Comenius University, Bratislava
Faculty of Mathematics and Physics

Time and Space Complexity of Reversible Pebbling

Master Thesis

Diplomant:
Richard Královič

Consultant:
Prof. RNDr. Branislav Rovan, PhD.

Čestne prehlasujem, že som diplomovú prácu vypracoval samostatne s použitím materiálov, ktoré uvádzam v zozname použitej literatúry.

Bratislava, 2004 .....

Acknowledgement

I would like to thank Prof. RNDr. Peter Ružička, CSc.(†) and to my supervisor Prof. RNDr. Branislav Rovan, PhD. for introducing me into the area of reversible pebbling and for their valuable advice and consultations.

Contents

Chapter 1

Introduction

Computer science exploits a multitude of computation models for solving various kinds of problems. These models often represent the problem by the values to be computed, which may depend on each other. In order to compute a given value, another values may have to be be computed first. These dependencies form the computational structure of the problem. Values are computed according to the rules of the computation model (e.g. deterministically or nondeterministically), and stored in memory cells. The complexity of the problem is usually measured in the terms of time and space, i.e. how many computation steps and how many memory cells are needed to compute all of the values. It is often impossible to achieve optimal time and space complexity simultaneously, in which cases it is reasonable to study tradeoffs between the time and space complexity.

It is possible, and sometimes very useful, to model such computations in an abstract way. For this purpose a graph-theoretic model called *pebble game* was introduced. In the pebble game, a problem is represented by a directed acyclic graph. Values to be computed are represented by the vertices of this graph, and an edge from a vertex b to a vertex a represents the fact that for computing the value a, the value b has to be already known. A computation consists of by placing and removing pebbles (representing memory locations) on/from the vertices. A pebble placed on a certain vertex denotes the fact that the value of this vertex is already computed and stored in the memory.

In order to obtain a time-efficient space-restricted computation for a problem represented as a directed acyclic graph, it is often useful to study the time-space complexity of the pebbling game on the corresponding class of graphs. Possible applications include code generation for arithmetic expression evaluation, implementation of recursive programs, or many problems in linear algebra.

Within the above mentioned framework a directed acyclic graph (dag) corresponds to a specific instance of the problem. In order to prove results about the problem in general, whole classes of directed acyclic graphs have to be considered. For example a simple sequential algorithm for computing the maximum of an array of length n can be described by a chain graph with n vertices. Hence, for studying the computation of maximum in general, the class of all chain graphs has to be considered. Another important classes dags representing the structure of many important problems are the complete binary trees and the butterflies. For example recursive "divide and conquer" algorithms (e.g. recursive mergesort) can be described by the complete binary tree topology and numerical problems such as Fast Fourier Transform can be described by the butterfly topology.

Pebble game was originally designed to study deterministic computations. In connection with another models of computation, various modifications of the standard pebble game have been introduced. For example, pebble game with black and white pebbles can be used for modelling nondeterministic computations, pebble game with two players for modelling alternating computations, pebble game with red and blue pebbles for modelling input-output complexity analysis, pebble game with labels for database serializability testing, etc. For an overview of these modifications see [11]. The standard pebble game played on dynamic graphs was studied as a model of incremental computations (see [12] and [13]). Modifications of the pebble game for various computation models provide a unified view of these models and allow for comparing their power.

The most profoundly studied computation models are determinism and nondeterminism. However, the interest in reversible computations has been growing recently. While in the deterministic computation each state of the computation has to uniquely define the following state of the computation, in reversible computation each state has to uniquely define both the *following* and the *preceding* state of the computation.

The model of reversible computations is interesting in connection with quantum computing. Since the basic laws of quantum physics are reversible, the quantum computation has to be reversible, too. Another motivation for studying the model of reversible computation follows from the fact that reversible operations are not known to require any heat dissipation. With the continuing miniaturisation of computing devices, reduction of the energy dissipation becomes a very important issue. Both these reasons for studying reversible computations are mentioned in [2], [5], [6] and [14].

Introduced by Charles H. Bennett in [1] and further analysed in literature in connection with the chain topology (e.g. in [4], [5] and [15]), reversible pebble game is a modification of the standard pebble game for modelling

reversible computations that provides a tool for analysing the time and space complexity and time-space trade-offs of reversible computations.

We present a technique to prove upper and lower bounds on the time and space complexities of the reversible pebble game. Using this technique we study the space complexity, the time complexity and time space tradeoff for the reversible pebble game on the chain topology. It is evident that the minimal space complexity of the standard pebble game on a chain of length n is $O(1)$, the minimal time complexity is $O(n)$ and the minimal space and time complexities can both be achieved simultaneously. As will be discussed later, it is not the case of the reversible pebble game. Concerning the reversible pebble game, the minimal space complexity on the chain topology is $O(\lg n)$ (proved in [4], [5], [6]) and the time complexity for space-optimal computation is $\Omega(n \lg n)$ for infinitely many n (published in [8] and [9]).

We also discuss the space complexity of the binary tree and butterfly topology. A well known result proved by Paterson and Hewitt (presented in [10]) states that the minimal space complexity for the standard pebble game on a complete binary tree of height h is $h + 1$. We show a tight space bound for reversible pebble game on a complete binary tree of the form $h + \Theta(\lg^* h)$ (published in [8] and [9]). These results give an evidence that more resources are needed for reversible computation in comparison with irreversible computation.

We also show an upper bound on the time and space complexity of the reversible pebble game based on the complexity of the standard pebble game, regardless of the topology: let time t and space p be sufficient for the standard pebble game on an arbitrary graph G and let time t' and space p' be sufficient for the reversible pebble game on a chain of length t. Then time t' and space $p \cdot p'$ are sufficient for the reversible pebble game on G. This result is a generalisation of the results by Charles Bennett (published in [1]) about reversible simulation of irreversible computations, expressed in terms of the reversible pebble game.

Chapter 2

Models of Pebble Games

In this chapter we introduce the basic concepts of pebbling games and give the definitions of the chain, binary tree and butterfly topologies. Furthermore, we define the standard pebble game as well as its modifications, pebble game with black and white pebbles and reversible pebble game.

2.1 Basic Topologies

A directed acyclic graph (shortly *dag*) is a graph with directed edges without multiple edges, loops and cycles. A dag can be used to represent the structure of an instance of a problem: the vertices represent the values to be computed and the edges represent the dependencies among them. A directed edge from a vertex b to a vertex a represents the fact that in order to compute a value a the value b have to be known.

A *topology* is a tuple $(\mathcal{G}, \mathcal{M})$, where $\mathcal{G}$ is a class of dags and $\mathcal{M} : \mathcal{G} \rightarrow \mathbb{N}$ is a function. A topology can be used to represent a general problem. The instances of this problem are represented by the members of $\mathcal{G}$ and the function $\mathcal{M}$ specifies the size of the individual instances. The rationale behind $\mathcal{M}$ is the fact that for the sake of computing the time and space complexities of the problem the size of the problem may be different from the number of vertices or number of edges of a particular member of $\mathcal{G}$.

The most basic yet very important topology is the *chain topology*. It can be used to model simple sequential computations where each value can be computed directly from the previous one.

Definition 2.1. *A* chain graph *(denoted as Ch_n) is a directed graph with n vertices and $n-1$ edges: $Ch_n = (V, E)$, where $V = \{1 \dots n\}$ and*

$$E = \big\{(i, i+1) \mid \forall i \in \{1 \dots n-1\}\big\}$$

The chain topology *is defined as* $(\mathcal{G}, \mathcal{M})$*, where* $\mathcal{G} = \{Ch_n \mid n \in \mathbb{N}\}$ *and* $\mathcal{M}(Ch_n) = n$*.*

For modelling of recursive computations that use a "divide and conquer" strategy, the complete binary tree topology can be used. Binary trees reflect the structure of problems where the global solution can be computed from the solutions of two subproblems.

Definition 2.2. *A* complete binary tree *of height* 1 *(denoted by* $\mathrm{Bt}(1)$*) is a directed graph containing one vertex and no edges. A* complete binary tree *of height* $h > 1$ *(denoted as* $\mathrm{Bt}(h)$*) consists of a root vertex* r*, a left subtree* L *and a right subtree* R*.* L *and* R *are complete binary trees of height* $h - 1$ *and there is a directed edge from root of* L *to* r *and from root of* R *to* r*.*

The binary tree topology *is defined as* $(\mathcal{G}, \mathcal{M})$*, where* $\mathcal{G} = \{\mathrm{Bt}(h) \mid h \in \mathbb{N}\}$ *and* $\mathcal{M}(\mathrm{Bt}(h)) = h$*.*

Let T be a complete binary tree. We denote the root vertex of T as $\mathrm{R}(T)$, the left subtree of T as $\mathrm{Lt}(T)$ and the right subtree of T as $\mathrm{Rt}(T)$. The definition of function $\mathcal{M}$ indicates that time and space complexities for pebbling games on binary tree topology will be expressed in terms of the height of the binary trees.

Another important class of graphs to study are the butterfly graphs. They posses the superconcentrator property and form an inherent structure of some important problems in numerical computations, such as the discrete FFT.

Definition 2.3. *A* butterfly graph *of order* d *(denoted by* $\mathrm{Bf}(d)$*) is a directed graph* $G = (V, E)$*, where* $V = \{1 \ldots d\} \times \{0 \ldots 2^{d-1} - 1\}$ *and*

$$E = \left\{ \big((i,j),(i+1,j)\big), \big((i,j),(i+1, j \oplus 2^{i-1})\big) | 1 \le i < d, 0 \le j \le 2^{d-1} - 1 \right\}$$

where $\oplus$ *denotes bitwise exclusive or.*

The butterfly topology *is defined as* $(\mathcal{G}, \mathcal{M})$*, where* $\mathcal{G} = \{\mathrm{Bf}(d) \mid d \in \mathbb{N}\}$ *and* $\mathcal{M}(\mathrm{Bf}(d)) = d$*.*

For an example of a butterfly graph, see Figure 2.1a. An important property of the butterfly graphs is that the butterfly graph $\mathrm{Bf}(d)$ can be decomposed into 2^{d-1} complete binary trees of height d. The root of the i-th tree is the vertex $(1, i)$ and this tree contains all vertices that can be reached from the root (see Figure 2.1b). Note that these trees are not disjoint.

This decomposition property implies that the space complexity of a pebbling game on the butterfly graph of order d equals to the the space complexity of the pebbling game on the complete binary tree of height d. This fact is discussed more precisely in section 5.4.

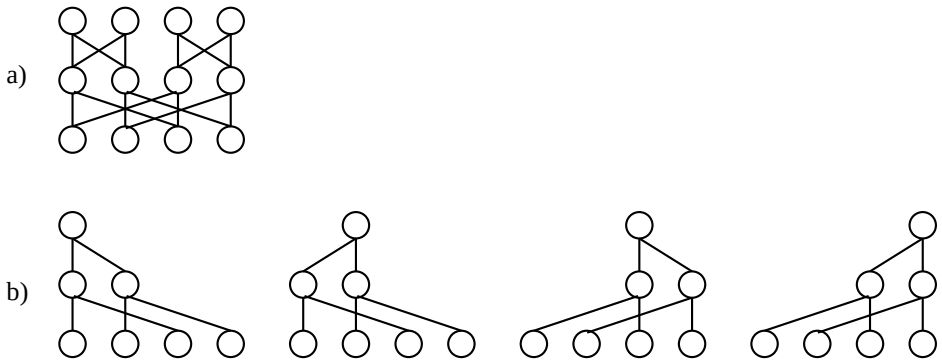

Figure 2.1: a) The butterfly graph of order 3. b) The decomposition of the butterfly graph of order 3 into 4 complete binary trees of height 3.

2.2 Common Concepts

All pebbling games are played on directed acyclic graphs. In general, a pebbling game may use k types of pebbles. The state of the pebble game played on a dag G is called a configuration.

Definition 2.4. *Let $G = (V, E)$ be a dag. A* configuration on G *is a function $V \to 2^{\{1...k\}}$ that specifies for each vertex of G a set of pebbles that lies on it.*

Since many variants of pebble game allow at most one pebble on a particular vertex in a given time, we use a simpler notation in these cases. Instead of writing $C(v) = \{i\}$ we write $C(v) = i$ and instead of $C(v) = \emptyset$ we write $C(v) = 0$.

An important parameter of a configuration is the number of pebbles it uses.

Definition 2.5. *Let $G = (V, E)$ be a dag and let C be a configuration of a pebble game that uses k types of pebbles on G. We denote the number of pebbles of type i used in a configuration C by*

$$\#_i(C) = |\{v | v \in V \wedge i \in C(v)\}|$$

and the total number of pebbles used in a configuration by

$$\#(C) = \sum_{i=1}^{k} \#_i(C) = \sum_{v \in V} |C(v)|$$

A configuration without any pebbles on G is called an empty configuration *and is denoted by* $\mathrm{E}(G)$.

Each variant of the pebble game specifies its own rules describing which moves are allowed in the game. An ordered pair of configurations on a dag G such that the second one follows from the first one by applying one of the rules is called a *transition*.

For our purposes a transition can be also a pair of two identical configurations. Such a transition is called *trivial*.

As a model of a real computation we define the computation in the context of a pebble game.

Definition 2.6. *Let G be a dag. A* computation $\mathcal{K}$ of length n on G *is a sequence $\mathcal{K}(1) \ldots \mathcal{K}(n)$ of configurations on G such that each successive pair of configurations forms a transition according to the rules of the pebble game.*

A computation $\mathcal{K}$ is a complete computation *if and only if the first and the last configurations of $\mathcal{K}$ are empty (i.e. $\#(\mathcal{K}(1)) = \#(\mathcal{K}(n)) = 0$) and for each vertex v there exists a configuration C in $\mathcal{K}$ such that v is covered by some pebble in C.*

A complete computation on G is an abstraction of a successful solution of the problem represented by G.

We are interested in the *space* and *time* complexities of a computation $\mathcal{K}$.

Definition 2.7. *The* space *of a computation $\mathcal{K}$ (denoted as $\mathrm{S}(\mathcal{K})$) is the number of pebbles needed to perform the computation – that is the maximum number of pebbles used over all configurations of $\mathcal{K}$:*

$$\mathrm{S}(\mathcal{K}) = \max_{i=1\ldots n} \#(\mathcal{K}(i))$$

where n is the length of the computation $\mathcal{K}$.

The time *of a computation $\mathcal{K}$ (denoted as $\mathrm{T}(\mathcal{K})$) is the number of non-trivial transitions in $\mathcal{K}$.*

Note that $\mathrm{T}(\mathcal{K})$ can be less than the length of $\mathcal{K}$ in case that $\mathcal{K}$ contains trivial transitions. By removing all trivial transitions from $\mathcal{K}$ we obtain a computation of length $1 + \mathrm{T}(\mathcal{K})$.

Now we extend the definition of space and time complexities to a dag G:

Definition 2.8. *The* minimal space *of the reversible pebble game on a dag G (denoted as $\mathrm{S}_{\min}(G)$) is the minimum of $\mathrm{S}(\mathcal{K})$ over all complete computations $\mathcal{K}$ on G.*

The time $\mathrm{T}(G, s)$ *of the reversible pebble game on the dag G with* at most s pebbles *is the minimum of $\mathrm{T}(\mathcal{K})$ over all complete computations $\mathcal{K}$ on G such that $\mathrm{S}(\mathcal{K}) \leq s$.*

We denote $\mathrm{T}(G, \infty)$ as $\mathrm{T}(G)$.

