \geq 0$, in canonical form we write

$$\{n \mid\} = n + 1 \text{ and } \{\mid -n\} = -n - 1.$$

These integer games also have the properties that one would expect.

Theorem 2.17. Integer games $A, B, C \in \mathbb{G}$ with respective values $a, b, c \in \mathbb{Z}$ have the following properties.

- $A \geq B$ if and only if $a \geq b$.
- $A + B = C$ if and only if $a + b = c$.

This means that the subgroup of integer games is isomorphic with the integers, i.e., $\mathbb{Z} \subseteq \mathbb{G}$. We use this isomorphism to refer to these short games by their integer value.

We have also mentioned (but not yet proven) that $\{0 \mid 1\} = \frac{1}{2}$. For this, one can notice that the game $\frac{1}{2}$ behaves just as the rational $\frac{1}{2}$.

Example 2.18. For example, $\frac{1}{2} + \frac{1}{2} = 1$, since

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2} &= \{0 \mid 1\} + \{0 \mid 1\} \\
&= \{0 + \{0 \mid 1\}, \{0 \mid 1\} + 0 \mid 1 + \{0 \mid 1\}, \{0 \mid 1\} + 1\} \\
&= \{\{0 \mid 1\} \mid 1 + \{0 \mid 1\}\} \\
&= \{\{0 \mid \{0 \mid \}\} \mid \{0 \mid \} + \{0 \mid 1\}\} \\
&= \{0 \mid \{0 + \{0 \mid 1\}, \{0 \mid \} + 0 \mid \{0 \mid \} + 1\}\} && \text{(by reversibility)} \\
&= \{0 \mid \{\{0 \mid 1\}, \{0 \mid \} \mid 2\}\} \\
&= \{0 \mid \{\{0 \mid \} \mid 2\}\} && \text{(by domination)} \\
&= \{0 \mid \} && \text{(by reversibility)} \\
&= 1.
\end{aligned}$$

Similarly, one can show that $\{0 \mid \frac{1}{2}\} = \frac{1}{4}$, and by iterating that $\frac{1}{2^n} = \{0 \mid \frac{1}{2^{n-1}}\}$ for $n \geq 1$. Since $\mathbb{G}$ is closed under addition, any sum of such rationals is also a short game, i.e., $m \cdot \frac{1}{2^n} = \frac{m}{2^n} \in \mathbb{G}$, with $\frac{m}{2^n} = \frac{1}{2^{n'}} + \frac{1}{2^{n''}} + \ldots$ or $\frac{m}{2^n} = -\frac{1}{2^{n'}} - \frac{1}{2^{n''}} - \ldots$ for some odd m and $n \geq 1$. Rationals of this form, which are written in their lowest terms and have a power of 2 in the denominator, are called *dyadic rationals*, and they form a group denoted by $\mathbb{D}$ [3]. These are the only kinds of rational values that can occur in short games, because their game trees have finite depth. The properties from Theorem 2.17 still apply to dyadic rational games, so we also have an isomorphism between the subgroup of dyadic rational games and the dyadic rationals, i.e., $\mathbb{D} \subseteq \mathbb{G}$. Again, we use this isomorphism to refer to these short games by their dyadic rational value. In canonical form, dyadic rational games can be written as follows.

Theorem 2.19. Given an odd integer m and an integer $n \geq 1$, in canonical form we write

$$\frac{m}{2^n} = \left\{ \frac{m-1}{2^n} \middle| \frac{m+1}{2^n} \right\}.$$

Proof. Suppose that $G = \{\mathcal{G}^L \mid \mathcal{G}^R\} = \frac{m}{2^n} \in \mathbb{G}$, and assume by symmetry that $m > 0$. Then we can also write

$$G = m \cdot \frac{1}{2^n} = \overbrace{\frac{1}{2^n} + \ldots + \frac{1}{2^n}}^{m \text{ times}}.$$

From the definition of disjunctive sum (Definition 2.3), we know that a player can move in exactly one of the components. If Left moves in a component, which is of the form

$\frac{1}{2^n} = \{0 \mid \frac{1}{2^{n-1}}\}$, then she moves to 0. If right moves in a component, then he moves to $\frac{1}{2^{n-1}}$. Thus, we obtain

$$G = \left\{ (m-1) \cdot \frac{1}{2^n} \middle| (m-1) \cdot \frac{1}{2^n} + \frac{1}{2^{n-1}} \right\} = \left\{ \frac{m-1}{2^n} \middle| \frac{m+1}{2^n} \right\}.$$

□

We can also prove that this is indeed the unique canonical form by showing that no simplifications, by neither domination nor reversibility, can be applied, based on the proof from [3].

Proof. Assume that $x = \frac{m}{2^n}$ and that it and all its followers are in canonical form, meaning that all dyadic rational followers are in the form given in Theorem 2.19. For domination, Left and Right both have only one option, so clearly there cannot be any dominated options. For reversibility, let us consider x^{LR}. We know that $x^L = \frac{m-1}{2^n} = \frac{m'}{2^{n'}}$ for some odd m' and $n' < n$, since $m-1$ is even. Then $x^{LR} = \frac{m'+1}{2^{n'}} = \frac{m-1}{2^n} + \frac{1}{2^{n'}} > \frac{m}{2^n} = x$, and thus there are no reversible Left options. Similarly, it can be shown that $x^{RL} < x$, so there are no reversible Right options either. □

If we determine the canonical form of a game, and we see that it is in the form of an integer (Definition 2.16), a dyadic rational (Theorem 2.19), or 0, then we say that the game is a *number*, and we can refer to it by its number value. Figures 2.1a, 2.1b, and 2.1c show examples of HACKENBUSH positions with number values 1, −1, and 0, respectively. Otherwise, if a game is in canonical form, but not in the form of a number, then we say that it is *not a number*. For example, Figure 2.1d shows a position that we denote $* = \{0 \mid 0\}$ (pronounced "star"). Let us also denote games of the form $G = \{x \mid x\}$, where x is a number, as $G = x + * = x*$. This notation is inspired by the convention of writing fractions $a + \frac{b}{c}$ as $a\frac{b}{c}$. The value x of a short game G is also said to be its *equivalence class*, because $G = G'$ for any $G' = x$. A theorem that can be useful for determining the number value of a short game (if it exists) is the *simplest number theorem* [2], which is given as follows.

Theorem 2.20. A short game $G = \{\mathscr{G}^L \mid \mathscr{G}^R\}$, where $G^L < G^R$ for all $G^L \in \mathscr{G}^L$ and $G^R \in \mathscr{G}^R$, is the *simplest number* x lying strictly between the largest G^L and the smallest G^R, as given by:

- The integer $x = n$ smallest in absolute value, for which $G^L < n < G^R$, if such an integer exists.
- The fraction $x = \frac{i}{2^j}$ for which $G^L < \frac{i}{2^j} < G^R$ with the smallest j, otherwise.

3 Quantum-inspired combinatorial games

A *quantum-inspired combinatorial game* is a combinatorial game in which the domain of game states and the move functions have been adapted to allow for superposed game states and superposed moves, respectively. We call these games "quantum-inspired", because they are *not* based on quantum mechanics, with game states defined on continuous qubits (quantum bits) and moves defined as unitary transformations. Rather, they are combinatorial games (which have discrete game states and moves) with aspects inspired by concepts from quantum mechanics; namely superposition, entanglement, collapse (also referred to as measurement), and (constructive) interference.

When defining a quantum-inspired variant of a classical combinatorial game, we first need a more explicit description of which classical moves are legal on a given game state, and when classical moves are considered to be distinct, according to the ruleset. For this reason we assign a *label* to each move option (for both Left and Right). The new formulation that we introduce is similar to those introduced in [4] and [6], except that we explicitly define labelled short games, where they must be used, and why they cannot be simplified (unlike "normal" non-labelled short games). We refer to positions in labelled short games as *game states* (instead of "positions") for clarity, and to emphasise that we cannot abstract away from what the visual representation of the state "looks like" (which *is* possible for non-labelled short games).

Definition 3.1. Let $G = \{\mathscr{G}^L \mid \mathscr{G}^R\} \in \mathbb{G}$ be a short game that can be generated by following a *ruleset* $\mathcal{R}$. Then, let $\widehat{G}$ denote a corresponding *labelled short game*, in which each move option $G' \in \mathscr{G}^L \cup \mathscr{G}^R$ has been assigned a label $\sigma \in \Sigma = \{1, \ldots, n\}$, for some integer $n \geq 1$, as defined by $\mathcal{R}$, and where this labelling of move options is applied to each G' to yield $\widehat{G}'$, recursively.

The finite set Σ is called the *move alphabet*.

Definition 3.2. We say that two game states $\widehat{G}$ and $\widehat{H}$ are *equivalent*, written $\widehat{G} \equiv \widehat{H}$, if for each Left (Right) labelled move option of $\widehat{G}$ there is exactly one equivalent Left (Right) labelled move option of $\widehat{H}$ that has been assigned the same label, and vice versa.

For ease of notation, we may also write $\widehat{G} \cong \widehat{H}$ or $\widehat{G} \cong H$, and $\widehat{G} = \widehat{H}$ or $\widehat{G} = H$ to mean isomorphism and equality in the combinatorial game sense, respectively. For example, $\widehat{G} \cong H$ means "the short game G obtained from removing the labelling from $\widehat{G}$ is isomorphic to the short game H", and $\widehat{G} = \widehat{H}$ means "the short games G and H obtained from removing the labellings from $\widehat{G}$ and $\widehat{H}$, respectively, have the same value". However, whenever we are considering a set of game states, we mean elements to be unique up to equivalence.

Any non-empty short game G generated by a given ruleset has a countably infinite number of distinct (non-equivalent) corresponding labelled short games $L = \{\widehat{G}_1, \widehat{G}_2, \ldots\}$, which are all defined by the ruleset. Any labelled move option $\widehat{H}$ of $\widehat{G} \in L$ may be recursively assigned a new label, as long as the new label does not already occur in any follower of $\widehat{G}$. We also call the repeated application of this process (zero or more times) *relabelling*. In particular, notice that it is possible to "swap" any two labels in this way, and that for our given G any $\widehat{G}$ can be relabelled to any other $\widehat{G}' \in L$.

Theorem 3.3. Given two game states $\widehat{G}$ and $\widehat{H}$, if any one can be relabelled to be equivalent to the other, then $\widehat{G} \cong \widehat{H}$.

This relates to the concept of *transpositions* from combinatorial game theory, which are positions with the same value reached through different sequences of moves [2]. Our theorem ensures isomorphism, however, which is stronger than equality. The theorem can be proven as follows.

Proof. By Definition 3.2, each Left (Right) labelled move option of $\widehat{G}$ has exactly one corresponding isomorphic Left (Right) labelled move option in $\widehat{H}$, and vice versa. □

The ruleset defines a labelling of options of a game state, which is unique up to relabelling.

Definition 3.4. A *ruleset* is defined by

$$\mathcal{R} = (\rho_L, \rho_R),$$

where $\rho_L, \rho_R \colon \mathbb{G}^{\mathcal{R}} \times \Sigma \to \bigcup_{\widehat{G} \in \mathbb{G}^{\mathcal{R}}} \{\{\widehat{G}\}\} \cup \{\varnothing\}$ are the move functions for Left and Right, respectively, and $\mathbb{G}^{\mathcal{R}}$ is the (countably infinite) set of all labelled short games generated and labelled by following $\mathcal{R}$. If from a game state $\widehat{G} \in \mathbb{G}^{\mathcal{R}}$ a player $p \in \{L, R\}$ plays an illegal move $\sigma \in \Sigma$, then $\rho_p(\widehat{G}, \sigma) = \varnothing$. Otherwise, the result is a singleton set $\rho_p(\widehat{G}, \sigma) = \{\widehat{G}'\}$ containing the resulting game state $\widehat{G}'$. If σ is legal for both players, then we also impose that $\rho_L(\widehat{G}, \sigma) = \rho_R(\widehat{G}, \sigma)$ (for both players, the same move must lead to the same game state), and recursively that $\rho_L(\widehat{G}', \sigma) = \rho_R(\widehat{G}', \sigma)$ for all followers $\widehat{G}'$ of $\widehat{G}$. This ensures that if a move that was at any point legal for both players becomes illegal, then it will also be illegal for both players. If σ is only legal for exactly one $p \in \{L, R\}$, then we impose for the opposing player $\bar{p}$ that $\rho_{\bar{p}}(\widehat{G}', \sigma) = \varnothing$, recursively for all followers $\widehat{G}'$ of $\widehat{G}$ (the move must never become legal for the other player).

The reason to use singleton sets in this way, is to allow for more natural definitions of superposed game states and superposed move functions, which we will give below. We have for impartial rulesets that $\rho_L = \rho_R$, and for partizan rulesets that $\rho_L \neq \rho_R$. An *empty game state* is a game state on which no moves are legal, for Left nor Right.

Definition 3.5. Given a game state $\widehat{G}$ and player $p \in \{L, R\}$, the set of *legal moves* is given by

$$\mathrm{P}_p(\widehat{G}) = \{\sigma \colon \sigma \in \Sigma, \rho_p(\widehat{G}, \sigma) \neq \varnothing\}.$$

Let us introduce some additional useful notation for moving and determining legal moves.

Definition 3.6. The *spanning move function* $\rho \colon \mathbb{G}^{\mathcal{R}} \times \Sigma \to \bigcup_{\widehat{G} \in \mathbb{G}^{\mathcal{R}}} \{\{\widehat{G}\}\} \cup \{\varnothing\}$ spans across both players' legal moves, and is defined as

$$\rho(\widehat{G}, \sigma) = \rho_L(\widehat{G}, \sigma) \cup \rho_R(\widehat{G}, \sigma).$$

The result will still always be a singleton set or $\varnothing$, because if σ is legal for both players, then it must hold that $\rho_L(\widehat{G}, \sigma) = \rho_R(\widehat{G}, \sigma)$, if it is legal for exactly one player, then $\rho(\widehat{G}, \sigma) = \rho_L(\widehat{G}, \sigma)$ or $\rho(\widehat{G}, \sigma) = \rho_R(\widehat{G}, \sigma)$, and if it is legal for neither player, then $\rho_p(\widehat{G}, \sigma) = \varnothing$ for $p \in \{L, R\}$. Note that we are still considering both impartial and partizan rulesets. The spanning move function simply "selects" a player p for which a given move is legal (if there exists such a p).

Definition 3.7. Given a game state $\widehat{G} \in \mathbb{G}^{\mathcal{R}}$ the set of legal moves *for any player* is given by

$$\mathrm{P}(\widehat{G}) = \{\sigma \colon \sigma \in \Sigma, \rho(\widehat{G}, \sigma) \neq \varnothing\} = \mathrm{P}_L(\widehat{G}) \cup \mathrm{P}_R(\widehat{G}).$$

A ruleset can be said to have the *dead-ending* property [7].

Definition 3.8. A ruleset $\mathcal{R}$ is *dead-ending* if for all labelled short games $\widehat{G}$ and all $\sigma \in \mathrm{P}(\widehat{G})$, with $\rho(\widehat{G}, \sigma) = \{\widehat{G}'\}$, it holds that $\mathrm{P}(\widehat{G}') \subseteq \mathrm{P}(\widehat{G})$.

This means that making a move cannot "reveal" another move to become legal in any following game state. If a move becomes illegal, it will remain illegal for the remainder of play. In this thesis we will mostly be concerned with $\mathcal{R} =$ Hackenbush, which is dead-ending, but other dead-ending rulesets will be considered as well. A simple example of a non-dead-ending ruleset is Maze. It is played on a grid oriented at a 45° angle, with certain edges highlighted as walls, and a piece that can be moved one or more cells southwest or southeast, by Left or Right, respectively [2]. At a Maze game state, Left may initially not have any move option, but Right's move may "open up" an option for Left. Figure 3.1 shows such an example.

