Universitat Politècnica de Catalunya

Departament de Llenguatges i Sistemes Informàtics

Programa de Doctorat de Software

Ph.D. Dissertation

Complexity Measures for Resolution

Juan Luis Esteban

Advisors:

María Luisa Bonet

Jacobo Torán

Pasmo sempre quando acabo qualquer coisa. Pasmo e desolo-me. O meu instinto de perfeição deveria inibir-me de acabar, deveria inibir-me até de dar começo. Mas distraio-me e faço. O que consigo é um produto, em mim, não de uma aplicação de vontade, mas de uma cedência dela. Começo porque não tenho força para pensar, acabo porque não tenho alma para suspender.

LIVRO DO DESASSOSSEGO. Trecho 152

Contents

Chapter 1

Introduction and definitions

The most important problem in Computational Complexity can be briefly stated as: P = NP? The class P is formed by the problems that can be solved in deterministic polynomial time and NP is the class of problems that can be solved in nondeterministic polynomial time. Details and definitions can be found in [8, 51]. The problem is of the utmost importance both theoretically and practically. Polynomial time deterministic algorithms are universally considered as efficient algorithms, so a problem in P will have a fast algorithm, while a problem not in P will not. Many natural and practically important problems are shown to lay in NP but it is not known whether they belong to P or not, see [34]. The widely accepted conjecture is that P $\neq$ NP, but despite the tremendous amount of work done by many clever minds the question is still unsettled.

In 1979, Cook and Reckhow proposed in [27] a plan to solve the P $\neq$ NP question. They proved a relationship between the complexity of propositional proof systems, that is, the size, number of symbols, of proofs of propositional tautologies, and the question whether NP = coNP or not. The class coNP is the set of problems whose complement lies in NP. The main result in this paper was: if there is a propositional proof system such that for any tautology $\mathbb{T}$ we can give a proof of $\mathbb{T}$ of polynomial size in the size of $\mathbb{T}$, then NP = coNP. They call such a desirable system *super*. Cook and Reckhow's result can be restated as:

Theorem 1 *[27]* NP = coNP *iff there exists a propositional proof system that is super.*

As P = NP implies NP = coNP, to show P $\neq$ NP is sufficient to prove that there is no

supersystem, that is, for all propositional proof systems there is a tautology $\mathbb{T}$ whose shortest proof is at least superpolynomial in the size of $\mathbb{T}$. This result is very difficult and they had little hopes in proving it directly, therefore in [27] they proposed the following plan, known as Cook's program:

> Try to find families of tautologies hard to prove for progressively more powerful propositional proof systems until having sufficient knowledge to prove P $\neq$ NP.

This program has created a new and fruitful branch of Complexity Theory, called Proof Complexity. Since the work of Cook and Reckhow many important results have been obtained following the the lines of their program. Nevertheless many problems remain open and the fundamental question whether P $\neq$ NP is still unsolved.

Proof complexity is also relevant to the study of efficiency issues for Automated Theorem Proving. Proof Complexity has strong relations with other branches of Complexity Theory. Results about Circuit Complexity have been successfully used to get results about Proof Complexity, giving then a new impulse to the study of Circuit Complexity.

The aim of this dissertation is to make contributions to the study of the complexity of a certain proof system by the name of Resolution and several other proof systems related to it. Resolution was proven long ago not to be a super proof system [38], so any new result about Resolution will hardly be of interest to the advancement of Cook's program. Nevertheless, we think that is very important to understand completely the power of any proof system, especially one so widely used as Resolution. That means studying in depth the complexity of Resolution. Although there is plenty of papers devoted to Resolution, there are still interesting open problems that we believe that should be solved independently of the existence of Cook's program. For example, we can mention whether Resolution is automatizable or not. So, is it possible to find an algorithm that produces Resolution proofs not much longer than the shortest proofs? Sometime, the shortest proofs are extremely long, but at least, can we find mechanically these proofs? This algorithm would be useful because when there were short proofs it would find one very fast. Another interesting open problem comes from the fact that currently we know that there formulas that require long Resolution proofs and others do not, but we do not know exactly why this happen. It would be very

interesting to know why some formulas are hard for Resolution. So, in our opinion there are still a lot of work, interesting and difficult, to be done about Resolution.

This introductory chapter is structured as follows. In Section 1.1 we define the proof systems that are considered in this work. We define in Section 1.2 the complexity measures that are studied in this work. In Section 1.3 we define some important formulas for Proof Complexity. We do so because these formulas are used in several parts of this work. Besides we believe that having defined proof systems, complexity measures and formulas; the discussion of results about Proof Complexity in Section 1.5 makes more sense. In Section 1.4 we explain some results about Circuit Complexity that will be used in this work, for example the feasible monotone interpolation property which is defined in Section 1.4.2.

1.1 Proof systems

The central notion of Proof System Complexity is PROOF SYSTEM. The widely accepted definition of what a proof system is was given by Cook and Reckhow in [27]:

Definition 2 *Let* TAUT *be the set of propositional tautologies. A function* $f : \{0,1\}^* \to$ TAUT *is a* PROOF SYSTEM *iff* f *is a polynomial time computable surjective function.*

So, if $f(x) = \mathbb{T}$ we say that x is an f-proof of the tautology $\mathbb{T}$, that is, $x \in \{0,1\}^*$ encodes the proof of the tautology $\mathbb{T}$ in the system f. It is important that any alleged f-proof can be checked efficiently, that is, in polynomial time on the size of the proof. f is surjective because f must be complete, any $\mathbb{T} \in$ TAUT must have at least one f-proof.

To get NP $\neq$ coNP following Cook's program, it must be proved that for every propositional proof system f, there is a class of tautologies $\mathbb{T}$ such that any x which holds $f(x) = \mathbb{T}$, has exponential, or at least superpolynomial, size on the size of $\mathbb{T}$.

In order to compare the efficiency of different proof systems, Cook and Reckhow proposed:

Definition 3 *Let* $f_1, f_2 : \{0,1\}^* \to$ TAUT *be proof systems. Then* f_1 POLYNOMIALLY SIMULATES f_2 *if there is a polynomial time computable function* $g : \{0,1\}^* \to \{0,1\}^*$ *such that* $f_1(g(x)) = f_2(x)$ *for all* x.

So f_1 polynomially simulates f_2 iff there is an algorithm that translates proofs in f_2 into proofs in f_1 which are at most polynomially longer than the original proofs in f_2.

It can happen that two proof systems are not comparable, that means that one system is faster, produces shorter proofs, for some tautologies and the other system is faster for some other tautologies. It also is possible that one proof system is better than another, that is, it is never slower and sometime or most of the time is faster. In this work we will show that certain proof systems are much better than others. We define this in the following way:

Definition 4 *Let* $f_1, f_2 : \{0,1\}^* \to \text{TAUT}$ *be proof systems. Then* f_1 DOMINATES f_2 *if* f_1 *polynomially simulates* f_2 *and* f_1 *is almost exponentially separated from* f_2. *That last means that there is a formula* $\mathbb{T}$ *on* n *variables with polynomial* f_1*-proofs but requires almost exponential* f_2*-proofs. We say that a a proof* $\mathcal{R}$ *for a formula* $\mathbb{T}$ *on* n *variables is* ALMOST EXPONENTIAL *if its size is at least* $2^{\Omega(n/\log n)}$.

The complexity of a proof system can be measured in different ways. The most common are SIZE and LENGTH. Size is the number of symbols in the proof, length is the number of lines in the proof. In some proof systems both measures are polynomially related, in the case that this does not hold the preferred measure is size.

The proofs produced by any proof system can be represented in several ways. We will consider two ways: TREELIKE proofs and DAGLIKE proofs. In a treelike proof any line, that is, any intermediate formula, can be used only once. If we need the same formula more than once, it should be derived again. In a daglike proof, any line can be used as many times as needed without having to rederive it. It is easier to get lower bounds for treelike proofs, so in order to study a proof system it is a good idea sometimes to study first the treelike version and then proceed with the daglike or unrestricted version. For some systems treelike size and daglike size are polynomially related, on the other hand, for other systems the difference is proved to be exponential, that is, for certain tautologies the system produces polynomial size proofs but requires exponential size treelike proofs.

Definition 5 *We group the definitions of several terms of common use in Proof Complexity.*

- *A* BOOLEAN VARIABLE *is a variable that can take only the values* TRUE *or* FALSE*. We will usually refer to boolean variables just by variables. We will denote variables as roman lowkey letters as a, b, c, etc.*

- *A* LITERAL *is a boolean variable or its negation. For a variable v the* POSITIVE *literal will be denoted by v and the* NEGATIVE *literal by $\bar{v}$.*

- *A* CLAUSE *is a disjunction of literals. A clause is normally represented like $l_1 \vee \ldots \vee l_n$ where l_i for $i \in [n]$ are the literals occurring in the clause, but sometimes we will omit the $\vee$ symbols, in this case a clause looks like $l_1 \ldots l_n$. We will denote clauses as roman capital letters such as A, B, C, etc.*

- *A k-*TERM *is a conjunction of up to $k \geq 1$ literals.*

- *A k-*CLAUSE *is a disjunction of k-terms. A clause is then a 1-clause.*

- *A* CNF FORMULA *is a conjunction of clauses. CNF stands for Conjunctive Normal Form. A CNF formula looks like $C_1 \wedge \cdots \wedge C_n$ where C_i for $i \in [n]$ are the clauses in the formula. As we are always dealing with CNF formulas we may call them just* FORMULAS*. Also we may represent formulas as a list of clauses $C_1, \ldots, C_n$. To denote the names of the formulas we will use math capital letters such as $\mathbb{F}$, $\mathbb{PHP}$, etc.*

- *An* ASSIGNMENT *to a formula $\mathbb{F}$ is a mapping from the variables in $\mathbb{F}$ to the values* TRUE *and* FALSE*. When the assignment is not total we may call it a* PARTIAL ASSIGNMENT*. We will denote assignments with short greek letters such as α, β, etc. A set of assignments will be denoted by greek capital letters, such as Γ, Δ, etc.*

- *Given a formula $\mathbb{F}$ and a partial assignment α, the* RESTRICTION *of $\mathbb{F}$ to α, denoted $\mathbb{F}_\alpha$ or also $\mathbb{F}|_\alpha$, is the formula obtained after changing in $\mathbb{F}$ the variables mapped in α by its values and simplifying the resulting formula.*

- *A formula $\mathbb{F}$ is unsatisfiable if no assignment to the variables in $\mathbb{F}$ satisfies $\mathbb{F}$, that is, no assignment makes $\mathbb{F}$* TRUE*.*

- *The* EMPTY CLAUSE*, denoted by λ, is the clause with no literals and is unsatisfiable.*

- *A* REFUTATION *for a formula $\mathbb{F}$ is a proof of the unsatisfiability of $\mathbb{F}$. A refutation will be denoted by $\mathcal{R}$, a treelike refutation by $\mathcal{T}$.*

- *A* DERIVATION *of a formula $\mathbb{F}'$ from a formula $\mathbb{F}$ is a proof of $\mathbb{F}'$ from $\mathbb{F}$. A derivation will be denoted by $\mathcal{R}$, a treelike derivation by $\mathcal{T}$.*

- *A* REFUTATIONAL *proof system is a proof system for refuting formulas.*

Our results concern mainly to refutational proof systems such as Resolution and extensions and restrictions of Resolution. Another refutational proof system studied in this work is the Cutting Planes proof system, which is also related to Resolution.

1.1.1 Resolution

RESOLUTION is a refutation proof system for CNF formulas introduced by Robinson in [60]. The only inference rule is the Resolution rule:

$$\frac{C \vee x \qquad D \vee \bar{x}}{C \vee D}\ .$$

From clauses $C \vee x$ and $D \vee \bar{x}$ we get the clause $C \vee D$ which is called RESOLVENT. In this example we say that variable x is CUT or ELIMINATED. A Resolution refutation of a CNF formula $\mathbb{F}$ is a derivation of λ from $\mathbb{F}$ using the resolution rule. Resolution is a sound and complete refutation system: a set of clauses has a Resolution refutation if and only if it is unsatisfiable.

A Resolution refutation of a CNF formula $\mathbb{F}$ is a list of clauses $C_1, \ldots, C_n$ such that C_n is λ, and for all $i \in [n]$, C_i is either a clause in $\mathbb{F}$ or a resolvent from two clauses C_j and C_k where $1 < j < k < i \leq n$. Any refutation in the form of a list of clauses can be transformed into a daglike refutation. The graph will have n nodes, each labeled by a clause. For a node C_i we will draw edges from the parent clauses C_j and C_k to C_i. An initial clause will have no incoming edges and the node λ will have no outgoing edges. If the graph is a tree we will have a treelike refutation.

Several restriction of Resolution have been proposed. These restrictions forbid to apply the Resolution rule under certain conditions, but maintaining the completeness. The idea

behind the restriction is to help to find refutations by limiting the search space. Some of the more studied restrictions are:

- REGULAR Resolution: Viewing the refutations as graph, in any path from λ to any initial clause, no variable is eliminated twice.
- ORDERED Resolution: There exists an arbitrary ordering of the variables in the formula, such that if a variable x is eliminated before a variable y on any path from an initial clause to λ, then x is before y in the ordering. As no variable is eliminated twice on any path, ordered Resolution is a restriction of regular Resolution. This system is also known as DAVIS-PUTNAM Resolution.
- NEGATIVE Resolution: To apply the Resolution rule, one of the two clauses should consist only of negative literals.

1.1.2 Extensions of Resolution

Resolution was generalized by Krajíček in [48]. The new proof system R(k) allows disjunctions of conjunctions of up to k literals and provides rules to work with them.

The inference rules are:

1. $\wedge$-INTRODUCTION
$$\frac{A \vee \bigwedge_{i\in I} l_i \quad B \vee \bigwedge_{i\in J} l_i}{A \vee B \vee \bigwedge_{i\in I\cup J} l_i}$$
2. k-CUT
$$\frac{A \vee \bigwedge_{i\in I} l_i \quad B \vee \bigvee_{i\in I} \bar{l}_i}{A \vee B}$$
3. WEAKENING
$$\frac{A}{A \vee \bigwedge_{i\in I} l_i}$$

where A and B are k-clauses, I, J are sets of indices such that $|I \cup J| \leq k$, where l_i for $i \in \{I \cup J\}$ are literals. Notice that R(1) is Resolution with a Weakening rule.

We will follow the notation in [48], so Resolution will be denoted by R(1), treelike Resolution by R*(1) and treelike R(k) by R*(k).

1.1.3 Cutting Planes

The CUTTING PLANES proof system, CP for short, is a refutational system for CNF formulas, as Resolution is. It works with linear inequalities. The initial clauses are transformed into linear inequalities in the following way:

$$\bigvee_{i=1}^{j} x_i \vee \bigvee_{i=1}^{k} \bar{y}_i \qquad \rightsquigarrow \qquad \sum_{i=1}^{j} x_i + \sum_{i=1}^{k} (1 - \bar{y}_i) \geq 1$$

We translate the boolean value TRUE into 1 and FALSE into 0. A CP refutation of a set $\mathcal{E}$ of inequalities is a derivation of $0 \geq 1$ from the inequalities in $\mathcal{E}$ and the axioms $x \geq 0$ and $-x \geq -1$ for every variable x, using the CP rules which are basic algebraic manipulations as addition of two inequalities, multiplication of an inequality by a positive integer and the following division rule:

$$\frac{\sum_{i \in I} a_i x_i \geq k}{\sum_{i \in I} \frac{a_i}{b} x_i \geq \left\lceil \frac{k}{b} \right\rceil} ,$$

where b is a positive integer that evenly divides all a_i, $i \in I$.