Finally, we extend the definition of space and time complexities to the topology:

Definition 2.9. *Let $\mathcal{T} = (\mathcal{G}, \mathcal{M})$ be a topology. The* minimal space function $\mathrm{S}_{\min}(n)$ *of $\mathcal{T}$ is the maximum of $\mathrm{S}_{\min}(G)$ over all dags $G \in \mathcal{G}$ such that $\mathcal{M}(G) = n$.*

The time function $\mathrm{T}(n, s)$ *is the maximum of $\mathrm{T}(G, s)$ over all dags $G \in \mathcal{G}$ such that $\mathcal{M}(G) = n$.*

We denote $\mathrm{T}(n, \infty)$ as $\mathrm{T}(n)$.

2.3 Standard Pebble Game

The standard pebble game has been designed to model deterministic computations. In a deterministic computation, a value can be computed if and only if all its preceding values are known. Furthermore, a value can be erased at any time. Based on these facts we formulate the rules of the standard pebble game.

Definition 2.10. The standard pebble game *is a pebbling game that uses only one kind of pebbles. One move of the game consists of applying one of the following rules:*

1. *A pebble may be placed on a vertex whose all direct predecessors are covered by pebbles.*

2. *A pebble may be removed from any vertex.*

From these rules it follows that if two configurations form a transition they may differ in the state of at most one vertex; in case of a nontrivial transition they differ in the state of exactly one vertex. The rules of the standard pebble game are not symmetrical: if configurations C_1 and C_2 form a transition, C_2 and C_1 do not necessary form a transition, too.

2.4 Pebble Game with Black and White Pebbles

In the nondeterministic computation it is possible to "guess" any value. However, to ensure the correctness of the computation it is necessary to check whether the guess was correct.

Designed for modelling nondeterministic computations, the pebble game with black and white pebbles is played with black pebbles representing the

values computed in a deterministic way and white pebbles representing the values guessed nondeterministically. Hence, the rules of the pebble game with black and white pebbles are following:

Definition 2.11. The pebble game with black and white pebbles *is a pebbling game that uses two kind of pebbles – black and white. One move of the game consists of applying one of the following rules:*

1. *A black pebble may be placed on a vertex whose all direct predecessors are pebbled.*
2. *A black pebble may be removed from any vertex.*
3. *A white pebble may be placed on any vertex.*
4. *A white pebble may be removed from a vertex whose all direct predecessors are pebbled.*

We consider a vertex pebbled, if and only if there is a pebble of any kind placed on it.

Rules 1 and 2 deal with deterministic computation and deletion of deterministically computed values in a similar fashion as in the standard pebble game. Rule 3 models nondeterministic guess of a value and Rule 4 ensures the correctness of a nondeterministic guesses. A guessed value may be deleted only after being checked, i.e all preceding values must be known prior to deletion.

2.5 Reversible Pebble Game

A reversible computation can compute a new value in the same manner as a deterministic computation. Erasing an already computed value from the memory poses, an the other hand, a serious problem, since deleting information is not a reversible operation. If, however, all preceding values are in memory, given value can be deleted safely because it can be recomputed from its predecessors.

The reversible pebble game intended for modelling reversible computations is defined by a simple modification of the standard pebble game.

Definition 2.12. The reversible pebble game *is a pebbling game that uses one kind of pebbles. One move of the game consists of applying one of the following rules:*

1. *A pebble may be placed on a vertex whose all direct predecessors are covered by pebbles.*

2. *A pebble may be removed from a vertex whose all direct predecessors are covered by pebbles.*

The reversible pebble game differs from the standard pebble game in the rule 2 – in the standard pebble game, pebbles can be removed from any vertex at any time. This modification indicates that in reversible computation it is equally difficult to compute and to delete a value.

Similarly as in the standard pebble game, configurations that form a nontrivial transition always differ in a state of exactly one vertex.

An important property of a transition in a reversible pebble game is its *symmetry*. From the rules of the game it follows that if C_1 and C_2 form a transition, then also C_2 and C_1 form a transition.

Chapter 3

Operations on Computations

In the next chapters we shall deal mostly with reversible pebble game, although also some fundamental results concerning standard pebble game will be presented. For proving upper and lower bounds on time and space complexities of the reversible pebble game it is useful to reason formally about reversible computations. In order to do so we develop an algebraic model of computations. In this section we introduce some operations for constructing and modifying computations.

3.1 Basic operations

For changing the state of a particular vertex in a configuration we use the operation *Put*. $Put(C, v, 1)$ denotes a configuration obtained from a configuration C by pebbling a vertex v. Similarly $Put(C, v, 0)$ denotes a configuration obtained from C by unpebbling a vertex v. In the sequel we present the formal definition of Put.

Definition 3.1. *Let $G = (V, E)$ be a dag and let C be a configuration on G. Let $u, v \in V$ and $h \in \{0, 1\}$. Then* $\mathrm{Put}(C, v, h)$ *is a configuration on G defined as follows:*

- $\mathrm{Put}(C, v, h)(u) = C(u)$, *if* $u \neq v$;
- $\mathrm{Put}(C, v, h)(u) = h$, *if* $u = v$.

It follows from the symmetry of the rules of the reversible pebble game that reversing a reversible computation does not violate the reversible computation property. Thus, we can introduce an operator $\mathrm{Rev}(\mathcal{K})$ that reverses the order of configurations in the computation $\mathcal{K}$:

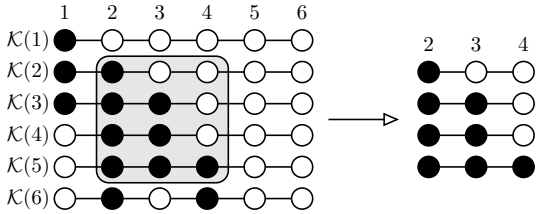

Figure 3.1: An example of a restriction – $\mathrm{Rst}(\mathcal{K}, 2, 5, \{2, 3, 4\})$

Definition 3.2. *Let $\mathcal{K}$ be a computation on G of length n. Then the* reverse *of the computation $\mathcal{K}$ (denoted as* $\mathrm{Rev}(\mathcal{K})$*) is a computation on G defined as follows:*

$$\mathrm{Rev}(\mathcal{K})(i) = \mathcal{K}(n + 1 - i)$$

Any computation on a graph G can be applied on any graph G' that is isomorphic to G. We will denote a computation $\mathcal{K}$ *applied to* the graph G' as $\mathcal{K}|G'$:

Definition 3.3. *Let $\mathcal{K}$ be a computation of length n on a dag G and let G' be a dag isomorphic to G with φ being the isomorphism between G and G'. A computation $\mathcal{K}$* applied to *the graph G' (denoted as $\mathcal{K}|G'$) is a computation on G' of length n such that $(\mathcal{K}|G')(i)(v) = \mathcal{K}(i)(\varphi(v))$ for all $1 \leq i \leq n$ and for all vertices v of G.*

3.2 Restriction

An important property of reversible computations is the following one: Let G be a dag, G' be a subgraph of G, and $\mathcal{K}$ be a computation on G. If we remove all vertices not in G' from all configurations of $\mathcal{K}$ we obtain a computation on G'. The correctness of such construction is immediate – no rule of reversible pebble game can be violated by removing a vertex from all configurations of a computation. Another important fact is that removing some configurations from the beginning and the end of a reversible computation does not violate any property of a reversible computation, either.

Hence, we can introduce an operator for a "restriction" of a computation: $\mathrm{Rst}(\mathcal{K}, i, j, V')$ denotes a computation obtained from $\mathcal{K}$ by discarding vertices other than those in V' and configurations other than those numbered i to j. We use $\mathrm{Rst}(\mathcal{K}, i, j)$ when no vertices should be removed.

For an example of a restriction see Figure 3.1. A formal definition of restriction follows:

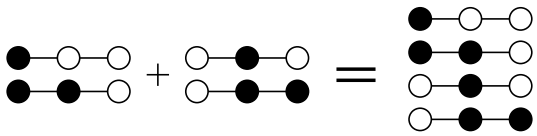

Figure 3.2: An example of a join

Definition 3.4. *Let $G = (V, E)$ be a dag, $V' \subseteq V$. Let $\mathcal{K}$ be a computation of length n on G. A* Restriction $\mathcal{K}' = \mathrm{Rst}(\mathcal{K}, i, j, V')$ *of the computation $\mathcal{K}$ to an interval $\{i \dots j\}$ $(1 \leq i \leq j \leq n)$ and to a subgraph $G' = (V', E \cap (V' \times V'))$ is a computation $\mathcal{K}'$ of length $j - i + 1$ on G' defined as follows:*

$$(\forall k \in \{1, \dots, j - i + 1\})(\forall v \in V')\mathcal{K}'(k)(v) = \mathcal{K}(i + k - 1)(v)$$

We use the notation $\mathrm{Rst}(\mathcal{K}, i, j)$ *when no vertices are removed, i.e.*

$$\mathrm{Rst}(\mathcal{K}, i, j) = \mathrm{Rst}(\mathcal{K}, i, j, V)$$

for the graph $G = (V, E)$.

3.3 Join and Merge

In the following section we introduce two operations that are in some sense inverse to restriction.

Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be computations on a dag G such that the last configuration of $\mathcal{K}_1$ and the first configuration of $\mathcal{K}_2$ form a transition. Then we can execute $\mathcal{K}_1$ first and subsequently execute $\mathcal{K}_2$. In this way we obtain a new computation $\mathcal{K}_1 + \mathcal{K}_2$ called *a join* of computations $\mathcal{K}_1$ and $\mathcal{K}_2$.

For an example of a join, see Figure 3.2. The formal definition is as follows:

Definition 3.5. *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be computations on a dag G and let $\mathcal{K}_1$ and $\mathcal{K}_2$ have length n_1 and n_2 respectively. Let $\mathcal{K}_1(n_1)$ and $\mathcal{K}_2(1)$ form a transition. Then the* join *of computations C_1 and C_2 (denoted by $\mathcal{K}_1 + \mathcal{K}_2$) is a computation on G of length $n_1 + n_2$ defined as follows:*

- $(\mathcal{K}_1 + \mathcal{K}_2)(i) = \mathcal{K}_1(i)$, *if* $i \leq n_1$
- $(\mathcal{K}_1 + \mathcal{K}_2)(i) = \mathcal{K}_2(i - n_1)$, *if* $i > n_1$

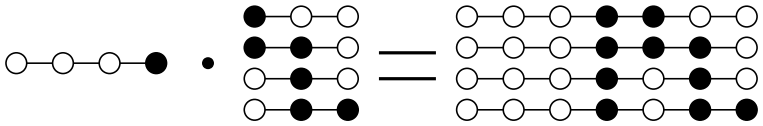

Figure 3.3: An example of a merge

It is clear that this definition is correct. Configurations $(\mathcal{K}_1+\mathcal{K}_2)(n_1)$ and $(\mathcal{K}_1+\mathcal{K}_2)(n_1+1)$ form a transition by the assumption. All other successive pairs of configurations form transitions because $\mathcal{K}_1$ and $\mathcal{K}_2$ are computations.

A configuration C can be identified with a computation $\mathcal{K}$ of length 1 such that $\mathcal{K}(1) = C$ and subsequently we may speak about a join of a computation with a configuration or vice versa.

The join of two computations is an inverse operation to restriction by removing configurations. Next we introduce an inverse operation to the restriction performed by removing vertices.

Let V_1, V_2 be disjoint sets of vertices of a dag G. Let C be a configuration on a subgraph of G induced by V_1 and let $\mathcal{K}$ be a computation on a subgraph induced by V_2. We construct a computation on a subgraph of G induced by $V_1 \cup V_2$ by adding C to each configuration in $\mathcal{K}$. This computation will be denoted as $C \cdot \mathcal{K}$ – *a merge* of computation $\mathcal{K}$ with configuration C.

To ensure that the constructed computation is correct we enforce the following constraint: if there is an edge in G from $u \in V_1$ to $v \in V_2$ the vertex u must be pebbled in C.

For an example of merge, see Figure 3.3. The formal definition of the merge operation follows.

Definition 3.6. *Let $G = (V, E)$ be a dag, $V_1 \subseteq V$, $V_2 \subseteq V$, $V_1 \cap V_2 = \emptyset$. Let C be a configuration on the graph $(V_2, E \cap (V_2 \times V_2))$. Let*

$$\{(w, v) | v \in V_1 \wedge w \in V_2 \wedge C(w) = 0 \wedge (w, v) \in E\} = \emptyset$$

Let $\mathcal{K}$ be a computation of length n on the graph $(V_1, E \cap (V_1 \times V_1))$. The computation $\mathcal{K}$ merged with the configuration C *(denoted by $\mathcal{K} \cdot C$) is a computation on the graph $(V_1 \cup V_2, E \cap ((V_1 \cup V_2) \times (V_1 \cup V_2)))$ of length n defined as follows:*

- $(\mathcal{K} \cdot C)(i)(v) = \mathcal{K}(i)(v)$, *if* $v \in V_1$
- $(\mathcal{K} \cdot C)(i)(v) = C(v)$, *if* $v \in V_2$

To see that this definition is correct analyse the possibilities to violate the rules of the reversible pebble game by adding the same configuration C to all configurations of some computation $\mathcal{K}$. The only way to introduce a violation of the rules by adding C is to add a non-pebbled direct predecessor of a vertex with a pebble laid on or removed from in $\mathcal{K}$, which is prohibited.

Chapter 4

Chain Topology

As have been already mentioned in section 2.1, the simplest topology for a pebble game is a *chain*. This topology is an abstraction of a simple straightforward computation where the result of a step $n+1$ can be computed from the result of the step n alone.

The chain topology is fundamental since it helps to understand the time and space complexities of most basic computation concepts in reversible setting. Furthermore, each deterministic computation can be viewed as a sequence of configurations, such that each next configuration can be computed from the previous one. Thus, the chain topology plays an important role in the analysis of reversible simulation of irreversible computation.

The analysis of the standard pebble game on the chain topology is easy. It is obvious that for pebbling a chain of length $n > 1$ at least two pebbles are needed. Furthermore, the computation needs at least $2n$ computation steps, a pebble must be placed to and removed from each vertex.

Now we show that each chain of length n can be pebbled by a computation of length $2n$ that uses 2 pebbles. Start by pebbling the vertex 1. Then we sequentially pebble vertex $i+1$ and unpebble vertex i for each $1 \leq i < n$. Finally, we unpebble vertex n.

Also, in the standard pebble game played on the chain topology it is possible to achieve optimal time and space complexity simultaneously, so no tradeoff analysis is required.

In this section the time and space complexity of the reversible pebble game on the chain topology is analysed. We discuss the optimal space complexity – the minimal space function $\mathrm{S}_{\min}(n)$. We also discuss lower and upper bounds on the optimal time complexity of the space optimal pebbling – the time function $\mathrm{T}(n, \mathrm{S}_{\min}(n))$ and bounds on the time-space tradeoff for the chain topology.

4.1 Optimal Space Complexity

For determining the space complexity of the reversible pebble game on the chain topology we examine the maximum length of the chain that can be pebbled by p pebbles. We denote this length by $\mathrm{S}^{-1}(p)$. It holds that $\mathrm{S}^{-1}(p) = \max\{m \mid \exists \mathcal{K} \in \mathbb{C}_{Ch(m)} : \mathrm{S}(\mathcal{K}) \leq p\}$ where $\mathbb{C}_{Ch(m)}$ is the set of all complete computations on $Ch(m)$.

The reversible pebble game on the chain topology was studied in connection with reversible simulation of irreversible computations. C. H. Bennett suggested in [1] a pebbling strategy whose special case has space complexity $\Theta(\lg n)$. The space optimality of this algorithm was proved in [4], [5] and [6]. In the sequel we present an alternative proof using our proof technique.