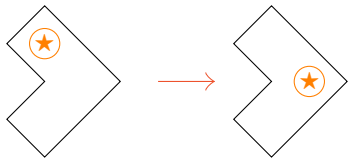


Figure 3.1: A MAZE game state where Left does not initially have any move option (left), but Right's move gives Left a move option (right).

In HACKENBUSH, a move naturally corresponds one-to-one to the removal of an edge[2]. Whenever two options correspond to the same move (i.e., "remove edge σ"), they are assigned the same label $\sigma \in \Sigma$, as defined by the ruleset. This means that for any game state $\widehat{G}$, a label corresponds to either exactly one Left (Right) option $\widehat{G}'$, or to no Left (Right) option at all. Labels that do not correspond to any Left (Right) option at a given game state, correspond to *illegal* moves. We also require any labelling to be *persistent*, which ensures that the "interpretation" of any label does not change as play progresses. In combinatorial game theory, it is desirable to have games for which the rules can be defined concisely, while at the same time making optimal play non-trivial. Thus, in order to have a concise definition of a game's rules, it is necessary for move labels to have an interpretation that is consistent across all reachable game states [4]. Formally, we have the following.

Definition 3.9. A ruleset $\mathcal{R}$ is said to be *persistent* if, given an arbitrary subset $M \subseteq \Sigma$ of moves, and some game state $\widehat{G} \in \mathbb{G}^{\mathcal{R}}$, applying all moves $\sigma \in M$ in any order as a sequence (by repeated application of the spanning move function) always results in either $\{\widehat{G}'\}$ or $\varnothing$, where all $\widehat{G}'$ are equivalent.

This means that moves always "accumulate" in the same way, leading to the same result, regardless of the order in which they are applied. Note that HACKENBUSH is persistent. We also know that this ruleset is dead-ending, but in general, if a ruleset is dead-ending it does not necessarily mean that it is also persistent (nor the other way around). In fact, for the dead-ending and persistence properties, there exist rulesets that have both properties, only one of the two, and neither.

As an example of a dead-ending ruleset that is not persistent, take CLEAR THE POND. It is played on a finite strip of squares, where each square may be empty or occupied by a blue or red piece, and where a blue (red) piece can be moved to the first empty square to its right (left) by Left (Right), or off the strip if no such empty square exists [2]. A move for either player is labelled by which piece of their colour they wish to move, where their pieces are indexed starting from the left. This ruleset is dead-ending, because the number of pieces can only ever decrease (pieces can be moved off the strip, but there is no mechanism for the number of pieces to increase). However, CLEAR THE POND is not persistent, because different sequences of legal moves can lead to different game states. An example is illustrated in Figure 3.2.

An example of a non-dead-ending, but persistent ruleset is MAZE. This ruleset is persistent, because a move (indicating direction and distance) is legal as long as it does not attempt to move the piece through a wall. Thus, any sequence of legal moves can always be "summed up", as if the piece only moved in an L-shape on a grid without any walls (all Left moves first, followed by all Right moves), to give the resulting game state.

Finally, an example of a ruleset which is not dead-ending nor persistent is KONANE. It is played on a rectangular grid, where each cell may be empty or occupied by a blue or red piece, and where a blue (red) piece can jump over an orthogonally adjacent red (blue) piece onto an empty cell, removing the piece that was jumped over [2]. A move for either player is labelled by which piece of their colour they wish to move, and in which cardinal

[2]For certain other combinatorial games, such as NIM, there is more room for discussion, regarding what constitutes a "natural labelling" when defining the ruleset. In Section 3.2 we elaborate on this discussion.

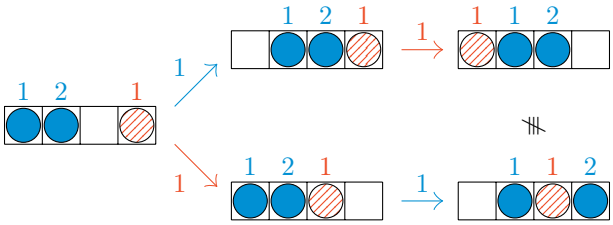


Figure 3.2: An example Clear the Pond position where Left moves first, followed by Right (top), and where Right moves first, followed by Left (bottom). These are both sequences of legal moves, but they do not lead to equivalent game states.

direction, where their pieces are indexed row-wise from the top left to the bottom right. It is not dead-ending, because players can "open up" moves for each other, as illustrated in Figure 3.3. Also, Konane is not persistent[3], because applying two legal moves in different orders can lead to different game states, as shown in the example in Figure 3.4.

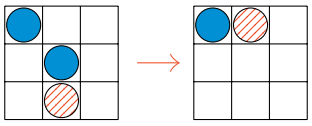


Figure 3.3: A Konane game state where Left does not have any move option initially (left), but where she does have a move option after Right's move (right).

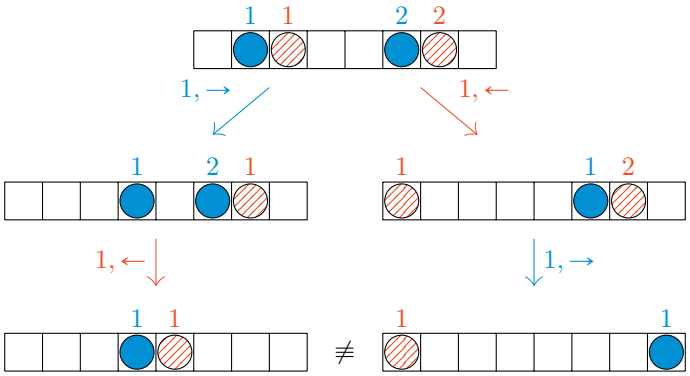


Figure 3.4: A Konane game state where Left first moves her piece with label 1 east, after which Right moves his piece with label 1 west (left subtree), and where Right first moves his piece with label 1 west, followed by Left moving her piece with label 1 east (right subtree). These two legal sequences of moves do not lead to equivalent game states.

Notice that, compared to our original Definition 2.1, in Definition 3.1 the notion of moving has a different connotation. Rather than indicating the position that should be transitioned to from the current position, we now indicate what we want to *change* about the current game state in order to obtain the new game state, and we label this change as $\sigma \in \Sigma$. Which changes σ comprise legal moves from any given game state $\widehat{G} \in \mathbb{G}^{\mathcal{R}}$, is defined by the ruleset $\mathcal{R}$. In the world of classical combinatorial games it is not customary to introduce a move alphabet Σ, because it is typically easier to simply describe a move by the resulting position, as we did in Definition 2.1. However, the move alphabet will help us in providing a more "natural" definition of quantum-inspired

[3]It should be noted that the way in which we identify a piece when playing a move makes it so that the ruleset is not persistent. We could also define Konane differently, for example by giving each piece a unique number, and keeping track of each unique piece's position throughout play, which would then make the ruleset persistent. In Section 3.2 we elaborate on the discussion surrounding different definitions of the "same" classical ruleset, which define different labellings.

combinatorial games, which we will do below.

We follow the general framework for transforming a combinatorial game into a quantum-inspired variant, introduced by Dorbec and Mhalla [4], and which has been used in later work [6, 8]. They introduce five *flavours* of quantum-inspired combinatorial games; $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{C}'$, and $\mathcal{D}$. The flavours differ in terms of when it is allowed to play an unsuperposed move (move of width 1), and we define each of them below (Definition 3.13), after we have defined superposed game states, superposed moves, and superposed move functions.

For the definitions below, we fix a (classical) ruleset $\mathcal{R} = (\rho_L, \rho_R)$. In the framework, a *superposed game state* is a set of distinct classical game states (also called *realisations*) under a flavour $f \in \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{C}', \mathcal{D}\}$. The definition is as follows.

Definition 3.10. A *superposed game state*

$$\{\widehat{G}_1, \ldots, \widehat{G}_n\}^f$$

is a set of $n \geq 0$ distinct classical game states $\widehat{G}_1, \ldots, \widehat{G}_n \in \mathbb{G}^{\mathcal{R}}$ that all have the *same* labelling (not just any labelling) as defined by $\mathcal{R}$, with a flavour f. We say that $\{\widehat{G}_1, \ldots, \widehat{G}_n\}^f$ is n-wide. We also call a 1-wide superposed game state $\{\widehat{G}_1\}^f$ a classical game state under flavour f. The set of all possible superposed game states is denoted $\mathbb{G}^{\mathcal{R}^f} = \left\{A^f : A \in 2^{\mathbb{G}^{\mathcal{R}}}\right\}$ (all possible sets of game states under a flavour f). The empty superposed game state $\varnothing^f$ never occurs in a (labelled) short game tree. Rather, it represents the result of an illegal superposed move.

Notice that a superposed game state itself is also a game state, which has been labelled as defined by the ruleset $\mathcal{R}^f$ (the quantum-inspired variant of $\mathcal{R}$ under flavour f). This means that, if one would like, it is possible to define a quantum-inspired quantum-inspired combinatorial game, or a quantum-inspired quantum-inspired quantum-inspired combinatorial game, and so on.

Analogously to a superposed game state, a *superposed move* is a set of distinct classical moves.

Definition 3.11. A *superposed move*

$$\{\sigma_1, \ldots, \sigma_m\}$$

is a set of $m \geq 1$ distinct classical moves $\sigma_1, \ldots, \sigma_m \in \Sigma$, which has *move width* m. A move of width 1 is a classical move, which we denote by $\{\sigma_1\}$ or simply σ_1. We use $\Sigma_w = \left\{\{\sigma_1, \ldots, \sigma_m\} : \{\sigma_1, \ldots, \sigma_m\} \in 2^{\Sigma}, 1 \leq |\{\sigma_1, \ldots, \sigma_m\}| \leq w\right\}$ to denote the set of all possible superposed moves of width $\leq w$.

Notice that if $\Sigma = \{1, \ldots, n\}$ and $w \geq n$, then Σ_w contains all possible superposed moves. For the remainder of this thesis, we do not make any assumptions on whether w is set to some finite value or if it is unbounded, because usually this does not matter. For results that are specific to some w, we will state it explicitly.

A superposed move $\{\sigma_1, \ldots, \sigma_m\}$ is legal if each of its classical moves $\sigma_1, \ldots, \sigma_m$ are legal on at least one (not necessarily the same) realisation of the superposed game state $\{\widehat{G}_1, \ldots, \widehat{G}_n\}^f$. The move functions for the players $\{L, R\} \ni p$ are now of the form $\rho_p^f : \mathbb{G}^{\mathcal{R}^f} \times \Sigma_w \to \mathbb{G}^{\mathcal{R}^f}$, and are defined as follows.

Definition 3.12. Given a player $p \in \{L, R\}$, a superposed game state $\widehat{S} \equiv \{\widehat{G}_1, \ldots, \widehat{G}_n\}^f$, and a superposed move $\{\sigma_1, \ldots, \sigma_m\}$, the *superposed move function* ρ_p^f is defined as

$$\rho_p^f(\widehat{S}, \{\sigma_1, \ldots, \sigma_m\}) \equiv \begin{cases} \left(\bigcup_{1 \leq i \leq n, 1 \leq j \leq m} \rho_p(\widehat{G}_i, \sigma_j)\right)^f & \text{if } 1 < m \leq w, \text{ and all } \sigma_j \text{ are legal on at least one } \widehat{G}_i; \\ \left(\bigcup_{1 \leq i \leq n} \rho_p(\widehat{G}_i, \sigma_1)\right)^f & \text{if } m = 1, \text{ and } \sigma_1 \text{ is legal according to } f; \\ \varnothing^f & \text{otherwise.} \end{cases}$$

Intuitively, making a superposed move means trying to apply all given classical moves to all given classical game states in a "Cartesian product"-style fashion. If all classical moves in the superposed move are illegal on a realisation that we are trying to apply them to, then we also say that the realisation *collapses*. If the superposed move is illegal, then it leads to the empty superposed game state.

In terms of relabelling, note that in a superposed game state any single realisation may not simply be relabelled, because this could introduce additional legal superposed moves (or remove them), which may affect winnability. Instead, all realisations should be relabelled using the same relabelling function.

Let us now define the five flavours of quantum-inspired combinatorial games.

Definition 3.13. The five flavours, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{C}'$, and $\mathcal{D}$, of quantum-inspired combinatorial games are defined as follows.

- Flavour $\mathcal{A}$: unsuperposed moves are never allowed.
- Flavour $\mathcal{B}$: unsuperposed moves are not allowed, except if there exists only one legal classical move across all realisations, then the player can play this unsuperposed move.
- Flavour $\mathcal{C}$: unsuperposed moves are not allowed, except if the move is legal across all realisations.
- Flavour $\mathcal{C}'$: unsuperposed moves are not allowed, except if the move is legal across all realisations where the player still has at least one legal classical move.
- Flavour $\mathcal{D}$: unsuperposed moves are always allowed.

The lattice of permissivity of the different flavours is shown in Figure 3.5. Flavours $\mathcal{B}$ and $\mathcal{C}$ cannot be compared mutually, because flavour $\mathcal{B}$ allows for unsuperposed moves that may be illegal in some realisations, while flavour $\mathcal{C}$ allows for unsuperposed moves, even if other classical moves are available.

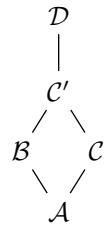


Figure 3.5: The lattice of permissivity for the different quantum flavours [4].

Under flavours $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}'$, and $\mathcal{D}$, empty realisations do not have any impact on which (unsuperposed or superposed) move options are available to the player. Empty realisations can safely be "ignored" in a superposed game state. Under flavour $\mathcal{C}$, however, the presence of an empty realisation means that whatever unsuperposed move the player may want to play, it will be illegal on (at least) one realisation; namely the empty realisation. This means that under flavour $\mathcal{C}$, a player may have different options depending on whether the current superposed game state contains an empty realisation.

3.1 Playing quantum-inspired combinatorial games

Quantum-inspired combinatorial games are played using superposed moves, which are sets of classical moves. A superposed move is legal if and only if each of its classical moves is legal in a *possibilistic* sense. This means that it must be possible for the current superposed game state to "collapse" (to be measured in the physical sense, so that one of the realisations becomes "reality") in such a way that the classical move becomes legal, meaning that the classical move should be legal in at least one realisation.

It is possible to play a quantum-inspired variant of a combinatorial game, without needing to keep track of all realisations, the number of which potentially grows exponentially with the number of turns. The approach proposed in [4] is for a player on their turn to announce the set of classical moves they want to play. If the other player suspects that this superposed move is illegal, then they can challenge the player that moved to prove that their superposed move was legal. This can be proven by, for each classical move in the superposed move, providing a classical run respecting all superposed moves that have been played thus far. This way, they prove that all classical moves they played were legal in at least one realisation. If the challenged player manages to prove the legality of their superposed move, then they win, otherwise they lose. Note that the option for a player to challenge their opponent does not affect winnability. If a player is able to win in the absence of challenges, then they are also able to win if they are challenged during play.

This approach works for the more "straightforward" flavours $\mathcal{A}$ and $\mathcal{D}$. However, for flavours $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{C}'$, the set of realisations must be inspected to be able to determine whether any unsuperposed move is legal. For example, under flavour $\mathcal{B}$, if a player plays an unsuperposed move, and they are challenged by their opponent, then they must be able to prove that all other unsuperposed moves would have been illegal. Proving that a classical move is illegal requires the player to consider all possible classical runs respecting the superposed moves made thus far, effectively constructing the entire set of realisations.

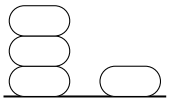
3.2 Labelling moves

For defining a quantum-inspired combinatorial game, the way in which we define the ruleset of the classical game is important, because we obtain our labelling directly from this ruleset. This labelling then, in turn, determines which classical moves in a superposed move we consider to be distinct. We say that the ruleset Hackenbush is *natural*, because it defines a "natural" labelling; namely one where a move (label) corresponds one-to-one to the removal of an edge. So-called *placement games* have a similar property, where a move corresponds one-to-one to placing a piece on a board [7]. However, many other combinatorial games, such as Nim, do not have this property. We also call these games *nonnatural*. The ruleset for the game of Nim can be given in multiple ways, each defining a different labelling. We illustrate differences in labellings (and their effects on the quantum-inspired variants of the rulesets) by defining two rulesets for Nim, which we call Nim-Subtract and Nim-Cut, respectively.