It can be shown that a set of inequalities has a CP refutation iff it has no $\{0,1\}$-solution. Any assignment satisfying the original clauses is actually a $\{0,1\}$-solution. It is also well known that CP polynomially simulates resolution [28], and this simulation preserves treelikeness. To unify the notation we will denote treelike CP by CP*.

1.2 Complexity measures

In this section we present in detail the complexity measures that will be used in this work. We will use calligraphic letters to denote these measures.

1.2.1 Size and length

In R(1) and CP size and length are polynomially related which for us means that are equivalent and we will use both words indistinctly.

The LENGTH of a R(k) or R*(k) refutation $\mathcal{R}$ is the number of k-clauses in $\mathcal{R}$. The length of refuting a formula $\mathbb{F}$ in R(k), denoted by $\mathcal{L}_k(\mathbb{F})$, is the minimal length of all R(k)

refutations for $\mathbb{F}$. The length of R*(k) refutations is denoted by $\mathcal{L}^*_k(\mathbb{F})$. Similarly, the length of refuting $\mathbb{F}$ in CP or CP* is denoted by $\mathcal{L}_{CP}(\mathbb{F})$ or $\mathcal{L}^*_{CP}(\mathbb{F})$.

1.2.2 Width

WIDTH is a recent complexity measure defined by Ben-Sasson and Wigderson in [17]. The width of a clause C, $\mathcal{W}(C)$ is the number of literals in C. The width of a set of clauses $\mathcal{C}$, $\mathcal{W}(\mathcal{C})$, is the maximum width of the clauses in $\mathcal{C}$. Note that a set of clauses can be for example a formula or a refutation. The width of refuting a formula $\mathbb{F}$, $\mathcal{W}(\mathbb{F} \vdash \lambda)$, is the minimal width of all the refutations for $\mathbb{F}$.

In [17] there were also proved interesting relations between size and width.

Theorem 6 *For an unsatisfiable CNF formula $\mathbb{F}$ with n variables,*

- $\mathcal{L}^*_1(\mathbb{F}) \geq 2^{\mathcal{W}(\mathbb{F}\vdash\lambda)-\mathcal{W}(\mathbb{F})}$
- $\mathcal{L}_1(\mathbb{F}) \geq \exp(\Omega((\mathcal{W}(\mathbb{F} \vdash \lambda) - \mathcal{W}(\mathbb{F}))^2/n))$

Observe that we have not defined width for R(k) in general because as R(1) refutations are R(k) refutations, the width for R(k) cannot be bigger that the width for R(1). On the other hand is very easy to transform a R(k) refutation into a R(1) refutation without increasing the width, but possibly increasing the size. That means that the width for R(k) refutations cannot be smaller than the width for R(1) refutations.

1.2.3 Space

Another measure for the complexity of R(1) refutations is the amount of SPACE it needs. This measure was defined in [44] in the following way:

Definition 7 *[44] An unsatisfiable CNF formula $\mathbb{F}$ has* R(1) *refutation bounded by space k if there is a series of CNF formulas $\mathbb{F}_1, \ldots, \mathbb{F}_s$, such that $\mathbb{F} = \mathbb{F}_1$, $\lambda \in \mathbb{F}_s$, in any $\mathbb{F}_i$ there are at most k clauses, and for each $i < s$, $\mathbb{F}_{i+1}$ is obtained from $\mathbb{F}_i$ by deleting, if wished, some of its clauses and adding the resolvent of two clauses of $\mathbb{F}_i$.*

Intuitively this expresses the idea of keeping a set of active clauses in the refutation, and producing from this set a new one by copying clauses from the previous set and resolving one pair of clauses, until λ is included in the set. Initially the set of active clauses consists of all the clauses of $\mathbb{F}$, and the space needed is the maximum number of clauses that are simultaneously active in the refutation.

The above definition has the important drawback that the space needed in a refutation can never be less than the number of clauses in the formula being refuted. This is so because the formula is the first one in the sequence used to derive λ. Making an analogy with a more familiar computation model, like the Turing machine, this is the same as saying that the space needed cannot be less than the size of the input being processed. To be able to study problems in which the working space is smaller than the size of the input, the space needed in the input tape is usually not taken into consideration. We do the same for the case of R(1) and introduce the following alternative definition for the space needed in a refutation.

Definition 8 *([30]) An unsatisfiable CNF formula* $\mathbb{F}$ *has* R(1) *refutation bounded by space* k *if there is a series of CNF formulas* $\mathbb{F}_1, \ldots, \mathbb{F}_s$*, such that* $\mathbb{F}_1 \subseteq \mathbb{F}$*,* $\lambda \in \mathbb{F}_s$*, in any* $\mathbb{F}_i$ *there are at most* k *clauses, and for each* $i < s$*,* $\mathbb{F}_{i+1}$ *is obtained from* $\mathbb{F}_i$ *by deleting some of its clauses, or adding the resolvent of two clauses of* $\mathbb{F}_i$*, or adding some of the clauses of* $\mathbb{F}$*.*

So we can give a definition of the space for refuting a formula.

Definition 9 *The* SPACE *needed for refuting in* R(1) *an unsatisfiable formula is the minimum* k *for which the formula has a refutation bounded by space* k*.*

In the new definition it is allowed to add initial clauses to the set of active clauses at any stage in the refutation. Therefore these clauses do not need to be stored and do not consume much space since in any moment at most two of them are needed simultaneously. The only clauses that consume space are the ones derived at intermediate stages. The space for refuting a formula can now range from constant to lineal.

There is another natural way to look at this definition using pebble games on graphs, a traditional model used for space measures in Complexity Theory and for register allocation

problems, see [62]. As said in Subsection 1.1.1, R(1) refutations can be represented as directed acyclic graphs of indegree two, in which the nodes are labeled with the clauses.

Definition 10 *In a directed acyclic graph G the* SOURCE *nodes are the nodes with no incoming edges, that is, with no predecessors and the* TARGET *nodes are the nodes with no outgoing edges, that is, with no successors.*

In a graph representing a R(1) refutation, the source nodes are the initial clauses, all the other nodes have indegree two, and the only target node is λ.

The space required for the R(1) refutation of a CNF formula $\mathbb{F}$, as expressed in Definition 8, corresponds to the minimum number of pebbles needed in the following game played on the graphs of refutations of $\mathbb{F}$.

Definition 11 *Given a connected directed acyclic graph with one target the aim of the* PEBBLE GAME *is to put a pebble on the target of the graph, following this set of rules:*

1) A pebble can be placed in any source node.

2) Any pebble can be removed from any node at any time.

3) A node can be pebbled provided all its parent nodes are pebbled.

3') If all the parent nodes of node are pebbled, instead of placing a new pebble on it, one can shift a pebble from a parent node.

An important concept that will be used often in this work is:

Definition 12 *The* PEBBLING NUMBER *of a directed acyclic graph G is the minimal number of pebbles needed to put a pebble in a target node of G following the rules of the pebbling game. The pebbling number of a G is denote by $\mathcal{P}(G)$.*

There are several variations of this simple pebble game in the literature. In [68] it is shown that the inclusion of rule 3' in the game can at most decrease by one the number of pebbles needed to pebble a graph, but in the worst case the saving is obtained at the price of squaring the number of moves needed in the game. We include rule 3' so that the number of

pebbles coincides exactly with the space in Definition 8. This fact is stated in the following straightforward Lemma.

Lemma 13 *Let $\mathbb{F}$ be an unsatisfiable CNF formula. The space needed in a* R(1) *refutation $\mathcal{R}$ of $\mathbb{F}$ is $\mathcal{P}(\mathcal{R})$.*

Definition 11 allows us to use techniques introduced for the estimation of the number of pebbles required for pebbling certain graphs, for computing the space needed in R(1) refutations. However the estimation of the number of pebbles needed in the refutation of a formula is harder than the estimation of the number of pebbles needed for a graph, since in the first case one has to consider all the possible refutation graphs for the formula.

We also give the formulation of space in [2] which is equivalent to ours, but some results in this work are proved using the terminology in [2]. We start by defining a CONFIGURATION.

Definition 14 *A* CONFIGURATION *is set of k-clauses.*

We use calligraphic letters to denote configurations. We will use the word configuration to denote the set of clauses pebbled in an stage of a pebbling strategy.

The formulation of [2] uses the concept of configuration.

Definition 15 *A* R(1) *refutation of a formula $\mathbb{F}$ can be viewed as a list of configurations $\mathcal{C} = \mathcal{C}_0, \ldots, \mathcal{C}_s$ such that $\mathcal{C}_0 = \emptyset$, $\mathcal{C}_s = \lambda$ and each $\mathcal{C}_t$ for $t \in [s]$ is obtained by one of the following rules:*

- AXIOM DOWNLOAD*: $\mathcal{C}_t = \mathcal{C}_{t-1} \cup C$ for some clause $C \in \mathbb{F}$;*
- MEMORY ERASING*: $\mathcal{C}_t = \mathcal{C}_{t-1} - C$ for some $C \in \mathcal{C}_{t-1}$;*
- INFERENCE ADDING*: $\mathcal{C}_t = \mathcal{C}_{t-1} \cup C$, for some C obtained by the Resolution rule applied to two clauses in $\mathcal{C}_{t-1}$.*

So, the definition of space of a refutation $\mathcal{C}$ is:

Definition 16 *Given a refutation $\mathcal{C}$ as a list of configurations, the* SPACE *of $\mathcal{C}$ is the maximal length of a configuration in $\mathcal{C}$.*

If we call $\mathcal{C}_i$ to the set of pebbled clauses in the ith pebbling step, it is clear that both Definition 11 and Definition 16 are equivalent.

We will denote the space for refuting $\mathbb{F}$ in R(k) as $\mathcal{S}_k(\mathbb{F})$, and for R*($k$), by $\mathcal{S}^*_k(\mathbb{F})$.

1.3 Formulas

We present the main CNF formulas that are studied in this work. In Section 1.5 we will state results concerning these formulas, so putting our own results in context.

1.3.1 Pigeonhole Principle

The most studied CNF formula is the PIGEONHOLE PRINCIPLE, usually shortened to $\mathbb{PHP}$. It expresses that it is not possible to have a one-to-one mapping from m objects to n places, when $m > n$. Let the variable $x_{i,j}$ denote that the ith-object is placed in the jth-place. We can write a CNF formula for the Pigeonhole Principle as follows:

$$x_{i,1}x_{i,2}\dots x_{i,n} \qquad i \in [m] \tag{1.1}$$

$$\bar{x}_{i,k}\bar{x}_{j,k} \qquad 1 \leq i < j \leq m, 1 \leq k \leq n \tag{1.2}$$

Clauses 1.1 say that every object must be placed somewhere. Clauses 1.2 say that in every place there is at most one object.

When $m \geq 2n$ the formula is known as WEAK pigeonhole principle. Haken in [38] gave the first exponential size lower bound for R(1). Recently Razborov has dedicated a survey [59] to the Pigeonhole Principle collecting and commenting all the results concerning several proof systems and variations of the standard Pigeonhole Principle.

1.3.2 Tseitin Formulas

These formulas where defined in [65]. Let $G = (V, E)$ be a connected undirected graph with n vertices, and let $m : V \to \{0, 1\}$ be a marking of the vertices of G satisfying the property

$$\sum_{v \in V} m(x) = 1 (\text{mod } 2).$$

For such a graph we can define an unsatisfiable CNF formula $\mathbb{T}(G,m)$ in the following way: The formula has E as set of variables, and is the conjunction of the translation in CNF of the formulas $\mathbb{PAR}_v$ for $v \in V$, where

$$\mathbb{PAR}_v = \begin{cases} e_1(v) \oplus \cdots \oplus e_d(v) & \text{if} \quad m(v) = 1 \\ \overline{e_1(v) \oplus \cdots \oplus e_d(v)} & \text{if} \quad m(v) = 0 \end{cases}$$

Here $e_1(v), \ldots, e_d(v)$ are the edges, variables, incident with vertex v. If d is the maximum degree of a node in G, $\mathbb{T}(G,m)$ contains at most $n2^{d-1}$ many clauses, each one with at most d many literals. The number of variables in the formula is bounded by $\frac{dn}{2}$.

$\mathbb{T}(G,m)$ captures the combinatorial principle that for all graphs the sum of the degrees of the vertices is even. When the marking m is odd, $\mathbb{T}(G,m)$ is unsatisfiable. Suppose on the contrary that there were a satisfying assignment $\alpha : E \to \{0,1\}$. For every vertex v, the number of edges of v that have been assigned value 1 by α has the same parity as $m(x)$, and therefore

$$\sum_{v \in V} \sum_{(v,w) \in E} \alpha((v,w)) = \sum_{v \in V} m(v) \equiv 1 (\mod 2)$$

but in the left hand sum in the equality, every edge is counted twice and therefore this sum must be even, which is a contradiction. Tseitin in [65] gave the first exponential size lower bound for a nontrivial proof system, concretely regular R(1). Usually the marking will be omitted and the formula will be denoted as $\mathbb{T}(G)$ in the understanding that we are considering an odd marking. Note that applying an assignment to $\mathbb{T}(G,m)$ has curious consequences with m. Let e be the edge joining nodes u and v, and let restrict $\mathbb{T}(G,m)$ with $e = 1$, that is, $\mathbb{T}(G,m)_{e=1}$. The new formula is $\mathbb{T}(G',m')$, where $G' = (V, E - e)$ and $m'(u)$ is $m(u)$ toggled and $m'(v)$ is $m(v)$ toggled.

1.3.3 Graph Tautologies

The Graph Tautologies, $\mathbb{GT}_n$ are unsatisfiable CNF formulas based on directed graphs with n nodes. Let variable $x_{i,j}$ mean that there is an edge from node i to node j.

$$\bar{x}_{i,j}\bar{x}_{j,k}x_{i,k} \qquad i,j,k \in [n], i \neq j \neq k \tag{1.3}$$

$$\bar{x}_{i,j}\bar{x}_{j,i} \qquad i,j \in [n], i \neq j \tag{1.4}$$

$$x_{1,i} \ldots x_{i-1,i}x_{i+1,i} \ldots x_{n,i} \qquad i \in [n] \tag{1.5}$$

Clauses 1.3 say that when there is an edge from node i to node j and an edge to node j to node k, then there is an edge from node i to node k. Clauses 1.4 say that there are no cycles of size two. Graphs that satisfy these clauses must have a node with no incoming edges. Clauses 1.5 force that all nodes have an incoming edge, thus getting an unsatisfiable formula. This formula have short resolution proofs, see [64], nevertheless Bonet and Galesi in [19] proved a width lower bound of n. This result is very important because it shows that a width lower bound of the square root in the number of variables does not give a superpolynomial size R(1) lower bound. The space for this formula is also n, see [2].

1.3.4 Pebbling Contradictions

The Pebbling Contradictions are formulas based in directed acyclic graphs of indegree 2 and the Pebbling Game, recall Definitions 11 and 12. Let us call w to the variable representing node w. The meaning of variable w is that the node can be pebbled. Remember that a source node is a node with no predecessors and a target node is a node with no successors.