In proofs of time-space complexity of reversible pebble game on chain the concept of the *middle vertex* is useful:

Definition 4.1. *Let $\mathcal{K}$ be a complete computation on chain of length n that uses p pebbles. Let us consider the pebbled vertex with the minimal number in each configuration of $\mathcal{K}$. Let x be the maximum of all these numbers. Formally, x is defined as follows: (In this case, the minimum of an empty set is defined as 0.)*

$$x = \max\{\min\{j | j \in \{1 \ldots n\} \wedge \mathcal{K}(i)(j) = 1\} | i \in \{1 \ldots l\}\}$$

Then x will be called the middle vertex of computation $\mathcal{K}$.

We prove an important property of the middle vertex:

Lemma 4.2. *Let $\mathcal{K}$ be a complete computation on a chain of length n that uses at most p pebbles. Let x be a middle vertex of $\mathcal{K}$. It holds that $n - x \leq \mathrm{S}^{-1}(p-1)$. Furthermore, it holds that $x - 1 \leq \mathrm{S}^{-1}(p-1)$.*

Proof. Let l be the length of $\mathcal{K}$. Let us consider the computation $\mathcal{K}_1 = \mathrm{Rst}(\mathcal{K}, 1, l, \{x+1 \ldots n\})$. It is clear that $\mathcal{K}$ is a complete computation on the graph $Ch(n)$ restricted to vertices $\{x+1 \ldots n\}$ which is isomorphic to $Ch(n-x)$. From the fact that $\mathrm{S}(\mathcal{K}) \leq p$ and from the definition of the middle vertex it follows that $\mathrm{S}(\mathcal{K}_1) \leq p-1$. So, from the assumptions of the theorem, we have $n - x \leq \mathrm{S}^{-1}(p-1)$ (see also Figure 4.1).

The definition of x ensures the existence of some a such that in the configuration $\mathcal{K}(a)$ the vertex x is pebbled and no vertex with number less than x is pebbled: $x = \min\{j | j \in \{1 \ldots n\} \wedge \mathcal{K}(a)(j) = 1\}$.

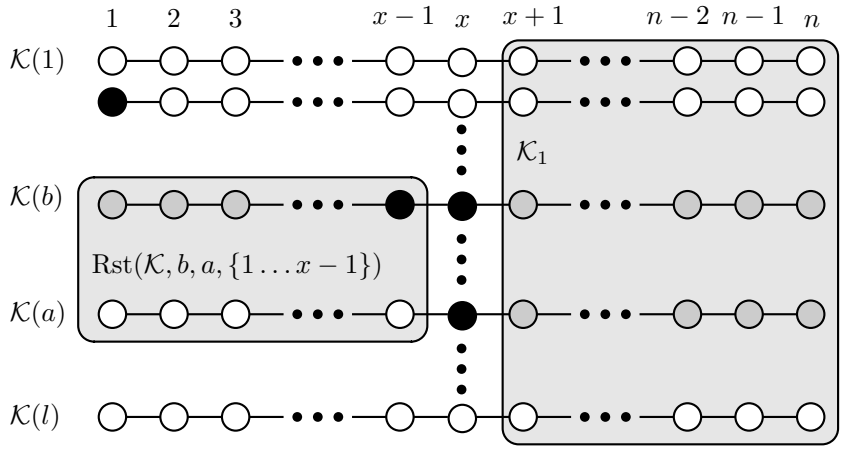

Figure 4.1: Proof of Lemma 4.2. Gray circles are vertices with unknown state.

Let b be the minimal number such that vertex x is pebbled in all configurations $\mathcal{K}(b), \ldots, \mathcal{K}(a)$:

$$b = \min\{i | i \in \{1, \ldots, a\} \wedge (\forall j \in \{i, \ldots, a\})\mathcal{K}(j)(x) = 1\}$$

Since $\mathcal{K}$ is a complete computation we have $\mathcal{K}(1)(x) = 0$ and hence $b > 1$. Also, $\mathcal{K}(b-1)(x) = 0$ from the minimality of b. Because $\mathcal{K}$ is a computation it holds that $\mathcal{K}(b-1)(x-1) = \mathcal{K}(b)(x-1) = 1$.

Let $\mathcal{K}_2$ be a computation defined as follows:

$$\mathcal{K}_2 = \mathrm{Rev}(\mathrm{Rst}(\mathcal{K}, b, a, \{1 \ldots x-1\})) + \mathrm{Rst}(\mathcal{K}, b, a, \{1 \ldots x-1\})$$

It is easy to see that $\mathcal{K}_2$ is a complete computation on the graph $Ch(x-1)$. Furthermore, because $\mathrm{S}(\mathcal{K}) \leq p$ and $\mathcal{K}_2$ uses at least one pebble less than $\mathcal{K}$, it holds $\mathrm{S}(\mathcal{K}_2) \leq p-1$ and from the assumptions of the theorem we have $x - 1 \leq \mathrm{S}^{-1}(p-1)$. □

We prove an upper bound on the function $\mathrm{S}^{-1}(p)$ which implies a lower bound on $\mathrm{S}_{\min}(p)$.

Lemma 4.3. $\mathrm{S}^{-1}(p+1) \leq 2\mathrm{S}^{-1}(p) + 1$

Proof. Assume the statement of Lemma 4.3 does not hold. Then there exists a complete computation $\mathcal{K}$ on a chain of length $n > 2\mathrm{S}^{-1}(p) + 1$, such that $\mathrm{S}(\mathcal{K}) \leq p + 1$. Let l be the length of the computation $\mathcal{K}$.

Let x be the middle point of $\mathcal{K}$. From Lemma 4.2 it follows that $n - x \leq \mathrm{S}^{-1}(p)$ and $x - 1 \leq \mathrm{S}^{-1}(p)$. These two inequalities immediately imply the fact that $n \leq 2\mathrm{S}^{-1}(p) + 1$ which is a contradiction. □

Next we prove a lower bound $\mathrm{S}^{-1}(p)$ which implies an upper bound on $\mathrm{S}_{\min}(p)$.

Lemma 4.4. $\mathrm{S}^{-1}(p+1) \geq 2\mathrm{S}^{-1}(p) + 1$

Proof. The definition of $\mathrm{S}^{-1}(p)$ ensures the existence of a complete computation $\mathcal{K}_1$ of length l on the chain of length $n = \mathrm{S}^{-1}(p)$ such that $\mathrm{S}(\mathcal{K}_1) \leq p$. Let G_1 be a graph obtained from the chain of length n by renaming all its vertices from $1 \ldots n$ to $n+2 \ldots 2n+1$. Clearly the graphs G_1 and Ch_n are isomorphic. Let $G_2 = (\{n+1\}, \emptyset)$. Hence, Ch_n, G_1 and G_2 are subgraphs of Ch_{2n+1}. Because $\mathcal{K}_1$ is a complete computation on Ch_n, there exists a configuration $\mathcal{K}_1(i)$ such that $\mathcal{K}_1(i)(n) = 1$.

Now we can construct a complete computation $\mathcal{K}_2$ on the graph Ch_{2n+1} such that $\mathrm{S}(\mathcal{K}_2) \leq \mathrm{S}(\mathcal{K}_1) + 1 \leq p + 1$ and thus conclude the proof.

At first we pebble vertices $1, \ldots, n$ of the chain by the computation

$$\mathrm{Rst}(\mathcal{K}_1, 1, i) \cdot \mathrm{E}(G_2) \cdot \mathrm{E}(G_1)$$

This part of the computation uses at most $\mathrm{S}(\mathcal{K}_1)$ pebbles. Then we pebble the vertex $n+1$ and unpebble all other vertices by the computation

$$\mathrm{Rst}(\mathcal{K}_1, i, l) \cdot \mathrm{Put}(\mathrm{E}(G_2), n+1, 1) \cdot \mathrm{E}(G_1)$$

In this phase we use at most $\mathrm{S}(\mathcal{K}_1) + 1$ pebbles. In the third part we will pebble vertices $n+2 \ldots 2n+1$ by a computation

$$(\mathcal{K}_1|G_1) \cdot \mathrm{E}(G_1) \cdot \mathrm{Put}(\mathrm{E}(G_2), n+1, 1)$$

that uses exactly $\mathrm{S}(\mathcal{K}_1) + 1$ pebbles. Finally, we remove all pebbles analogically to the first two steps. So the computation $\mathcal{K}_2$ is as follows (see also Figure 4.2):

$$\begin{aligned}
\mathcal{K}_2 &= \mathrm{Rst}(\mathcal{K}_1, 1, i) \cdot \mathrm{E}(G_2) \cdot \mathrm{E}(G_1) + \\
&+ \mathrm{Rst}(\mathcal{K}_1, i, l) \cdot \mathrm{Put}(\mathrm{E}(G_2), n+1, 1) \cdot \mathrm{E}(G_1) + \\
&+ (\mathcal{K}_1|G_1) \cdot \mathrm{E}(Ch_n) \cdot \mathrm{Put}(\mathrm{E}(G_2), n+1, 1) + \\
&+ \mathrm{Rst}(\mathcal{K}_1, 1, i) \cdot \mathrm{Put}(\mathrm{E}(G_2), n+1, 1) \cdot \mathrm{E}(G_1) + \\
&+ \mathrm{Rst}(\mathcal{K}_1, i, l) \cdot \mathrm{E}(G_2) \cdot \mathrm{E}(G_1)
\end{aligned}$$

The computation $\mathcal{K}_2$ uses at most $\mathrm{S}(\mathcal{K}_1) + 1 \leq p + 1$ pebbles. □

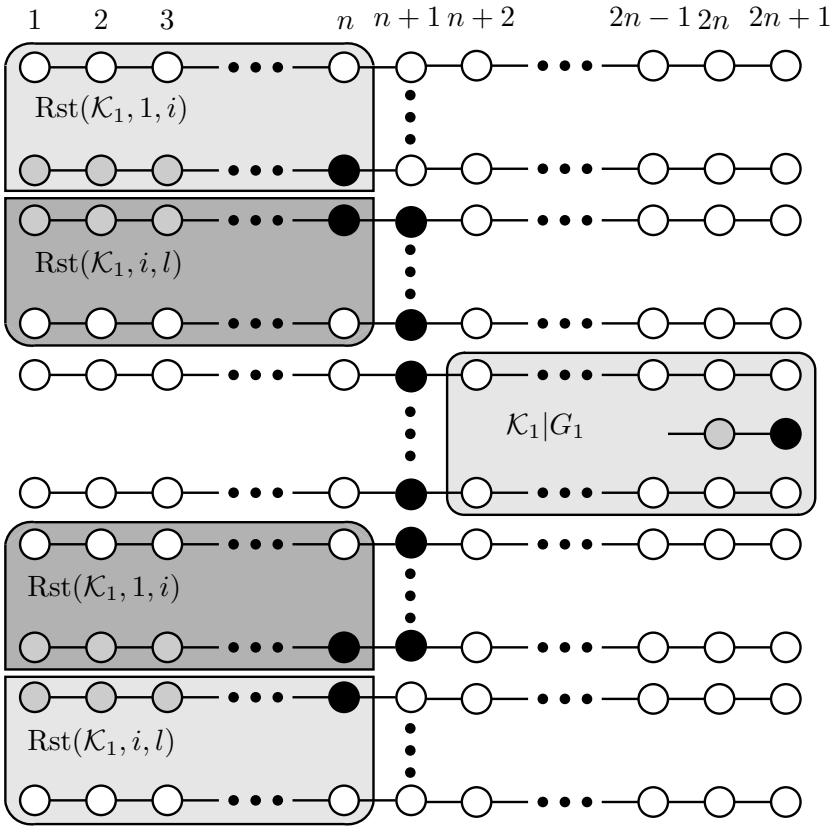

Figure 4.2: A computation $\mathcal{K}_2$ where $n = \mathrm{S}^{-1}(p)$.

Corollary 4.5. *It holds that* $\mathrm{S}^{-1}(p) = 2^p - 1$. *It follows that for the minimal space function of the chain topology* $\mathrm{S}_{\min}(n)$ *it holds*

$$\mathrm{S}_{\min}(n) = \lceil \lg(n+1) \rceil = \Theta(\lg n)$$

where lg *denotes the logarithm of base 2.*

Proof. From Lemma 4.3 and Lemma 4.4 it follows that $\mathrm{S}^{-1}(p+1) = 2\mathrm{S}^{-1}(p)+1$. It is obvious that $\mathrm{S}^{-1}(1) = 1$. By solving this recurrence we obtain $\mathrm{S}^{-1}(p) = 2^p - 1$. The minimal space function is clearly monotonous, and so from the definition of S^{-1} it follows that $\mathrm{S}_{\min}(n) = p$ such that $\mathrm{S}^{-1}(p-1) < n \leq \mathrm{S}^{-1}(p)$. It is easy to verify that $\mathrm{S}_{\min}(n) = \lceil (\mathrm{S}^{-1}(n))^{-1} \rceil$. □

4.2 Optimal Time and Space Complexity

To show that the optimal time and the optimal space complexity cannot be achieved simultaneously we examine the following question: Given the minimal number of pebbles needed to perform the complete computation, what is the minimal time needed to perform it? This minimal time for space optimal reversible computation is described by the function $\mathrm{T}(n, \mathrm{S}_{\min}(n))$.

An upper bound on time for the space optimal reversible computation on a chain topology follows from the results presented in [5]. As for lower bounds, we prove that for infinitely many n it holds $\mathrm{T}(n, \mathrm{S}_{\min}(n)) = \Omega(n \lg n)$ (published in [8] and [9]). Hence, the function $\mathrm{T}(n, \mathrm{S}_{\min}(n))$ must be larger than the optimal time $\mathrm{T}(n)$, and so it makes sense to discuss tradeoffs between the time and space complexity.

This result, although discovered independently, uses a similar idea to that presented in [7].

An important fact is that any time optimal computation must be symmetric. The time of its subcomputation to the point where the vertex with the maximal number is pebbled is half of the time of the whole computation.

Lemma 4.6. *Let* $\mathcal{K}$ *be a complete computation of length* l *on* Ch_n, $\mathrm{S}(\mathcal{K}) = \mathrm{S}_{\min}(n)$, $\mathrm{T}(\mathcal{K}) = \mathrm{T}(n, \mathrm{S}_{\min}(n))$. *Let* $i = \min\{i | i \in \{1 \ldots l\} \wedge \mathcal{K}(i)(n) = 1\}$. *Then it holds that*

$$\mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i)) = \mathrm{T}(\mathrm{Rst}(\mathcal{K}, i, l)) = \frac{\mathrm{T}(\mathcal{K})}{2}$$

This lemma obviously follows from the reversibility of $\mathcal{K}$. However, we present a detailed proof, too.

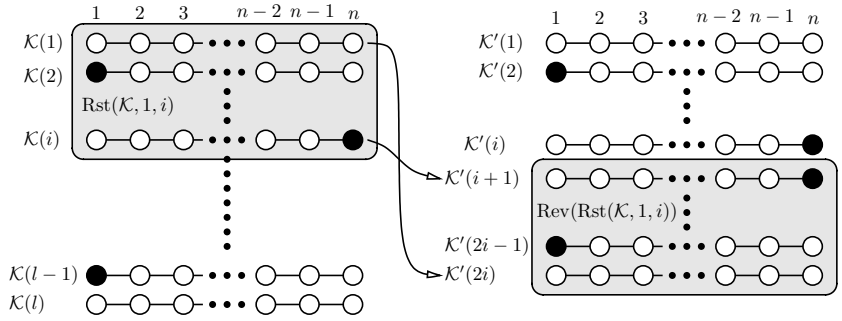

Figure 4.3: Proof of Lemma 4.6

Proof of Lemma 4.6. Assume that the first equality does not hold. Without loss of generality we can assume that $T(Rst(\mathcal{K}, 1, i)) < T(Rst(\mathcal{K}, i, l))$ (otherwise we can consider $Rev(\mathcal{K})$ instead of $\mathcal{K}$). Let us consider the computation $\mathcal{K}' = Rst(\mathcal{K}, 1, i) + Rev(Rst(\mathcal{K}, 1, i))$ (see Figure 4.3). $\mathcal{K}'$ is a complete computation on Ch_n. Clearly $S(\mathcal{K}') \leq S(\mathcal{K})$ and $T(\mathcal{K}') < T(\mathcal{K})$. But this contradicts the fact that $T(\mathcal{K}) = T(n, S_{\min}(n))$.