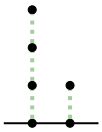
Definition 3.14. Nim-Subtract is played on a number of heaps of tokens. On their turn, a player selects *one* heap, from which they take (subtract) a number of tokens.

Definition 3.15. Nim-Cut is played on a number of heaps of tokens. On their turn, a player selects *one* heap, for which they specify the height at which to cut, reducing the heap to a size strictly smaller than its current size.

Intuitively, a move in Nim-Subtract can be seen as taking a number of tokens off the top of a heap. On the other hand, in Nim-Cut a move can be seen as selecting a token in a heap, and removing this token, along with any tokens above it. Note that both of these rulesets are dead-ending and persistent. Also notice that Nim-Cut can be simulated in Red-Green-Blue Hackenbush by constructing a green stalk of appropriate length for each Nim heap. We also say that the game of Nim-Cut is contained in the game of Red-Green-Blue Hackenbush. A Nim game state (ruleset Nim-Subtract or Nim-Cut) can be illustrated as in Figure 3.6, where it is shown with its corresponding Red-Green-Blue Hackenbush game state. Classically, Nim-Subtract and Nim-Cut always behave the same way, in the sense that they generate isomorphic short games when starting from any given heap configuration. However, as we will illustrate below, from Nim-Subtract we obtain a more interesting quantum-inspired variant than from Nim-Cut, because for a single heap the latter case essentially reduces back to the classical game.



(a) A two-heap Nim-Subtract or Nim-Cut game state with 3 and 1 token(s), respectively.



(b) A Red-Green-Blue Hackenbush game state with two green stalks of length 3 and 1, respectively.

Figure 3.6: A Nim game state with its corresponding Red-Green-Blue Hackenbush game state.

We can play an "unlabelled" move (the resulting game state could be seen as the "label" of the move) to the game state with 2 and 1 token(s) as shown in Figure 3.7.

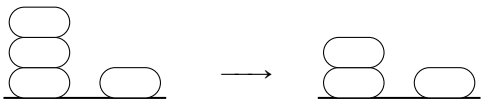


Figure 3.7: Moving to the Nim game state with 2 and 1 token(s), respectively.

In Nim-Subtract we would label this move by (i, n), where i is the heap number and n is the number of tokens being removed, as shown in Figure 3.8.

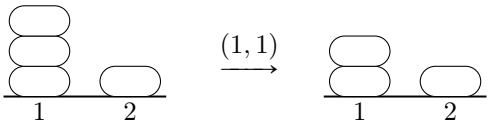


Figure 3.8: Applying move $(1, 1)$ to a Nim-Subtract game state, which removes one token from the first heap.

In Nim-Cut we would label this move differently, namely as (i, j). Here, i still indicates the heap number, but j indicates the height of the token that should be removed, along with any tokens above it. An example is shown in Figure 3.9.

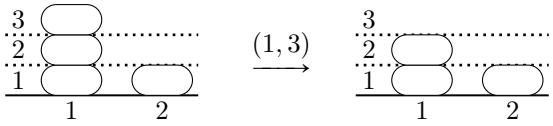
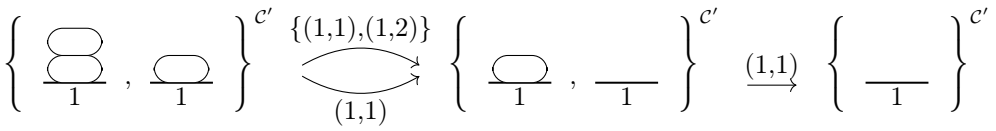


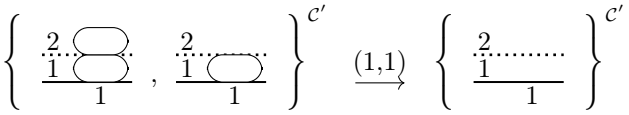
Figure 3.9: Applying move $(1, 3)$ to a Nim-Cut game state, which removes token 3 and any tokens above it from the first heap.

The outcome classes of superposed game states are not always equal between Quantum Nim-Subtract and Quantum Nim-Cut. We illustrate an example of a superposed game state for a single heap under flavour $\mathcal{C}'$ in Figure 3.10. This superposed game state can be reached starting from a classical game state of a single three-token heap and removing one and two tokens in superposition, for example.

In Quantum Nim-Subtract (Figure 3.10a), the second player always wins, because the first player is not able to immediately move to the classical empty game state. Rather, the first player must play the superposed move $\{(1, 1), (1, 2)\}$, or the only classical move that is valid in all non-empty realisations $(1, 1)$. On the following turn, the second player is able to move to the classical empty game state by move $(1, 1)$. However, in Quantum Nim-Cut (Figure 3.10b), the first player is able to win by simply playing $(1, 1)$ immediately.



(a) In Quantum Nim-Subtract, the second player always wins (this is a $\mathscr{P}$-position).



(b) In Quantum Nim-Cut, the first player always wins (this is an $\mathscr{N}$-position).

Figure 3.10: Winning strategies for an example superposed game state, in Quantum Nim-Subtract and Quantum Nim-Cut, both under flavour $\mathcal{C}'$. This illustrates the difference between two ways of labelling Nim moves in the quantum-inspired case.

If we consider Quantum Nim-Cut under flavour $\mathcal{D}$ with a single heap, then we can notice that this ruleset effectively reduces back to classical Nim-Cut. Any superposed game state can be collapsed into a classical game state by means of a classical move. This is due to the property that no label ever repeats along any path in a Nim-Cut labelled short game tree, which we call *non-repeating*. This means that whenever we play a move, it becomes illegal for the remainder of play. It follows that, in a superposed game state, any realisations on which we try to apply such a move "for a second time" collapse. Nim-Subtract, on the other hand, is *repeating*. Whenever a move is "repeated", it does not necessarily mean that the realisation collapses. This means that, in general, we obtain superposed game states consisting of more realisations, which cannot be collapsed as easily, making the game more interesting.

3.3 Covered realisations

We call empty realisations that we are able to ignore *covered realisations*. The notion of covering can be generalised to other realisations in a superposed game state, beyond the trivial case of the empty game state [4], as we will discuss below.

Definition 3.16. Given a set of game states $\{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_n\} \in 2^{\mathbb{G}^{\mathcal{R}}}$, we say that $\widehat{G}_1$ is *covered* by $\{\widehat{G}_2, \ldots, \widehat{G}_n\}$ if for all $\sigma \in \mathrm{P}(\widehat{G}_1)$ with $\rho(\widehat{G}_1, \sigma) = \{\widehat{G}'_1\}$, there exists some $\widehat{G}_i, 2 \leq i \leq n$ on which σ is also legal, and game state $\widehat{G}'_1$ is covered by $\bigcup_{2 \leq i \leq n} \rho(\widehat{G}_i, \sigma)$.

Note that an empty game state $\widehat{G}_1$ (meaning that $\mathrm{P}(\widehat{G}_1) = \varnothing$) is covered by any non-empty set of other realisations.

Theorem 3.17. For flavour $f \in \{\mathcal{A}, \mathcal{B}, \mathcal{D}\}$, if we have a superposed game state $\{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_n\}^f \in \mathbb{G}^{\mathcal{R}^f}$ such that $\widehat{G}_1$ is covered by $\{\widehat{G}_2, \ldots, \widehat{G}_n\}$, then $\{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_n\}^f \equiv \{\widehat{G}_2, \ldots, \widehat{G}_n\}^f$.

This theorem means that a covered realisation does not add any superposed move options. It can be left out of the superposed game state without affecting the labelled short game tree.

Note that this theorem does not hold for flavour $\mathcal{C}'$, whereas it is allowed to ignore empty realisations under that flavour. Take for example $\widehat{S} \equiv \{\widehat{G}_1, \widehat{G}_2, \widehat{G}_3\}^{\mathcal{C}'}$, with $\mathcal{R} =$ Hackenbush, two moves $\sigma_1, \sigma_2 \in \Sigma$, with $\mathrm{P}_L(\widehat{G}_1) = \{\sigma_1\}$, $\mathrm{P}_L(\widehat{G}_2) = \{\sigma_2\}$, and $\mathrm{P}_L(\widehat{G}_3) = \{\sigma_1, \sigma_2\}$, and where σ_1 and σ_2 are two grounded blue edges (since moves correspond one-to-one to edges in Hackenbush). We have that σ_1, the only move that is legal on $\widehat{G}_1$, is also legal on a game state in $\{\widehat{G}_2, \widehat{G}_3\}$; namely on $\widehat{G}_3$, and applying σ_1 to $\widehat{G}_1$ leads to the empty game state which is covered (by $\rho_L(\widehat{G}_3, \sigma_1)$), so $\widehat{G}_1$ is covered by $\{\widehat{G}_2, \widehat{G}_3\}$. However, we cannot say that $\widehat{S} \equiv \widehat{S}'$, with $\widehat{S}' \equiv \{\widehat{G}_2, \widehat{G}_3\}^{\mathcal{C}'}$, because the classical move σ_2 is not legal on $\widehat{S}$ (because σ_2 is not legal on $\widehat{G}_1$, while Left does have

a legal classical move σ_1 on that realisation), while it is legal on $\widehat{S}'$. We now give the proof of Theorem 3.17, based on the proof from [4].

Proof. The equivalence can be proven by induction on the birthday of game state $\widehat{G}_1$. If $\widehat{G}_1$ is empty, then it has birthday 0. In this case it is clear that $\widehat{S} \equiv \{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_n\}^f$ has the same legal move options as $\widehat{S}' \equiv \{\widehat{G}_2, \ldots, \widehat{G}_n\}^f$ for $f \in \{\mathcal{A}, \mathcal{B}, \mathcal{D}\}$, meaning that $\widehat{S} \cong \widehat{S}'$, and $\widehat{S} \equiv \widehat{S}'$ since both have the same labelling. Now for the induction step, assume that the theorem holds for all game states which have a birthday smaller than some arbitrary $\widehat{G}_1$'s birthday. If we apply some legal move $\{\sigma_1, \ldots, \sigma_m\} \in \Sigma_w$ to our superposed game state $\widehat{S} \equiv \{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_n\}^f$, then we get the superposed game state $\widehat{T} \equiv \left(\bigcup_{1 \leq i \leq n, 1 \leq j \leq m} \rho(\widehat{G}_i, \sigma_j)\right)^f$. Here, for every $\sigma_j \in \{\sigma_1, \ldots, \sigma_m\}$ with $\rho(\widehat{G}_1, \sigma_j) = \{\widehat{G}'_1\}$, we have that $\widehat{G}'_1$ is covered by $\bigcup_{2 \leq i \leq n} \rho(\widehat{G}'_i, \sigma_j)$ (since $\widehat{G}_1$ is covered by $\{\widehat{G}_2, \ldots, \widehat{G}_n\}$), meaning that $\widehat{T} \cong \widehat{T}'$, with $\widehat{T}' \equiv \left(\bigcup_{2 \leq i \leq n, 1 \leq j \leq m} \rho(\widehat{G}_i, \sigma_j)\right)^f$. Thus, $\widehat{S}'$ has exactly the same legal move options as $\widehat{S}$, so $\widehat{S} \cong \widehat{S}'$, and since both superposed game states have the same labelling $\widehat{S} \equiv \widehat{S}'$. □

Definition 3.18. A dead-ending ruleset $\mathcal{R}$ is said to be *consistent* if, given two game states $\widehat{G}$ and $\widehat{H}$ such that $\mathrm{P}(\widehat{G}) \subseteq \mathrm{P}(\widehat{H})$, for all $\sigma \in \mathrm{P}(\widehat{G})$, with $\rho(\widehat{G}, \sigma) = \{\widehat{G}'\}$ and $\rho(\widehat{H}, \sigma) = \{\widehat{H}'\}$, it holds that $\mathrm{P}(\widehat{G}') \subseteq \mathrm{P}(\widehat{H}')$.

This means that, given $\mathrm{P}(\widehat{G}) \subseteq \mathrm{P}(\widehat{H})$, whenever a move σ causes some moves σ' (possibly including σ) to become illegal in the follower $\widehat{H}'$ of $\widehat{H}$, then all of these σ' must also become illegal in $\widehat{G}'$ (the follower of $\widehat{G}$). Note that many dead-ending rulesets, including Hackenbush, Nim-Subtract, and Nim-Decrease, also have the consistency property. In the case of Hackenbush, whenever the removal of an edge σ also causes the removal of some other edges σ', then these σ' must also be removed in a game state that does not contain any additional edges that could support them. A similar argument holds for Nim-Decrease, with tokens requiring support from tokens at lower levels. In the case of Nim-Subtract, if multiple tokens are removed from a heap, then removing this same number of tokens from a smaller heap cannot result in a larger heap. However, it is possible to design a ruleset that is dead-ending, but not consistent. For example, a variant of Nim-Subtract with a move counter, which ends after a set number of moves have been played. This ruleset is still dead-ending. However, if we have a game state $\widehat{G}$ of a single heap of size 2 or greater that ends after 2 turns, and another game state $\widehat{H}$ of a single larger heap that ends after 1 turn, then $\mathrm{P}(\widehat{G}) \subseteq \mathrm{P}(\widehat{H})$, but if we apply the move to remove one token to both game states, we end up with $\widehat{G}'$ that still has a legal move, and $\widehat{H}'$ that does not have any legal move, so $\mathrm{P}(\widehat{G}') \not\subseteq \mathrm{P}(\widehat{H}')$.

Definition 3.19. Given two game states $\widehat{G}$ and $\widehat{H}$, we say that $\widehat{G}$ is *weakly covered* by $\widehat{H}$ if all moves $\sigma \in \Sigma$ that are legal on $\widehat{G}$ are also legal on $\widehat{H}$, i.e., $\mathrm{P}(\widehat{G}) \subseteq \mathrm{P}(\widehat{H})$.

In the example of Nim-Subtract with a move counter that we have just discussed, a smaller heap may be weakly covered, but depending on the move counter it may not be covered. For example, if the immediate follower of the covered realisation has move options, while the immediate follower of the covering realisation does not have any move options.

Theorem 3.20. Given a consistent ruleset $\mathcal{R}$ and set of game states $\{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_n\}$ such that $\widehat{G}_1$ is weakly covered by some $\widehat{G}_i, 2 \leq i \leq n$, then $\widehat{G}_1$ is covered by $\{\widehat{G}_2, \ldots, \widehat{G}_n\}$.