Let $G = (V, E)$. A node w can be pebbled if all its parents nodes are pebbled. We can represent it with the clause $\bar{u}\bar{v}w$ where u and v are the parents of w in G. If w is a source the clause becomes just w and we call it SOURCE CLAUSE, otherwise it is called a PEBBLING CLAUSE. In order to obtain a contradiction we add for each target node $t \in V$ the TARGET CLAUSE $\bar{t}$. We denote this contradiction by $\mathbb{PEB}(G)$.

An interesting result about $\mathbb{PEB}(G)$ appeared in [14]. It is proven that $\mathbb{PEB}(G)$ cannot have R(1) refutations with both constant space and constant width. It is easy to find R(1),

in fact $\mathrm{R}^*(1)$, refutations with constant space but the width is $\Theta(\mathcal{P}(G))$ and also $\mathrm{R}(1)$ refutations with constant width, but then the space is $\Theta(\mathcal{P}(G))$.

These contradictions can be generalized in the following way. The contradiction $\mathbb{PEB}_k^l(G)$ where $l, k \geq 1$ is obtained from $\mathbb{PEB}(G)$ by introducing $k \cdot l$ variables $v_{i,j}$, $i \in [l]$, $j \in [k]$ for each variable v in $\mathbb{PEB}(G)$. Each variable v is replaced by

$$\bigwedge_{i \in [l]} \bigvee_{j \in [k]} v_{i,j}.$$

The resulting formula is then transformed into CNF using de Morgan's laws, and distributivity. Hence, each source clause s in $\mathbb{PEB}(G)$ will correspond to the $\mathbb{PEB}_k^l(G)$ source clauses

$$s_{i,1} \vee \cdots \vee s_{i,k}$$

for $i \in [l]$. Each target clause $\bar{t}$ in $\mathbb{PEB}(G)$ will correspond to the $\mathbb{PEB}_k^l(G)$ target clauses.

$$\bar{t}_{1,j_1} \vee \cdots \vee \bar{t}_{l,j_l}$$

for $j_1, \ldots, j_l \in [k]$. And each pebbling clause $\bar{u} \vee \bar{v} \vee w$ in $\mathbb{PEB}(G)$ will correspond to the $\mathbb{PEB}_k^l(G)$ pebbling clauses

$$\bar{u}_{1,j_1} \vee \cdots \vee \bar{u}_{l,j_l} \vee \bar{v}_{1,m_1} \vee \cdots \vee \bar{v}_{l,m_l} \vee w_{i,1} \vee \cdots \vee w_{i,k}$$

for $j_1, \ldots, j_l, m_1, \ldots, m_l \in [k]$, $i \in [l]$. Clearly, $\mathbb{PEB}_k^l(G)$ is a contradiction since $\mathbb{PEB}(G)$ is. Moreover $\mathbb{PEB}_k^l(G)$ has small $\mathrm{R}^*(k)$ refutations. Ben-Sasson *et al.* considered in [17] the formulas $\mathbb{PEB}_2^1(G)$ to give a quasioptimal size separation between $\mathrm{R}(1)$ and $\mathrm{R}^*(1)$. These formulas have exponential $\mathrm{R}^*(1)$ refutations but constant width polynomial size $\mathrm{R}(1)$ refutations.

1.3.5 Random Formulas

Let $\mathbf{F}_m^n$ be the probability distribution obtained by selecting m clauses of size exactly 3 independently, uniformly at random from the set of all $2^3 \cdot \binom{n}{3}$ clauses of size 3 built on n distinct variables. $\mathbb{F} \sim \mathbf{F}_m^n$, means that $\mathbb{F}$ is selected at random from this distribution. A random 3-CNF formula is a formula $\mathbb{F} \sim \mathbf{F}_m^n$.

1.4 Circuit Complexity

In this section we introduce concepts about Circuit Complexity that will be used in several places of this work. As said in the Introduction, Circuit Complexity results are often used in Proof Complexity. The reason is that under certain circumstances R(1) or CP refutations can be transformed into circuit computing a function related to the formula being refuted. The size of the circuit is similar to that of the original refutation, so size lower bounds for circuits can be translated into size lower bounds for refutations, see for example [20, 39].

1.4.1 Monotone circuits

Definition 17 *A* BOOLEAN FUNCTION *in the boolean variables $x_1, \ldots, x_n$ is a map $f : \{0,1\}^n \to \{0,1\}$.*

Definition 18 *A* MONOTONE BOOLEAN FUNCTION *f is a boolean function such that for any two inputs a and b, when $a \leq b$ holds that $f(a) \leq f(b)$.*

Definition 19 *A* BOOLEAN CIRCUIT *is a directed acyclic graph of indegree 2, where source nodes are labeled by variables and boolean constants, and nonsource nodes are called* gates *and are labeled with the boolean function computed at that gate.*

Definition 20 *A* MONOTONE BOOLEAN CIRCUIT *is a boolean circuit computing a monotone boolean function using monotone gates.*

Definition 21 *A* MONOTONE BOOLEAN FORMULA *is a fanout 1 monotone boolean circuit.*

The class of real circuits was introduced by Pudlák[54] and are a generalization of boolean circuits.

Definition 22 *A* MONOTONE REAL CIRCUIT *is a circuit of fanin 2 computing with real numbers where every gate computes a nondecreasing real function. The circuits output 0 or 1 on every input of zeroes and ones only. A* MONOTONE REAL FORMULA *is a fanout 1 monotone real circuit.*

We list the main complexity measures for circuits.

Definition 23 *Complexity measures for circuits.*

1. *The* SIZE *of a circuit is the number of gates.*

2. *The* SIZE *of a function is the minimal size of its circuits. We will denote boolean (monotone) circuit size by $\mathcal{S}_{\mathbb{B}}$ ($\mathcal{S}_{\mathbb{MB}}$), and real (monotone) circuit size by $\mathcal{S}_{\mathbb{R}}$ ($\mathcal{S}_{\mathbb{MR}}$).*

3. *The* DEPTH *of a circuit is the length of the longest path from the target to a source.*

4. *The* DEPTH *of a function is the minimal depth of its circuits. We will denote boolean (monotone) circuit depth by $\mathcal{D}_{\mathbb{B}}$ ($\mathcal{D}_{\mathbb{MB}}$), and real (monotone) circuit depth by $\mathcal{D}_{\mathbb{R}}$ ($\mathcal{D}_{\mathbb{MR}}$).*

5. *A circuit is called* FORMULA *if every gate has fanout at most 1. The size of monotone real formulas is denoted by $\mathcal{S}^*_{\mathbb{MR}}$.*

Lower bounds on the size of monotone real circuits were given by Pudlák [54], Cook and Haken [39] and Fu [32]. Rosenbloom [61] shows that they are strictly more powerful than monotone boolean circuits, since every slice function can be computed by a linear size, logarithmic depth monotone real circuit, whereas most slice functions require exponential size boolean circuits. On the other hand, Jukna [42] gives a general lower bound criterion for monotone real circuits, and uses it to show that certain functions in $P/poly$ require exponential size monotone real circuits. Hence the computing power of monotone real circuits and boolean circuits is incomparable.

1.4.2 The feasible monotone interpolation property

For the separations from CP* to CP and from R*(1) to R(1) in Section 2.1 we use the following version of feasible monotone interpolation property. Theorem 24 relates the size of CP refutations with the size of monotone real circuits and also the size of CP* refutations with the size of monotone real formulas.

Theorem 24 *([54]) Let $\vec{p}, \vec{q}, \vec{r}$ be disjoint vectors of variables, and let $A(\vec{p}, \vec{q})$ and $B(\vec{p}, \vec{r})$ be sets of inequalities in the indicated variables such that the variables $\vec{p}$ either have only nonnegative coefficients in $A(\vec{p}, \vec{q})$ or have only nonpositive coefficients in $B(\vec{p}, \vec{r})$.*

Suppose there is a CP *refutation $\mathcal{R}$ of $A(\vec{p}, \vec{q}) \cup B(\vec{p}, \vec{r})$. Then there is a monotone real circuit $C(\vec{p})$, called the* INTERPOLANT, *of size $O(|\mathcal{R}|)$ such that for any vector $\vec{a} \in \{0,1\}^{|\vec{p}|}$*

$$C(\vec{a}) = 0 \quad \rightarrow \quad A(\vec{a}, \vec{q}) \text{ is unsatisfiable}$$
$$C(\vec{a}) = 1 \quad \rightarrow \quad B(\vec{a}, \vec{r}) \text{ is unsatisfiable}$$

Furthermore, if $\mathcal{R}$ is treelike, then $C(\vec{p})$ is a monotone real formula.

Skipping the condition that the variables $\vec{p}$ either have only nonnegative coefficients in $A(\vec{p}, \vec{q})$ or have only nonpositive coefficients in $B(\vec{p}, \vec{r})$, the interpolant is real circuit or a real formula if $\mathcal{R}$ was a CP refutation or a CP* refutation.

For the case of R(1) or R*(1) refutations a simpler version of Theorem 24 suffices. The interpolant will be a monotone boolean circuit instead a monotone real circuit. This version of Theorem 24 will be also used in Section 2.2 to separate R(2) from R(1).

Also note that it is not stated in the original formulation of Theorem 24 in [54] that treelike refutations produce formulas instead of circuits, but this can be checked easily in the construction of the interpolant from the refutation.

1.5 Overview of results in the area

In this section we give an overview of results in Proof Complexity and put our results in context, explaining its significance and the relations with previous and posterior works of others.

Haken in [38] was the first in proving exponential R(1) size lower bounds. He showed that $\mathbb{PHP}_n^{n+1}$ requires exponential size R(1) refutations. The proof techniques of Haken where extended in [25] to prove that $\mathbb{PHP}_n^{n^{2-\epsilon}}$ requires exponential size R(1) refutations. Only recently it has been proved exponential lower bounds for $\mathbb{PHP}_n^m$ where $m \geq n^2$ [56, 58]. Urquhart [66] proved exponential R(1) size lower bounds for $\mathbb{T}(G)$ for a suitable family

of graphs G. Chvátal and Szemerédi [26] showed that in some sense, almost all classes of unsatisfiable CNF formulas require exponential size R(1) refutations. In [13, 17] there are simplified proofs of these results. All these exponential lower bounds are bad news for Automated Theorem Proving, since they mean that often the time used in finding refutations will be exponentially long in the size of the formula, just because the shortest refutations are also exponentially long in the size of the formula. Refinements of R(1) have been also studied. It is important to know if these refinements produce longer refutations than R(1), because these refinements are often used in Automated Theorem Proving. Goerdt in [35, 36, 37] gave superpolynomial separations between R(1) and several refinements: for ordered, negative and regular R(1) respectively. In [18] an exponential separation between ordered R(1) and R(1) is proved, in fact between ordered and negative R(1). Recently exponential separations have been proved between R(1) against regular and negative R(1), in [3, 22] respectively. It is also important to study efficiency issues for R*(1) because it is widely used in Automatic Theorem Proving. Its importance stems from the close relationship between the complexity of R*(1) refutations and the runtime of a certain class of satisfiability testing algorithms, the so-called DLL Algorithms, see [55, 12]. Superpolynomial separations between R*(1) and R(1) can be found in [67]. In [18], see Section 2.1, this separation is proven to be exponential, thus showing that finding R*(1) refutations is not an efficient strategy for finding R(1) refutations. In [16] this result is improved by giving a nearly optimal separation between R(1) and R*(1). All the separation results of [18] improve to exponential the previously known superpolynomial ones, and these exponential separations harden the known results showing inefficiency of several widely used strategies for finding proofs, especially for R(1).

There are also exponential size lower bounds for CP. Impagliazzo *et al.* [40] proved exponential size lower bounds for CP*. Bonet *et al.* [20] proved an exponential size lower bound for a subsystem of CP, where the coefficients appearing in the inequalities are polynomially bounded in the size of the formula being refuted. This is an important result because all known CP refutations fulfill this property. Finally, Pudlák [54] and Cook and Haken [39] gave general circuit complexity results from which exponential lower bounds for CP follow. To this day it is still unknown whether CP with bounded coefficients polynomially simulates CP. Since there is an exponential speedup of CP over R(1), see [28], it would be nice to find

an efficient algorithm for finding CP refutations and a question to ask is whether trying to find CP* refutations would be an efficient strategy for finding CP refutations. Johannsen [41] gave a superpolynomial separation, with a lower bound of the form $\Omega(n^{\log n})$, between CP* and CP. This was previously known for CP* with bounded coefficients in [20]. In [18] this separation is improved to exponential, this means that trying to find CP* refutations is not a good strategy for finding CP refutations. The separations between R*(1) and CP* against R(1) and CP respectively are obtained using the feasible monotone interpolation property introduced by Krajíček [46]. Closely related ideas appeared previously in the mentioned works that gave lower bounds for fragments of CP, see [40, 20]. The interpolation method applied to CP, translates proofs of certain formulas to monotone real circuits, a class of circuits which generalize boolean circuits. This transformation has two important features: it preserves the size, that is, the size of the circuit is of the order of the size of the proof from which it is built; and it preserves the structure, that is, treelike proofs give rise to treelike circuits. So it gives a way to reduce the problem of proving size lower bounds for CP* to that of giving lower bounds for the size of monotone real formulas. To use this method, in [18], we extend to monotone real circuits a result from [57] for monotone boolean circuits.

In [6] we give results about R(2). R(k) can be viewed either as an extension of R(1) or as a restriction of bounded-depth Frege. In [6, 4] it is proven that $\mathbb{PHP}_n^{cn}$ requires exponential size R(2) refutations. This is, to our knowledge, the first exponential lower bound for the weak Pigeonhole Principle in a subsystem of bounded-depth Frege that extends R(1). We state other results about $\mathbb{PHP}$ in several proof systems. For a complete treatment refer to [59]. Beame et al. [10] proved that $\mathbb{PHP}_n^{n+c}$ requires exponential size proofs in bounded-depth Frege systems and it is open whether lower bounds can be proved when the number of pigeons in greater than $n+c$. Regarding upper bounds, Buss [23] gave polynomial size proofs of $\mathbb{PHP}_n^{n+1}$ in unrestricted Frege systems. It is also known that $\mathbb{PHP}_n^{2n}$ has quasipolynomial size proofs in bounded-depth Frege [52, 49]. Also in [49] there is a quasipolynomial upper bound for depth-0.5 LK, which is equivalent to R($\log n$), when we allow conjunctions of up to polylog literals. As a consequence there is an exponential separation between R(2) and R($\log n$). In [6, 4] using techniques from [11], it is also obtained an exponential R(2) size lower bound for Random Formulas with a certain clause density. Again, this is the first

exponential lower for Random Formulas for a proof system stronger than R(1). This result may be considered as a first step towards proving them hard for bounded-depth Frege.

Another important question to ask is whether R(2) is more powerful than R(1). In [6] we prove that R(1) cannot polynomially simulate R(2), and therefore R(2) is superpolynomially more efficient than R(1). As a corollary, we see that R(2) does not have the feasible monotone interpolation property, solving this way a conjecture of Krajíček [48]. These results are in Section 2.2. This separation between R(1) and R(2) has been improved to slightly exponential in [5] using a different formula.

Another motivation for working with the system R(2) is to see how useful it can be for Automated Theorem Proving. Given that it is more efficient than R(1), because at least there is a superpolynomial separation, it might be a good idea to try to find good heuristics to find proofs in R(2).