We have proven that $T(Rst(\mathcal{K}, 1, i)) = T(Rst(\mathcal{K}, i, l))$. Clearly, $T(\mathcal{K}) = T(Rst(\mathcal{K}, 1, i)) + T(Rst(\mathcal{K}, i, l))$, so the second equality is obvious. □

In the sequel we prove a lower bound on the time for the space optimal computation. We consider only chains of length $S^{-1}(p)$. A chain has length $S^{-1}(p)$ for some p if and only if it can be pebbled by p pebbles but each longer chain requires more than p pebbles. We prove a recurrent inequality that gives a lower bound for $T(S^{-1}(p), p)$:

Theorem 4.7. $T(S^{-1}(p+1), p+1) \geq 2S^{-1}(p) + 2 + 2T(S^{-1}(p), p)$

Proof. Let $\mathcal{K}$ be a time optimal complete computation on $Ch_{S^{-1}(p+1)}$ such that $S(\mathcal{K}) = p + 1$. Clearly $T(\mathcal{K}) = T(S^{-1}(p+1), p+1)$. We prove that $T(\mathcal{K}) \geq 2S^{-1}(p) + 2 + 2T(S^{-1}(p), p)$.

Let l be the length of $\mathcal{K}$ and $n = S^{-1}(p)$. According to the Theorem 4.5 it holds $S^{-1}(p+1) = 2n + 1$. We define i as follows:

$$i = \min\{i | i \in \{1 \ldots l\} \wedge \mathcal{K}(i)(2n+1) = 1\}$$

By Lemma 4.6 it holds $T(Rst(\mathcal{K}, 1, i)) = \frac{1}{2}T(\mathcal{K})$. We prove that the time needed for the first half of $\mathcal{K}$ is at least $n + 1 + T(n, p)$. The main idea of the proof is to partition the computation into parts such that each transition in the computation occurs in at most one part. The time of the computation then has to be at least the sum of the times of the parts.

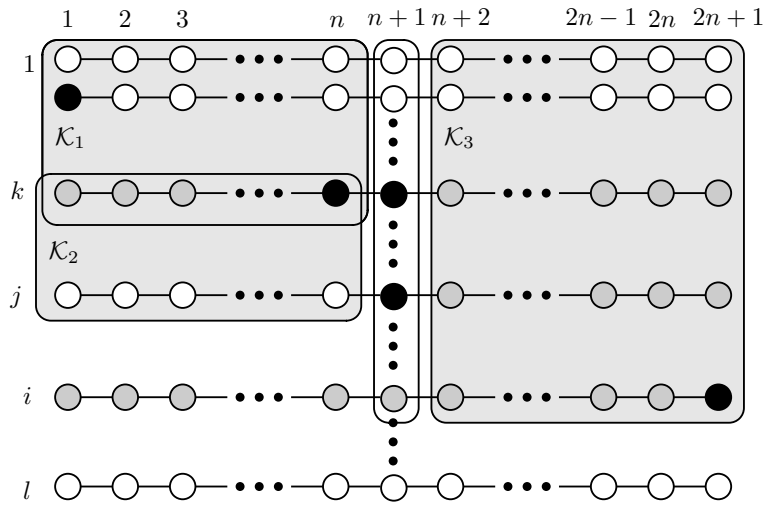

Figure 4.4: Computations $\mathcal{K}$, $\mathcal{K}_1$, $\mathcal{K}_2$ and $\mathcal{K}_3$. Gray circles are vertices with unknown state.

From Lemma 4.2 and Theorem 4.5 it follows that for each k at least one of the first $n+1$ vertices in $\mathcal{K}(k)$ is pebbled and that there exists j such that the first pebbled vertex in $\mathcal{K}(j)$ is the vertex number $n+1$. We can assume w.l.o.g. that $j \leq i$ (otherwise we can replace $\mathcal{K}$ by $\mathrm{Rev}(\mathcal{K})$). Let k be a configuration such that $\mathcal{K}(k-1)(n+1) = 0$ and $(\forall q : k \leq q \leq j)\mathcal{K}(q)(n+1) = 1$. Clearly $\mathcal{K}(k-1)(n) = \mathcal{K}(k)(n) = 1$.

Let $\mathcal{K}_1 = \mathrm{Rst}(\mathcal{K}, 1, k, \{1 \ldots n\})$, $\mathcal{K}_2 = \mathrm{Rst}(\mathcal{K}, k, j, \{1 \ldots n\})$ and $\mathcal{K}_3 = \mathrm{Rst}(\mathcal{K}, 1, i, \{n+2 \ldots 2n+1\})$. See Figure 4.4.

It holds that

$$\mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i)) = \mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i, \{1 \ldots n\})) + \mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i, \{n+1\})) + \mathrm{T}(\mathcal{K}_3)$$

because each nontrivial transition has to occur in exactly one of the three parts of $\mathcal{K}$. Analogically it holds that

$$\mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i, \{1 \ldots n\}) = \mathrm{T}(\mathcal{K}_1) + \mathrm{T}(\mathcal{K}_2) + \mathrm{T}(\mathrm{Rst}(\mathcal{K}, j, i, \{1 \ldots n\}))$$

Thus, it holds that

$$\mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i)) \geq \mathrm{T}(\mathcal{K}_1) + \mathrm{T}(\mathcal{K}_2) + \mathrm{T}(\mathcal{K}_3) + \mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i, \{n+1\}))$$

Trivially, $\mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i, \{n+1\})) \geq 1$ and $\mathrm{T}(\mathcal{K}_1) \geq n$.

$\mathrm{Rev}(\mathcal{K}_2) + \mathcal{K}_2$ is a complete computation on Ch_n. Since $(\forall q : k \leq q \leq j)$ $\mathcal{K}(q)(n+1) = 1$, it holds that $\mathrm{S}(\mathrm{Rev}(\mathcal{K}_2) + \mathcal{K}_2) = p$. Hence,

$$2\mathrm{T}(\mathcal{K}_2) = \mathrm{T}(\mathrm{Rev}(\mathcal{K}_2) + \mathcal{K}_2) \geq \mathrm{T}(n, p)$$

Similarly, $\mathcal{K}_3 + \mathrm{Rev}(\mathcal{K}_3)$ is a complete computation on a graph isomorphic to Ch_n. Due to Lemma 4.2, $\mathrm{S}(\mathcal{K}_3 + \mathrm{Rev}(\mathcal{K}_3)) = p$. Hence, we have

$$2\mathrm{T}(\mathcal{K}_3) = \mathrm{T}(\mathcal{K}_3 + \mathrm{Rev}(\mathcal{K}_3)) \geq \mathrm{T}(n, p)$$

Putting everything together yields

$$\mathrm{T}(\mathrm{Rst}(\mathcal{K}, 1, i, \{1 \ldots n\}) \geq 1 + n + \mathrm{T}(n, p)$$

Thus, $\mathrm{T}(\mathcal{K}) \geq 2\mathrm{S}^{-1}(p) + 2 + 2\mathrm{T}(\mathrm{S}^{-1}(p), p)$. □

Corollary 4.8. $\mathrm{T}(n, \mathrm{S}_{\min}(n)) = O(n^{\log_2 3})$, $\mathrm{T}(n, \mathrm{S}_{\min}(n)) \neq o(n \lg n)$,

Proof. The upper bound, presented also in [5], can be easily obtained from the analysis of the proof of Lemma 4.4. The pebbling strategy implied by the proof ensures that $\mathrm{T}(2^{p+1}-1, p+1) \leq 3\mathrm{T}(2^p-1, p)+2$. Since $\mathrm{T}(2^1-1, 1) = 2$, it holds that $\mathrm{T}(2^p - 1, p) \leq 3^p - 1$. It is clear that for each n such that $2^{p-1} - 1 < n \leq 2^p - 1$ it holds that $\mathrm{T}(n, p) \leq \mathrm{T}(2^p - 1, p)$. Hence, it holds that $\mathrm{T}(n, \mathrm{S}_{\min}(n)) = O(n^{\log_2 3})$.

By solving the recurrence from the preceding theorem we obtain that $T(n, \mathrm{S}_{\min}(n)) = \Omega(n \lg n)$ for $n = 2^p - 1$. Since this function is a restriction of $T(n, \mathrm{S}_{\min}(n))$, the function $T(n, \mathrm{S}_{\min}(n))$ cannot be[1] $o(n \lg n)$. □

4.3 Upper Bound on Time-Space Tradeoff

In the previous section the time complexity of a reversible pebbling for space optimal computations was analysed. Now we discuss the time complexity for computations that are not space optimal.

Bennett's pebbling strategy introduced in [1] can be used to derive upper bounds on the time-space tradeoff for the chain topology. This strategy pebbles a chain of length ml^k in time $m(2l-1)^k$ using $m + k(l-1)$ pebbles. Choosing $m = 1$ and $k = \lg_l n$ for a fixed l we obtain a tradeoff in the following form: space $O(\lg n)$ versus time $O(n^{\frac{\lg(2l-1)}{\lg l}})$. Choosing $m = 1$ and $l = \sqrt[k]{n}$ for a fixed k we obtain a tradeoff in the form: space $O(\sqrt[k]{n})$ versus time $O(n)$. See also [3] where these results are formulated in terms of reversible simulation of irreversible computation with the emphasis on obtaining the constant factors of the asymptotic terms.

[1] Note that the function $T(n, \mathrm{S}_{\min}(n))$ is not monotonic, so we cannot formulate the result in a form $T(n, \mathrm{S}_{\min}(n)) = \Omega(n \lg n)$. We can only state that $T(n, \mathrm{S}_{\min}(n)) = \Omega(n \lg n)$ holds for infinitely many n.

As an example of our proof technique we present an alternative proof of the second tradeoff.

It is obvious that for any complete computation $\mathcal{K}$ on Ch_n it holds $\mathrm{T}(\mathcal{K}) \geq 2n$, because each vertex has to be pebbled at least once and unpebbled at least once. So it is easy to see that the space of a complete computation $\mathcal{K}$ such that $\mathrm{T}(\mathcal{K}) = 2n$ is exactly n.

We will analyse the space complexity of complete computations on Ch_n that are running in time at most $c \cdot n$. We denote $\mathrm{S}^{-1}(c, p)$ the maximal length of a chain that can be pebbled by p pebbles in time at most c times its length. Formally, $\mathrm{S}^{-1}(c, p) = \max\{n | \exists \mathcal{K} \in \mathbb{C}_{Ch(n)}, \mathrm{S}(\mathcal{K}) \leq p \wedge \mathrm{T}(\mathcal{K}) \leq cn\}$ where $\mathbb{C}_{Ch(n)}$ is the set of all complete computations on Ch_n.

Theorem 4.9. *It holds that* $\mathrm{S}^{-1}(2^k, p) \geq \binom{p}{k}$.

Proof. We prove the statement by induction on p. The base case $\mathrm{S}^{-1}(2^k, 1) \geq \binom{1}{k}$ holds trivially.

For $k = 1$ the inequality $\mathrm{S}^{-1}(2^1, p) \geq \binom{p}{1}$ holds. (It is easy to make a complete computation $\mathcal{K}$ on Ch_p satisfying $\mathrm{S}(\mathcal{K}) = p$ and $\mathrm{T}(\mathcal{K}) = 2p$.)

Let $p > 1$ and $k > 1$. Let us assume by induction that $\mathrm{S}^{-1}(2^{k-1}, p-1) \geq \binom{p-1}{k-1}$ and $\mathrm{S}^{-1}(2^k, p-1) \geq \binom{p-1}{k}$. We prove that $\mathrm{S}^{-1}(2^k, p) \geq \binom{p}{k}$.

Let $n = \mathrm{S}^{-1}(2^{k-1}, p-1)$ and let $m = \mathrm{S}^{-1}(2^k, p-1)$. We construct a complete computation $\mathcal{K}$ on a chain of length $n + 1 + m$ as follows: pebble n vertices using $p-1$ pebbles in time $2^{k-1}n$ and put the p-th pebble on the vertex number $n+1$. Then pebble next m vertices (i.e. vertices numbered $n+2, \ldots, n+1+m$) using $p-1$ pebbles in time $2^k m$ with the p-th pebble placed on the vertex $n+1$. Finally, pebble first n vertices again and remove the p-th pebble from the vertex $n+1$.

To formally describe this computation, let $\mathcal{K}_1$ be a complete computation on Ch_n such that $\mathrm{S}(\mathcal{K}_1) \leq p-1$ and $\mathrm{T}(\mathcal{K}_1) \leq 2^{k-1}n$. Let us denote the length of $\mathcal{K}_1$ by l_1. Let i be such that $\mathcal{K}_1(i)(n) = 1$. Let $\mathcal{K}_2$ be a complete computation on Ch_m such that $\mathrm{S}(\mathcal{K}_2) \leq p-1$ and $\mathrm{T}(\mathcal{K}_2) \leq 2^k m$.

Let G_1 be a graph $(\{n+1\}, \emptyset)$. Let G_2 be a graph obtained from Ch_n by renaming its vertices to $n+2, \ldots, n+1+m$.