Proof. Again, we use induction on the birthday of game state $\widehat{G}_1$. Assume that $\widehat{G}_1$ is weakly covered by $\widehat{G}_i \in \{\widehat{G}_2, \ldots, \widehat{G}_n\}$. If $\widehat{G}_1$ is the empty game state, then it is trivially both weakly covered by $\widehat{G}_i$ and covered by any non-empty set of other realisations $\{\widehat{G}_2, \ldots, \widehat{G}_n\}$, so the theorem holds. Now, for our induction hypothesis, assume that the theorem holds for all game states that have a birthday smaller than some $\widehat{G}_1$'s

birthday. We want to show that for all $\sigma \in \mathrm{P}(\widehat{G}_1)$ with $\rho(\widehat{G}_1, \sigma) = \{\widehat{G}'_1\}$, there exists some $\widehat{G}_i, 2 \leq i \leq n$ on which σ is also legal, and $\widehat{G}'_1$ is covered by $\bigcup_{2 \leq i \leq n} \rho(\widehat{G}_i, \sigma)$. Since $\widehat{G}_1$ is weakly covered, there must exist some $\widehat{G}_i, 2 \leq i \leq n$ such that $\mathrm{P}(\widehat{G}_1) \subseteq \mathrm{P}(\widehat{G}_i)$, meaning that any move that is legal on $\widehat{G}_1$ is also legal on $\widehat{G}_i$. Also, since all $\widehat{G}'_1$ have a smaller birthday than $\widehat{G}_1$, it will suffice to show that all $\widehat{G}'_1$ are weakly covered by some $\widehat{G}'_j \in \bigcup_{2 \leq j \leq n} \rho(\widehat{G}_j, \sigma)$ for $\sigma \in \mathrm{P}(\widehat{G}_1)$, by our induction hypothesis. Now, if a move $\sigma \in \Sigma$ causes some moves σ' to become illegal in $\widehat{G}_i$, meaning that all $\sigma' \notin \mathrm{P}(\widehat{G}'_i)$, with $\rho(\widehat{G}_i, \sigma) = \{\widehat{G}'_i\}$, then these moves must also be illegal in $\widehat{G}'_1$, because $\widehat{G}_1$ is weakly covered by $\widehat{G}_i$ and $\mathcal{R}$ is consistent. This proves that $\mathrm{P}(\widehat{G}'_1) \subseteq \mathrm{P}(\widehat{G}'_i)$, so $\widehat{G}'_1$ is weakly covered by $\widehat{G}'_i$. □

Theorem 3.20 tells us that certain realisations can safely be left out of a superposed game state without having to recursively check for coverage. As we have mentioned before, this theorem does not apply to quantum-inspired variations of other combinatorial games in general, because not all rulesets have the dead-ending property. We used Maze as an example. Not all dead-ending rulesets are necessarily consistent either, as we have also discussed.

3.4 Quantum Hackenbush

Let us now gain some intuition for how Quantum Hackenbush works by looking at some examples, and deriving some basic results. Given a Hackenbush position G, we can construct a width-1 Quantum Hackenbush superposed game state $\{\widehat{G}\}^f$, under some flavour f. We have already shown a small example labelled game tree starting from a classical game state in Figure 1.2.

To start, notice that, if legal, applying the same width-2 move $\{\sigma_1, \sigma_2\}$ twice effectively removes both edges "classically". The result is as if the two edges σ_1 and σ_2 were removed one by one, in nondeterministic order. All realisations from the original game state that did not contain both σ_1 and σ_2 collapse, and we are left with a superposed game state where no realisation contains σ_1 or σ_2.

We also have that, for n grounded edges of the same colour, a player will always be able to play a total of n moves, under any flavour, by following the strategy that we now define.

Definition 3.21. We say that a player follows *chain-extension* if they always play a width-2 move whenever one is legal, and if they always include exactly one classical move that they have not included in any previously played superposed move (a "new" classical move), whenever this is possible.

If only an unsuperposed move is legal, then the player can play this move (which can only be on their last turn). If the only superposed move that is legal is one that includes more than one "new" classical move (which will always be on their first turn), or no "new" classical moves at all (which can only be on their last turn), then the player can also play this move.

Take for example, for a superposed game state $\{\widehat{G}\}^f$ under any flavour f, with $\mathrm{P}(\widehat{G}) = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, and where each move corresponds to a grounded blue edge. A 4-move chain-extension sequence for Left would be $\{\sigma_1, \sigma_2\}$–$\{\sigma_2, \sigma_3\}$–$\{\sigma_3, \sigma_4\}$–$\{\sigma_1, \sigma_4\}$.

For quantum-inspired combinatorial games in general, a weakly covered realisation cannot be left out of a superposed game state without changing its legal superposed move options. However, this *is* possible for Quantum Hackenbush superposed game states under flavour $\mathcal{A}$, $\mathcal{B}$, or $\mathcal{D}$, because Hackenbush is consistent (Theorem 3.20).

From Section 4 onwards, we will study a few classes of restricted Hackenbush game states in-depth. For such classes of game states, it is simpler to analyse certain properties, such as the outcome class or value. Analysing such classes of restricted game states can

reveal algebraic structures, which may be used for the purpose of inspiring the discovery of similar structures in classes of more general game states.

4 Results for Stalks game states

The first class of restricted HACKENBUSH positions that we study, we call *Stalks*. A position in this class consists of several independent (non-connected) acyclic paths of $\ell \geq 1$ edges (stalks), each originating from the ground.

Let us define a subclass of Stalks called *Single Stalk*, which are the positions containing no more than one stalk. Classically, the value of any stalk can easily be read off by interpreting the stalk as a *HACKENBUSH string*. A simple rule for determining the value of a HACKENBUSH string was found in [9], and it is given as follows.

Theorem 4.1. The value of any HACKENBUSH string can be determined by first, starting from the ground node, counting the number of edges m until the first colour change. If the grounded edge is blue then this number has a positive sign (m), and if it is red the number has a negative sign $(-m)$. Then, for every nth edge after the first colour change (the first edge of a different colour has $n = 1$), add $\frac{1}{2^n}$ for a blue edge, and subtract $\frac{1}{2^n}$ for a red edge.

Following this theorem, the value of a stalk G of length ℓ has the form

$$G = (\pm m) + \left(\pm\frac{1}{2^1}\right) + \ldots + \left(\pm\frac{1}{2^{\ell-m}}\right).$$

In [9], it was also proven that the optimal move on a HACKENBUSH string is given as follows.

Theorem 4.2. When playing on a HACKENBUSH string, the optimal move for either player is to remove the edge of their colour which is farthest from the ground.

This also makes sense knowing how the value of a HACKENBUSH string G can be determined (Theorem 4.1), because if another move were to be optimal, the edges of the player's colour farther from the ground would not contribute to the value of G.

Let us define another subclass of Stalks, which we will call *Hollyhocks* (the name was first coined in [1]). A hollyhock is a stalk, with the additional restriction that if the grounded edge is blue then all other edges above it must be red, and vice versa. We call a hollyhock whose grounded edge is blue a *blue hollyhock*, and a hollyhock whose grounded edge is red a *red hollyhock*. Figure 4.1 shows a general blue and red hollyhock. The value of a blue hollyhock is $\frac{1}{2^{\ell-1}}$, where ℓ is the length (number of edges) of the hollyhock. A red hollyhock has value $-\frac{1}{2^{\ell-1}}$. Proofs can be constructed in a way similar to Example 2.18, or by using the rule from Theorem 4.1. In combinatorial game theory, the value of a position consisting of several independent components is fully determined by the values of the individual components [3]. Thus, a Hollyhocks position can be written as a disjunctive sum

$$G \cong H_\ell + \ldots + H_{\ell'} - H_k - \ldots - H_{k'},$$

where H_ℓ denotes a blue hollyhock of length ℓ, and $-H_k$ denotes a red hollyhock of length k. Note that it is always allowed to reorder hollyhocks, because the disjunctive sum is commutative.

(a) A blue hollyhock of length ℓ.

(b) A red hollyhock of length k.

Figure 4.1: Blue and red hollyhocks.

Additionally, let us define *Short Hollyhocks* to be the subclass of Hollyhocks positions consisting only of hollyhocks of length $\ell \leq 2$. Short Hollyhocks positions are more commonly called sums of *halves*, and *wholes*. The terms "half" and "whole" refer to the values of such components, because $H_2 = \frac{1}{2}$ and $H_1 = 1$ (and of course $-H_2 = -\frac{1}{2}$ and $-H_1 = -1$).

Finally, let us also define an even more restricted subclass of Hollyhocks, which we name *Blue Meadow*. A Blue Meadow position consists of n blue hollyhocks H_2 of length 2, along with m red "weeds" $-H_1$ of length 1. It can be written as

$$G \cong n \cdot H_2 - m \cdot H_1.$$

The position in Figure 1.1 is a Blue Meadow position consisting of two blue hollyhocks and a single red weed.

4.1 A program for analysing Quantum Hackenbush

In the following subsections, we show multiple tables containing outcome classes or values of certain Hackenbush classical game states under different flavours. These outcome classes and values were calculated by a program for analysing Quantum Hackenbush superposed game state trees, which we named `quantum-hackenbush-suite`[4]. This program was written in C++, and significant parts of its implementation were inspired by `CGSynch`[5], which is a program for analysing combinatorial and synchronised games [10].

`quantum-hackenbush-suite` takes as an input a description of a Hackenbush position G, and constructs a width-1 superposed game state $\widehat{S} \equiv \{\widehat{G}\}^f$ under a given flavour f based on this. The program repeatedly applies simplifications to the short game S obtained by removing the labelling from $\widehat{S}$. The program uses domination and reversibility until it reaches a canonical form, which it then outputs. The program can also determine the birthday or outcome class of S. For these properties, the canonical form does not need to be determined first. The outcome class in particular can be determined efficiently, because only alternating turns need to be considered when searching the tree, rather than needing to consider repeated turns by the same player, which is necessary for determining the value.

To improve performance, the program uses three "databases" (hash maps) to store intermediate results, preventing the same computation from being performed multiple times. It stores intermediate results for Hackenbush game states, the superposed game states made up of these, and the short games obtained by unlabelling superposed game states. Specifically, it keeps references to the children of any game state, superposed game state, or short game. For short games it also stores the result of operations such as taking the inverse, comparing to another short game, or determining the canonical form. To further improve performance, the program uses an efficient representation for Short Hollyhocks and Single Stalk game states. For such game states containing k edges, the program uses an array representation that scales with $\mathcal{O}(k)$, which is more efficient (in terms of both space and time) than the $\mathcal{O}(k^2)$ adjacency matrix representation it uses for general game states.

4.2 Values of single stalks

It turns out that the rule from Theorem 4.1 can be applied directly to any Single Stalk game state under flavour $f \in \{\mathcal{B}, \mathcal{C}', \mathcal{D}\}$, which we will prove below. For flavour $\mathcal{A}$, we will prove that a simple modification of the rule can be used. Let us start with a preliminary theorem.

[4] `quantum-hackenbush-suite`'s source code can be found at https://github.com/jpleunes/quantum-hackenbush-suite.

[5] `CGSynch`'s source code can be found at https://github.com/xlenstra/CGSynch.

Theorem 4.3. Given a single-stalk position G and a flavour $f \in \{\mathcal{A}, \mathcal{B}, \mathcal{D}\}$, any follower $\{\widehat{G}'_1, \ldots, \widehat{G}'_n\}^f$ of $\{\widehat{G}\}^f$ is equivalent to $\{\widehat{G}'_j\}^f$, where $\widehat{G}'_j \in \{\widehat{G}'_1, \ldots, \widehat{G}'_n\}$ is the realisation consisting of the tallest stalk.

Proof. Any realisation consisting of a smaller stalk is weakly covered by $\widehat{G}'_j$, so it is covered, according to Theorem 3.20. Since these realisations are covered, we have that $\{\widehat{G}'_1, \ldots, \widehat{G}'_n\}^f \equiv \{\widehat{G}'_j\}^f$, according to Theorem 3.17. □

We can now prove that the classical rule can also be used under flavour $f \in \{\mathcal{B}, \mathcal{C}', \mathcal{D}\}$.

Theorem 4.4. Under flavour $f \in \{\mathcal{B}, \mathcal{C}', \mathcal{D}\}$, any single-stalk position G has the same value as in the classical case, i.e., $\{\widehat{G}\}^f = G$.

Proof. Under flavour $\mathcal{D}$, using Theorem 4.3 we know that any superposed move leads to a superposed game state that is equivalent to the classical game state where only the classical move corresponding to the edge farthest from the ground was played. This means that superposed moves do not introduce any additional options.

Under flavour $\mathcal{C}'$, the player can always play a superposed move containing the edge σ farthest from the ground which exists in any realisation, or if there exists exactly one edge across all realisations, then the player can classically remove this edge. Playing a superposed move containing edge σ leads to a superposed game state containing a realisation which contains all edges below σ. Thus, effectively, either player has exactly one move corresponding to each edge of their colour, just as in the classical case (this follows from Theorem 4.2, which states that it is optimal for the player to not cause any edges of their colour to fly away).

Under flavour $\mathcal{B}$, Theorem 4.3 again tells us that a superposed move leads to an equivalent classical game state, where the edge from the superposed move that is farthest from the ground has been removed. This means that a player is effectively able to play any classical move, except that they cannot remove the edge of their colour that is closest to the ground if there exist multiple edges of their colour. This does not affect the value of any classical game state that we encounter, however, because when playing on a HACKENBUSH string the optimal move is always to remove the edge of your colour that is farthest from the ground (Theorem 4.2). When there exists exactly one edge of a player's colour, then the player can play a classical move removing this edge, according to flavour $\mathcal{B}$. □

We can also prove that a small modification of the classical rule can be used under flavour $\mathcal{A}$.

Theorem 4.5. Under flavour $\mathcal{A}$, the value of a HACKENBUSH string can be determined by following Theorem 4.1, but ignoring the first blue edge, if it exists, and the first red edge, if it exists.

Proof. The proof is the same as for Theorem 4.4, flavour $\mathcal{B}$, except that when there exists exactly one edge of a player's colour, then the player does not have any move option. This means that a player will never be able to move to a classical game state where the edge closest to the ground, of their colour, has been removed. Thus, these edges do not contribute to the value, and can be ignored when following the rule from Theorem 4.1. □

Under flavour $\mathcal{C}$, it is possible to construct classical game states that are not a number. Figure 4.2 shows an example. Values of single stalks of different lengths, encoded as bit strings, under flavour $\mathcal{C}$ are given in Table 4.1. Certain patterns appear in the non-number values that arise. For example, the HACKENBUSH string G that encodes the decimal 4 always seems to have value $(\ell - 3)*$, where $\ell \geq 4$ is the length of G. Also, the G that encodes the decimal $2^{\ell-1} - 4$ always appears to be the last non-number value in a column, with value $\frac{1}{2^{\ell-4}}*$. It is also interesting to note that any non-number canonical form of

a G of length ℓ seems to appear as an immediate follower of some other non-number canonical form of a G' of length $\ell+1$, which may imply that there exists some recurrence relation for the non-number values shown in the table.

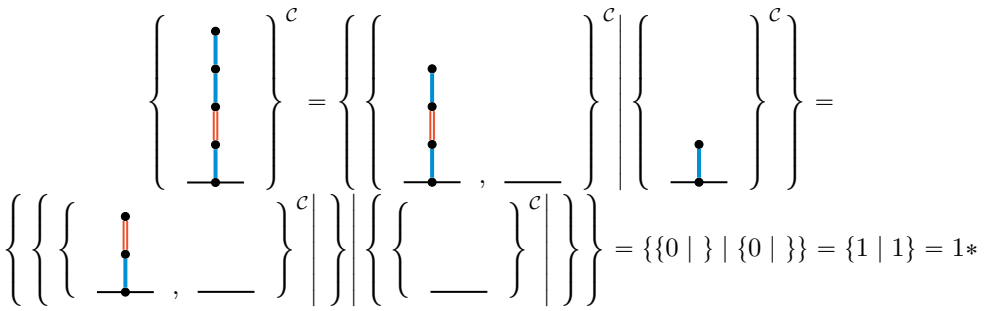


Figure 4.2: A classical stalk that is not a number under flavour $\mathcal{C}$. Only non-dominated options are shown for brevity.