In [29] there are some results about R$^*(k)$. It is proven that R$^*(k)$ forms a hierarchy regarding proof size, see Subsection 2.4.2. That means that there are formulas that require exponential size R$^*(k)$ refutation whereas they have polynomial size R$^*(k+1)$ refutations. This separation holds also between R$^*(2)$ and R$^*(1)$. In [29] it is also proven that R(1) dominates R$^*(k)$, see Subsection 2.4.3. This is a particular case of a simulation from [45], but we show that an increment by factor 2, independent of k, suffices. In [63] it is proved that R(k) forms a hierarchy regarding proof size, thus extending the result in [29], and that the Pigeon Principle with certain parameters and Random Formulas with certain initial width require exponential size R(k) refutations, thus extending the results in [6] which hold only for R(2).

Width is a complexity measure introduced in [17]. Under certain circumstances width lower bounds can provide exponential size R(1) lower bounds and proving a width lower bound should be easier than proving a R(1) size lower bound directly. In [17], previously known size R(1) lower bounds for formulas such as $\mathbb{PHP}$ and $\mathbb{T}(G)$ were proved in an unified way using the concept of width. In [19] it was proved that a width lower bound of the square root in the number of variables does not imply a superpolynomial R(1) size lower bounds, solving an open problem in [17]. Another interesting result about width is the combinatorial characterization of [7]. A Player-Adversary game over CNF formulas can be used to find

width bounds, this simplifies the task of proving width lower bounds and consequently the task of proving size lower bounds. In [30] a relationship between $\mathrm{R}^*(1)$ space and width was proved and was later extended to $\mathrm{R}(1)$ space also in [7].

Space is also a recent complexity measure. It was introduced in [30] along with general results about relationships between space and size and some space lower bounds for well studied formulas as $\mathbb{PHP}$ and $\mathbb{T}(G)$. Independently in [2] appeared an equivalent definition of space and the lower bounds in [30] were proved using this formulation along with a new space lower bound for $\mathbb{GT}_n$. The authors of [2] made a difference between what they call clause space, which is what we call just space, and variable space, in which it is taken into account not the number of clauses but the minimal number of literals that must be kept simultaneously in order to carry out the refutation. They also extended space to other proof systems. In [15] it was proved a new space lower bound for Random Formulas. In [29] all the previously known space lower bounds have been proved in an simpler and unified way using the concept of dynamical satisfiability also introduced in [29]. Besides this concept allows to extend all these lower bounds to $\mathrm{R}(k)$. This results appear in Section 3.5. The concept of dynamical satisfiability is very similar to the combinatorial characterization of width in [7], but it was found independently. As happened with respect to size, in [29] it is also shown that $\mathrm{R}^*(k)$ forms a hierarchy respect to space, see Section 3.4. So, there are formulas that require nearly linear space for $\mathrm{R}^*(k)$ whereas they have constant space $\mathrm{R}^*(k+1)$ refutations. In [31] a combinatorial characterization of $\mathrm{R}^*(1)$ space is proved, see Section 3.2. As in the case of the width characterization in [7] it is also via a Player-Adversary game over CNF formulas. It would be interesting to find a combinatorial characterization for $\mathrm{R}(1)$ space. An interesting open problem about $\mathrm{R}^*(1)$ space is the space for $\mathbb{PEB}_2^1(G)$ for a graph G with a high pebbling number. In [16] is proved an exponential $\mathrm{R}^*(1)$ size lower bound for these formulas. That lower bound implies by a result in [30] nearly linear $\mathrm{R}^*(1)$ space lower bounds that can also be obtained via the combinatorial characterization in [31]. $\mathbb{PEB}_2^1(G)$ has $\mathrm{R}(1)$ refutations with both polynomial size and constant width, but not much is known about the space. In [31] using this formula the first separation between $\mathrm{R}(1)$ space and $\mathrm{R}^*(1)$ space is given, see Section 3.3. This separation shows that the characterization of $\mathrm{R}^*(1)$ space is not valid for $\mathrm{R}(1)$ space. It would be interesting to prove a matching space lower bound

for $\mathbb{PEB}_2^1(G)$ or find an smaller space upper bound. As $\mathbb{PEB}_2^1(G)$ has constant space $\mathrm{R}^*(2)$ refutations using the dynamical satisfiability concept only a constant space lower bound can be proved, so if it happens that $\mathbb{PEB}_2^1(G)$ requires nonconstant space $\mathrm{R}(1)$ refutations, the dynamical satisfiability concept will not be a tight characterization of $\mathrm{R}(1)$ space.

1.6 Summary of results and organization of this work

After giving an overview of results in Proof Complexity we will comment our results separately to show clearly our contribution to the field of Proof Complexity. The rest of this work is divided into three chapters. Chapter 2 is devoted to the results about size regarding the Proof Systems presented in Section 1.1, which include size lower and upper bounds that when they are related provide separations between different proof systems or the treelike and the general version of the same proof system. In Chapter 3 we present results mainly about space complexity, including also space lower and upper bounds and relationships between space and other complexity measures. Chapter 4 shows a summary of the results in this work compared to previous and posterior related results. We also list some open problems related to our work.

Regarding size, in Section 2.1 we improve separations between treelike and general versions of $\mathrm{R}(1)$ and CP. To do so we extended a size lower bound from [57] for monotone boolean circuits to monotone real circuits. The results appeared in [18]. This kind of separations is interesting because some Automated Theorem Provers rely on the treelike version of proof systems, so the separations show that is not always a good idea to restrict to the treelike version. What we do is to prove CP^* exponential lower bounds for certain formulas via the feasible monotone interpolation property, see Theorem 24, which clearly are also lower bounds for $\mathrm{R}^*(1)$. To get the separation we show $\mathrm{R}(1)$ polynomial size upper bounds for the same formulas, which clearly are also upper bounds for CP. In fact we can separate not only $\mathrm{R}^*(1)$ from $\mathrm{R}(1)$, also we can separate $\mathrm{R}^*(1)$ from certain restrictions of $\mathrm{R}(1)$ like regular $\mathrm{R}(1)$ and negative $\mathrm{R}(1)$. The separation result for $\mathrm{R}^*(1)$ and $\mathrm{R}(1)$ was later improved in [16].

The rest of Chapter 2 is devoted to $\mathrm{R}(k)$ and $\mathrm{R}^*(k)$. After the apparition of $\mathrm{R}(k)$ which is

a system lying between R(1) and bounded-depth Frege it was important to study how powerful it is and its relation both with R(1) and bounded-depth Frege. In Section 2.2 we show that R(2) is strictly more powerful than R(1). We give a R(2) polynomial size upper bound for a certain Clique-Coclique principle reducing it to a Pigeonhole Principle with parameters that ensures polynomial size proofs. But as R(1) has the feasible monotone interpolation property, and it is known that monotone circuits separating cliques from cocliques require superpolynomial size, then R(1) refutations for this Clique-Coclique principle also require superpolynomial size. This separation answers an open problem by Krajíček, namely we show that R(2) does not have the feasible monotone interpolation property. This result appeared in [6]. The separation between R(1) and R(2) was later improved in [5]. In Section 2.3 we present an unpublished result that shows that R(2) lower bounds for $\mathbb{PHP}_n^{n^{1.5}}$ provides R(1) lower bounds for $\mathbb{PHP}_n^{n^2}$. This was a new attempt of solving a long standing open problem, the R(1) size for $\mathbb{PHP}_n^{n^2}$. Of course we do not know whether this approach would have made the proof easier, but as the problem was solved while we were working a it, see [56, 58], we abandoned this approach. Section 2.4, deals with R$^*(k)$. It was known that R(2) was more powerful that R(1) and R*(2) more powerful than R*(1), so a natural question was to find out whether we can separate successive levels of R(k) or R$^*(k)$. The answer is yes. We show exponential separations between successive levels of what we can call now the R$^*(k)$ hierarchy and Segerlind *et al.* [63] showed separations for the R(k) hierarchy. We also prove that R(1) simulates R$^*(k)$ which is a particular case of a theorem by Krajíček [45], but we can make the simulation shorter than the general simulation.

In Chapter 3 we show our results concerning R(k) space. R(1) space was defined in [30] improving a definition from [44]. Also in [2] there is an equivalent formulation of R(1) space. In Section 3.1 we give general results for R(1) and R*(1) space that appeared mainly in [30]. In Section 3.2 a combinatorial characterization of R*(1) space is proved. This result appeared in [31]. As in the case of the width characterization in [7] it is also via a Player-Adversary game over CNF formulas. It would be interesting to find a combinatorial characterization for R(1) space. In Section 3.3 we give another result from [31], namely the first space separation from R(1) to R*(1). We show that $\mathbb{PEB}_2^1(G)$ requires less space for R(1) than for R*(1), at least one third less. In Section 3.4 we show a result from [29]. As happened with

respect to size, $\mathrm{R}^*(k)$ forms a hierarchy respect to space. So, there are formulas that require nearly linear space for $\mathrm{R}^*(k)$ whereas they have constant space $\mathrm{R}^*(k+1)$ refutations. In Section 3.5 we present another result from [29]. We extend all previously known R(1) space lower bounds to $\mathrm{R}(k)$ in a simpler and unified way, that holds for R(1) as well, using the concept of dynamical satisfiability presented also in [29].

Chapter 2

Size

In this chapter we present the results relative to proof size. In Section 2.1 we prove several separations. The first, in Subsection 2.1.4, is an exponential separation between CP* and R(1) which of course implies an exponential separation between R*(1) and R(1) and also between CP* and CP. This separation was later improved in [16]. The second separation in Subsection 2.1.5 is between CP* and regular R(1). In Section 2.2 we present a superpolynomial separation between R(1) and R(2). We give a polynomial R(2) refutation of $\mathbb{PHP}$ with certain parameters, whereas there are only superpolynomial R(1) refutations for $\mathbb{PHP}$ with the same parameters. This solves an open question posed by Krajíček in [48], namely that R(2) has not the feasible monotone interpolation property, see Theorem 24. In Section 2.3 we prove that R(2) size lower bounds for $\mathbb{PHP}_n^{n^{1.5}}$ can be translated into R(1) size lower bounds for $\mathbb{PHP}_n^{n^2}$. This was a way of solving a longstanding open problem: the size complexity of $\mathbb{PHP}_n^{n^2}$, but as it was solved in [58, 56] later, we abandoned this work. In Section 2.4 we prove that R*(k) forms a hierarchy with respect to size and also we show that R(1) dominates R*(k) only by doubling the size of the refutation.

2.1 Size separation between CP* and R(1)

The main result of this section is an exponential separation between CP* and CP and also between R*(1) and R(1). The separations are obtained through the feasible monotone

interpolation property, see Section 1.4.2. To apply this property we had to extend the lower bounds for monotone boolean circuits of [57] to monotone real circuits, see Section 1.4.1. The results in this section are part of [18] and also have appeared in Galesi's dissertation [33]. I include in this work the results in which I had a significative part explaining also some results in [18, 33] which are needed to understand this section.

Before discussing how the results are obtained we need to define some new concepts that will be used only in this section.

2.1.1 Real Communication Complexity

Definition 25 *A $R \subseteq X \times Y \times Z$ is a* MULTIFUNCTION *if for every pair $(x, y) \in X \times Y$, there is a $z \in Z$ with $(x, y, z) \in R$.*

We view such a multifunction as a search problem, that is, given input $(x, y) \in X \times Y$, the goal is to find a $z \in Z$ such that $(x, y, z) \in R$.

Definition 26 *A* DETERMINISTIC COMMUNICATION PROTOCOL *P over $X \times Y \times Z$ specifies the exchange of information bits between two players, I and II, that receive as inputs respectively $x \in X$ and $y \in Y$ and finally agree on a value $P(x, y) \in Z$ such that $(x, y, P(x, y)) \in R$.*

Definition 27 *The* DETERMINISTIC COMMUNICATION COMPLEXITY *of R, $CC(R)$, is the number of bits communicated between players I and II in the optimal protocol for R.*

Definition 28 *A* REAL COMMUNICATION PROTOCOL *over $X \times Y \times Z$ is executed by two players I and II who exchange information by simultaneously playing real numbers and then comparing them according to the natural order of $\mathbb{R}$.*

This generalizes ordinary deterministic communication protocols in the following way: in order to communicate a bit, the sender plays this bit, while the receiver plays a constant between 0 and 1, so that he can determine the value of the bit from the outcome of the comparison.

Formally, such a protocol P is specified by a binary tree, where each internal node v is labeled by two functions $f_v^I : X \to \mathbb{R}$, giving player I's move, and $f_v^{II} : Y \to \mathbb{R}$, giving player

II's move, and each leaf is labeled by an element $z \in Z$. On input $(x, y) \in X \times Y$, the players construct a path through the tree according to the following rule:

> At node v, if $f_v^I(x) > f_v^{II}(y)$, then the next node is the left son of v, otherwise the right son of v.

The value $P(x, y)$ computed by P on input (x, y) is the label of the leaf reached by this path.

A real communication protocol P solves a search problem $R \subseteq X \times Y \times Z$ if for every $(x, y) \in X \times Y$, $(x, y, P(x, y)) \in R$ holds.

Definition 29 *The* REAL COMMUNICATION COMPLEXITY $CC_{\mathbb{R}}(R)$ *of a search problem R is the minimal depth of a real communication protocol that solves R.*

Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function, let $X := f^{-1}(1)$ and $Y := f^{-1}(0)$, and let the multifunction $R_f \subseteq X \times Y \times [n]$ be defined by

$$(x, y, i) \in R_f \quad \text{iff} \quad x_i = 1 \text{ and } y_i = 0$$

Definition 30 *The* KARCHMER-WIGDERSON *game for f is defined as follows: Player I receives an input $x \in X$ and Player II an input $y \in Y$. They have to agree on a position $i \in [n]$ such that $(x, y, i) \in R_f$. We will call R_f is the Karchmer-Wigderson game for the function f.*

There is a relation between the real communication complexity of R_f and the depth of a monotone real circuit or the size of a monotone real formula computing f, similar to the boolean case:

Lemma 31 (Krajíček [47]) *Let f be a monotone boolean function. Then*

1. $CC_{\mathbb{R}}(R_f) \leq \mathcal{D}_{\mathbb{MR}}(f)$;
2. $CC_{\mathbb{R}}(R_f) \leq \log_{3/2} \mathcal{S}^*_{\mathbb{MR}}(f)$.

For a proof see [47] or [41]. Notice that by (2) a linear lower bound for the real communication complexity of R_f gives an exponential lower bound for the size of the smallest monotone real formula computing f.

2.1.2 DART games and Structured Protocols

Raz and McKenzie [57] introduced a special kind of communication games, called DART games, and a special class of communication protocols, the structured protocols, for solving them.