The computation $\mathcal{K}$ can be defined as follows (see also Figure 4.5):

$$\begin{aligned}
\mathcal{K} = {} & \mathrm{Rst}(\mathcal{K}_1, 1, i) \cdot \mathrm{E}(G_1) \cdot \mathrm{E}(G_2) + \\
& + \mathrm{Rst}(\mathcal{K}_1, i, l_1) \cdot \mathrm{Put}(\mathrm{E}(G_1), n+1, 1) \cdot \mathrm{E}(G_2) + \\
& + (\mathcal{K}_2 | G_2) \cdot \mathrm{E}(Ch_n) \cdot \mathrm{Put}(\mathrm{E}(G_1), n+1, 1) + \\
& + \mathrm{Rev}(\mathrm{Rst}(\mathcal{K}_1, i, l_1)) \cdot \mathrm{Put}(\mathrm{E}(G_1), n+1, 1) \cdot \mathrm{E}(G_2) + \\
& + \mathrm{Rev}(\mathrm{Rst}(\mathcal{K}_1, 1, i)) \cdot \mathrm{E}(G_1) \cdot \mathrm{E}(G_2)
\end{aligned}$$

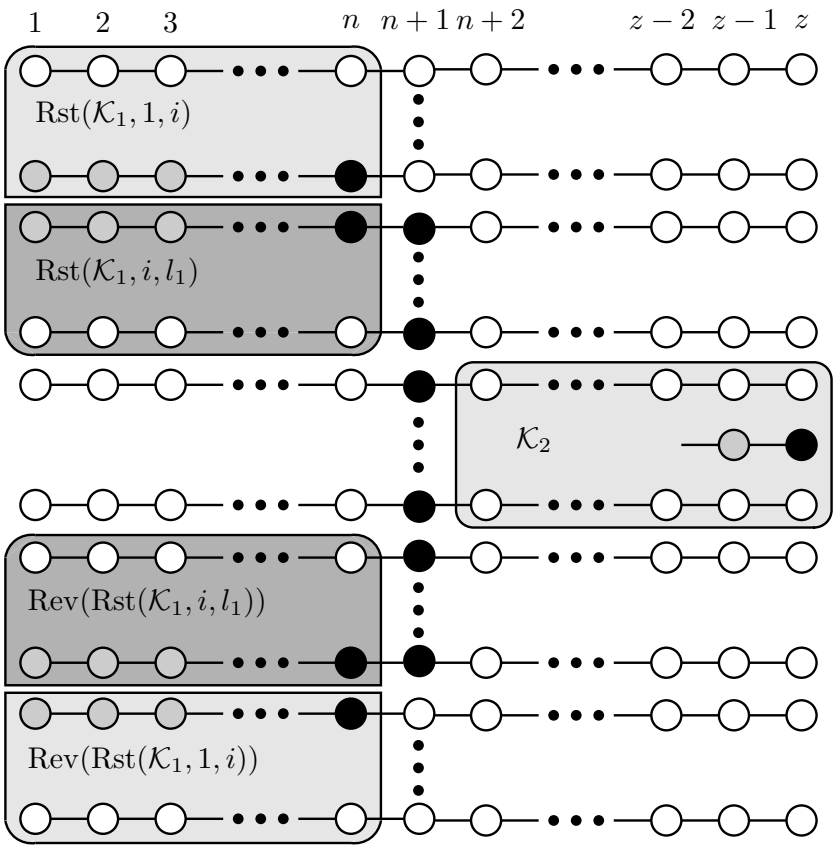

Figure 4.5: A computation $\mathcal{K}$ where $n = \mathrm{S}^{-1}(2^{k-1}, p-1)$, $m = \mathrm{S}^{-1}(2^k, p-1)$ and $z = n + 1 + m$.

Clearly $\mathcal{K}$ is a complete computation on Ch_{n+1+m}, $\mathrm{S}(\mathcal{K}) \leq p$ and it holds that

$$\begin{aligned} \mathrm{T}(\mathcal{K}) &\leq 2\mathrm{T}(\mathcal{K}_1) + 2 + \mathrm{T}(\mathcal{K}_2) \leq 2^k n + 2 + 2^k m \leq 2^k(n+1+m) = \\ &= 2^k(\mathrm{S}^{-1}(2^{k-1}, p-1) + 1 + \mathrm{S}^{-1}(2^k, p-1)) \end{aligned}$$

Thus, $\mathrm{S}^{-1}(2^k, p) \geq \mathrm{S}^{-1}(2^{k-1}, p-1) + 1 + \mathrm{S}^{-1}(2^k, p-1)$.

By applying the induction hypothesis we have

$$\mathrm{S}^{-1}(2^k, p) \geq \binom{p-1}{k-1} + \binom{p-1}{k} = \binom{p}{k}$$

□

Corollary 4.10. *Let k be fixed. $O(\sqrt[k]{n})$ pebbles are sufficient for a complete $O(n)$ time computation on Ch_n.*

Chapter 5

Binary Tree Topology

In this section we discuss the space complexity of the standard and reversible pebble game on complete binary trees. This topology represents a class of problems where the result can be computed from two different subproblems. The definition of the binary tree topology has been presented in 2.2.

As mentioned in Section 2.2, we denote the minimal number of pebbles needed to perform a complete computation on $\mathrm{Bt}(h)$ by $\mathrm{S}_{\min}(h)$. To simplify some of the proofs we also consider the minimal number of pebbles needed to perform a computation from the empty configuration to a configuration where only the root is pebbled.

Definition 5.1. *Let $\mathcal{K}$ be a computation of length l on $\mathrm{Bt}(h)$. Let $\mathcal{K}(1) = \mathrm{E}(\mathrm{Bt}(h))$ and $\mathcal{K}(l) = \mathrm{Put}(\mathrm{E}(\mathrm{Bt}(h)), \mathrm{R}(\mathrm{Bt}(h)), 1)$. Then $\mathcal{K}$ is called a* semicomplete computation.

The minimal number of pebbles needed to perform a semicomplete computation on $\mathrm{Bt}(h)$ (i.e. $\min\{\mathrm{S}(\mathcal{K})\}$, where $\mathcal{K}$ is a semicomplete computation) will be denoted as $\mathrm{S}'_{\min}(h)$.

5.1 Optimal Space Complexity of the Standard Pebble Game

The exact value of the space complexity of the standard pebble game played on the binary tree topology has been proved by Paterson and Hewitt in [10]. The main idea of this proof – a concept of open and closed configurations – is used in the analysis of the optimal space complexity of the reversible pebble game, too. In this section the proof is commemorated.

Definition 5.2. *Let C be a configuration on a complete binary tree T. C is called* open*, if and only if there exists a path P from the root vertex of T to some leaf of T, such that no vertex of P is pebbled.*

Configuration C that is not open is called closed*.*

In the sequel we prove that the minimal space complexity of the standard pebble game played on a complete binary tree $\mathrm{Bt}(h)$ of height h is at least $h+1$.

Theorem 5.3. *For the space complexity of the standard pebble game on a binary tree topology it holds that $\mathrm{S}_{\min}(h) \geq h+1$ for $h > 1$.*

Proof. Let $\mathcal{K}$ be a space optimal complete computation on $\mathrm{Bt}(h)$. We prove that $\mathrm{S}(\mathcal{K}) \geq h+1$. Because $\mathcal{K}(1)$ is an empty configuration, it is an open configuration. Since $\mathcal{K}$ is a complete computation, there exists some configuration $\mathcal{K}(a)$ in computation $\mathcal{K}$, such that root vertex of $\mathrm{Bt}(h)$ is pebbled in $\mathcal{K}(a)$. Thus, configuration $\mathcal{K}(a)$ is closed (see also Figure 5.1).

Let $\mathcal{K}(i)$ be the last open configuration in $\mathcal{K}$ such that $i < a$. There exists some path from the root vertex of $\mathrm{Bt}(h)$ to some leaf of $\mathrm{Bt}(h)$ such that no vertex of this path is pebbled. We may assume without loss of generality that this path is the rightmost path of $\mathrm{Bt}(h)$, i.e. it is formed by vertices $\mathrm{R}(\mathrm{Bt}(h))$, $\mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(h)))$, …, $\mathrm{R}(\mathrm{Rt}^{h-1}(\mathrm{Bt}(h)))$.

Let $\mathcal{K}(j)$ be first computation such that $i < j < a$ and that the vertex $\mathrm{R}(\mathrm{Rt}^{h-2}(\mathrm{Bt}(h)))$ is pebbled in $\mathcal{K}(j)$. Because $\mathcal{K}$ is a computation, both predecessors of vertex $\mathrm{R}(\mathrm{Rt}^{h-2}(\mathrm{Bt}(h)))$ have to be pebbled. As the configuration $\mathcal{K}(j)$ is closed, there must be at least one vertex pebbled in each of the subtrees $\mathrm{Lt}(\mathrm{Bt}(h))$, $\mathrm{Lt}(\mathrm{Rt}(\mathrm{Bt}(h)))$, …, $\mathrm{Lt}(\mathrm{Rt}^{h-3}(\mathrm{Bt}(h)))$.

Hence, there are at least $h-2+3 = h+1$ pebbled vertices in the configuration $\mathcal{K}(j)$ and the inequality $\mathrm{S}(\mathcal{K}) \geq h+1$ holds. □

We have proven that the standard pebble game played on a binary tree of height $h > 1$ requires at least $h+1$ pebbles. Next we prove that $h+1$ pebbles suffice.

Theorem 5.4. *For the space complexity of the standard pebble game on a binary tree topology it holds that $\mathrm{S}_{\min}(h) \leq h+1$ for $h > 1$.*

Proof. We prove by induction that $h+1$ pebbles are sufficient to construct a semicomplete computation on a complete binary tree of height h. The base case is trivial – three pebbles suffice to perform a semicomplete computation on $\mathrm{Bt}(2)$.

Assume by induction that it is possible to perform a semicomplete computation on $\mathrm{Bt}(h)$ using at most $h+1$ pebbles. We can construct a semicomplete

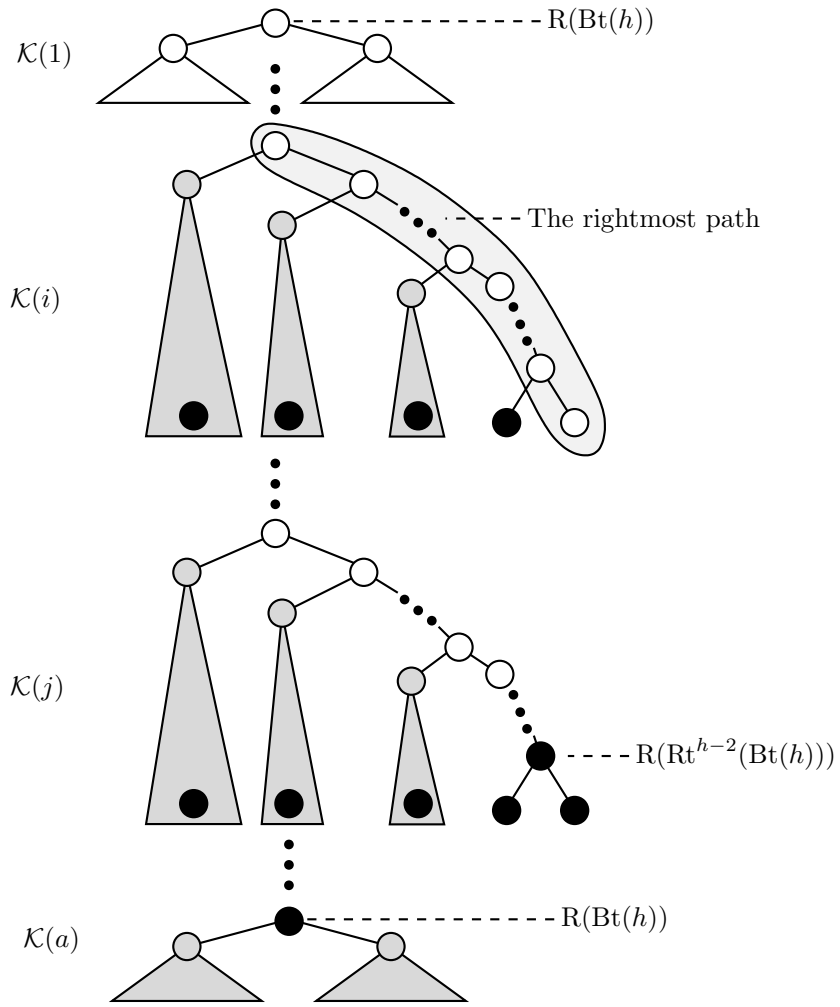

Figure 5.1: The proof of the lower bound on space complexity of the standard pebble game played on a binary tree.

computation on $\mathrm{Bt}(h+1)$ using $h+2$ pebbles as follows. At first we perform a semicomplete computation on the left subtree of $\mathrm{Bt}(h+1)$. This stage requires $h+1$ pebbles. Next, we perform a semicomplete computation on the right subtree of $\mathrm{Bt}(h+1)$, while the root vertex of the left subtree remains pebbled. This can be done with $h+2$ pebbles. Finally, we pebble the root vertex of $\mathrm{Bt}(h+1)$ and remove the pebbles from $\mathrm{R}(\mathrm{Lt}(\mathrm{Bt}(h+1)))$ and $\mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(h+1)))$. This part of the computation requires 3 pebbles. Hence, $h+2$ pebbles suffice to perform a semicomplete computation on $\mathrm{Bt}(h+1)$.

The fact $\mathrm{S}_{\min}(h) \leq h+1$ is now obvious. A complete computation on $\mathrm{Bt}(h)$ can be obtained by performing a semicomplete computation on $\mathrm{Bt}(h)$ and removing the pebble from the root vertex. □

5.2 Relationship between Space Complexities of Complete and Semicomplete Computations

In the following sections of this chapter we analyse the space complexity of the reversible pebble game on the binary tree topology. As we rely heavily on the concept of semicomplete computations, it is important to understand the relationship between space complexities of complete and semicomplete reversible computations on binary trees.

In the following two lemmas we prove relations between $\mathrm{S}_{\min}(h)$ and $\mathrm{S}'_{\min}(h)$ showing that the difference between the number of pebbles needed to perform a complete and a semicomplete computation is small. Hence, we can estimate the space complexity of the binary tree topology by calculating the number of pebbles needed to perform a semicomplete computation.

Lemma 5.5. $\mathrm{S}_{\min}(h) + 1 \geq \mathrm{S}'_{\min}(h) \geq \mathrm{S}_{\min}(h)$

By performing a complete computation without removing a pebble from the root vertex we can obtain a semicomplete computation with space complexity $\mathrm{S}_{\min}(h)+1$, hence $\mathrm{S}_{\min}(h)+1 \geq \mathrm{S}'_{\min}(h)$. By joining a semicomplete computation with its reverse we obtain a semicomplete computation, hence $\mathrm{S}'_{\min}(h) \geq \mathrm{S}_{\min}(h)$. The formal proof of the lemma follows.

Proof of Lemma 5.5. Let $\mathcal{K}$ be a semicomplete computation on $\mathrm{Bt}(h)$ such that $\mathrm{S}(\mathcal{K}) = \mathrm{S}'_{\min}(h)$. It holds that $\mathcal{K} + \mathrm{Rev}(\mathcal{K})$ is a complete computation on $\mathrm{Bt}(h)$ and $\mathrm{S}(\mathcal{K} + \mathrm{Rev}(\mathcal{K})) = \mathrm{S}(\mathcal{K}) \geq \mathrm{S}_{\min}(h)$. Hence $\mathrm{S}'_{\min}(h) \geq \mathrm{S}_{\min}(h)$.

Now we prove the first inequality. Let $\mathcal{K}$ be a complete computation on $\mathrm{Bt}(h)$, such that $\mathrm{S}(\mathcal{K}) = \mathrm{S}_{\min}(h)$. Let l be the length of the computation $\mathcal{K}$, let $\mathcal{K}(i)(\mathrm{R}(\mathrm{Bt}(h))) = 1$. Consider a computation

$$\begin{aligned}\mathcal{K}_2 &= \mathrm{Rst}(\mathcal{K}, 1, i) + \\ &+ \mathrm{Rst}(\mathcal{K}, i, l, \mathrm{Lt}(\mathrm{Bt}(h)) \cup \mathrm{Rt}(\mathrm{Bt}(h))) \cdot \mathrm{Put}(\mathrm{R}(\mathrm{Bt}(h)), \mathrm{R}(\mathrm{Bt}(h)), 1)\end{aligned}$$

Clearly, $\mathcal{K}_2$ is a semicomplete computation on $\mathrm{Bt}(h)$ and $\mathrm{S}(\mathcal{K}_2) \leq \mathrm{S}(\mathcal{K}) + 1$. Thus, it holds

$$\mathrm{S}_{\min}(h) + 1 = \mathrm{S}(\mathcal{K}) + 1 \geq \mathrm{S}(\mathcal{K}_2) \geq \mathrm{S}'_{\min}(h)$$

□

Lemma 5.6. $\mathrm{S}'_{\min}(h+1) = \mathrm{S}_{\min}(h) + 2$

Proof. We start by proving that $\mathrm{S}_{\min}(h) + 2 \geq \mathrm{S}'_{\min}(h+1)$. The previous lemma states that $\mathrm{S}_{\min}(h) + 1 \geq \mathrm{S}'_{\min}(h)$. Let $\mathcal{K}_1$ be a semicomplete computation on $\mathrm{Bt}(h)$ that uses no more than $\mathrm{S}_{\min}(h) + 1$ pebbles. Let $\mathcal{K}_2$ be a complete computation on $\mathrm{Bt}(h)$ of length l that uses no more than $\mathrm{S}_{\min}(h)$ pebbles.