	4	5	6	7
$4_{10} = 0000100_2$	$1*$	$2*$	$3*$	$4*$
$8_{10} = 0001000_2$		$\{2 \mid 1\}$	$\{3 \mid 2\}$	$\{4 \mid 3\}$
$9_{10} = 0001001_2$		$\{1 \mid 1, 1*\}$	$\{2 \mid 2, 2*\}$	$\{3 \mid 3, 3*\}$
$11_{10} = 0001011_2$		$\frac{1}{2}*$	$1\frac{1}{2}*$	$2\frac{1}{2}*$
$12_{10} = 0001100_2$		$\frac{1}{2}*$	$1\frac{1}{2}*$	$2\frac{1}{2}*$
$16_{10} = 0010000_2$			$\{3 \mid 1\}$	$\{4 \mid 2\}$
$17_{10} = 0010001_2$			$\{2 \mid 1, \{2 \mid 1\}\}$	$\{3 \mid 2, \{3 \mid 2\}\}$
$18_{10} = 0010010_2$			$\{1\frac{1}{2} \mid 1, 1*\}$	$\{2\frac{1}{2} \mid 2, 2*\}$
$19_{10} = 0010011_2$			$\{1 \mid 1, \{1 \mid 1, 1*\}\}$	$\{2 \mid 2, \{2 \mid 2, 2*\}\}$
$20_{10} = 0010100_2$			$\frac{3}{4}*$	$1\frac{3}{4}*$
$22_{10} = 0010110_2$			$\{\frac{1}{2}, \frac{1}{2}* \mid \frac{1}{2}\}$	$\{1\frac{1}{2}, 1\frac{1}{2}* \mid 1\frac{1}{2}\}$
$23_{10} = 0010111_2$			$\{\frac{1}{2} \mid \frac{1}{4}\}$	$\{1\frac{1}{2} \mid 1\frac{1}{4}\}$
$24_{10} = 0011000_2$			$\{\frac{3}{4} \mid \frac{1}{2}\}$	$\{1\frac{3}{4} \mid 1\frac{1}{2}\}$
$25_{10} = 0011001_2$			$\{\frac{1}{2} \mid \frac{1}{2}, \frac{1}{2}*\}$	$\{1\frac{1}{2} \mid 1\frac{1}{2}, 1\frac{1}{2}*\}$
$27_{10} = 0011011_2$			$\frac{1}{4}*$	$1\frac{1}{4}*$
$28_{10} = 0011100_2$			$\frac{1}{4}*$	$1\frac{1}{4}*$
$32_{10} = 0100000_2$				$\{4 \mid 1\}$
$33_{10} = 0100001_2$				$\{3 \mid 1, \{3 \mid 1\}\}$
$34_{10} = 0100010_2$				$\{2\frac{1}{2} \mid 1, \{2 \mid 1\}\}$
$35_{10} = 0100011_2$				$\{2 \mid 1, \{2 \mid 1, \{2 \mid 1\}\}\}$
$36_{10} = 0100100_2$				$\{1\frac{3}{4} \mid 1, 1*\}$
$37_{10} = 0100101_2$				$\{1\frac{1}{2} \mid 1, 1*, \{1\frac{1}{2} \mid 1, 1*\}\}$
$38_{10} = 0100110_2$				$\{\{1 \mid 1*\} \mid 1, \{1 \mid 1, 1*\}\}$
$39_{10} = 0100111_2$				$\{1 \mid 1, \{1 \mid 1, \{1 \mid 1, 1*\}\}\}$
$40_{10} = 0101000_2$				$\{\frac{7}{8} \mid \frac{3}{4}\}$
$41_{10} = 0101001_2$				$\{\frac{3}{4} \mid \frac{3}{4}, \frac{3}{4}*\}$
$43_{10} = 0101011_2$				$\frac{5}{8}*$
$44_{10} = 0101100_2$				$\{\frac{1}{2}, \{\frac{1}{2}, \frac{1}{2}* \mid \frac{1}{2}\} \mid \frac{1}{2}\}$
$45_{10} = 0101101_2$				$\{\frac{1}{2}, \frac{1}{2}* \mid \frac{3}{8}\}$
$46_{10} = 0101110_2$				$\{\frac{1}{2}, \{\frac{1}{2} \mid \frac{1}{4}\} \mid \frac{1}{4}\}$
$47_{10} = 0101111_2$				$\{\frac{1}{2} \mid \frac{1}{8}\}$
$48_{10} = 0110000_2$				$\{\frac{7}{8} \mid \frac{1}{2}\}$
$49_{10} = 0110001_2$				$\{\frac{3}{4} \mid \frac{1}{2}, \{\frac{3}{4} \mid \frac{1}{2}\}\}$
$50_{10} = 0110010_2$				$\{\frac{5}{8} \mid \frac{1}{2}, \frac{1}{2}*\}$
$51_{10} = 0110011_2$				$\{\frac{1}{2} \mid \frac{1}{2}, \{\frac{1}{2} \mid \frac{1}{2}, \frac{1}{2}*\}\}$
$52_{10} = 0110100_2$				$\frac{3}{8}*$
$54_{10} = 0110110_2$				$\{\frac{1}{4}, \frac{1}{4}* \mid \frac{1}{4}\}$
$55_{10} = 0110111_2$				$\{\frac{1}{4} \mid \frac{1}{8}\}$
$56_{10} = 0111000_2$				$\{\frac{3}{8} \mid \frac{1}{4}\}$
$57_{10} = 0111001_2$				$\{\frac{1}{4} \mid \frac{1}{4}, \frac{1}{4}*\}$
$59_{10} = 0111011_2$				$\frac{1}{8}*$
$60_{10} = 0111100_2$				$\frac{1}{8}*$

Table 4.1: Canonical form values of HACKENBUSH strings of different lengths (horizontal) that represent different binary encodings (vertical) under flavour $\mathcal{C}$, for any finite or unbounded w. The grounded edge corresponds to the leftmost bit, with blue edges corresponding to 0s and red edges corresponding to 1s. For any length, we do not consider binary strings that have 1 as the leftmost bit, because the value of each of these HACKENBUSH strings can be obtained by taking the negative of the value of the inverted binary string. Only values that differ from the classical case are shown.

4.3 Outcome classes can differ between flavours

Let us consider Blue Meadow positions, which are of the form $G \cong n \cdot H_2 - m \cdot H_1$. In classical HACKENBUSH, the value (and thus also the outcome class) of a Blue Meadow position can easily be determined by considering each stalk separately. If $n = 2m$, then

the position is a $\mathscr{P}$-position, since $2m \cdot H_2 - m \cdot H_1 = 2m \cdot \frac{1}{2} - m \cdot 1 = 0$. Otherwise, if $n < 2m$ the position is an $\mathscr{R}$-position, or if $n > 2m$ it is an $\mathscr{L}$-position.

However, in Quamtum Hackenbush superposed moves enable interactions between the stalks, meaning that the stalks are no longer non-interacting components, so they cannot be analysed separately anymore. This can result in a classical game state receiving a different outcome class as compared to the classical case. In Table 4.2 we see the outcome classes for small values of n and m for the different flavours $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{C}'$, and $\mathcal{D}$, with a maximum move width of $w = 2$. Except for flavours $\mathcal{B}$ and $\mathcal{C}'$, all flavours are pairwise different, in terms of the outcome classes that are shown.

	0	1	2	3	4	5
0	$\mathscr{P}$	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
1	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
2	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
3	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
4	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
5	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
6	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
7	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$			
8						

(a) Flavour $\mathcal{A}$.

	0	1	2	3	4	5
0	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
1	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
2	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
3	$\mathscr{L}$	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
4	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
5	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
6	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
7	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$			
8						

(b) Flavour $\mathcal{B}$.

	0	1	2	3	4	5
0	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
1	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
2	$\mathscr{L}$	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
3	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
4	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
5	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
6	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
7	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$			
8						

(c) Flavour $\mathcal{C}$.

	0	1	2	3	4	5
0	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
1	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
2	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
3	$\mathscr{L}$	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
4	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
5	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
6	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
7	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$			
8						

(d) Flavour $\mathcal{C}'$.

	0	1	2	3	4	5
0	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
1	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
2	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
3	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
4	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
5	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
6	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
7	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{R}$	$\mathscr{R}$	$\mathscr{R}$
8	$\mathscr{L}$	$\mathscr{L}$	$\mathscr{L}$			

(e) Flavour $\mathcal{D}$.

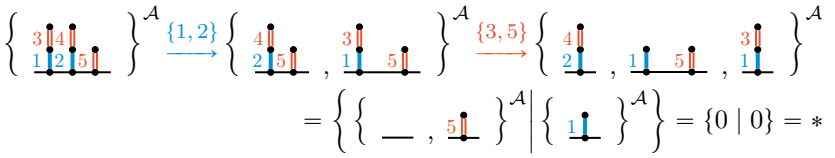
Table 4.2: Outcome classes of sums of n blue halves (vertical) and m red wholes (horizontal), under all flavours, with maximum superposed move width $w = 2$. Marked in blue are the outcome classes that differ from classical Hackenbush.

4.4 Short Hollyhocks game states can have $*$-followers

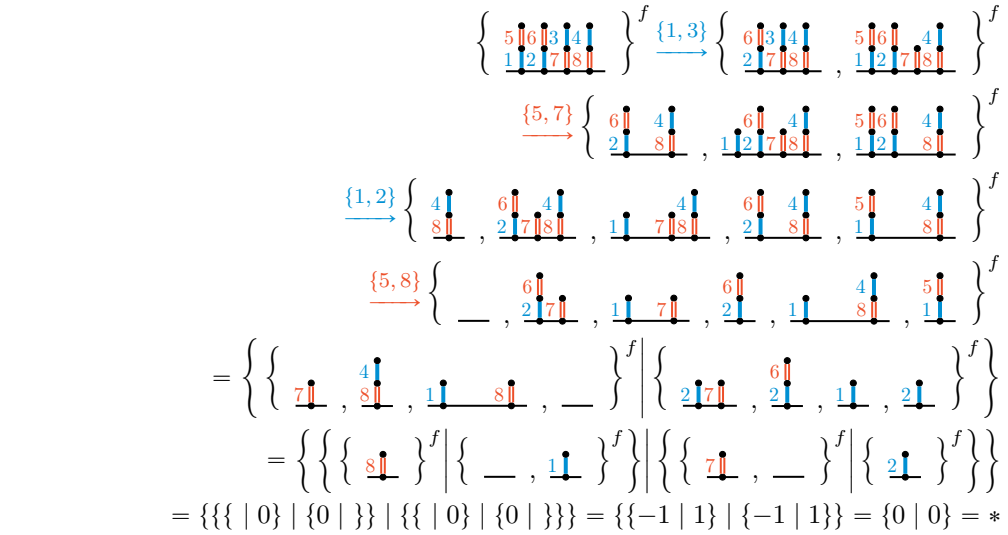
In classical Hackenbush, there are no $\mathscr{N}$-positions. This can be shown in a way similar to that in [1], by noticing that removing a blue edge from a position G causes some subgraph to be removed, which can in turn be viewed as a Hackenbush position G' with a single grounded blue edge. This means that the Left option G^L of G has value $G^L = G - G' < G$, and a symmetrical inequality can be shown for Right options. Thus,

since $G^L < G < G^R$ for all Left options G^L and Right options G^R of G, the simplest number theorem (Theorem 2.20) can be applied, so all HACKENBUSH positions must be a number.

In QUANTUM HACKENBUSH, however, it is possible to encounter positions with value $*$ (which have outcome class $\mathcal{N}$) in the superposed game state tree. Figure 4.3 shows, for every flavour, an example superposed game state with value $*$, along with how it can be reached from a classical Short Hollyhocks game state.



(a) Reaching a $*$ under flavour $\mathcal{A}$. From the $*$, Left can play $\{1,2\}$ to move to 0, and Right can play $\{3,5\}$ to move to 0.



(b) Reaching a $*$ under flavour $f \in \{\mathcal{B}, \mathcal{C}'\}$. From the $*$, Left can play $\{1,4\}$ to move to 0, and Right can play $\{6,7\}$ to move to 0.

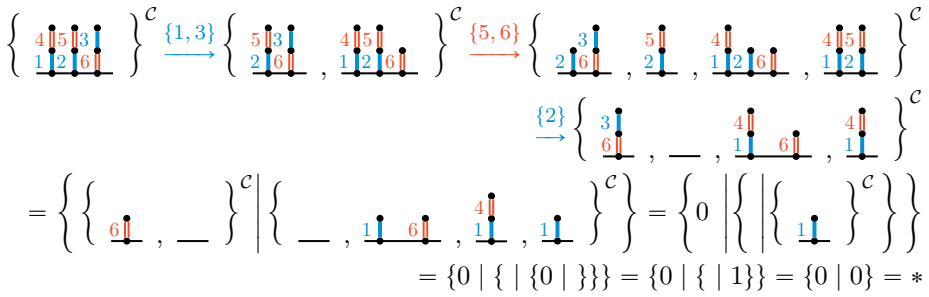
Figure 4.3: A path leading to a superposed game state with value $*$, starting from some classical Short Hollyhocks game state, under every flavour. Only non-dominated options are shown for brevity. (Continues on next page.)

4.5 Values of Blue Meadow game states

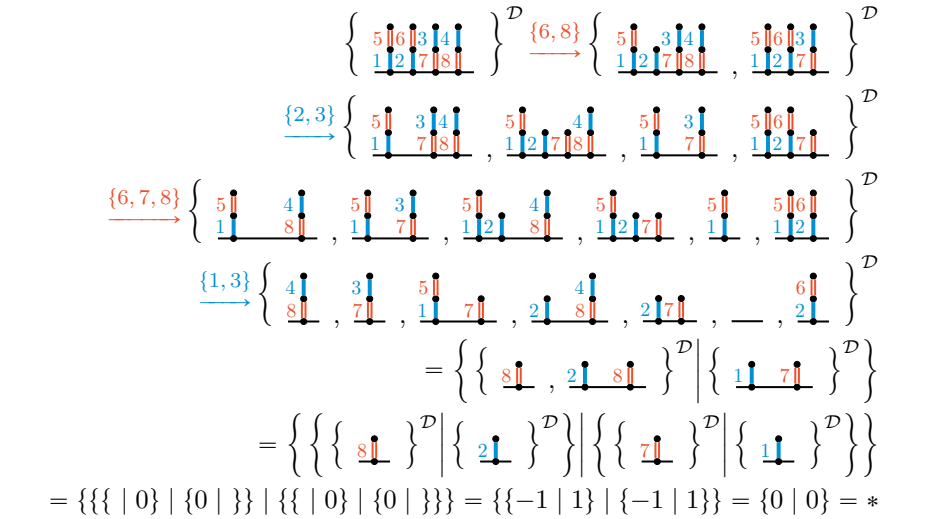
Let us again consider Blue Meadow positions $G \cong n \cdot H_2 - m \cdot H_1$. We know that classically $G = n \cdot \frac{1}{2} - m$. However, $\{\widehat{G}\}^f = G$ does not hold in general. Canonical form values of small Blue Meadow classical game states under the different flavours are shown in Table 4.3.

We were not able to calculate as many values for unbounded w compared to $w = 2$, because these computations are move intensive (in terms of both time and space), due to the increased number of superposed move options. It is interesting to note that flavours $\mathcal{B}$, $\mathcal{C}'$, and $\mathcal{D}$ seem to have identical values when considering unbounded w. For $w = 2$, flavours $\mathcal{B}$ and $\mathcal{C}'$ also seem to have identical values, except for $3 \cdot H_2$ ($n = 3$ and $m = 0$).

Theorem 4.6. The value of a Blue Meadow classical game state under any flavour f can never be greater than its classical value, except under flavour $\mathcal{A}$ when there is



(c) Reaching a $*$ under flavour $\mathcal{C}$. From the $*$, Left can play $\{1,3\}$ to move to 0, and Right can play $\{4,6\}$ to move to 0.



(d) Reaching a $*$ under flavour $\mathcal{D}$. From $*$, Left can play $\{4\}$ to move to 0, and Right can play $\{5\}$ to move to 0.

Figure 4.3: A path leading to a superposed game state with value $*$, starting from some Short Hollyhocks classical game state, under every flavour. Only non-dominated options are shown for brevity. (Continued.)

exactly one red hollyhock, i.e., $\{-H_1\}^{\mathcal{A}} = 0 > -H_1 = -1$.

Proof. The ability to play superposed moves can only be advantageous for Right (not for Left), because this way he is, in some cases, able to "avoid" his edges from being removed in all realisations, because Left may not (within a single move) be able to remove a corresponding blue edge in all realisations. Left, on the other hand, will not be able to play a total number of moves that is greater than in the classical case, because each turn she must remove one blue edge from each realisation.