Definition 32 *For $m, k \in \mathbb{N}$,* DART(m,k) *is the set of communication games specified by a relation $R \subseteq X \times Y \times Z$ such that:*

- *$X = [m]^k$. The inputs for Player I are k-tuples of elements $x_i \in [m]$.*
- *$Y = (\{0,1\}^m)^k$. The inputs for Player II are k-tuples of binary colorings y_i of $[m]$.*
- *For all $i = 1, \ldots, k$ let $e_i = y_i(x_i) \in \{0,1\}$, that is, the x_i-th bit in the m-bits string y_i. The relation $R \subseteq X \times Y \times Z$ defining the game only depends on $e_1, \ldots, e_k$ and z, hence we can describe $R(x,y,z)$ as $R((e_1, \ldots, e_k), z)$.*
- *$R((e_1, \ldots, e_k), z)$ can be expressed as a DNF Search Problem. There exists a DNF tautology F_R defined over the variables $e_1, \ldots, e_k$ such that Z is the set of terms of F_R, and $R((e_1, \ldots, e_k), z)$ holds if and only if the term z is satisfied by the assignment $(e_1, \ldots, e_k)$.*

Definition 33 *A* STRUCTURED PROTOCOL *for a* DART *game is a communication protocol for solving the search problem R, where player I gets input $x \in X$, player II gets input $y \in Y$, and in each round, player I reveals the value x_i for some i, and II replies with $y_i(x_i)$.*

Definition 34 *The* STRUCTURED COMMUNICATION COMPLEXITY *of $R \in$* DART(m,k)*, denoted by $SC(R)$, is the minimal number of rounds in a structured protocol solving R.*

In [57] it is was proven that $CC(R) \leq SC(R) \cdot \Omega(\log m)$. This is easy to generalize to real communication complexity.

Lemma 35 *For a* DART *game R, $CC_{\mathbb{R}}(R) \leq SC(R) \cdot \Omega(\log m)$.*

Proof. Observe that at each structured round the two players transmit $\lceil \log m \rceil + 1$ bits. The first player transmits a number in $[m]$ and the second answers with a bit. Observe that w.l.o.g. we can assume that both players know the structure of the protocol of the game. Therefore at each round they both know what is the coordinate i of the inputs they have to talk about and they have no need to transmit it, so the result follows. Q.E.D.

Theorem 36 is a generalization to real communication complexity of a result of [57]. It is necessary to produce lower bounds for monotone real circuits. The proof of Theorem 36 can be found in [18, 33].

Theorem 36 *Let $m, k \in \mathbb{N}$. For every relation $R \in \mathrm{DART}(m,k)$, where $m \geq k^{14}$,*

$$CC_{\mathbb{R}}(R) \geq SC(R) \cdot \Omega(\log m) .$$

From Lemma 35 and Theorem 36 follows:

Corollary 37 *[18] $CC_{\mathbb{R}}(R) = SC(R) \cdot \Omega(\log m)$.*

Another corollary to Theorem 36, is that for DART games, real communication protocols are no more powerful than deterministic communication protocols.

Corollary 38 *Let $m, k \in \mathbb{N}$. For $R \in \mathrm{DART}(m,k)$ with $m \geq k^{14}$, $CC_{\mathbb{R}}(R) = \Theta(CC(R))$.*

Proof. $CC(R) \geq CC_{\mathbb{R}}(R) \geq SC(R) \cdot \Omega(\log m) \geq \Omega(CC(R))$. Q.E.D.

At this point we must define the following concepts:

Definition 39 *A* MINTERM *(respectively a* MAXTERM*) of a boolean function $f : \{0,1\}^n \to \{0,1\}$ is a set of inputs $x \in \{0,1\}^n$ such that $f(x) = 1$ (respectively $f(x) = 0$) and for each $y \in \{0,1\}^n$ obtained from x by changing a bit from 1 to 0 (respectively by changing a bit from 0 to 1) it holds that $f(y) = 0$ (respectively $f(y) = 1$).*

We will apply the feasible monotone interpolation property, see Subsection 1.4.2 on a formula $A(\vec{p}, \vec{q}) \cup B(\vec{p}, \vec{r})$ such that $A(\vec{p}, \vec{q})$ will encode that $\vec{p}$ is a minterm of f and $B(\vec{p}, \vec{r})$ will encode that $\vec{p}$ is maxterm of f. Given a CP* refutation of $A(\vec{p}, \vec{q}) \cup B(\vec{p}, \vec{r})$, the interpolant provided by Theorem 24 will be a monotone real formula $C(\vec{p})$ which computes the function

f. The fact that $C(\vec{p})$ is a monotone real treelike circuit if the refutation $\mathcal{R}$ is treelike is not part of the original theorem, but can be directly obtained from the proof of the theorem in [54]. The reason is that the underlying graphs of the refutation and the circuit are the same. Therefore if we are able to prove exponential lower bound for the size of the treelike monotone real circuits computing f we immediately obtain an exponential lower bound for CP* and *a fortiori* for R(1) refutations for $A(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$.

To get the separation we need a monotone boolean function such that:

- has exponential lower bounds for monotone real formulas computing it and
- the corresponding $A(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ have polynomial size R(1) refutations, and therefore also polynomial size CP refutations.

For the monotone boolean function f we consider the monotone function GEN_n of n^3 inputs $t_{a,b,c}$, $a,b,c \in [n]$ defined as follows:

Definition 40 $\text{GEN}_n(\vec{t}) = 1$ *iff* $\vdash n$, *where for* $c \in [n]$, $\vdash c$ *means* c *is generated, which is defined recursively by* $\vdash c$ *iff* $c = 1$ *or there are* $a, b \leq n$ *with* $\vdash a$, $\vdash b$ *and* $t_{a,b,c} = 1$.

Sometimes we will write $a, b \vdash c$ for $t_{a,b,c} = 1$.

We will prove exponential lower bounds for the size of treelike monotone real circuits computing GEN_n in Section 2.1.3. The formulas $A(\vec{p},\vec{q})$ and $B(\vec{p},\vec{r})$ expressing respectively a minterm and a maxterm of GEN_n, with short R(1) refutations are presented in Section 2.1.4.

2.1.3 Lower bounds for Real Communication Complexity

We want to prove a $\Omega(n^\epsilon)$ lower bound for the real communication complexity of the Karchmer-Wigderson game associated to GEN_n. We will consider a DART game related to the GEN_n function. In this game the generation will proceed in a pyramidal way. We first define a set that will ease the definition of the game.

Definition 41 *For* $d \in \mathbb{N}$, *let* $P_d = \{\, (i,j)\, ;\, 1 \leq j \leq i \leq d \,\}$.

Following [57], we define the DART game $\text{PYR}(m,d)$, related to the GEN_n, with parameters $m = d^{28}$ and $n = \binom{d+1}{2} m + 2$, so that $d \approx n^{1/30}$.

Definition 42 *We regard the indices as elements of P_d, so that the inputs for the two players I and II in the* PYR(m,d) *game are respectively sequences of elements $x_{i,j} \in [m]$ and $y_{i,j} \in \{0,1\}^m$ with $(i,j) \in P_d$, and we picture these as laid out in a pyramidal form with $(1,1)$ at the top and (d,j), $1 \leq j \leq d$ and the bottom. The goal of the game is to find either an element colored 0 at the top of the pyramid, or an element colored 1 at the bottom of the pyramid, or an element colored 1 with the two elements below it colored 0. That is we have to find indices (i,j) such that one of the following holds:*

1. *$i = j = 1$ and $y_{1,1}(x_{1,1}) = 0$, or*
2. *$y_{i,j}(x_{i,j}) = 1$ and $y_{i+1,j}(x_{i+1,j}) = 0$ and $y_{i+1,j+1}(x_{i+1,j+1}) = 0$, or*
3. *$i = d$ and $y_{d,j}(x_{d,j}) = 1$.*

Observe that, setting $e_{i,j} = y_{i,j}(x_{i,j})$ for $1 \leq j \leq i \leq d$, this search problem can be defined as a DNF search problem given by the following DNF tautology:

$$\bar{e}_{1,1} \vee \bigvee_{1 \leq j \leq i \leq d-1} (e_{i,j} \wedge \bar{e}_{i+1,j} \wedge \bar{e}_{i+1,j+1}) \vee \bigvee_{1 \leq j \leq d} e_{d,j}$$

Therefore, PYR(m,d) is a game in DART$(m, \binom{d+1}{2})$.

Theorem 43 *For some $\epsilon > 0$ and sufficiently large n $CC_{\mathbb{R}}(R_{\text{GEN}_n}) \geq \Omega(n^{\epsilon})$.*

Proof. The theorem follows from the following results:

$$\begin{aligned}
&d \leq SC(\text{PYR}(m,d)) && \text{(Lemma 44)}\\
&\Omega(\log m) SC(\text{PYR}(m,d)) \leq CC_{\mathbb{R}}(\text{PYR}(m,d)) && \text{(Theorem 36)}\\
&CC_{\mathbb{R}}(\text{PYR}(m,d)) \leq CC_{\mathbb{R}}(R_{\text{GEN}_n}) && \text{(Lemma 45)}
\end{aligned}$$

Q.E.D.

Lemma 44 is proved in [57]. In [18, 33], Theorem 36 is proven for every DART game R. Lemma 45 is an adaptation of a proof in [57]. A lower bound on the structured communication complexity of PYR(m,d) was proved in [57]:

Lemma 44 ([57]) *$SC(\text{PYR}(m,d)) \geq d$.*

Lemma 45 shows that the real communication complexity of the game PYR(m,d) is bounded by the real communication complexity of the Karchmer-Wigderson game for GEN_n for a suitable n. The proof is adapted from [57].

Lemma 45 *Let $d, m \in \mathbb{N}$ and let $n := m \cdot \binom{d+1}{2} + 2$, then $CC_{\mathbb{R}}(\text{PYR}(m,d)) \leq CC_{\mathbb{R}}(R_{\text{GEN}_n})$.*

Proof. We prove that any protocol P solving the Karchmer-Wigderson game for GEN_n can be used to solve the PYR(m,d) game. Recall that PYR(m,d) is a $DART(m, \binom{d+1}{2})$ game, so the two players I and II receive inputs respectively of the form $(x_{1,1}, \ldots, x_{d,d})$ where $x_{i,j} \in [m]$ for all $(i,j) \in P_d$ and $(y_{1,1}, \ldots, y_{d,d})$ where $y_{i,j} \in \{0,1\}^m$ for all $(i,j) \in P_d$.

From their respective inputs for the PYR(m,d) game, Player I and II compute respectively a minterm $t^x_{a,b,c}$ and a maxterm $t^y_{a,b,c}$, for GEN_n and then they play the Karchmer-Wigderson game applying the protocol P.

As in [57] we consider fixed the element 1 as a bottom generator and the element n as the element we want to generate. We interpret the remaining $n = \binom{d+1}{2}m$ elements between 2 and $n-1$ as triples (i,j,k), where $(i,j) \in P_d$ and $k \in [m]$.

Now player I computes from his input $(x_{1,1}, \ldots, x_{d,d})$ an input $t^x_{a,b,c}$ for GEN_n such that $\text{GEN}_n(t^x_{a,b,c}) = 1$ by setting the following (recall that $a, b \vdash c$ means $t_{a,b,c} = 1$):

$$
\begin{array}{ll}
1, 1 \vdash g_{d,j} & \text{for } 1 \leq j \leq d \\
g_{1,1}, g_{1,1} \vdash n & \\
g_{i+1,j}, g_{i+1,j+1} \vdash g_{i,j} & \text{for } (i,j) \in P_{d-1}
\end{array}
$$

where $g_{i,j} := (i,j,x_{i,j}) \in \{2, \ldots, n-1\}$ and all the other bits $t^x_{a,b,c} = 0$. This completely determines $t^x_{a,b,c}$ and obviously $\text{GEN}_n(t^x_{a,b,c}) = 1$ since we have forced a generation of n (in a pyramidal form).

Likewise Player II computes from his input $(y_{1,1}, \ldots, y_{d,d})$ a coloring col of the elements from $[n]$ by setting $col(1) = 0$, $col(n) = 1$ and $col((i,j,k)) = y_{i,j}(k)$ (the k-th bit of $y_{(i,j)}$). From this coloring, he computes an input $t^y_{a,b,c}$ by setting $t^y_{a,b,c} = 1$ iff it is not the case that $col(c) = 1$ and $col(a) = col(b) = 0$. Obviously $\text{GEN}_n(t^y_{a,b,c}) = 0$.

Playing the Karchmer-Wigderson game P for GEN_n now yields a triple (a, b, c) such that $t^x_{a,b,c} = 1$ and $t^y_{a,b,c} = 0$. By definition of t^y, this means that $col(a) = col(b) = 0$ and $col(c) = 1$, and by definition of t^x one of the following cases must hold:

- $a = b = 1$ and $c = g_{d,j}$ for some $j \leq d$. By definition of col, $y_{d,j}(x_{d,j}) = 1$.
- $c = n$ and $a = b = g_{1,1}$. In this case, $y_{1,1}(x_{1,1}) = 0$.
- $a = g_{i+1,j}$, $b = g_{i+1,j+1}$ and $c = g_{i,j}$. Then we have $y_{i,j}(x_{i,j}) = 1$, and $y_{i+1,j}(x_{i+1,j}) = y_{i+1,j+1}(x_{i+1,j+1}) = 0$.

In any case, the players have solved $\text{PYR}(m, d)$ without additional communication. Q.E.D.

From Theorem 43 we obtain consequences for monotone real circuits analogous to those obtained in [57] for monotone boolean circuits. An immediate consequences of Theorem 43 and Lemma 31 is that any treelike monotone real circuit computing the boolean function GEN_n must have exponential size.

Theorem 46 *$\mathcal{S}^*_{\mathbb{MR}}(\text{GEN}_n) = 2^{\Omega(n^\epsilon)}$ for some $\epsilon > 0$.*

Consider now the following definition

Definition 47 *Let $\vec{t}$ be an input to GEN_n. We say that n is generated in a* DEPTH-d PYRAMIDAL FASHION *by $\vec{t}$ if there is a mapping $m : P_d \to [n]$ such that the following hold (recall that $a, b \vdash c$ means $t_{a,b,c} = 1$):*

$$
\begin{array}{ll}
1, 1 \vdash m(d, j) & \text{for every } j \leq d \\
m(i+1, j), m(i+1, j+1) \vdash m(i, j) & \text{for every } (i, j) \in P_{d-1} \\
m(1, 1), m(1, 1) \vdash n &
\end{array}
$$

We need a function related to GEN_n in order to produce a unsatisfiable CNF formula to get the size separations.

Definition 48 *Call PYR_n the boolean function that outputs 1 on every input to GEN_n for which n is generated in a depth-d pyramidal fashion, and outputs 0 on all inputs where GEN_n is 0.*

We can obtain an analogous of Theorem 46 also for the simpler case in which the generation is restricted to be only in a pyramidal form.

Corollary 49 $\mathcal{S}^*_{\mathbb{MR}}(\text{PYR}_n) = 2^{\Omega(n^\epsilon)}$ *for some* $\epsilon > 0$.

Proof. Observe that in Lemma 45 Player I from its input, builds an input for GEN_n which forces a depth-d pyramidal generation. So Lemma 36 can be easily adapted to PYR_n to prove that $CC_{\mathbb{R}}(\text{PYR}(m,d)) \leq CC_{\mathbb{R}}(R_{\text{PYR}_n})$. Lemma 44 and Theorem 36 imply that $CC_{\mathbb{R}}(R_{\text{PYR}_n}) \geq \Omega(n^\epsilon)$, for some $\epsilon > 0$. Finally Lemma 31 gives the Theorem. Q.E.D.

The other consequences drawn from Theorem 36 and Lemma 44 in [57] apply to monotone real circuits as well. We just state without proof the following result:

Theorem 50 *There are constants* $0 < \epsilon, \gamma < 1$ *such that for every function* $d(n) \leq n^\epsilon$, *there is a family of monotone functions* $f_n : \{0,1\}^n \to \{0,1\}$ *that can be computed by monotone boolean circuits of size* $n^{O(1)}$ *and depth* $d(n)$, *but cannot be computed by monotone real circuits of depth less than* $\gamma \cdot d(n)$.