Using the computations $\mathcal{K}_1$ and $\mathcal{K}_2$ we construct a semicomplete computation $\mathcal{K}_3$ on $\mathrm{Bt}(h+1)$ that uses at most $\mathrm{S}_{\min}(h) + 2$ pebbles.

Let i be the minimal number such that $\mathcal{K}_2(i)(\mathrm{R}(\mathrm{Bt}(h))) = 1$. We define the computation $\mathcal{K}_3$ as follows (see also Figure 5.2a):

$$\begin{aligned}\mathcal{K}_3 = \quad & \big(\mathcal{K}_1 | \mathrm{Lt}(\mathrm{Bt}(h+1))\big) \cdot \mathrm{E}\big(\mathrm{Rt}(\mathrm{Bt}(h+1)) \cup \mathrm{R}(\mathrm{Bt}(h+1))\big) + \\ & + \big(\mathrm{Rst}(\mathcal{K}_2, 1, i) | \mathrm{Rt}(\mathrm{Bt}(h+1))\big) \cdot \\ & \quad \cdot \mathrm{Put}\Big(\mathrm{E}\big(\mathrm{Lt}(\mathrm{Bt}(h+1)) \cup \mathrm{R}(\mathrm{Bt}(h+1))\big), \mathrm{R}\big(\mathrm{Lt}(\mathrm{Bt}(h+1))\big), 1\Big) + \\ & + \big(\mathrm{Rst}(\mathcal{K}_2, i, l) | \mathrm{Rt}(\mathrm{Bt}(h+1))\big) \cdot \\ & \quad \cdot \mathrm{Put}\Big(\mathrm{E}\big(\mathrm{Lt}(\mathrm{Bt}(h+1))\big), \mathrm{R}\big(\mathrm{Lt}(\mathrm{Bt}(h+1))\big), 1\Big) \cdot \\ & \quad \cdot \mathrm{Put}\Big(\mathrm{E}\big(\mathrm{R}(\mathrm{Bt}(h+1))\big), \mathrm{R}(\mathrm{Bt}(h+1)), 1\Big) + \\ & + \Big(\mathrm{Rst}\big(\mathcal{K}_2, 1, i, \mathrm{Bt}(h) \setminus \mathrm{R}(\mathrm{Bt}(h))\big) \mid \mathrm{Lt}(\mathrm{Bt}(h+1)) \setminus \mathrm{R}\big(\mathrm{Lt}(\mathrm{Bt}(h+1))\big)\Big) \cdot \\ & \quad \cdot \mathrm{Put}\Big(\mathrm{E}\big(\mathrm{R}(\mathrm{Lt}(\mathrm{Bt}(h+1)))\big), \mathrm{R}\big(\mathrm{Lt}(\mathrm{Bt}(h+1))\big), 1\Big) \cdot \\ & \quad \cdot \mathrm{Put}\Big(\mathrm{E}\big(\mathrm{R}(\mathrm{Bt}(h+1)) \cup \mathrm{Rt}(\mathrm{Bt}(h+1))\big), \mathrm{R}(\mathrm{Bt}(h+1)), 1\Big) +\end{aligned}$$

$$+\big(\mathrm{Rst}(\mathcal{K}_2, i, l)|\mathrm{Lt}(\mathrm{Bt}(h+1))\big)\cdot$$
$$\cdot\,\mathrm{Put}\Big(\mathrm{E}\big(\mathrm{R}(\mathrm{Bt}(h+1))\cup\mathrm{Rt}(\mathrm{Bt}(h+1))\big), \mathrm{R}(\mathrm{Bt}(h+1)), 1\Big)$$

Clearly, $\mathcal{K}_3$ is a semicomplete computation on $\mathrm{Bt}(h+1)$ and it holds that

$$\mathrm{S}(\mathcal{K}_3) \leq \max(\mathrm{S}(\mathcal{K}_1), \mathrm{S}(\mathcal{K}_2)+2) \leq \mathrm{S}_{\min}(h)+2$$

Hence, the first inequality holds.

In the sequel we prove that $\mathrm{S}_{\min}(h)+2 \leq \mathrm{S}'_{\min}(h+1)$. Let $\mathcal{K}$ be a semicomplete computation of length l on $\mathrm{Bt}(h+1)$, such that $\mathrm{S}(\mathcal{K}) = \mathrm{S}'_{\min}(h+1)$.

We show that a complete computation on $\mathrm{Bt}(h)$ that uses at most $\mathrm{S}'_{\min}(h+1)-2$ pebbles can be obtained by an appropriate restriction of $\mathcal{K}$.

Let i be the minimal number such that

$$(\forall k : i \leq k \leq l)\mathcal{K}(k)(\mathrm{R}(\mathrm{Bt}(h+1))) = 1$$

Let j be the minimal number such that $i \leq j \leq l$ and it holds:

$$\#\Big(\mathrm{Rst}\big(\mathcal{K}, j, j, \mathrm{Lt}(\mathrm{Bt}(h+1))\big)\Big) = 0 \quad \vee \quad \#\Big(\mathrm{Rst}\big(\mathcal{K}, j, j, \mathrm{Rt}(\mathrm{Bt}(h+1))\big)\Big) = 0$$

We may assume without loss of generality that $\#(\mathrm{Rst}(\mathcal{K}, j, j, \mathrm{Lt}(\mathrm{Bt}(h+1)))) = 0$.

Let us consider the following computation $\mathcal{K}_2$ (see also Figure 5.2b):

$$\mathcal{K}_2 = \mathrm{Rst}\big(\mathcal{K}, i, j, \mathrm{Lt}(\mathrm{Bt}(h+1))\big)$$

$\mathcal{K}_2$ is a computation on a graph $\mathrm{Rt}(\mathrm{Bt}(h+1))$ which is isomorphic to $\mathrm{Bt}(h)$. Since the last configuration of $\mathcal{K}_2$ is empty and in the first configuration of $\mathcal{K}_2$ the root vertex is pebbled, it holds that $\mathrm{Rev}(\mathcal{K}_2)+\mathcal{K}_2$ is a complete computation on the tree of height h.

In all configurations $\mathcal{K}(k)$ for $i \leq k \leq j$ the root $\mathrm{R}(\mathrm{Bt}(h))$ is pebbled and there is at least one vertex pebbled in $\mathrm{Rt}(\mathrm{Bt}(h))$. Hence, the space of the computation $\mathcal{K}_2$ can not exceed $\mathrm{S}(\mathcal{K})-2$ and it holds that $\mathrm{S}_{\min}(h) \leq \mathrm{S}(\mathcal{K}_2) \leq \mathrm{S}'_{\min}(h+1)-2$.

□

As a corollary of the preceding lemmas we obtain the following useful facts:

Corollary 5.7. $\mathrm{S}'_{\min}(h)$ *equals to* h *plus the number of different numbers* i *such that* $i < h$ *and* $\mathrm{S}'_{\min}(i) = \mathrm{S}_{\min}(i)$. *Furthermore, for* $h_1 \geq h_2$ *it holds that* $\mathrm{S}'_{\min}(h_1) - \mathrm{S}'_{\min}(h_2) \geq h_1 - h_2$.

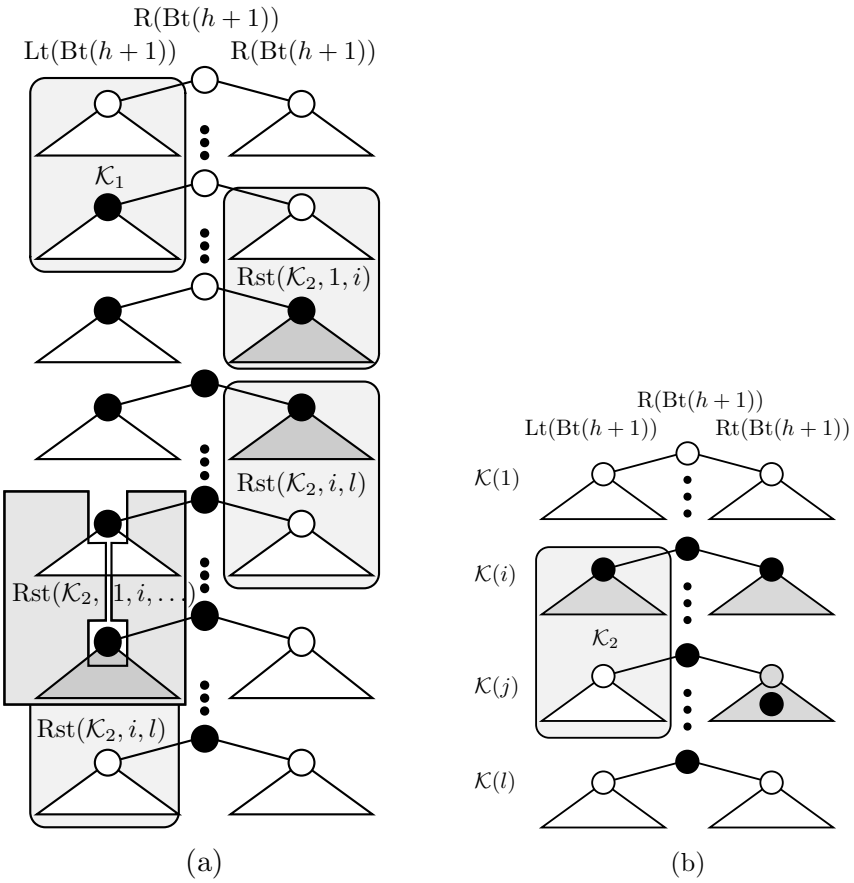

Figure 5.2: a) The proof of $S_{min}(h) + 2 \geq S'_{min}(h + 1)$ – the construction of the computation $\mathcal{K}_3$. b) The proof of $S_{min}(h) + 2 \leq S'_{min}(h + 1)$ – the computation $\mathcal{K}_2$.

5.3 Tight Space Bound for Binary Tree Topology

In this section we prove a tight space bound for the binary tree topology (also published in [8] and [9]). In the following considerations we use a function $\mathrm{S}'^{-1}_{\min}$. The value $h = \mathrm{S}'^{-1}_{\min}(p)$ denotes the maximal height of a binary tree that can be pebbled by a semicomplete computation that uses at most $h+p$ pebbles. Formally

$$\mathrm{S}'^{-1}_{\min}(p) = \max\{h | \exists \mathcal{K} \in \ \mathrm{Sc}_h \wedge \mathrm{S}(\mathcal{K}) = h + p\}$$

where Sc_h is the set of all semicomplete computations on $\mathrm{Bt}(h)$.

From the definition of $\mathrm{S}'^{-1}_{\min}(p)$ and Corollary 5.7 we easily obtain the following lemma. It enables us to obtain an upper bound on function $\mathrm{S}'_{\min}$ from the lower bound on $\mathrm{S}'^{-1}_{\min}$ and vice versa.

Lemma 5.8. *For each h, p such that $\mathrm{S}'_{\min}(h) = h + p$ it holds that $\mathrm{S}'^{-1}_{\min}(p - 1) < h \leq \mathrm{S}'^{-1}_{\min}(p)$.*

Furthermore let $f(p)$ be a nondecreasing function such that $\mathrm{S}'^{-1}_{\min}(p) \leq f(p)$. Then it holds that $\mathrm{S}'_{\min}(h) \geq h + f^{-1}(h)$ for each h.

Similarly, let $g(p)$ be a nondecreasing function such that $\mathrm{S}'^{-1}_{\min}(p) \geq g(p)$. Then it holds that $\mathrm{S}'_{\min}(h) \leq h + g^{-1}(h) + 1$ for each h.

Now we prove the upper (lower) bound on $\mathrm{S}'^{-1}_{\min}(p)$. From this lemma we obtain a lower (upper) bound on $\mathrm{S}'_{\min}(h)$ and hence a lower (upper) bound on $\mathrm{S}_{\min}(h)$, too. The following lemma proves the upper bound on the function $\mathrm{S}^{-1}(p)$.

Lemma 5.9. *Let $h_1 = \mathrm{S}'^{-1}_{\min}(p)$, $h_2 = \mathrm{S}'^{-1}_{\min}(p+1)$. Then the following inequality holds:*

$$h_2 - h_1 \leq 2^{h_1+p+1}$$

Proof. From the assumption $h_2 = \mathrm{S}'^{-1}_{\min}(p+1)$ it follows that there exists some semicomplete computation $\mathcal{K}$ of length l on $\mathrm{Bt}(h_2)$ such that $\mathrm{S}(\mathcal{K}) = h_2 + p + 1$ (see Figure 5.3a). Let i be the first configuration of $\mathcal{K}$ such that the root vertex is pebbled in $\mathcal{K}(j)$ for all $j \geq i$ (i.e. $i = \min\{i' | (\forall j \geq i')\mathcal{K}(j)(\mathrm{R}(\mathrm{Bt}(h_2))) = 1\}$).

Since $\mathcal{K}$ is a reversible computation, it holds that

$$\mathcal{K}(i)\Big(\mathrm{R}\big(\mathrm{Lt}(\mathrm{Bt}(h_2))\big)\Big) = \mathcal{K}(i)\Big(\mathrm{R}\big(\mathrm{Rt}(\mathrm{Bt}(h_2))\big)\Big) = 1$$

Thus, the configuration obtained from $\mathcal{K}(i)$ by unpebbling the root vertex (denoted as $\mathrm{Put}\big(\mathcal{K}(i), \mathrm{R}(\mathrm{Bt}(h_2)), 0\big)$) is a closed configuration. Since

$\mathrm{Put}\big(\mathcal{K}(l), \mathrm{R}(\mathrm{Bt}(h_2)), 0\big)$ is an empty configuration, it is also an open configuration. Let j be the minimal number such that $j \geq i$ and $\mathrm{Put}(\mathcal{K}(j), \mathrm{R}(\mathrm{Bt}(h_2)), 0)$ is open.

Since $\mathrm{Put}(\mathcal{K}(j), \mathrm{R}(\mathrm{Bt}(h_2)), 0)$ is open and $\mathrm{Put}(\mathcal{K}(j-1), \mathrm{R}(\mathrm{Bt}(h)), 0)$ is closed and $\mathcal{K}$ is a reversible computation, there exists exactly one path in $\mathcal{K}(j)$ from the root to a leaf such that no pebble is placed on it. We can assume without loss of generality that this path is $\mathrm{R}(\mathrm{Bt}(h_2))$, $\mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(h_2)))$, $\mathrm{R}(\mathrm{Rt}^2(\mathrm{Bt}(h_2)))$, $\ldots$, $\mathrm{R}(\mathrm{Rt}^{h_2-1}(\mathrm{Bt}(h_2)))$. We call this set of vertices the *rightmost path*.

We prove that for each k such that $h_2 \geq k \geq h_1 + 2$ and for each r such that $i \leq r < j$ it holds $\#(\mathcal{K}(r)(\mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2))))) > 0$. This part of the proof is illustrated by Figure 5.3b. We need this fact to "bind" $h_2 - h_1 - 1$ pebbles so that they can not be used on the vertices of the rightmost path.