Under flavour $\mathcal{A}$, any game state that has at most one edge of either colour has value 0, because neither player will be able to move. This is not necessarily true for any other flavour, because the players could play a classical move on the edge of their colour. The only exception for Theorem 4.6 is $\{-H_1\}^{\mathcal{A}}$, because this is the only Blue Meadow game state containing exactly a single red edge, which classically has a value smaller than 0, i.e., $-H_1 = -1 < 0$. □

Finally, under flavours $\mathcal{B}$, $\mathcal{C}'$, and $\mathcal{D}$, with unbounded w, all canonical form values seem to have birthday m if $n = 0$, and birthday $n + \max(1, m) - 1$ otherwise.

	0	1	2	3	4	5
0	0	0	-2	-3	-4	-5
1	0	-2	-3	-4	-5	-6
2	$\frac{3}{4}$	$\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
3	1	$\frac{3}{8}$	$-1\frac{1}{8}$	$-2\frac{1}{8}$	$-3\frac{1}{8}$	$-4\frac{1}{8}$
4	$1\frac{1}{2}$	$\frac{3}{4}$	$-\frac{1}{2}$	$-1\frac{1}{2}$		
5	2	$1\frac{1}{4}$				

(a) Flavour $\mathcal{A}$ with $w = 2$.

	0	1	2	3	4	5
0	0	0	-2	-3	-4	-5
1	0	-2	-3	-4	-5	-6
2	$\frac{3}{4}$	$\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
3	$\frac{7}{8}$	$\frac{3}{8}$	$-1\frac{1}{8}$	$-2\frac{1}{8}$		
4	$\frac{15}{16}$	$\frac{7}{16}$				
5						

(b) Flavour $\mathcal{A}$ with unbounded w.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	$\frac{3}{4}$	$-\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	$\frac{7}{8}$	0	-1	-2	-3	-4
4	$1\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$		
5	$1\frac{3}{4}$	$\frac{3}{4}$				

(c) Flavour $\mathcal{B}$ with $w = 2$.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	$\frac{3}{4}$	$-\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	$\frac{7}{8}$	$-\frac{1}{8}$	$-1\frac{1}{8}$	$-2\frac{1}{8}$		
4	$\frac{15}{16}$	$-\frac{1}{16}$				
5						

(d) Flavour $\mathcal{B}$ with unbounded w.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	1	0	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	$1\frac{1}{2}$	$\frac{1}{2}$	-1	-2	-3	-4
4	$1\frac{3}{4}$	$\frac{3}{4}$	$-\frac{1}{2}$	$-1\frac{1}{2}$		
5	$2\frac{1}{4}$	$1\frac{1}{4}$				

(e) Flavour $\mathcal{C}$ with $w = 2$.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	1	0	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	$1\frac{1}{4}$	$\frac{1}{4}$	$-1\frac{1}{8}$	$-2\frac{1}{8}$		
4	$1\frac{3}{8}$	$\frac{3}{8}$				
5						

(f) Flavour $\mathcal{C}$ with unbounded w.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	$\frac{3}{4}$	$-\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	1	0	-1	-2	-3	-4
4	$1\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$		
5	$1\frac{3}{4}$	$\frac{3}{4}$				

(g) Flavour $\mathcal{C}'$ with $w = 2$.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	$\frac{3}{4}$	$-\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	$\frac{7}{8}$	$-\frac{1}{8}$	$-1\frac{1}{8}$	$-2\frac{1}{8}$		
4	$\frac{15}{16}$	$-\frac{1}{16}$				
5						

(h) Flavour $\mathcal{C}'$ with unbounded w.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	$\frac{3}{4}$	$-\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	$1\frac{1}{4}$	$\frac{1}{4}$	$-\frac{3}{4}$	$-1\frac{3}{4}$	$-2\frac{3}{4}$	$-3\frac{3}{4}$
4	$1\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$		
5	2	1				

(i) Flavour $\mathcal{D}$ with $w = 2$.

	0	1	2	3	4	5
0	0	-1	-2	-3	-4	-5
1	$\frac{1}{2}$	$-\frac{1}{2}$	$-1\frac{1}{2}$	$-2\frac{1}{2}$	$-3\frac{1}{2}$	$-4\frac{1}{2}$
2	$\frac{3}{4}$	$-\frac{1}{4}$	$-1\frac{1}{4}$	$-2\frac{1}{4}$	$-3\frac{1}{4}$	$-4\frac{1}{4}$
3	$\frac{7}{8}$	$-\frac{1}{8}$	$-1\frac{1}{8}$	$-2\frac{1}{8}$		
4	$\frac{15}{16}$	$-\frac{1}{16}$				
5						

(j) Flavour $\mathcal{D}$ with unbounded w.

Table 4.3: Canonical form values of sums of n blue halves (vertical) and m red wholes (horizontal), under all flavours. Marked in blue are the values that differ for unbounded maximum superposed move width w, as opposed to $w = 2$.

5 Circus Tent game states are numbers

Another class of restricted HACKENBUSH positions that we study is that of *Circus Tent* positions. A position in this class consists of $n \geq 2$ legs, which are each a single blue edge with a single red edge on top. The tops of all legs are connected to each other. A general circus tent is shown in Figure 5.1. Let us denote an n-leg circus tent by C_n.

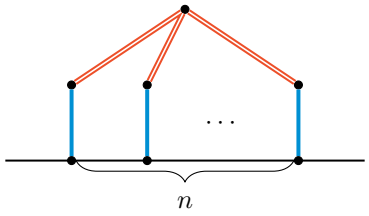


Figure 5.1: A circus tent with n legs.

Circus Tent is a subclass of the more general Redwood Furniture positions. Redwood Furniture is defined as all positions where no grounded edge is red, where all blue edges are grounded, each with its non-grounded side connected to a unique red edge, and where all edges are (indirectly) connected. A useful theorem for determining the value of a Redwood Furniture position is given as follows.

Theorem 5.1. A Redwood Furniture position G has value $\{0 \mid G^R\}$, where G^R is the smallest Right option of G.

A proof for this theorem can be found in [1]. We can use this theorem to prove the value of an arbitrary circus tent.

Theorem 5.2. A circus tent has value 1, i.e., $C_n = 1$ for all $n \geq 2$.

Proof. Let us prove the statement by induction on the number of legs n. For $n = 2$, we have $C_2 = \{\frac{1}{4} \mid 1\frac{1}{2}\} = 1$, as also illustrated in Figure 5.2. Now assume that $C_k = 1$ for any given $k \geq 2$. We want to prove that $C_{k+1} = 1$. If Right moves on C_{k+1}, then he removes a red edge from one of the legs. This leaves us with C_{k+1}^R consisting of a circus tent C_k with a separate blue edge, which has value $C_k + 1 = 1 + 1 = 2$. Since C_{k+1} is a Redwood Furniture position, we can apply Theorem 5.1 to obtain $C_{k+1} = \{0 \mid C_{k+1}^R\} = \{0 \mid 2\} = 1$. □

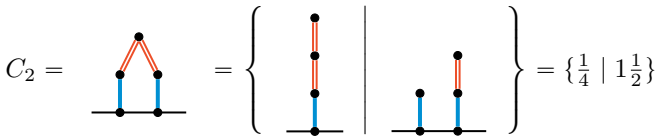


Figure 5.2: The first move on C_2. Left moves to a blue hollyhock of length 3, for which we know $H_3 = \frac{1}{2^{3-1}} = \frac{1}{4}$. Right moves to a sum of two hollyhocks $H_1 + H_2 = \frac{1}{2^{1-1}} + \frac{1}{2^{2-1}} = 1 + \frac{1}{2} = 1\frac{1}{2}$.

Theorem 5.3. Under flavour $f \in \{\mathcal{A}, \mathcal{B}, \mathcal{C}', \mathcal{D}\}$, any circus tent has value $\frac{1}{2}$, i.e., $\{\widehat{C}_n\}^f = \frac{1}{2}$ for $n \geq 2$.

Proof. First notice that Left will always be able to move, as long as she has had strictly fewer than n turns. She can follow chain-extension, but she can also follow any other strategy, as long as she is careful under flavour $\mathcal{A}$ to not leave exactly a single blue edge across all realisations.

Right will be able to move as long as he has had strictly fewer than n turns, there exists at least one blue edge in at least one realisation (which is the case if Left has had strictly fewer than n turns), he is careful under flavour $\mathcal{A}$ to not leave only a single red edge

across realisations, and he is careful to not give Left "free" moves. Right can ensure that he has move options under flavour $\mathcal{A}$ by following chain-extension. With this strategy, he also does not give Left free moves, because he ensures (except possibly on his last turn) that no red edge is removed from all realisations for which the corresponding blue edge may still be present in some realisation(s) (i.e., "classically separating" some blue edges from the remainder of the circus tent). Such separate blue edges give Left an advantage, because she does not need to play these edges in order to remove all red edges in all realisations. Thus, if Right follows chain-extension, he is always able to satisfy the requirements for being able to move, except for both players Left and Right having had strictly fewer than n turns. From now on, we only need to consider this single requirement.

We now prove that $\widehat{S} \equiv \{\widehat{C}_n\}^f = \frac{1}{2}$ based on the properties that we have just found. These properties are that, for any follower of $\widehat{S}$, if Left has had $k < n$ turns, then she can move, and if both Left and Right have had $k < n$ and $\ell < n$ turns, respectively, then Right can move. We can disregard ("reverse out") the first $2 \cdot (n-1)$ turns of play ($n-1$ turns for each player), because during these turns, either player is always able to respond to any move by the other player. This means that the follower $\widehat{S}'$ of $\widehat{S}$, for which $k = \ell = n-1$, has the same value as $\widehat{S}$, i.e., $\widehat{S}' = \widehat{S}$. For $\widehat{S}'$, we have that if Left moves, she moves to the empty position, and if Right moves, he moves to a position where Left has one final move (to the empty position) and Right has no moves. Thus, formally,

$$\widehat{S} = \widehat{S}' = \{0 \mid \{0 \mid \}\} = \{0 \mid 1\} = \frac{1}{2}.$$

□

Theorem 5.4. Under flavour $\mathcal{C}$, any circus tent has value 1, i.e., $\{\widehat{C}_n\}^{\mathcal{C}} = 1$ for $n \geq 2$.

Proof. Observe that Left is always able to move, as long as she has had strictly fewer than n turns. Right is able to move as long as both Left and Right have had strictly fewer than n turns, with one exception: if there exists exactly one red edge across all realisations, but this edge is not present in all realisations, then he is not allowed to play the unsuperposed move for this edge. In fact, Left can always force this situation to occur whenever she has had exactly $n-2$ turns and Right has had exactly $n-1$ turns (but not in any other case), which we will now prove.

Observe that if Right has had exactly $n-1$ turns, and Left has had strictly fewer than $n-1$ turns, then any realisation contains at most one red edge, since any superposed move by Right removes at least one red edge from every realisation ($n - x \leq 1$ for $x \geq n-1$). If Right has followed chain-extension, then there will be at least two realisations that each contain a different red edge (and possibly some other realisations that do not contain any red edge). Otherwise, it is possible that all realisations contain the same red edge. If Right has played poorly, it is also possible that no realisations contain any red edge, but we can assume that Right plays optimally. Now, if Left has had exactly $n-2$ turns, every realisation will contain exactly $n - (n-2) = 2$ blue edges (since all blue edges are grounded). Say that, thus far, Left has classically removed a blue edge every turn. This means that we either have exactly one realisation, or exactly two realisations, where both realisations contain a different red edge. If there is exactly one realisation, then Left can play an unsuperposed move, removing the blue edge corresponding to the final red edge. Otherwise, if there are exactly two realisations, Left can play any unsuperposed move, resulting in one realisation that no longer contains any red edge, and one realisation that contains exactly one red edge. In both cases, Left has moved to a position that has value 1, because from the new position Left is able to move again (to the empty position), but Right is not able to move. Left is not able to move to 1 from any position where she has had strictly fewer than $n-2$ turns, because she cannot play any (unsuperposed or superposed) move to ensure that two or more blue edges with their corresponding red edges are removed from all realisations, leaving only a single red edge across realisations.

Left is also not able to move to 1 from any position where Right has had strictly fewer than $n-1$ turns, because such a position has, for each blue edge, at least one realisation containing a red edge connected to that blue edge, along with at least one other red edge.

We now know for the follower of $\widehat{S} \equiv \{\widehat{C}_n\}^{\mathcal{C}}$ where Left has had exactly $k = n-2$ turns and Right has had exactly $\ell = n-1$ turns, that Left is able to move to 1. For any other follower, Left is able to move if she has had $k < n$ turns, and Right is able to move if Left and Right have had $k < n$ and $\ell < n$ turns, respectively. We can disregard the first $2 \cdot (n-2)$ turns of play, because the players are always able to respond to each other's moves during these turns. This means that follower $\widehat{S}'$ of $\widehat{S}$, with $k = \ell = n-2$, has the same value as $\widehat{S}$, i.e., $\widehat{S}' = \widehat{S}$. Formally, according to the above properties, we have

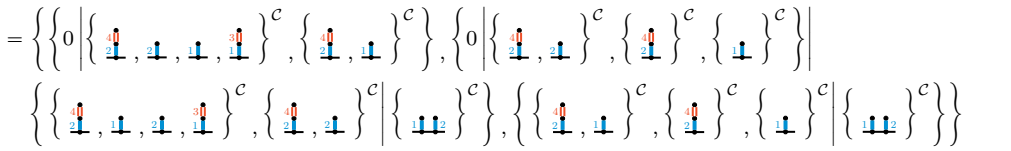
$$\begin{aligned}
\widehat{S} &= \widehat{S}' \\
&= \{\{0 \mid \{0 \mid \{0 \mid \}\}\} \mid \{1 \mid \{\{0 \mid \} \mid \}\}\} \\
&= \{\{0 \mid \{0 \mid 1\}\} \mid \{1 \mid \{1 \mid \}\}\} \\
&= \left\{\left\{0 \mid \frac{1}{2}\right\} \mid \{1 \mid 2\}\right\} \\
&= \left\{\frac{1}{4} \mid 1\frac{1}{2}\right\} = 1.
\end{aligned}$$

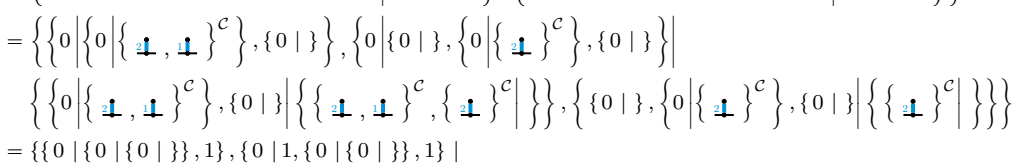
□

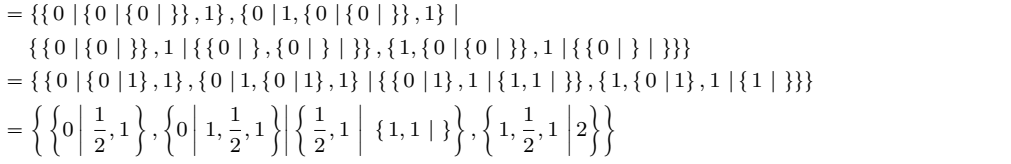
Example 5.5 illustrates how the value of a circus tent with $n = 2$ legs is calculated, under flavour $\mathcal{C}$.

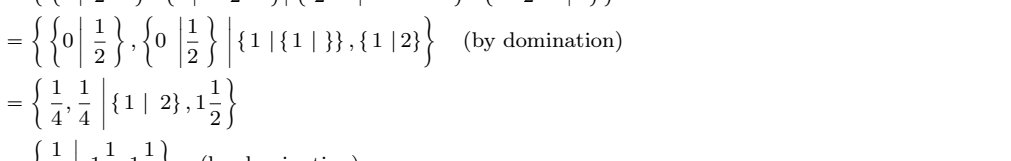
Example 5.5.