The method also gives a simpler proof of the lower bounds in [41], in the same way as [57] simplifies the lower bound of [43].

2.1.4 Separation between CP* and R(1)

As observed in Subsection 1.4.2, Theorem 24 allows to reduce the task of proving lower bounds for CP* to that of giving lower bounds for the size of treelike monotone real circuits. In this Section we build an unsatisfiable CNF formula $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{COL}(\vec{p},\vec{r})$ and we will obtain exponential lower bounds for CP* refutations using Corollary 49. That is, we build $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{COL}(\vec{p},\vec{r})$ in such a way that the interpolant provided by Theorem 24 is a monotone real treelike circuit computing the function GEN_n, where n is generated in a pyramidal form. After that we also show that $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{COL}(\vec{p},\vec{r})$ admits polynomially size R(1) refutations.

Let n and d be natural numbers whose values will be fixed below. Recall that $P_d := \{(i,j)\,;\, 1 \leq j \leq i \leq d\}$. The clauses in $\mathbb{GEN}(\vec{p},\vec{q})$ will encode the property that the inputs

$\vec{p}$ define a pyramidal generation, and therefore $\text{GEN}_n(\vec{p}) = 1$. The clauses in $\mathbb{COL}(\vec{p}, \vec{r})$ will say that that the inputs $\vec{p}$ define a coloring, so that $\text{GEN}_n(\vec{p}) = 0$ follows.

More precisely: the variables $p_{a,b,c}$ for $a, b, c \in [n]$ represent the input to GEN_n. The variables $q_{i,j,a}$ for $(i,j) \in P_d$ and $a \in [n]$ will encode a pyramidal structure for some mapping m defining a pyramidal generation, see Definition 47. The meaning of $q_{i,j,a}$ is that the mapping m is assigning the element $a \in [n]$ to the position (i,j) of the pyramid. The variables r_a for $a \in [n]$ represent a coloring of the elements in $[n]$ by $0, 1$ such that: 1 is colored 0, n is colored 1 and the elements colored 0 are closed under generation, that is, if in a triangle of the pyramid the two base elements are colored 0, then also the top must be colored 0. The set $\mathbb{GEN}(\vec{p}, \vec{q})$ is given by (2.1) - (2.4), and $\mathbb{COL}(\vec{p}, \vec{r})$ by (2.5) - (2.7).

$$\bigvee_{1 \le a \le n} q_{i,j,a} \qquad \text{for } (i,j) \in P_d \tag{2.1}$$

$$\bar{q}_{d,j,a} \vee p_{1,1,a} \qquad \text{for } 1 \le j \le d \text{ and } a \in [n] \tag{2.2}$$

$$\bar{q}_{1,1,a} \vee p_{a,a,n} \qquad \text{for } a \in [n] \tag{2.3}$$

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee p_{a,b,c} \qquad \text{for } (i,j) \in Pyr_{d-1} \text{ and } a, b, c \in [n] \tag{2.4}$$

$$\bar{p}_{1,1,a} \vee \bar{r}_a \qquad \text{for } a \in [n] \tag{2.5}$$

$$\bar{p}_{a,a,n} \vee r_a \qquad \text{for } a \in [n] \tag{2.6}$$

$$r_a \vee r_b \vee \bar{p}_{a,b,c} \vee \bar{r}_c \qquad \text{for } a, b, c \in [n] \tag{2.7}$$

If $\mathbb{GEN}(\vec{t}, \vec{q})$ is satisfiable for a fixed vector $\vec{t} \in \{0,1\}^{n^3}$, then n is generated in a depth-d pyramidal fashion, and if $\mathbb{COL}(\vec{t}, \vec{r})$ is satisfiable, then $\text{GEN}(\vec{t}) = 0$. Observe that the variables $\vec{p}$ occur only positively in $\mathbb{GEN}(\vec{p}, \vec{q})$ and only negatively in $\mathbb{COL}(\vec{p}, \vec{r})$. Hence from Theorem 24 and Corollary 49 we can prove size lower bounds for CP* refutations of $\mathbb{GEN}(\vec{p}, \vec{q}) \cup \mathbb{COL}(\vec{p}, \vec{r})$:

Theorem 51 $\mathcal{L}^*_{CP}(\mathbb{GEN}(\vec{p}, \vec{q}) \cup \mathbb{COL}(\vec{p}, \vec{r})) = 2^{\Omega(n^\epsilon)}$

On the other hand, there are polynomial size R(1) refutations of these clauses.

Theorem 52 $\mathcal{L}_1(\mathbb{GEN}(\vec{p}, \vec{q}) \cup \mathbb{COL}(\vec{p}, \vec{r})) = n^{O(1)}$

Proof. First we resolve clauses (2.2) and (2.5) to get

$$\bar{q}_{d,j,c} \vee \bar{r}_c \tag{2.8}$$

for $1 \leq j \leq d$ and $1 \leq c \leq n$.

Now we want to derive $\bar{q}_{i,j,c} \vee \bar{r}_c$ for every $(i,j) \in P_d$ and $1 \leq c \leq n$, by induction on i downward from d to 1. The induction base is just (2.8).

Now by induction we have

$$\bar{q}_{i+1,j,a} \vee \bar{r}_a \quad \text{and} \quad \bar{q}_{i+1,j+1,b} \vee \bar{r}_b \ ,$$

we resolve them against (2.7) to get $\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{p}_{a,b,c} \vee \bar{r}_c$ for $1 \leq a,b,c \leq n$ and then resolve them against (2.4) and get

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee \bar{r}_c$$

for every $1 \leq a,b \leq n$. All of these are then resolved against two instances of (2.1), and we get the desired $\bar{q}_{i,j,c} \vee \bar{r}_c$ for every $1 \leq c \leq n$.

Finally, we have in particular $\bar{q}_{1,1,a} \vee \bar{r}_a$ for every $1 \leq c \leq n$. We resolve them with (2.6) and get $\bar{q}_{1,1,a} \vee \bar{p}_{a,a,n}$ for every $1 \leq a \leq n$. These are resolved with (2.3) to get $\bar{q}_{1,1,a}$ for every $1 \leq a \leq n$. Finally, this clause is resolved with another instance of (2.3) (the one with $i = j = 1$) to get the empty clause. Q.E.D.

It is easy to check that the above refutation is an negative R(1) refutation. The following corollary is an easy consequence of the above theorems and known simulation results.

Corollary 53 *The clauses* $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{COL}(\vec{p},\vec{r})$ *exponentially separate* R*(1) *from* R(1) *and negative* R(1) *as well as* CP* *from* CP.

The R(1) refutation of $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{COL}(\vec{p},\vec{r})$ that appears in the proof of Theorem 52 is not regular. We do not know whether $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{COL}(\vec{p},\vec{r})$ has polynomial size regular R(1) refutations. To obtain a separation between R*(1) and regular R(1) we will modify the clauses $\mathbb{COL}(\vec{p},\vec{r})$.

2.1.5 Separation of CP* from regular R(1)

The clauses $\mathbb{COL}(\vec{p},\vec{r})$ are modified into clauses $\mathbb{RCOL}(\vec{p},\vec{r})$, so that $\mathbb{GEN}(\vec{p},\vec{q})\cup\mathbb{RCOL}(\vec{p},\vec{r})$ allow small regular R(1) refutations, but in such a way that the lower bound proof still applies. We replace the variables r_a by $r_{a,i,D}$ for $a \in [n]$, $1 \leq i \leq d$ and $D \in \{L,R\}$, giving the coloring of element a, with auxiliary indices i being a row in the pyramid and D distinguishing whether an element is used as a left or right predecessor in the generation process.

The set $\mathbb{RCOL}(\vec{p},\vec{r})$ is defined as follows:

$$\bar{p}_{1,1,a} \vee \bar{r}_{a,d,D} \qquad \text{for } a \in [n] \text{ and } D \in \{L,R\} \tag{2.9}$$

$$\bar{p}_{a,a,n} \vee r_{a,1,D} \qquad \text{for } a \in [n] \text{ and } D \in \{L,R\} \tag{2.10}$$

$$r_{a,i+1,L} \vee r_{b,i+1,R} \vee \bar{p}_{a,b,c} \vee \bar{r}_{c,i,D} \qquad \text{for } i < d,\ a,b,c \in [n] \text{ and } D \in \{L,R\} \tag{2.11}$$

$$\bar{r}_{a,i,D} \vee r_{a,i,\bar{D}} \qquad \text{for } 1 \leq i \leq d \text{ and } D \in \{L,R\} \tag{2.12}$$

$$\bar{r}_{a,i,D} \vee r_{a,j,D} \qquad \text{for } 1 \leq i,j \leq d \text{ and } D \in \{L,R\} \tag{2.13}$$

Due to the clauses (2.12) and (2.13), the variables $r_{a,i,D}$ are equivalent for all values of the auxiliary indices i, D. Hence a satisfying assignment for $\mathbb{RCOL}(\vec{p},\vec{r})$ still codes a coloring of $[n]$ such that elements a with $1,1 \vdash a$ are colored 0, the elements b with $b,b \vdash n$ are colored 1, and the 0-colored elements are closed under generation. Hence if $\mathbb{RCOL}(\vec{t},\vec{r})$ is satisfiable, then $\text{GEN}(\vec{t}) = 0$.

Hence any interpolant for the clauses $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{RCOL}(\vec{p},\vec{r})$ satisfies the assumptions of Corollary 49, and we can conclude

Theorem 54 $\mathcal{L}^*_{CP}(\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{RCOL}(\vec{p},\vec{r})) = 2^{\Omega(n^\epsilon)}$

On the other hand, we have the following upper bound on regular R(1) refutations of these clauses:

Theorem 55 *There are regular* R(1) *refutations of the clauses* $\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{RCOL}(\vec{p},\vec{r})$ *of size* $n^{O(1)}$.

Proof. First we resolve clauses (2.2) and (2.9) to get

$$\bar{q}_{d,j,a} \vee \bar{r}_{a,d,D} \tag{2.14}$$

for $1 \leq j \leq d$, $1 \leq a \leq n$ and $D \in \{L, R\}$. Next we resolve (2.3) and (2.10) to get

$$\bar{q}_{1,1,a} \vee r_{a,1,D} \tag{2.15}$$

for $1 \leq a \leq n$ and $D \in \{L, R\}$. Finally, from (2.4) and (2.11) we obtain

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee r_{a,i+1,L} \vee r_{b,i+1,R} \vee \bar{r}_{c,i,D} \tag{2.16}$$

for $1 \leq j \leq i < d$, $1 \leq a, b, c \leq n$ and $D \in \{L, R\}$.

Now we want to derive $\bar{q}_{i,j,a} \vee \bar{r}_{a,i,D}$ for every $(i, j) \in P_d$, $1 \leq a \leq n$ and $D \in \{L, R\}$, by induction on i downward from d to 1. The induction base is just (2.14).

For the inductive step, resolve (2.16) against the clauses

$$\bar{q}_{i+1,j,a} \vee \bar{r}_{a,i+1,L} \quad \text{and} \quad \bar{q}_{i+1,j+1,b} \vee \bar{r}_{b,i+1,R} \ ,$$

which we have by induction, to give

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee \bar{r}_{c,i,D}$$

for every $1 \leq a, b \leq n$. All of these are then resolved against two instances of (2.1), and we get the desired $\bar{q}_{i,j,c} \vee \bar{r}_{c,i,D}$.

Finally, we have in particular $\bar{q}_{1,1,a} \vee \bar{r}_{a,1,L}$, which we resolve against (2.15) to get $\bar{q}_{1,1,a}$ for every $a \leq n$. From these and an instance of(2.1) we get λ. Q.E.D.

Note that the refutation given in the proof of Theorem 55 is actually a ordered refutation: It respects the following elimination order

$$\begin{array}{l}
p_{1,1,1} \ \cdots \ p_{n,n,n} \\
r_{1,d,L} \ \ r_{1,d,R} \ \ \cdots \ \ r_{n,d,L} \ \ r_{n,d,R} \\
q_{1,d,1} \ \cdots \ q_{1,d,n} \ \cdots \ q_{d,d,1} \ \cdots \ q_{d,d,n} \\
r_{1,d-1,L} \ \cdots \ r_{n,d-1,R} \ \ q_{1,d-1,1} \ \cdots \ q_{d-1,d-1,n} \\
\vdots \\
r_{1,1,L} \ \ r_{1,1,R} \ \ q_{1,1,1} \ \cdots \ q_{1,1,n} \ .
\end{array}$$

Corollary 56 *The clauses* $\mathbb{GEN}(\vec{p}, \vec{q}) \cup \mathbb{RCOL}(\vec{p}, \vec{r})$ *exponentially separate the following proof systems:* $\mathrm{R}^*(1)$ *from regular* $\mathrm{R}(1)$ *and ordered* $\mathrm{R}(1)$.

The separation between R*(1) and R(1) was later improved in [16]. Recall the definition of the Pebbling Contradictions from Subsection 1.3.4. They show that for a certain graph G, $\mathcal{L}_1(\mathbb{PEB}_1^2(G)) = O(n)$, but $\mathcal{L}_1^*(\mathbb{PEB}_1^2(G)) = \exp(\Omega(n/\log n))$. They also prove that this separation is nearly optimal, because if for a formula $\mathbb{F}$, $\mathcal{L}_1(\mathbb{F}) = s$ then $\mathcal{L}_1^*(\mathbb{F}) = \exp(O(s \log\log s/\log s))$.