Assume that this conjecture is false and let k be the maximal number violating it. Let r be a maximal number such that $i \leq r < j$ and

$$\#\Big(\mathcal{K}(r)\big(\mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2)))\big)\Big) = 0 \quad \vee \quad \#\Big(\mathcal{K}(r)\big(\mathrm{Rt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2)))\big)\Big) = 0$$

Our assumption ensures that such r exists. Note that only one clause of the disjunction can be satisfied because $\mathcal{K}$ is a reversible computation. Without loss of generality let $\#\Big(\mathcal{K}(r)\big(\mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2)))\big)\Big) = 0$. Since $\mathrm{Put}\big(\mathcal{K}(r), \mathrm{R}(\mathrm{Bt}(h_2)), 0\big)$ is closed, at least one vertex from $\mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(h_2)))$, $\mathrm{R}(\mathrm{Rt}^2(\mathrm{Bt}(h_2)))$, $\ldots$, $\mathrm{R}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2)))$ is pebbled in the configuration $\mathcal{K}(r)$. All these vertices are unpebbled in the configuration $\mathcal{K}(j)$. Let q be the minimal number such that $q > r$ and all above mentioned vertices are unpebbled in $\mathcal{K}(q)$. Clearly $q \leq j$. Since $\mathcal{K}$ is a reversible computation it holds that

$$\mathcal{K}(q-1)\Big(\mathrm{R}\big(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2))\big)\Big) = 1$$

$$\mathcal{K}(q)\Big(\mathrm{R}\big(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2))\big)\Big) = 0$$

$$\mathcal{K}(q-1)\Big(\mathrm{R}\big(\mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2)))\big)\Big) = \mathcal{K}(q)\Big(\mathrm{R}\big(\mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2)))\big)\Big) = 1$$

Consider the computation $\mathcal{K}' = \mathrm{Rst}(\mathcal{K}, r, q-1, \mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2))))$. Computation $\mathcal{K}' + \mathrm{Rev}(\mathcal{K}')$ is a complete computation on $\mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2))))$ (this graph is isomorphic to $\mathrm{Bt}(k-1)$). Let us consider the space of this computation. For each $z \in \{r \ldots q-1\}$ in $\mathcal{K}(z)$ the root vertex of $\mathrm{Bt}(h_2)$ is pebbled. By the choice of k, there is at least one vertex pebbled in each of the following subgraphs: $\mathrm{Lt}(\mathrm{Bt}(h_2)), \mathrm{Lt}(\mathrm{Rt}(\mathrm{Bt}(h_2))), \ldots, \mathrm{Lt}(\mathrm{Rt}^{h_2-k-1}(\mathrm{Bt}(h_2)))$. By the choice of q, at least one vertex is pebbled in the upper part of the rightmost path – at least one of the following vertices is pebbled: $\mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(h_2)))$,

..., $\mathrm{R}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2)))$. By the choice of r, there is at least one vertex pebbled in $\mathrm{Rt}^{h_2-k+1}(\mathrm{Bt}(h_2))$. Hence, it holds $\mathrm{S}(\mathcal{K}' + \mathrm{Rev}(\mathcal{K}')) = \mathrm{S}(\mathcal{K}') \leq \mathrm{S}(\mathcal{K}) - (3 + h_2 - k) = k + p - 2$.

From our assumption it follows that the space of any semicomplete computation on $\mathrm{Bt}(k-1)$ is at least $k+p$. But from Lemma 5.5 it follows that the space of any complete computation on $\mathrm{Bt}(k-1)$ is at least $k+p-1$, which is a contradiction.

Consider the computation $\mathcal{K}_2$ (see Figure 5.3a):

$$\mathcal{K}_2 = \mathrm{Rst}(\mathcal{K}, i, j, \mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(h_2))) \cup \mathrm{R}(\mathrm{Rt}^2(\mathrm{Bt}(h_2))) \cup \ldots \cup \mathrm{R}(\mathrm{Rt}^{h_2-h_1-1}(\mathrm{Bt}(h_2))))$$

It is a computation on an upper part of the rightmost path, which is a graph isomorphic to $Ch(h_2 - h_1 - 1)$. The vertex $\mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(h_2)))$ is pebbled in the first configuration of $\mathcal{K}_2$ and no vertex is pebbled in the last configuration of $\mathcal{K}_2$. Hence, $\mathrm{Rev}(\mathcal{K}_2) + \mathcal{K}_2$ is a complete computation on a graph isomorphic to $Ch(h_2 - h_1 - 1)$.

Since for each $h_2 \geq k \geq h_1 + 2$ and for each $i \leq r \leq j$ it holds that $\#(\mathcal{K}(r)(\mathrm{Lt}(\mathrm{Rt}^{h_2-k}(\mathrm{Bt}(h_2))))) > 0$ and $\mathcal{K}(r)(\mathrm{R}(\mathrm{Bt}(h))) = 1$, we can estimate upper bound for the space of $\mathcal{K}_2$ to be: $\mathrm{S}(\mathcal{K}_2) \leq (h_2+p+1)-(1+h_2-h_1-1) = h_1+p+1$. Using the upper bound on the space of the chain topology (Theorem 4.5) we have $h_2 - h_1 - 1 \leq 2^{h_1+p+1} - 1$.

□

We have proven an upper bound on $\mathrm{S}'^{-1}_{\min}(p)$. Next we prove the lower bound:

Lemma 5.10. *Let $h_1 = \mathrm{S}'^{-1}_{\min}(p)$, $h_2 = \mathrm{S}'^{-1}_{\min}(p+1)$. Then the following inequality holds:*

$$h_2 - h_1 \geq 2^{h_1+p-2}$$

Proof. We prove by induction that for each $k \in \{h_1 + 1, \ldots, h_1 + 2^{h_1+p-2}\}$ there exists a semicomplete computation $\mathcal{K}$ on $Bt(k)$ such that $\mathrm{S}(\mathcal{K}) \leq k + p + 1$. This implies that $h_2 \geq h_1 + 2^{h_1+p-2}$.

The base case is $k = h_1 + 1$. By assumption there exists a semicomplete computation $\mathcal{K}'$ on $\mathrm{Bt}(h_1)$ such that $\mathrm{S}(\mathcal{K}') = h_1 + p$. After applying $\mathcal{K}'$ to $\mathrm{Lt}(\mathrm{Bt}(k))$ and $\mathrm{Rt}(\mathrm{Bt}(k))$, pebbling $\mathrm{R}(\mathrm{Bt}(k))$ and applying reversed $\mathcal{K}'$ to $\mathrm{Lt}(\mathrm{Bt}(k))$ and $\mathrm{Rt}(\mathrm{Bt}(k))$ we obtain a semicomplete computation on $\mathrm{Bt}(k)$ that uses at most $h_1 + p + 2 = k + p + 1$ pebbles.

Let us assume that the induction hypothesis holds for each $h \in \{h_1 + 1, \ldots, k-1\}$. We construct a computation $\mathcal{K}$ on $\mathrm{Bt}(k)$ as follows: At first we sequentially apply a space optimal semicomplete computation on subgraphs $\mathrm{Lt}(\mathrm{Bt}(k))$, $\mathrm{Lt}(\mathrm{Rt}(\mathrm{Bt}(k)))$, $\mathrm{Lt}(\mathrm{Rt}^2(\mathrm{Bt}(k)))$, ..., $\mathrm{Lt}(\mathrm{Rt}^{k-h_1-2}(\mathrm{Bt}(k)))$,

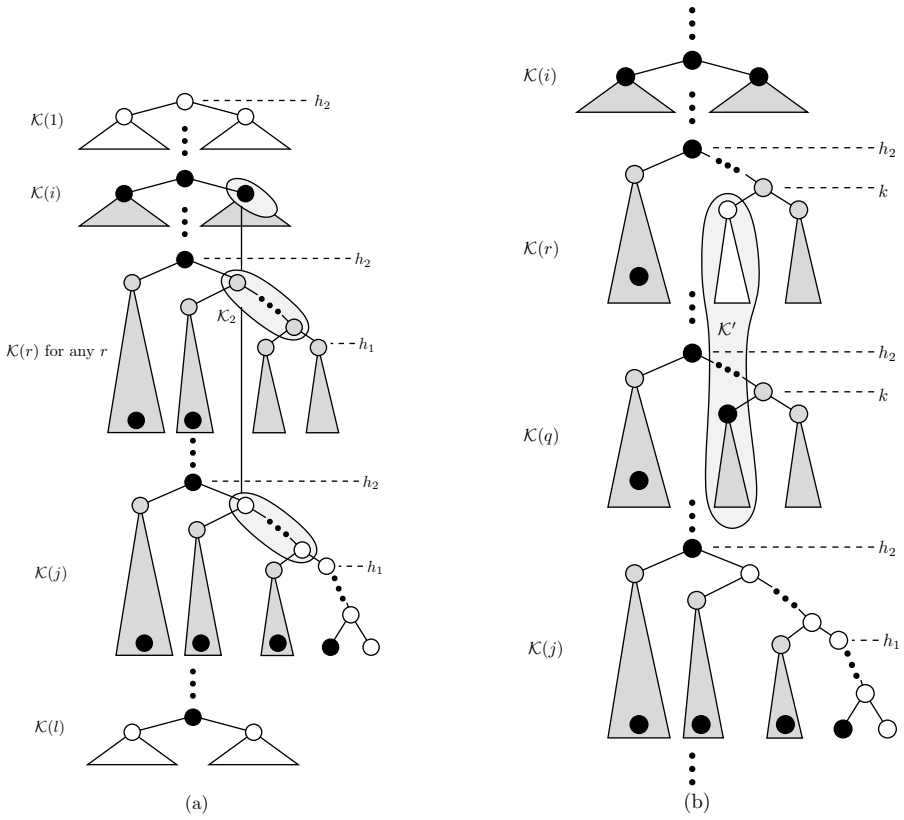

Figure 5.3: a) The main framework of the proof of Lemma 5.9. b) The proof of the auxiliary property by contradiction.

$\mathrm{Lt}(\mathrm{Rt}^{k-h_1-1}(\mathrm{Bt}(k)))$ and $\mathrm{Rt}^{k-h_1}(\mathrm{Bt}(k))$. By the induction hypothesis, the space of a semicomplete computation on $\mathrm{Lt}(\mathrm{Rt}^{i}(\mathrm{Bt}(k)))$ is at most $(k-i-1)+p+1$ for $i \leq k-h_1-2$. By the assumption, the space of a semicomplete computation on $\mathrm{Lt}(\mathrm{Rt}^{k-h_1-1}(\mathrm{Bt}(k)))$ and $\mathrm{Rt}^{k-h_1}(\mathrm{Bt}(k))$ is at most h_1+p, hence the space of this part of $\mathcal{K}$ is at most $k+p$.

In the second part of $\mathcal{K}$ we perform a space optimal complete computation on a chain consisting of the vertices $\mathrm{R}(\mathrm{Bt}(k))$, $\mathrm{R}(\mathrm{Rt}(\mathrm{Bt}(k)))$, $\mathrm{R}(\mathrm{Rt}^2(\mathrm{Bt}(k)))$, $\ldots$, $\mathrm{R}(\mathrm{Rt}^{k-h_1-1}(\mathrm{Bt}(k)))$. Due to the Theorem 4.5, the space of this part is less than $\lceil \log_2(k-h_1+1) \rceil + k - h_1 + 1 \leq \lceil \log_2(k-h_1) \rceil + k - h_1 + 2$. Since $k \leq h_1 + 2^{h_1+p-2}$, it holds that $\lceil \log_2(k-h_1) \rceil + k - h_1 + 2 \leq k+p$.

The third part of the computation $\mathcal{K}$ is the reversed first part.

Hence, $\mathcal{K}$ is a complete computation on $\mathrm{Bt}(k)$ and $\mathrm{S}(\mathcal{K}) \leq k+p$ which implies that $\mathrm{S}_{\min}(k) \leq k+p$. Using Lemma 5.5, it holds that $\mathrm{S}'_{\min}(k) \leq k+p+1$. Thus, there exists a semicomplete computation on $\mathrm{Bt}(k)$ with space less than $k+p+1$. □

We make some approximations and by putting everything together we obtain an asymptotically tight bound for $\mathrm{S}_{\min}(h)$:

Lemma 5.11. *For $p \geq 2$ it holds that $2^{\mathrm{S}'^{-1}_{\min}(p)} \leq \mathrm{S}'^{-1}_{\min}(p+1) \leq 2^{3\,\mathrm{S}'^{-1}_{\min}(p)}$.*

Proof. Let $h_1 = \mathrm{S}'^{-1}_{\min}(p)$, $h_2 = \mathrm{S}'^{-1}_{\min}(p+1)$. From Lemma 5.9 it follows that $h_2 - h_1 \leq 2^{h'+p+1}$, which is equivalent to $\mathrm{S}'^{-1}_{\min}(p+1) \leq 2^{\mathrm{S}'^{-1}_{\min}(p)+p+1} + \mathrm{S}'^{-1}_{\min}(p)$.

Definition of $\mathrm{S}'^{-1}_{\min}$ and Corollary 5.7 imply that $\mathrm{S}'^{-1}_{\min}(p) \geq p+1$. Thus $2^{\mathrm{S}'^{-1}_{\min}(p)+p+1} + \mathrm{S}'^{-1}_{\min}(p) \leq 2^{3\,\mathrm{S}'^{-1}_{\min}(p)}$ for $p \geq 1$. Hence, the second inequality holds.

From Lemma 5.10 it follows that $h_2 - h_1 \geq 2^{h_1+p-2}$, which is equivalent to $\mathrm{S}'^{-1}_{\min}(p+1) \geq 2^{\mathrm{S}'^{-1}_{\min}(p)+p-2} + \mathrm{S}'^{-1}_{\min}(p)$. Thus, for $p \geq 2$ it holds $\mathrm{S}'^{-1}_{\min}(p+1) \geq 2^{\mathrm{S}'^{-1}_{\min}(p)}$. □

Theorem 5.12. *For the minimal space function of the complete binary tree topology $\mathrm{S}_{\min}(h)$ it holds that $\mathrm{S}_{\min}(h) = h + \Theta(\lg^*(h))$, where the function $\lg^*(x)$ is defined as follows: $\lg^*(x) = 0$ for $x \leq 0$, $\lg^*(x) = 1 + \lg^*(\lg(x))$ otherwise.*

Proof. From the previous lemma it follows that $\mathrm{S}'^{-1}_{\min}(p) \leq \overbrace{8^{8^{\cdots 8}}}^{p}$ and that $\mathrm{S}'^{-1}_{\min}(p) \geq \overbrace{2^{2^{\cdots 2}}}^{p}$. Applying Lemma 5.8 we obtain $\mathrm{S}'_{\min}(h) = h + \Omega(\lg^*(h))$ and $\mathrm{S}'_{\min}(h) = h + O(\lg^*(h))$. Hence, $\mathrm{S}'_{\min}(h) = h + \Theta(\lg^*(h))$ holds. From Lemma 5.5 it follows, that $\mathrm{S}_{\min}(h) = h + \Theta(\lg^*(h))$. □

5.4 Extension to Butterflies

The butterfly topology has been introduced in Section 2.3. In this section the space complexity of a reversible pebble game on the butterfly graphs will be analysed.

Let $\mathrm{Bf}(d)$ be a butterfly graph of order d. As mentioned in Section 2.3, $\mathrm{Bf}(d)$ can be decomposed into $2^d - 1$ (not disjoint) complete binary trees of height d. We denote these trees as $T_1, \ldots T_{2^d-1}$.

The decomposition property implies that the minimal space complexity of a complete computation on the butterfly graph of order d cannot be lower than the minimal space complexity on the complete binary tree of height d:

Lemma 5.13. *It holds that* $\mathrm{S}(\mathrm{Bf}(d)) \geq \mathrm{S}(\mathrm{Bt}(d))$.