$\{\widehat{C}_2\}^{\mathcal{C}} \equiv$  $=$

$=$ 

$=$ 

$=$ 

$=$ 

$= \{\{0 \mid \{0 \mid \{0 \mid \}\}, 1\}, \{0 \mid 1, \{0 \mid \{0 \mid \}\}, 1\} \mid$
$\{\{0 \mid \{0 \mid \}\}, 1 \mid \{\{0 \mid \}, \{0 \mid \} \mid \}\}, \{1, \{0 \mid \{0 \mid \}\}, 1 \mid \{\{0 \mid \} \mid \}\}\}$

$= \{\{0 \mid \{0 \mid 1\}, 1\}, \{0 \mid 1, \{0 \mid 1\}, 1\} \mid \{\{0 \mid 1\}, 1 \mid \{1, 1 \mid \}\}, \{1, \{0 \mid 1\}, 1 \mid \{1 \mid \}\}\}$

$= \left\{\left\{0 \,\middle|\, \frac{1}{2}, 1\right\}, \left\{0 \,\middle|\, 1, \frac{1}{2}, 1\right\} \middle| \left\{\frac{1}{2}, 1 \,\middle|\, \{1, 1 \mid \}\right\}, \left\{1, \frac{1}{2}, 1 \,\middle|\, 2\right\}\right\}$

$= \left\{\left\{0 \,\middle|\, \frac{1}{2}\right\}, \left\{0 \,\middle|\, \frac{1}{2}\right\} \middle| \{1 \mid \{1 \mid \}\}, \{1 \mid 2\}\right\}$ (by domination)

$= \left\{\frac{1}{4}, \frac{1}{4} \,\middle|\, \{1 \mid 2\}, 1\frac{1}{2}\right\}$

$= \left\{\frac{1}{4} \,\middle|\, 1\frac{1}{2}, 1\frac{1}{2}\right\}$ (by domination)

$= \left\{\frac{1}{4} \,\middle|\, 1\frac{1}{2}\right\}$ (by domination)

$= 1.$

The example shows how, if Right plays the first move, Left is always able to move to a superposition where the final red edge does not exist in all realisations (or does not exist in any realisation at all). Note that in a set of options we do not write superposed game states that can be relabelled to one that is already given, because such superposed game states are isomorphic (Theorem 3.3).

6 Entanglement graph representation

When playing a quantum-inspired combinatorial game, the number of realisations is multiplied by a factor w (the maximum move width) every turn, in the worst case. For this section, we will restrict to $w = 2$ to make our discussion easier. The exponential growth in the number of realisations is not ideal if we want to keep track of the current superposed game state's set of realisations. However, since all realisations "originate" from the same classical starting game state, they will all be equivalent, except possibly for edges that have been included in previously played superposed moves. We want to use this redundancy to represent the current superposed game state more space-efficiently, as an *entanglement graph*. Unfortunately, when more moves have been played, it becomes possible for the realisations to have fewer mutual commonalities, making it more difficult to determine the result of any superposed move. The entanglement graphs that we describe can only be used for flavours $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$. They cannot be used for flavours $\mathcal{C}$ and $\mathcal{C}'$ directly, because these flavours require inspection of the realisations set in order to determine the legality of an unsuperposed move.

Our goal is to represent a superposed game state in space linear in the number of previously played moves t, rather than exponential, while being able to determine whether any given superposed move is legal (preferably in time exponential in t, or asymptotically faster). An entanglement graph $A = (V, e, I)$ is a set of vertices (Hackenbush edges) $V = \{\sigma_1, \ldots, \sigma_n\}$, a finite sequence of edges ("entanglements", or superposed moves) $e\colon \mathbb{N} \to 2^V$, where for $k \in [1, t]$, $e(k) = \{\sigma_i, \sigma_j\}$ consists of vertices $\sigma_i, \sigma_j \in V$ (possibly with $\sigma_i = \sigma_j$, for unsuperposed moves), and a *vertex illegality function* $I\colon V \to \{\top, \bot\}$. This vertex illegality function tells us for each $\sigma \in V$, whether e contains a (not necessarily contiguous) sequence of edges implying that σ cannot be used in any superposed move (corresponding to the case where Hackenbush edge σ no longer appears in any realisation). The vertex illegality function can be given in the form $I(\sigma) = C_1 \vee \ldots \vee C_m$, where each clause is of the form $C = p(\{\sigma_i, \sigma_j\}, \{\sigma_k, \sigma_\ell\}) \wedge \ldots \wedge p(\{\sigma'_i, \sigma'_j\}, \{\sigma'_k, \sigma'_\ell\})$. Here, we have for every $p(\{\sigma_i, \sigma_j\}, \{\sigma_k, \sigma_\ell\})$ that $p(\{\sigma_i, \sigma_j\}, \{\sigma_k, \sigma_\ell\}) = \top$ if and only if there exist $x, y \in [1, t]$ such that $e(x) = \{\sigma_i, \sigma_j\}$, $e(y) = \{\sigma_k, \sigma_\ell\}$, and $x < y$ (i.e., entanglement $\{\sigma_i, \sigma_j\}$ occurred before entanglement $\{\sigma_k, \sigma_\ell\}$), and otherwise $p(\{\sigma_i, \sigma_j\}, \{\sigma_k, \sigma_\ell\}) = \bot$. There is a vertex for each Hackenbush piece of some starting labelled short game $\widehat{G}$, and initially there are no entanglements, i.e., $t = 0$. For all vertices $\sigma \in V$, it then holds that $I(\sigma) = \bot$ (because there are no entanglements). When a move is played, it is appended to the end of sequence e. This new entanglement may cause $I(\sigma) = \top$ for some vertices σ, indicating that these vertices can no longer be used in superposed moves.

In general, for any $\sigma \in V$, the expression for the vertex illegality function $I(\sigma)$ can grow very quickly with the number of Hackenbush edges in the starting game state. This is because there are many ways in which a vertex could become illegal, each of which needs to be represented as a clause. However, for some classes of simple Hackenbush classical game states we do not need to construct this large expression. Instead, we can reason about *entanglement chains*, which are formed by multiple entanglements "connecting" vertices in a chain, for example by following the chain-extension strategy (Definition 3.21). For classical game states only consisting of a sum of wholes, if any entanglement chain forms a cycle, then all vertices of this cycle become illegal, along with all vertices of chains connected to this cycle. In this case, n moves must have been played on n Hackenbush edges, meaning that they can no longer appear in any realisation. Figure 6.1 shows an example.

We can also consider a classical game state consisting of two equal halves, for which the entanglement graph tree under flavour $\mathcal{A}$ is shown in Figure 6.2. If the red edges are untouched, but the blue edges become entangled, then the red edges also become entangled. This means that if Right then moves on these red edges, they become illegal, because they were already entangled. However, if the red edges are entangled first, followed by the blue edges, then the red edges do not become illegal. For two different halves, as shown in Figure 6.3, the edge of the starting player's colour farthest from the ground becomes illegal if crossing entanglements are made. Future research can focus on

developing (simple) rules for entanglement graphs of more complicated classical game states, such as arbitrary Blue Meadow game states, or more generally, Short Hollyhocks game states.

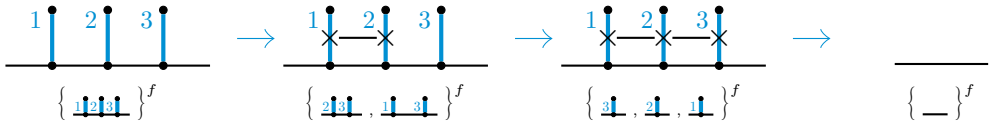


Figure 6.1: An example of an entanglement chain being formed, and all vertices being made illegal by the last entanglement. The line segments with crosses at their ends indicate which HACKENBUSH edges have occurred in a single move together ("entanglements"). Below each entanglement graph, the corresponding realisations set is shown.

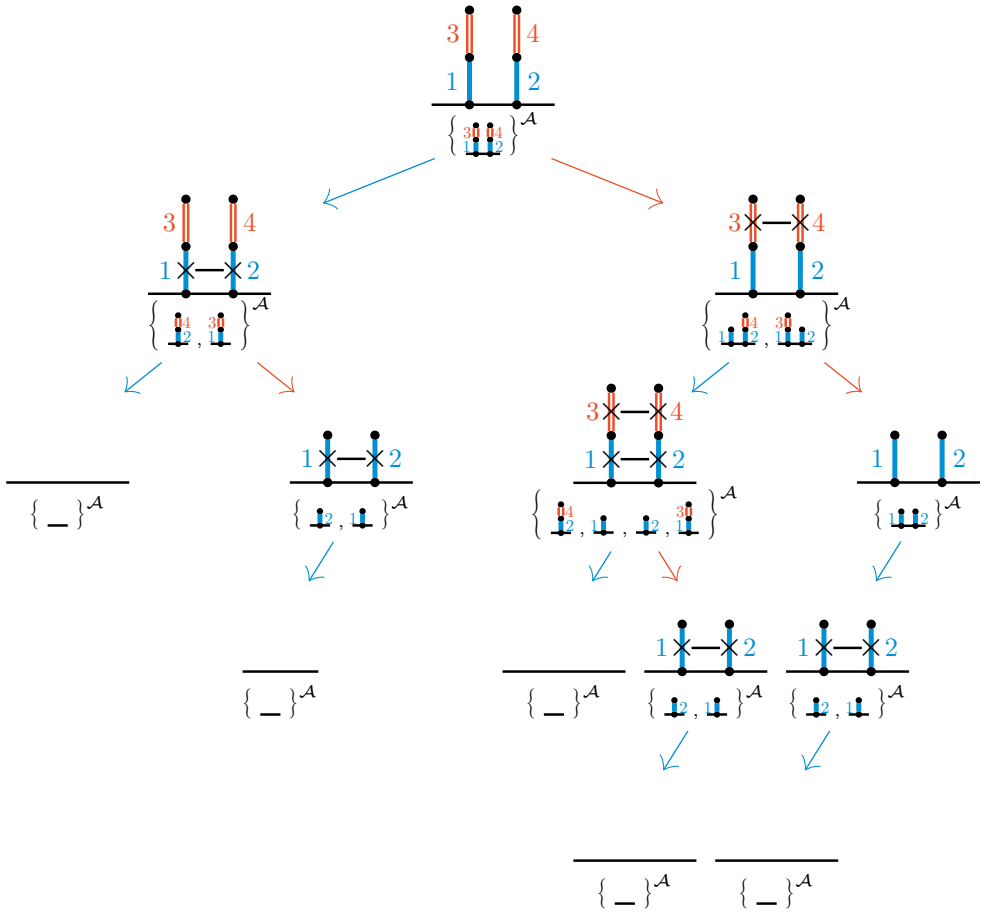


Figure 6.2: The entanglement graph tree for a classical game state of two equal halves, under flavour $\mathcal{A}$. Below each entanglement graph, the corresponding realisations set is shown.

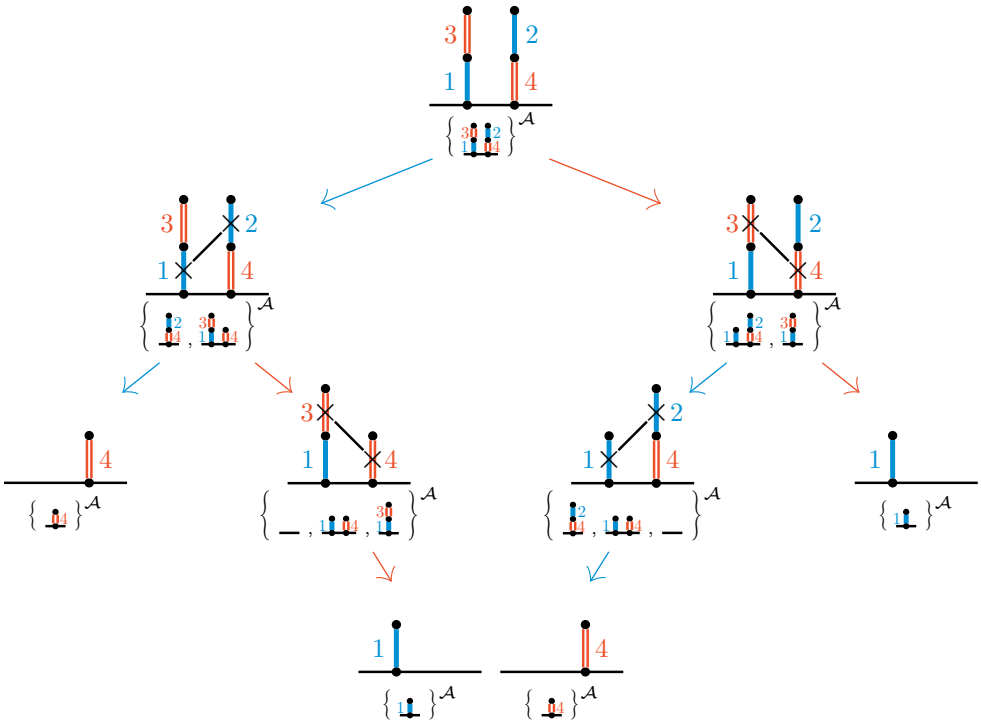


Figure 6.3: The entanglement graph tree for a classical game state of two different halves, under flavour $\mathcal{A}$. Below each entanglement graph, the corresponding realisations set is shown.

7 Relationship with quantum mechanics

As we have mentioned before, quantum-inspired combinatorial games relate to quantum mechanics by certain aspects of their design. These aspects serve as *metaphors* for concepts from quantum mechanics [11]; namely superposition, entanglement, collapse (measurement), and interference.

A game state in a quantum-inspired combinatorial game can be viewed as being in a superposition over its realisations, corresponding to the concept of a superposition of basis states in quantum computing. However, unlike in quantum computing, the realisations do not have any *amplitude* (making a superposed game state more similar to an equal-weight superposition, where each state has the same amplitude). Rather, we only care whether a realisation exists (has an amplitude strictly greater than 0) or not. Superposed moves can be viewed as applying gates to qubits (which represent the presence of Hackenbush edges, for example). These gates can generate entanglement between qubits. Illegal classical moves can be viewed as collapsing realisations, leaving only the realisations that can possibly be reached according to the classical ruleset. Finally, a phenomenon similar to interference can be constructed in, for example, Quantum Hackenbush (any flavour) by starting from a width-1 game state, and then applying the same width-2 move $\{\sigma_1, \sigma_2\}$ twice in a row. The first superposed move results in two realisations; one in which edge σ_1 has been removed, and another in which edge σ_2 has been removed. The second superposed move then removes edge σ_2 from the realisation in which it is still present, and edge σ_1 from the other realisation which still contains it, respectively. This results in two equivalent realisations, in which both σ_1 and σ_2 are no longer present. Thus, we are left with a superposed game state containing only one realisation. This "merging" of equivalent realisations in a superposed state reminds of constructive interference. In quantum computing, a simple example can be given using the Hadamard gate and a single-qubit starting state $|0\rangle$. Applying the Hadamard gate once results in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Applying it a second time results in $\frac{1}{2}(|0\rangle + |1\rangle + |0\rangle - |1\rangle) = |0\rangle$.

However, as we will explain below, it is not trivial to use "real" quantum mechanics, in the form of a quantum circuit, to simulate Quantum Hackenbush. As proposed in [4], a game state should be represented as a multi-qubit state, and moves should be implemented by certain unitary gates, but for this thesis we were not able to design a circuit that simulates Quantum Hackenbush, even when only considering game states consisting merely of independent grounded edges. Nonetheless, we will discuss our ideas for what such a circuit could look like, and the details of the design we tried to use.