2.2 R(2) has not the monotone interpolation property

In this section we prove that R(1) cannot polynomially simulate R(2). More precisely, we prove that a certain Clique-Coclique principle, as defined by Bonet, Pitassi and Raz in [20], has polynomial size R(2) refutations, but every R(1) refutation requires quasipolynomial size. The Clique-Coclique principle that we use, $\mathbb{CLIQUE}_{k,k'}^n$, is the conjunction of the following set of clauses:

$$x_{i,1} \vee \cdots \vee x_{i,n} \qquad 1 \leq l \leq k \tag{2.17}$$

$$\bar{x}_{l,i} \vee \bar{x}_{l,j} \qquad 1 \leq l \leq k,\ 1 \leq i,j \leq n,\ i \neq j \tag{2.18}$$

$$\bar{x}_{l,i} \vee \bar{x}_{l',i} \qquad 1 \leq l,l' \leq k,\ 1 \leq i \leq n,\ l \neq l' \tag{2.19}$$

$$y_{1,i} \vee \cdots \vee y_{k',i} \qquad 1 \leq i \leq n \tag{2.20}$$

$$\bar{y}_{l,i} \vee \bar{y}_{l',i} \qquad 1 \leq l,l' \leq k',\ 1 \leq i \leq n,\ l \neq l' \tag{2.21}$$

$$\bar{x}_{l,i} \vee \bar{x}_{l',j} \vee \bar{y}_{t,i} \vee \bar{y}_{t,j} \qquad 1 \leq l,l' \leq k,\ 1 \leq t \leq k',\ 1 \leq i,j \leq n,\ l \neq l',\ i \neq j \tag{2.22}$$

We start with a reduction from $\mathbb{CLIQUE}_{k,k'}^n$ to $\mathbb{PHP}_{k'}^k$ that can be carried over in R(2):

Theorem 57 *Let $k' < k \leq n$. If $\mathbb{PHP}_{k'}^k$ has* R(1) *refutations of size S, then $\mathbb{CLIQUE}_{k,k'}^n$ has* R(2) *refutations of size Sn^c for some constant $c > 0$.*

Proof. We use the following R(2) reduction to transform the formula $\mathbb{CLIQUE}_{k,k'}^n$ into $\mathbb{PHP}_{k'}^k$. The meaning of variable $p_{i,j}$ is that pigeon i sits in hole j. We perform the following substitutions:

$$p_{i,j} \equiv \bigvee_{l=1}^{n} (x_{i,l} \wedge y_{i,l}) \qquad\qquad \bar{p}_{i,j} \equiv \bigvee_{l=1, j' \neq j}^{n} (x_{i,l} \wedge y_{j',l})$$

First we show how to get clauses (1.1) from clauses (2.17) and (2.20). If we expand clause (1.1) for a certain i we have:

$$\begin{array}{c}(x_{i,1} \wedge y_{1,1}) \vee (x_{i,2} \wedge y_{1,2}) \vee (x_{i,3} \wedge y_{1,3}) \vee \cdots \vee (x_{i,n} \wedge y_{1,n}) \vee \\ (x_{i,1} \wedge y_{2,1}) \vee (x_{i,2} \wedge y_{2,2}) \vee (x_{i,3} \wedge y_{2,3}) \vee \cdots \vee (x_{i,n} \wedge y_{2,n}) \vee \\ (x_{i,1} \wedge y_{3,1}) \vee (x_{i,2} \wedge y_{3,2}) \vee (x_{i,3} \wedge y_{3,3}) \vee \cdots \vee (x_{i,n} \wedge y_{3,n}) \vee \\ \cdots \\ (x_{i,1} \wedge y_{k',1}) \vee (x_{i,2} \wedge y_{k',2}) \vee (x_{i,3} \wedge y_{k',3}) \vee \cdots \vee (x_{i,n} \wedge y_{k',n})\end{array} \tag{2.23}$$

We apply successively for $1 \leq j \leq k'$ the $\wedge$-introduction rule to clauses $y_{1,1} \vee \cdots \vee y_{k',1}$ and $x_{i,1} \vee \cdots \vee x_{i,n}$ along variables $x_{i,1}$ and $y_{j,1}$ and get:

$$(x_{i,1} \wedge y_{1,1}) \vee (x_{i,1} \wedge y_{2,1}) \vee \cdots \vee (x_{i,1} \wedge y_{k',1}) \vee x_{i,2} \vee \cdots \vee x_{i,n} \tag{2.24}$$

Observe that the conjunctions in (2.24) form the first column in (2.23). To add the second column of (2.23) to (2.24) we apply successively for $1 \leq j \leq k'$ the $\wedge$-rule to clauses $y_{1,2} \vee \cdots \vee y_{k',2}$ and (2.24) along variables $x_{i,2}$ and $y_{j,2}$ and get:

$$\begin{array}{c}(x_{i,1} \wedge y_{1,1}) \vee (x_{i,1} \wedge y_{2,1}) \vee \cdots \vee (x_{i,1} \wedge y_{k',1}) \vee \\ (x_{i,2} \wedge y_{1,2}) \vee (x_{i,2} \wedge y_{2,2}) \vee \cdots \vee (x_{i,2} \wedge y_{k',2}) \vee x_{i,3} \vee \cdots \vee x_{i,n}\end{array} \tag{2.25}$$

Now it is clear how to get (2.23).

Now we will show how to get the initial clauses (1.2). Let us consider the clause $\bar{p}_{i,t} \vee \bar{p}_{j,t}$. We first generate $p_{i,1} \vee \cdots \vee p_{i,k'}$ and $p_{j,1} \vee \cdots \vee p_{j,k'}$. Let us rewrite them as:

$$(x_{i,1} \wedge y_{t,1}) \vee (x_{i,2} \wedge y_{t,2}) \vee (x_{i,3} \wedge y_{t,3}) \vee \cdots \vee (x_{i,n} \wedge y_{t,n}) \vee A \tag{2.26}$$

$$(x_{j,1} \wedge y_{t,1}) \vee (x_{j,2} \wedge y_{t,2}) \vee (x_{j,3} \wedge y_{t,3}) \vee \cdots \vee (x_{j,n} \wedge y_{t,n}) \vee B \tag{2.27}$$

where A is $p_{i,1} \vee \cdots \vee p_{i,t-1} \vee p_{i,t+1} \vee \cdots \vee p_{i,k'}$ and B is $p_{j,1} \vee \cdots \vee p_{j,t-1} \vee p_{j,t+1} \vee \cdots \vee p_{j,k'}$. For the sake of brevity we use $p_{i,j}$ as abbreviation of the 2-disjunction it denotes. It is clear that $\bar{p}_{i,t} \vee \bar{p}_{j,t}$ is $A \vee B$, that is:

$$p_{i,1} \vee \cdots \vee p_{i,t-1} \vee p_{i,t+1} \vee \cdots \vee p_{i,k'} \vee p_{j,1} \vee \cdots \vee p_{j,t-1} \vee p_{j,t+1} \vee \cdots \vee p_{j,k'}$$

Now we will get $A \vee B$ from (2.26), (2.27) and (2.22). We apply the cut rule to (2.27) and $\bar{x}_{i,1} \vee \bar{x}_{j,l} \vee \bar{y}_{t,1} \vee \bar{y}_{t,l}$ for $1 \leq l \leq n$, $l \neq 1$, and get:

$$\bar{x}_{i,1} \vee \bar{y}_{t,1} \vee (x_{j,1} \wedge y_{t,1}) \vee B \tag{2.28}$$

Solving it with $\bar{x}_{i,1} \vee \bar{x}_{j,1}$ we get $\bar{x}_{i,1} \vee \bar{y}_{t,1} \vee B$. Solving this clause with (2.26) we get

$$(x_{i,2} \wedge y_{t,2}) \vee \cdots \vee (x_{i,n} \wedge y_{t,n}) \vee A \vee B \tag{2.29}$$

Now we can get rid successively of $(x_{i,2} \wedge y_{t,2}), \ldots, (x_{i,n} \wedge y_{t,n})$ as we did with $(x_{i,1} \wedge y_{t,1})$.

It remains to show how to simulate a normal R(1) step. We have $p_{i,j} \vee A$ and $\bar{p}_{i,j} \vee B$ and we want to get $A \vee B$. We expand both clauses:

$$(x_{i,1} \wedge y_{j,1}) \vee (x_{i,2} \wedge y_{j,2}) \vee (x_{i,3} \wedge y_{j,3}) \vee \cdots \vee (x_{i,n} \wedge y_{j,n}) \vee A \tag{2.30}$$

$$\begin{gathered}
(x_{i,1} \wedge y_{1,1}) \vee (x_{i,2} \wedge y_{1,2}) \vee (x_{i,3} \wedge y_{1,3}) \vee \cdots \vee (x_{i,n} \wedge y_{1,n}) \vee \\
(x_{i,1} \wedge y_{2,1}) \vee (x_{i,2} \wedge y_{2,2}) \vee (x_{i,3} \wedge y_{2,3}) \vee \cdots \vee (x_{i,n} \wedge y_{2,n}) \vee \\
\cdots \\
(x_{i,1} \wedge y_{j-1,1}) \vee (x_{i,2} \wedge y_{j-1,2}) \vee (x_{i,3} \wedge y_{j-1,3}) \vee \cdots \vee (x_{i,n} \wedge y_{j-1,n}) \vee \\
(x_{i,1} \wedge y_{j+1,1}) \vee (x_{i,2} \wedge y_{j+1,2}) \vee (x_{i,3} \wedge y_{j+1,3}) \vee \cdots \vee (x_{i,n} \wedge y_{j+1,n}) \vee \\
\cdots \\
(x_{i,1} \wedge y_{k',1}) \vee (x_{i,2} \wedge y_{k',2}) \vee (x_{i,3} \wedge y_{k',3}) \vee \cdots \vee (x_{i,n} \wedge y_{k',n}) \vee B
\end{gathered} \tag{2.31}$$

If we get clauses $\bar{x}_{i,l} \vee \bar{y}_{j,l} \vee B$ for $1 \leq l \leq n$, we solve them all with (2.30) and get $A \vee B$ as desired. We will show how to get $\bar{x}_{i,1} \vee \bar{y}_{j,1} \vee B$. We solve (2.31) with $\bar{y}_{j,1} \vee \bar{y}_{l,1}$, $l \neq j$ of course. With these we get rid of the first column of (2.31) and we add a literal $\bar{y}_{j,1}$. We can get rid of the rest of columns by solving enough times with clauses $\bar{x}_{i,1} \vee \bar{x}_{i,l}$, $l \neq 1$, and we get $\bar{x}_{i,1} \vee \bar{y}_{j,1} \vee B$. Q.E.D.

We will use the feasible monotone interpolation property for R(1), see Subsection 1.4.2, together with the following result of Alon and Boppana [21] establishing a lower bound to the size of monotone boolean circuits that separate large cliques from small cocliques. In the following, $F(m, k, k')$ is the set of monotone functions that separate k-cliques from k'-cocliques on m nodes.

Theorem 58 [21] *If $f \in F(m,k,k')$ where $3 \leq k' \leq k$ and $k\sqrt{k'} \leq m/(8\log m)$, then*

$$\mathcal{S}_{\mathbb{MB}}(f) \geq \frac{1}{8}\left(\frac{m}{4k\sqrt{k'}\log m}\right)^{(\sqrt{k'}+1)/2},$$

Theorem 59 *Let $k = \sqrt{m}$ and $k' = (\log m)^2/8\log\log m$. Then,*

1. *$\mathbb{CLIQUE}^m_{k,k'}$ has R(2) refutations of size polynomial in m, and*
2. *every R(1) refutation of $\mathbb{CLIQUE}^m_{k,k'}$ has size at least $\exp(\Omega((\log m)^2/\sqrt{\log\log m}))$.*

Proof. Regarding 1, we have that $k'\log k' \leq \frac{1}{4}(\log m)^2$, and so $2^{\sqrt{k'\log k'}} \leq m^{1/2} = k$. On the other hand, Buss and Pitassi [24] proved that $\mathbb{PHP}^k_{k'}$ has R(1) refutations of size polynomial in k whenever $k \geq 2^{\sqrt{k'\log k'}}$. Therefore, by Theorem 57, $\mathbb{CLIQUE}^m_{k,k'}$ has R(2) refutations of size polynomial in m.

Regarding 2, we apply the feasible monotone interpolation theorem for R(1). We have

$$\frac{\log m}{3\sqrt{\log\log m}} \leq \sqrt{k'} \leq \log m.$$

Therefore, by Theorem 58, if $f \in F(m,k,k')$ is a monotone interpolant, then

$$\mathcal{S}_{\mathbb{MB}}(f) \geq \frac{1}{8}\left(\frac{m}{4\sqrt{m}(\log m)^2}\right)^{\frac{\log m}{6\sqrt{\log\log m}}} \geq \frac{1}{8}\left(\frac{m}{m^{3/4}}\right)^{\frac{\log m}{6\sqrt{\log\log m}}},$$

which is $\exp(\Omega((\log m)^2/\sqrt{\log\log m}))$. Q.E.D.

As a corollary, we solve an open problem posed by Krajíček [48].

Corollary 60 R(2) *does not have the feasible monotone interpolation property.*

2.3 R(2) and $\mathbb{PHP}^{n^2}_n$

At the time of writing [6] the question about the size of R(1) refutations of $\mathbb{PHP}^{n^c}_n$, where $c \geq 2$, was still not settled. Before [58, 56] appeared, R(2) could have been used to settle this question, al least for $\mathbb{PHP}^{n^2}_n$.

In [13] it was defined the monotone R(1) proof systems and shown that it was equivalent to R(1) for $\mathbb{PHP}^m_n$.

A monotone clause contains only positive literals. Let R, S, T be subsets of $\{1, \ldots, m\}$. Let $P_{R,j}$ (resp. S, T) the disjunction of the variables $p_{i,j}$, where $i \in R$ (resp. S, T). Let $C_1 = A \vee P_{R,j} \vee P_{S,j}$ and $C_2 = B \vee P_{R,j} \vee P_{T,j}$, where R, S, T are disjoint sets. The monotone inference rule with respect to hole j allows us to derive $C_3 = A \vee B \vee P_{R,j}$. A monotone R(1) refutation of $\mathbb{PHP}_n^m$ is a sequence of monotone clauses, where the final clause is λ and where every clause is either an initial clause of the form $p_{i,1} \vee \cdots \vee p_{i,n}$ where $i \in \{1, \ldots, m\}$ or follows from two previous clauses by the monotone R(1) rule.

Now we will show how to get a R(2) refutation of $\mathbb{PHP}_n^{n^{\frac{3}{2}}}$ from a monotone R(1) refutation of $\mathbb{PHP}_n^{n^2}$ of similar size. The idea comes from [49].

Lemma 61 *Let $\mathcal{R}$ a monotone* R(1) *refutation of $\mathbb{PHP}_n^{n^2}$, then there are a* R(2) *refutation $\mathcal{R}'$ of $\mathbb{PHP}_n^{n^{\frac{3}{2}}}$ of similar size.*

Proof. From the clauses in $\mathbb{PHP}_n^{n^{\frac{3}{2}}}$ we will show how to get the pigeon clauses in $\mathbb{PHP}_n^{n^2}$. These are the only clauses needed for monotone R(1). Then we will show how to perform the monotone R(1) rule. The pigeon clauses in $\mathbb{PHP}_n^{n^2}$ are:

$$P_{i,1} \vee \cdots \vee P_{i,n} \qquad\qquad 1 \leq i \leq n^2 \tag{2.32}$$

where $P_{i,j}$ is not a variable, but a conjunction, that we define next. Let us divide the set $\{1, \ldots, n^2\}$ into $n^{\frac{1}{2}}$ sets of $n^{\frac{3}{2}}$ elements. Let the set $A_i = \{n^{\frac{3}{2}}(i-1)+1, \ldots, n^{\frac{3}{2}}i\}$. Let us divide the set $\{1, \ldots, n^{\frac{3}{2}}\}$ into $n^{\frac{1}{2}}$ sets of n elements. Let the set $B_i = \{n(i-1)+1, \ldots, ni\}$. Let $i \in A_k$, let $A(i)$ the rank of i in the set A_k. Let $B(l)$ the l-th element in B_k, then:

$$P_{i,j} \equiv \bigvee_{l=1}^{n} (p_{A(i),l} \wedge p_{B(l),j}).$$

For example, $P_{n^2,1}$ is $(p_{n^{\frac{3}{2}},1} \wedge p_{n^{\frac{3}{2}}-n+1,1}) \vee (p_{n^{\frac{3}{2}},2} \wedge p_{n^{\frac{3}{2}}-n+2,1}) \vee \cdots \vee (p_{n^{\frac{3}{2}},n} \wedge p_{n^{\frac{3}{2}},1})$.