Proof. Let us assume the contrary. Let $\mathcal{K}$ be a complete computation of length l on $\mathrm{Bf}(d)$ such that $\mathrm{S}(\mathcal{K}) < \mathrm{S}(\mathrm{Bt}(d))$. By restricting this computation to any binary tree from the decomposition (for example to T_1) we obtain a complete computation on $\mathrm{Bt}(d)$ with space complexity less than $\mathrm{S}(\mathrm{Bt}(d))$. Formally, $\mathrm{S}(\mathrm{Rst}(\mathcal{K}, 1, l, T_1)) < \mathrm{S}(\mathrm{Bt}(d))$, which is a contradiction. □

On the other side, by sequentially applying complete computations to all binary trees obtained by the decomposition of the butterfly graph, we obtain a complete computation on it. Hence, we can construct a complete computation on the butterfly graph of order d with space complexity equal to the minimal space complexity of the binary tree of height d:

Lemma 5.14. *It holds that* $\mathrm{S}(\mathrm{Bf}(d)) \leq \mathrm{S}(\mathrm{Bt}(d))$.

Proof. Let $\mathcal{K}$ be a complete computation on $\mathrm{Bt}(d)$ such that $\mathrm{S}(\mathcal{K}) = \mathrm{S}(\mathrm{Bt}(d))$. Consider the computation $\mathcal{K}'$ defined as follows:

$$\mathcal{K}' = \mathcal{K}|T_1 + \mathcal{K}|T_2 + \ldots + \mathcal{K}|T_{2^d-1}$$

Computation $\mathcal{K}'$ is a complete computation on $\mathrm{Bf}(h)$ such that $\mathrm{S}(\mathcal{K}') = \mathrm{S}(\mathcal{K}) = \mathrm{S}(\mathrm{Bt}(d))$. □

Thus, the minimal space complexity of the butterfly topology equals to the minimal space complexity of the binary tree topology. This result was published also in [8] and [9].

Theorem 5.15. *For the minimal space for the butterfly graph of order d it holds that* $\mathrm{S}_{\min}(d) = d + \Theta(\lg^*(d))$.

Proof. This theorem is an immediate consequence of Lemmas 5.13 and 5.14 and Theorem 5.12. □

Chapter 6

Reversible simulation of irreversible computation

In the previous sections we have discussed the complexity aspects of the reversible pebble game on various topologies. In the sequel we bound the time and space complexity of the reversible pebble game by the time and space complexity of the standard irreversible pebble game, regardless of the topology.

This result (published in [9]) expresses the idea of the reversible simulation of irreversible computations, which was introduced by [1], in a pebbling abstraction.

Theorem 6.1. *Let G be an arbitrary dag. Let G can be pebbled by a standard (irreversible) computation with p pebbles in time t. Let $\mathcal{K}$ be a complete reversible computation on the chain Ch_t. Then G can be pebbled by a reversible computation with $p \cdot \mathrm{S}(\mathcal{K})$ pebbles in time at most $\mathrm{T}(\mathcal{K})$.*

Proof. By assumption there exists some irreversible computation $\mathcal{K}_s$ on the graph G, such that $\mathrm{T}(\mathcal{K}_s) = t$ and $\mathrm{S}(\mathcal{K}_s) = p$. Without loss of generality we can assume that the length of $\mathcal{K}$ is $\mathrm{T}(\mathcal{K}) + 1$. We construct a complete reversible computation $\mathcal{K}'$ on the graph $G = (V, E)$ of length $\mathrm{T}(\mathcal{K}) + 1$ as follows:

$$\big(\forall i : 1 \leq i \leq \mathrm{T}(\mathcal{K}) + 1\big)\big(\forall v \in V\big)$$

$$\mathcal{K}'(i)(v) = 1 \Leftrightarrow \big(\exists j : 1 \leq j \leq \mathrm{T}(\mathcal{K}_s)\big)\big(\mathcal{K}(i)(j) = 1 \wedge \mathcal{K}_s(j)(v) = 1\big)$$

Informally speaking, we define $\mathcal{K}'(i)$ as a "superposition" of all configurations in $\mathcal{K}_s$ that are "marked" in $\mathcal{K}(i)$ (see also Figure 6.1).

Now we prove that $\mathcal{K}'$ is a reversible computation. At first we prove that two successive configurations can not differ in two vertices.

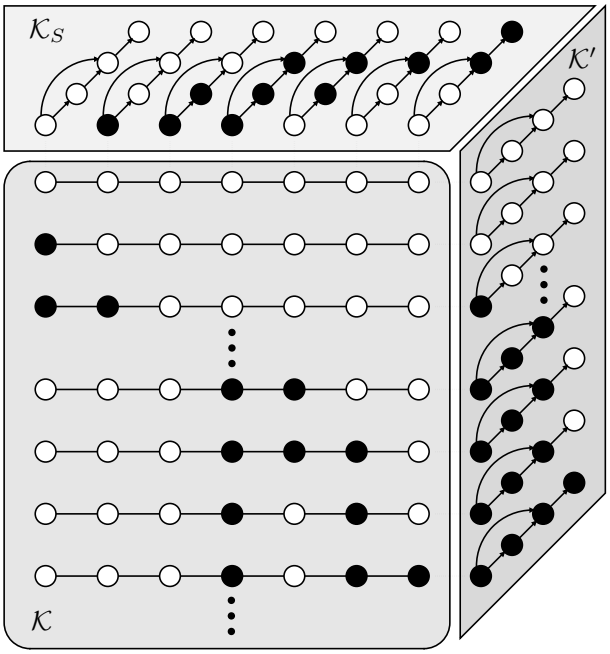

Figure 6.1: Reversible simulation of irreversible computation.

Assume that the conjecture does not hold: $\mathcal{K}'(i-1)(a) \neq \mathcal{K}'(i)(a)$ and $\mathcal{K}'(i-1)(b) \neq \mathcal{K}'(i)(b)$ for some i and some $a \neq b$. Clearly there exists at most one j such that $\mathcal{K}(i-1)(j) \neq \mathcal{K}(i)(j)$. From the definition of $\mathcal{K}'$ it follows that $\mathcal{K}_s(j)(a) = \mathcal{K}_s(j)(b) = 1$ (otherwise $\mathcal{K}'(i-1)(a) = \mathcal{K}'(i)(a)$ or $\mathcal{K}'(i-1)(b) = \mathcal{K}'(i)(b)$). Since $\mathcal{K}_s(1)$ is an empty configuration, $j \neq 1$. Hence, it holds that $\mathcal{K}_s(j-1)(a) = 1 \vee \mathcal{K}_s(j-1)(b) = 1$, otherwise $\mathcal{K}_s$ is not an (irreversible) computation. Without loss of generality $\mathcal{K}_s(j-1)(a) = 1$. Because $\mathcal{K}$ is a reversible computation, $\mathcal{K}(i-1)(j-1) = \mathcal{K}(i)(j-1) = 1$. Thus $\mathcal{K}'(i-1)(a) = \mathcal{K}'(i)(a) = 1$, which is a contradiction.

Next we prove that if two successive configurations in $\mathcal{K}'$ differ in a vertex a then all direct predecessors of a are pebbled in these configurations.

Again, let us assume that the conjecture does not hold. Hence, there exist $a, b \in V$ and i such that $(b, a) \in E$, $\mathcal{K}'(i-1)(a) \neq \mathcal{K}'(i)(a)$ and $\mathcal{K}'(i)(b) = 0$ (note that we have already proven that $\mathcal{K}'(i-1)(b) = \mathcal{K}'(i)(b)$). As already stated, there exists at most one j such that $\mathcal{K}(i-1)(j) \neq \mathcal{K}(i)(j)$. It also holds that $\mathcal{K}_s(j)(a) = 1$, $j \neq 1$ and $\mathcal{K}(i-1)(j-1) = \mathcal{K}(i)(j-1) = 1$. This implies that $\mathcal{K}_s(j-1)(a) = 0$. Since $\mathcal{K}_s$ is an (irreversible) computation, it holds that $\mathcal{K}_s(j-1)(b) = 1$. Hence, $\mathcal{K}'(i-1)(b) = \mathcal{K}'(i)(b) = 1$ which is a contradiction.

We have proven that $\mathcal{K}'$ is a reversible computation. Since $\mathcal{K}(1) = \mathcal{K}(\mathrm{T}(\mathcal{K})+1)$ are empty configurations, $\mathcal{K}'(1) = \mathcal{K}'(\mathrm{T}(\mathcal{K})+1)$ are empty configurations, too. For each vertex $v \in V$ there exists j such that $\mathcal{K}_s(j)(v) = 1$. For this j there exists some i such that $\mathcal{K}(i)(j) = 1$, thus it holds that $\mathcal{K}'(i)(v) = 1$. This implies that $\mathcal{K}'$ is a complete computation.

In each configuration of $\mathcal{K}'$ there can be at most $p \cdot \mathrm{S}(\mathcal{K})$ vertices pebbled. Hence, G can be pebbled by a reversible computation using $\mathrm{S}(\mathcal{K}') \leq \mathrm{S}(\mathcal{K}_s) \cdot \mathrm{S}(\mathcal{K}) = p \cdot \mathrm{S}(\mathcal{K})$ pebbles.

Since $\mathcal{K}'$ is a computation of length $\mathrm{T}(\mathcal{K}) + 1$, the time of $\mathcal{K}'$ is at most $\mathrm{T}(\mathcal{K})$. □

Applying various pebbling strategies for chain topology to Theorem 6.1, we obtain various upper bounds on time and space complexity of the reversible pebble game.

Corollary 6.2. *Let G be arbitrary an dag. Let G can be pebbled by the standard computation using p pebbles in time t. Then G can be pebbled by the reversible computation in time $O(t^{\log_2 3})$ using $p \cdot O(\log_2 t) = O(p^2)$ pebbles.*

Proof. This result can be obtained by applying the results from Theorem 4.5 and Corollary 4.8 to Theorem 6.1. It does not make sense to consider computations using p pebbles of length over 2^p. We can thus assume $\log_2 t \leq p$.

This result is a pebbling formulation of the results from [1] about reversible simulation of irreversible computations. □

Chapter 7

Conclusion

Reversible pebble game as a model of reversible computations has been introduced by C. H. Bennett. In this work, we have described a new technique for proving time and space complexity bounds for this game. Using this technique, alternative proofs of tight bounds on space complexity and time space tradeoff for chain topology have been presented, as well as a lower bound on the time for optimal space for the chain topology (for infinitely many n).

We have presented a tight optimal space bound for the binary tree topology and extended this result to the butterfly topology. We have also presented an upper bound on time and space complexity of the reversible pebble game based on the time and space complexity of the standard pebble game.

These results imply that reversible computations require more resources than standard irreversible computations, e.g. for a space complexity of the chain of length n it is $\Theta(1)$ vs. $\Theta(\lg n)$ and for a space complexity of the binary tree of height h it is $h + \Theta(\log^*(h))$ vs. $h + \Theta(1)$.

Bibliography

[1] Bennett, C. H.: Time-space trade-offs for reversible computation. SIAM J. Comput., 18(1989), pp. 766–776

[2] Buhrman, H., Tromp J. and Vitányi, P.: Time and space bounds for reversible simulation. Proc. ICALP 2001, LNCS 2076, Springer-Verlag, 2001

[3] Levine, R. Y. and Sherman, A. T.: A note on Bennett's time-space tradeoff for reversible computation. SIAM J. Computing, 19(4):673–677, 1990

[4] Li, M., Tromp, J. and Vitányi P. M. B.: Reversible simulation of irreversible computation. Physica D 120 (1998), pp. 168–176

[5] Li, M. and Vitányi P. M. B.: Reversibility and adiabatic computation: trading time and space for energy. Proceedings of the Royal Society of London Ser. A, 452:1–21, 1996.

[6] Li, M. and Vitányi P. M. B.: Reversible simulation of irreversible computation. Proc. 11th IEEE Conference on Computational Complexity, Philadelphia, Pennsylvania, May 24–27, 1996

[7] Knill, E.: An Analysis of Bennett's Pebble Game. LANL report LAUR-95-2258, May 1995

[8] Královič, Richard: Time and Space Complexity of Reversible Pebbling. Proc. SOFSEM 2001, LNCS 2234, pp. 292–304, Springer Verlag 2001

[9] Královič, Richard: Time and Space Complexity of Reversible Pebbling – accepted for publication in Theoretical Informatics and Applications

[10] Paterson, M. S. and Hewitt, C. E.: Comparative Schematology, MAC Conf. on Concurrent Systems and Parallel Computation, 1970, pp. 119–127

[11] Ružička, P.: Pebbling – The Technique for Analysing Computation Efficiency. SOFSEM'89, 1989, pp. 205–224

[12] Ružička, P. and Waczulík, J.: Pebbling Dynamic Graphs in Minimal Space. Theoretical Informatics and Applications, vol. 28, n. 6, 1994, pp. 557–565

[13] Ružička, P. and Waczulík, J.: On Time-Space Trade-Offs in Dynamic Graph Pebbling. MFCS'93, LNCS 711, Springer-Verlag, 1993, pp. 671–681

[14] Williams, R.: Space-Efficient Reversible Simulations, DIMACS REU report, July 2000

[15] Zavarský, A.: On the Cost of Reversible Computations: Time-Space Bounds on Reversible Pebbling. Manuscript, 1998

Abstrakt

Užitočným nástrojom na analýzu časovej a priestorovej zložitosti je pokrývacia hra, v ktorej je problém reprezentovaný orientovaným acyklickým grafom. Okrem jej štandardného variantu, používaného na analýzu deterministických výpočtov, boli študované aj jej rôzne modifikácie. Jednou z nich je aj reverzibilná pokrývacia hra, ktorá umožňuje analyzovať zložitosť reverzibilných výpočtov, dôležitých najmä v súvislosti s kvantovými algoritmami.

V tejto práci je prezentovaná technika pre dôkazy horných i dolných ohraničení časovej a priestorovej zložitosti reverzibilnej pokrývacej hry. Ďalej sú uvedené alternatívne dôkazy pre optimálnu priestorovú zložitosť reverzibilnej pokrývacej hry na reťazových grafoch ($\Theta(\log n)$) a horný odhad pre vzťah medzi časovou a priestorovou zložitosťou (čas $O(n)$ a priestor $O(\sqrt[k]{n})$). Pomocou prezentovanej techniky je dokázané aj tvrdenie, že priestorovo optimálny reverzibilný výpočet na reťazových grafoch potrebuje čas aspoň $O(n \log n)$ pre nekonečne veľa rôznych hodnôt n.

Ďalšia časť práce sa zaoberá priestorovou zložitosťou reverzibilnej pokrývacej hry na úplných binárnych stromoch a motýľových grafoch. V tejto časti je dokázané tvrdenie, že optimálna priestorová zložitosť reverzibilnej pokrývacej hry na úplnom binárnom strome výšky h aj na motýľovom grafe rádu h je $\Theta(h + \log^* h)$.

V poslednej časti práce je prezentovaná technika, ktorou je možné získať horný odhad pre časovú a pamäťovú zložitosť reverzibilnej hry pre ľubovoľný acyklický graf G. Horný odhad je získaný na základe zložitosti štandardnej pokrývacej hry na grafe G a zložitosti reverzibilnej pokrývacej hry na reťazových grafoch.

Práca je rozšírením článkov [8] a [9], v ktorých boli spomínané výsledky publikované.

Výsledky práce dokazujú, že reverzibilné výpočty majú vyššiu časovú a pamäťovú zložitosť ako štandardné deterministické výpočty.