7.1 Designing a Quantum Hackenbush circuit

A seemingly natural approach would be to model a Hackenbush game state as a fermionic system, which is used in physics to simulate the *creation* and *annihilation* of particles [12]. Here we have *sites*, each of which may contain a particle or not. For our purposes, one could, for example, say that each Hackenbush edge corresponds to exactly one site, and that a particle being present at that site corresponds to the edge being present. To be able to simulate a fermionic system, a fermion-to-qubit mapping can be used. For example, Qiskit Cold Atom[6] uses a qubit for each site, where $|0\rangle$ means that the site is unoccupied, and $|1\rangle$ means that the site is occupied. Of course, to be able to play the game, it should be possible to remove edges. However, since all quantum computations must be *reversible*, it is not possible to simply set any qubit to $|0\rangle$, disregarding the state that it was in previously. Instead, we can add an ancillary qubit, to which we can "swap occupation". Superposed moves should also be possible. For these, we need a way to split fermions across sites. This way, with some probability the site for one of the two edges would be measured as being empty while the other one is occupied, or the other way around with complementary probability. Splitting a fermion creates a superposition of basis states. Fermions can be split by applying the

[6] Qiskit Cold Atom's documentation can be found at https://qiskit-community.github.io/qiskit-cold-atom/.

swap Hamiltonian multiplied by a factor $\frac{1}{2}$. The above approach would only work for flavour $\mathcal{A}$ or $\mathcal{D}$, because for these flavours the current superposed game state (represented as a superposition of basis states) does not need to be inspected to determine whether a classical move is legal; classical moves are simply never legal or always legal.

On their turn, if a player suspects that their opponent's latest move was illegal, they can challenge their opponent, as we discussed in Section 3.1. To verify whether the move was illegal, we can measure the ancilla that is now supposed to contain the occupancy of the edge(s) that was (were) last removed. If this ancilla is in the $|0\rangle$ state, then we know that the move was illegal (no edges were removed in any realisation). Note that after t turns have been played, any particle has been split at most t times, meaning that in the worst case we only have a $\frac{1}{2^t}$ probability of measuring a $|1\rangle$ in a single measurement. Thus, in order to have a probability of at least $1-\epsilon$ that we do not miss any basis state in which the ancilla is $|1\rangle$, the minimum number of times we should measure $\lceil k \rceil$ is calculated as shown below.

$$\begin{aligned}
1-\left(1-\frac{1}{2^t}\right)^k &\geq 1-\epsilon \\
\left(1-\frac{1}{2^t}\right)^k &\leq \epsilon \\
\ln\left(\left(1-\frac{1}{2^t}\right)^k\right) &\leq \ln(\epsilon) \\
k\cdot\ln\left(1-\frac{1}{2^t}\right) &\leq \ln(\epsilon) \\
k &\geq \frac{\ln(\epsilon)}{\ln\left(1-\frac{1}{2^t}\right)}
\end{aligned}$$

If we never measure a $|1\rangle$, then we say that the latest move was illegal, so the current player wins.

7.1.1 Move gate design

If we want to simulate a game state consisting of n grounded edges, then we can start by initialising an $(n+m)$-qubit state where the first n qubits each correspond to one of the labelled edges and are set to $|1\rangle$. The remaining m qubits are ancillas that are set to $|0\rangle$. For implementing moves, the idea is that on their turn, the player selects a gate that swaps some occupancy to an ancilla. There could be one ancilla qubit per turn, to ensure that the ancilla we are swapping to is always $|0\rangle$, preventing occupations from being swapped from an ancilla qubit back to an edge qubit (if the player attempts to remove an edge that has already been removed). A Hackenbush game state consisting of n edges has birthday n, meaning that a total of n moves can be played on it. This means that we would need $m=n$ ancillas. For classical move gates, the occupancy for a single edge of the player's colour is swapped to an ancilla. For a width-2 superposed move gate, the occupation for one of the edges is first swapped to the ancilla, after which a "half swap" is performed between the two edges. The design described above has the problem that, for any two untouched edges, applying the same superposed move twice does not result in an occupancy of 0 for both edges, as we would want. Rather, they will both have an occupancy of $\frac{1}{4}$, since

$$\left|\left(\frac{1}{\sqrt{2}}\right)^2\right|^2 = \frac{1}{4}.$$

The recurrence relation for applying the move gate $M_{i,j,k}$, for two given edges $i \neq j$ and ancilla index k, some number of times $t \geq 1$ in a row, is given by

$$|\Psi^t\rangle = |\psi_1^t \dots \psi_{n+t}^t\rangle = M_{x,y,n+t}(|\Psi^{t-1}\rangle|0\rangle)$$
$$= \frac{1}{\sqrt{2}}(|\psi_1^{t-1}\dots\psi_{x-1}^{t-1}\ 0_x\ \psi_{x+1}^{t-1}\dots\psi_{n+t-1}^{t-1}\rangle + |\psi_1^{t-1}\dots\psi_{y-1}^{t-1}\ 0_y\ \psi_{y+1}^{t-1}\dots\psi_{n+t-1}^{t-1}\rangle)|1\rangle,$$

with $|\Psi^0\rangle = |1_1 \dots 1_n\rangle$, and two constant edges $x \neq y$.

The main aspect of quantum-inspired combinatorial games that is difficult to simulate is that realisations "disappear" when an illegal move is played on them. The problem above was caused by this aspect not being simulated. If we represent a superposed game state by a superposition of basis states, in the way that we described above, then we would want a move gate U for a move that is not legal on a classcial game state represented by a basis state $|\psi\rangle$ to lead this basis state (with its trailing ancilla state $|a\rangle$) to a state $U|\psi a\rangle = 0$, with amplitude 0. Such a move gate U cannot exist, because any gate must be unitary, meaning that it must preserve the norm of any state that it is applied to.

For future research, it may be interesting to see whether it is possible to simulate "disappearing" realisations using some form of measurement-based quantum computing. Some inspiration may be taken from Quantum Chess [11], but note that in that paper measurements were used to add an element of non-determinism to the game, which is not the goal in our case.

7.1.2 Dependencies between edges as Rydberg interactions

Another question that could be addressed in future research is that of simulating HACKENBUSH game states with dependencies between edges. For example, in Short Hollyhocks game states, the top edge of any half is dependent on the bottom edge of that half. If a player selects the bottom edge of a half, then the corresponding top edge must also be removed, if it was still present. In a fermionic system, a similar phenomenon can occur by means of *Rydberg interactions*. A *Rydberg blockade* uses these interactions to entangle two qubits in such a way that they will always have the same measurement result [13]. No Rydberg blockade should be used when removing the top edge of a half, however.

8 Conclusions, conjectures, and further research

We started by introducing HACKENBUSH, along with the classical combinatorial game theory that is relevant for our purposes. Then, we defined quantum-inspired combinatorial games, based on the original paper that proposed them [4]. These have five "flavours", which each define different conditions for when classical moves are allowed. We have formally defined properties of classical rulesets, and how these manifest in the quantum-inspired case. We have also shown that for dead-ending consistent rulesets, such as HACKENBUSH, certain realisations are covered, and thus can be left out. This result we then used to prove values of single stalks in QUANTUM HACKENBUSH under all flavours, except flavour $\mathcal{C}$. Then, we discussed interesting results that were obtained using our `quantum-hackenbush-suite` program. After that, we went back to theoretical analysis, but this time of circus tents under all flavours. Finally, we discussed two ideas that we have attempted to tackle for this thesis, but for both of which we encountered certain difficulties, so we leave them for future research.

To conclude, quantum-inspired combinatorial games are intriguing for the additional complexity that they introduce, in terms of determining optimal moves, and for the way in which they serve as "metaphors" for concepts from quantum mechanics, giving players a feel for the perhaps counter-intuitive nature of acting on quantum systems. We have shown an extensive variety of properties and strategies that can arise when allowing for superposed moves, such as non-number values, or even $\mathscr{N}$-positions arising, while in classical HACKENBUSH only numbers can be constructed. Superposed moves may also change the outcome class of a position, or they may allow a player to "abuse" the rules of the specific flavour that is being played under, shifting the advantage further in their favour. This thesis is a promising starting point for further analysis of QUANTUM HACKENBUSH, or other quantum-inspired combinatorial games, where similar diverging properties may be found as compared to the classical ruleset.

8.1 Conjectures

In our experimental results, we have observed certain patterns, based on which we have written the following conjectures. It may be interesting to prove these in future research.

Conjecture 8.1. All Blue Meadow classical game states are numbers, under any flavour.

This conjecture would imply that none of these positions are first player wins, for example.

Conjecture 8.2. Under flavour $f \in \{\mathcal{B}, \mathcal{C}', \mathcal{D}\}$, with unbounded w, for any Blue Meadow position $G \cong n \cdot H_2 - m \cdot H_1$, the canonical form value of $\{\widehat{G}\}^f$ has birthday m if $n = 0$, and birthday $n + \max(m, 1) - 1$ otherwise.

Proving this conjecture would be interesting, because it would illustrate how superposed moves allow Right to play moves on followers of $\{\widehat{G}\}^f$ without Left having an immediate answer, in the sense that Right is always able to "avoid" his edges from being removed in all realisations (except for his last turn).

Conjecture 8.3. All Short Hollyhocks classical game states are numbers, under any flavour.

This is a generalisation of Conjecture 8.1. It would mean that taller stalks, such as the ones in the Single Stalks game states we have studied, are necessary to construct non-number classical game states.

Conjecture 8.4. There are no classical game states with outcome class $\mathscr{N}$, under any flavour.

This would be in line with the $*$-positions for each flavour that we have seen in Section 4.4, which have all been superposed game states of width greater than 1. It would mean that it is not possible to have classical starting game state from which either player can win if they start.

Conjecture 8.5. Under flavour $\mathcal{C}$, the canonical form value of a Single Stalk classical game state of length ℓ is an immediate follower of the canonical form value of some Single Stalk classical game state of length $\ell + 1$.

Proving this conjecture could be interesting, because it might give more insight into the patterns shown in Table 4.1.

Conjecture 8.6. Any HACKENBUSH classical game state has the same value under any of the flavours $f \in \{\mathcal{B}, \mathcal{C}', \mathcal{D}\}$, for unbounded w.

This would mean that, for $\mathcal{R} =$ HACKENBUSH and unbounded w, there is no difference between these flavours. Thus, in that case there would never be any benefit to playing a classical move before you absolutely have to, because this is the only possible way to play under flavour $\mathcal{B}$. If this conjecture is proven, it also makes proving properties for HACKENBUSH classical game states under these flavours easier, in the case of unbounded w, because proving the property for any of these flavours would prove it for all.

8.2 Further research

In addition to attempting to prove the above conjectures, we have numerous ideas that could be studied in future research. Two of these we have already discussed in separate sections; namely representing QUANTUM HACKENBUSH superposed game states as entanglement graphs (Section 6) and simulating QUANTUM HACKENBUSH using a quantum circuit (Section 7).

The performance of programs for computing the outcome class, birthday, or value of a classical game state in QUANTUM HACKENBUSH could also be improved by pruning superposed game states that can be relabelled to another superposed game state in the tree. These superposed game states are isomorphic (Theorem 3.3). For Short Hollyhocks classical game states this is relatively easy, because for stalks that are "untouched" (all edges of the stalk exist in all realisations), we only need to consider superposed moves on combinations of different types of short hollyhocks (blue half, red half, blue whole, and red whole). This is possible, because stalks are disjoint components. A preliminary experiment, using an incorrect implementation[7], only showed a noticeable speedup for smaller numbers of halves and wholes, but a significant reduction in memory usage was observed for larger numbers.

Lastly, we have a few suggestions for interesting research directions that we have not mentioned earlier. First, there is finding a simple rule for determining the value of any Single Stalk classical game state under flavour $\mathcal{C}$, with a recurrence relation for the non-number values, which we already alluded to in Section 4.2. Finding and proving simple rules for the values of Short Hollyhocks classical game states under different flavours is another possibility. Additionally, general (taller) Hollyhocks classical game states may have interesting properties. We have also not yet studied general Redwood Furniture classical game states, we have only considered Circus Tent classical game states specifically. Adding green edges (QUANTUM RED-GREEN-BLUE HACKENBUSH) would also require a whole different analysis, possibly in some way relating to the results for QUANTUM NIM-SUBTRACT that were found in [4]. It may also be fun to come up with different ways of altering combinatorial games in a quantum-inspired fashion. For example: superposed moves may be assigned an age (number of turns), after which one of its classical moves becomes the move that was played "in reality". Another idea is to select one realisation whenever a certain event occurs (e.g., one of the two players does not have any move), similar to the cyclic entanglement rule of quantum tic-tac-toe [14]. Yet another idea is to keep track of the "weight" (amplitude) of each realisation, and to allow measurements to occur by some rule, which then probabilistically selects one

[7] This implementation considered two superposed game states to be isomorphic if they were at the same depth in the tree, and they contained the same number of realisations, and these realisations were equivalent in terms of the number of blue halves, red halves, blue wholes, and red wholes in each realisation. It was found that too many parts of the superposed game state tree were pruned based on this rule, because for larger numbers of halves and wholes the program outputted incorrect values.

realisation based on these weights. Finally, it may be interesting to study the relationship between quantum-inspired combinatorial games and other variants of combinatorial games, such as synchronised games, and *quantum games*, which are played using quantum strategies [15]. For example: what happens to the winnability if both players move in a width-2 superposed move? There may also be a relationship between synchronised games and quantum games, because in both cases a Nash equilibrium can be determined for the players' strategies.

References

[1] E. R. Berlekamp, J. H. Conway, and R. K. Guy. "Winning ways for your mathematical plays, Volume 1". In: *(No Title)* (2001).

[2] M. Albert, R. Nowakowski, and D. Wolfe. *Lessons in play: An introduction to combinatorial game theory*. 2nd ed. AK Peters/CRC Press, 2019.

[3] A. N. Siegel. *Combinatorial game theory*. Vol. 146. American Mathematical Soc., 2013.

[4] P. Dorbec and M. Mhalla. "Toward quantum combinatorial games". In: *arXiv preprint arXiv:1701.02193* (2017). URL: https://arxiv.org/abs/1701.02193.

[5] E. D. Demaine and R. A. Hearn. "Playing games with algorithms: Algorithmic combinatorial game theory". In: *arXiv preprint cs/0106019* (2001).

[6] K. Burke, M. Ferland, and S.-H. Teng. "Quantum combinatorial games: Structures and computational complexity". In: *arXiv preprint arXiv:2011.03704* (2020). URL: https://arxiv.org/abs/2011.03704.

[7] R. Milley and G. Renault. "Dead ends in misere play: The misere monoid of canonical numbers". In: *Discrete Mathematics* 313.20 (2013), pp. 2223–2231.

[8] K. W. Burke, M. Ferland, and S.-H. Teng. "Quantum-inspired combinatorial games: Algorithms and complexity". In: *11th International Conference on Fun with Algorithms (FUN 2022)*. 2022. URL: https://doi.org/10.4230/LIPIcs.FUN.2022.11.

[9] T. van Roode. "Partizan forms of Hackenbush combinatorial games." Master thesis. University of Calgary, 2002.

[10] X. Lenstra. "A program for analysing combinatorial and synchronized games". Bachelor thesis. LIACS, Leiden University, 2022. URL: https://theses.liacs.nl/2358.

[11] C. Cantwell. "Quantum chess: Developing a mathematical framework and design methodology for creating quantum games". In: *arXiv preprint arXiv:1906.05836* (2019). URL: https://arxiv.org/abs/1906.05836.

[12] D. Tong. *Lecture notes in quantum field theory*. https://www.damtp.cam.ac.uk/user/tong/qft.html. Accessed: 3 July 2024. 2006.

[13] E. Urban, T. A. Johnson, T. Henage, L. Isenhower, D. Yavuz, T. Walker, and M. Saffman. "Observation of Rydberg blockade between two atoms". In: *Nature Physics* 5.2 (2009), pp. 110–114.

[14] A. Goff. "Quantum tic-tac-toe: A teaching metaphor for superposition in quantum mechanics". In: *American Journal of Physics* 74.11 (2006), pp. 962–973.

[15] J. Eisert, M. Wilkens, and M. Lewenstein. "Quantum games and quantum strategies". In: *Physical Review Letters* 83.15 (1999), p. 3077.