We will show a R(2) derivation of (2.32) for i from $\mathbb{PHP}_n^{n^{\frac{3}{2}}}$. The expansion of the clause looks like this:

$$\begin{aligned}
&(p_{k_1,1} \wedge p_{k_2,1}) \vee (p_{k_1,2} \wedge p_{k_2+1,1}) \vee \cdots \vee (p_{k_1,n} \wedge p_{k_2+n-1,1}) \vee \\
&(p_{k_1,1} \wedge p_{k_2,2}) \vee (p_{k_1,2} \wedge p_{k_2+1,2}) \vee \cdots \vee (p_{k_1,n} \wedge p_{k_2+n-1,2}) \vee \\
&(p_{k_1,1} \wedge p_{k_2,3}) \vee (p_{k_1,2} \wedge p_{k_2+1,3}) \vee \cdots \vee (p_{k_1,n} \wedge p_{k_2+n-1,3}) \vee \\
&\qquad\qquad \cdots \\
&(p_{k_1,1} \wedge p_{k_2,n}) \vee (p_{k_1,2} \wedge p_{k_2+1,n}) \vee \cdots \vee (p_{k_1,n} \wedge p_{k_2+n-1,n})
\end{aligned} \tag{2.33}$$

Conjuncting successively $p_{k_1,1} \vee \cdots \vee p_{k_1,n}$ and $p_{k_2,1} \vee \cdots \vee p_{k_2,n}$ over variables $p_{k_1,1}$ and $p_{k_2,i}$ for $1 \leq i \leq n$ we get

$$(p_{k_1,1} \wedge p_{k_2,1}) \vee (p_{k_1,1} \wedge p_{k_2,2}) \vee (p_{k_1,1} \wedge p_{k_2,3}) \vee \cdots \vee (p_{k_1,1} \wedge p_{k_2,n}) \vee \tag{2.34}$$

$$\vee p_{k_1,2} \vee \cdots \vee p_{k_1,n} \tag{2.35}$$

This is the first column of the expansion plus the literals that will allow us to get the rest of the columns. To get the second column we conjunct successively (2.35) and $p_{k_2+1,1} \vee \cdots \vee p_{k_2+1,n}$ over variables $p_{k_1,2}$ and $p_{k_2+1,i}$ for $1 \leq i \leq n$, and so on.

Now we will show how to get rid of two R(2) clauses such as $P_{i,j}$ and $P_{k,j}$ with the invaluable help of the hole clauses for the hole j in $\mathbb{PHP}_n^{n^{\frac{3}{2}}}$. This will be used extensively in simulating the monotone R(1) rule.

Let $P_{i,j}$ be the R(2) clause $(p_{k_1,1} \wedge p_{k_2,j}) \vee (p_{k_1,2} \wedge p_{k_2+1,j}) \vee \cdots \vee (p_{k_1,n} \wedge p_{k_2+n-1,j})$ and $P_{k,j}$ be $(p_{k_3,1} \wedge p_{k_4,j}) \vee (p_{k_3,2} \wedge p_{k_4+1,j}) \vee \cdots \vee (p_{k_3,n} \wedge p_{k_4+n-1,j})$. If $k_2 \neq k_4$ then we can solve $(p_{k_3,1} \wedge p_{k_4,j}) \vee (p_{k_3,2} \wedge p_{k_4+1,j}) \vee \cdots \vee (p_{k_3,n} \wedge p_{k_4+n-1,j})$ with $\bar{p}_{k_4+i,j} \vee \bar{p}_{k_2,j}$ for $0 \leq i < n$ and get $\bar{p}_{k_2,j}$. Solving it with $(p_{k_1,1} \wedge p_{k_2,j}) \vee (p_{k_1,2} \wedge p_{k_2+1,j}) \vee \cdots \vee (p_{k_1,n} \wedge p_{k_2+n-1,j})$ we get rid of the first conjunctant. To eliminate the second conjunctant, that is, $(p_{k_1,2} \wedge p_{k_2+1,j})$ we solve $(p_{k_3,1} \wedge p_{k_4,j}) \vee (p_{k_3,2} \wedge p_{k_4+1,j}) \vee \cdots \vee (p_{k_3,n} \wedge p_{k_4+n-1,j})$ with $\bar{p}_{k_4+i,j} \vee \bar{p}_{k_2+1,j}$ to get $\bar{p}_{k_2+1,j}$. When $k_2 = k_4$ we cannot do that. Let us show how to overcome the problem. To eliminate the first conjunction of $P_{i,j}$, namely $(p_{k_1,1} \wedge p_{k_2,j})$ we solve $(p_{k_3,1} \wedge p_{k_4,j}) \vee (p_{k_3,2} \wedge p_{k_4+1,j}) \vee \cdots \vee (p_{k_3,n} \wedge p_{k_4+n-1,j})$ with $\bar{p}_{k_4+i,j} \vee \bar{p}_{k_2,j}$ for $1 \leq i < n$, and get $(p_{k_3,1} \wedge p_{k_4,1}) \vee \bar{p}_{k_2,j}$. As in fact $k_2 = k_4$ we can rewrite it as $(p_{k_3,1} \wedge p_{k_2,1}) \vee \bar{p}_{k_2,j}$. But now we can solve it with $\bar{p}_{k_1,1} \vee \bar{p}_{k_3,1}$ getting $\bar{p}_{k_1,1} \vee \bar{p}_{k_2,j}$. With this we can eliminate the first conjunction from $(p_{k_1,1} \wedge p_{k_2,j})$. The rest of conjunction can be eliminated in a similar way.

The last part is to show how to simulate a monotone R(1) rule over a hole, say j. We have clauses $C_1 = A \vee P_{R,j} \vee P_{S,j}$ and $C_2 = B \vee P_{R,j} \vee P_{T,j}$. Let us suppose that $T = \{1\}$, so $P_{T,j} = P_{1,j}$. Do not get confused with the notation. $P_{T,j}$, with T a set, is a disjunction of pigeons that share the same hole. $P_{i,j}$ with i a number is pigeon in $[n^2]$, this pigeon is in fact a R(2) clause and it is formed from literals in $\mathbb{PHP}_n^{n^{\frac{3}{2}}}$, which are denoted with a p instead of a P. In this case we can solve $P_{1,j}$ with all the pigeons in $P_{S,j}$ and get as desired $C_3 = A \vee B \vee P_{R,j}$. In general $|T| < n^2$, because now we are working with $\mathbb{PHP}_n^{n^2}$. For

simplicity $C_1 = A \vee P_{S,j}$ and $C_2 = B \vee P_{T,j}$, we omit the common holes. Without loss of generality let $T = \{1, \ldots i\}$. We make i copies of C_1 and solve the k-th with $P_{k,j}$. Now we will call $P_{k,j}$ just k because we know we are talking about a pigeon and we know the hole is j. We get i clauses like these.

$$\begin{aligned} &A \vee B \vee \{2, 3, 4, \ldots, i-1, i\} \\ &A \vee B \vee \{1, 3, 4, \ldots, i-1, i\} \\ &A \vee B \vee \{1, 2, 4, \ldots, i-2, i\} \\ &\ldots \\ &A \vee B \vee \{1, 2, 3, \ldots, i-2, i-1\} \end{aligned} \tag{2.36}$$

The j-th and the $j+1$-th clauses only differ in the j element. We solve this element for all the pairs of consecutive clauses and get $i-1$ clauses. Again the j-th and the $j+1$-th clauses only differ in the j element. We solve again pairs of consecutive clauses. At the end of this process we get clauses $A \vee B \vee P_{i,j}$ and $A \vee B \vee P_{1,j}$. From these we get $A \vee B$ as desired. Q.E.D.

2.4 Size and R*(k)

In this Section we prove results concerning R*(k) size. The first one is an exponential separation between successive levels of R*(k). We separate exponentially R*(k) from R*(k+1). We first give in Subsection 2.4.1 polynomial size R*(k) refutations for $\mathbb{PEB}_k^l(G)$, to do so we prove a general proposition for Horn formulas. In Subsection 2.4.2 we give the R*(k) lowers bounds needed to establish the separation. Last, in Subsection 2.4.3 we show that R(1) dominates R*(k).

2.4.1 Upper bounds for Generalized Pebbling Contradictions

To give the upper bound we will transform $\mathbb{PEB}_k^l(G)$ into a Horn formula using the R(k) $\wedge$-introduction rule and then apply the following proposition:

Proposition 62 *Let $\mathbb{F}$ be an unsatisfiable Horn formula. Then there is a linear* R*(1) *refutation of $\mathbb{F}$ that uses any input clause from $\mathbb{F}$ at most once.*

Proof. We will show how to construct the treelike refutation for $\mathbb{F}$. It is well known [44] that the following method can be used to decide the unsatisfiability of a Horn formula $\mathbb{F}$:

> Let $M_0 = \emptyset$. The set M_{d+1} is obtained from M_d by adding some atom $x \notin M_d$ from $\mathbb{F}$ such that there is a clause $A_d = \neg y_1 \vee \cdots \vee \neg y_l \vee x$ in F with $x_1, \ldots, x_k \in M_d$, $l \geq 0$. If no more atom can be added to M_d according to the above rule then $\mathbb{F}$ is unsatisfiable iff there is a clause $\neg x_1 \vee \cdots \vee \neg x_k$ in $\mathbb{F}$ such that $x_1, \ldots, x_k \in M_d$.

Actually when $\mathbb{F}$ is unsatisfiable, this method performs a unit R(1) refutation of $\mathbb{F}$. The treelike form of this refutation may however be of exponential size. Now using the sets M_i from the above construction (in order of decreasing i) the following algorithm produces a linear R*(1) refutation of $\mathbb{F}$.

> Start with the clause $C_1 = \neg x_1 \vee \cdots \vee \neg x_k$ such that $x_1, \ldots, x_k \in M_{d_1}$, where d_1 is the final index. Now we will subsequently derive clauses C_i and indices d_i for $i = 1, 2, \ldots$ such that C_i is a disjunction of some negated variables from M_{d_i}. Obtain d_i and C_i from d_{i-1} and C_{i-1} as follows: Let $d_i < d_{i-1}$ be the minimal index such that all variables in C_{i-1} are fully contained in M_{d_i+1}. This means that in order to construct M_{d_i+1} from M_{d_i} a variable x from C_i had been added such that there is a clause in F of the form $A_{d_i} = \neg y_1 \vee \cdots \vee \neg y_l \vee x$. C_{i+1} is obtained from C_i by resolving with A_{d_i} on x. Notice that all variables in C_i are contained in M_{d_i}. Continue like this until C_i is λ.

Since $M_1 = \emptyset$ and $d_{i+1} < d_i$, λ will be derived in at most d_1 steps. Moreover since any clause A_{d_i} is different, any input clause is used at most once. Q.E.D.

With this we can give the desired upper bound.

Theorem 63 *There is a* R*(k) *refutation of* $\mathbb{PEB}_k^l(G)$ *that involves less than twice the number of clauses in* $\mathbb{PEB}_k^l(G)$.

Proof. For some node v and $i \in [l]$ let v_i denote the clause $v_{i,1} \vee \cdots \vee v_{i,k}$ and let $\bar{v}_i$ denote the k-term $\bar{v}_{i,1} \wedge \cdots \wedge \bar{v}_{i,k}$. Hence, each source clause $s_{i,1} \vee \cdots \vee s_{i,k}$ is denoted by s_i for $i \in [l]$.

From the target clauses

$$\bar{t}_{1,j_1} \vee \cdots \vee \bar{t}_{l,j_l}$$

with $j_1, \ldots, j_l \in [k]$, we derive by solely using the $\wedge$-introduction rule, and using each of these clauses once, the k-clause

$$(\bar{t}_{1,1} \wedge \cdots \wedge \bar{t}_{1,k}) \vee \cdots \vee (\bar{t}_{l,1} \wedge \cdots \wedge \bar{t}_{l,k})$$

which is abbreviated by

$$\bar{t}_1 \vee \cdots \vee \bar{t}_l \,.$$

In a similar way we derive from the pebbling clauses

$$\bar{u}_{1,j_1} \vee \cdots \vee \bar{u}_{l,j_l} \vee \bar{v}_{1,m_1} \vee \cdots \vee \bar{v}_{l,m_l} \vee w_{i,1} \vee \cdots \vee w_{i,k}$$

for $j_1, \ldots, j_l, m_1, \ldots, m_l \in [k]$ the k-clause

$$\bar{u}_1 \vee \cdots \vee \bar{u}_l \vee \bar{v}_1 \vee \cdots \vee \bar{v}_l \vee w_i \,.$$

Observe, that we arrive at the Horn formula $\mathbb{PEB}_1^l(G)$ if we consider the formulas v_i and $\bar{v}_i$ as variables and their negations. Moreover any R*(1) refutation of this Horn formula is essentially a R*(k) refutation, since a cut involving v_i and $\bar{v}_i$ corresponds to a cut in R*(k). Proposition 62 yields a treelike refutation of this Horn formula of $\mathbb{PEB}_1^l(G)$ that uses each input clause at most once. Combining these refutations we obtain a R*(k) refutation of $\mathbb{PEB}_k^l(G)$ that uses each input clause at most once. Since we did not use weakening, the number of nodes in the refutation tree is less then twice the number of leaves. Hence the stated bound follows. Q.E.D.

Note that $\mathbb{PEB}_2^1(G)$ are the Pebbling Contradictions in [16]. By Theorem 63 and the lower bound of [16] we get an almost exponential separation between R*(1) and R*(2).

Corollary 64 R*(2) *dominates* R*(1).

2.4.2 Lower bounds for Generalized Pebbling Contradictions

In this section we show that any R*(k) refutation of $\mathbb{PEB}_{k+1}^l(G)$ with $l \geq k$ is of size at least $2^{(\mathcal{P}(G)-3)/k}$. Recall that $\mathcal{P}(G)$ is the pebbling number of G, see Definition 12. To obtain the lower bound we generalize a game introduced in [55] to prove lower bounds for R*(1). It is a 2-Player game where the two players build a partial assignment, one variable per round.

Here we extend the rules of this game such that at each round the two players can play with up to k variables at once.

A game on contradictions

The game $G_k(\mathbb{F})$ is a 2-Player game played on the unsatisfiable CNF formula $\mathbb{F}$. The aim of the first player, the Prover, is to build an assignment that falsifies an initial clause of $\mathbb{F}$. The aim of the second player, the Delayer, is to get the maximal number of points.

At each round the Prover asks for a set L of up to k yet unassigned literals in $\mathbb{F}$. The Delayer answers with a partial, possibly total, assignment ρ to the variables in L. If ρ falsifies either the conjunction or the disjunction of the literals in L, then the round is over. Otherwise the Prover extends ρ to a total assignment over the variables in L and the Delayer scores one point.

We show that each $R^*(k)$ refutation yields a strategy for the Prover in which the Delayer scores a number of points at most logarithmic in the size of the refutation. Actually already a special type of decision tree, called k-decision tree, here, for $\mathbb{F}$ can be used by the Prover to obtain a good strategy.

It is well known, see [16], that a $R^*(1)$ refutation of a CNF formula $\mathbb{F}$ can be transformed into a binary decision tree T of the same size such that for any assignment to $\mathbb{F}$, T yields a falsified clause of $\mathbb{F}$. In T each inner node is labeled by a variable and the decision how to continue the path at an inner node is determined by the assignment to its variable. So any total assignment will lead to a leaf node of T associated to a clause that is falsified by that assignment. Here we consider binary decision trees where each inner node is labeled by a k-term. The decision how to continue a path at an inner node is determined by the value of its k-term. We call such a tree a k-decision tree for $\mathbb{F}$. Similar to the well known result for $k = 1$ one obtains the following result for any $k \geq 1$.

Proposition 65 *If $\mathbb{F}$ has a $R^*(k)$ refutation of size S, then $\mathbb{F}$ has k-decision tree of size $\leq S$.*

Proof. We will describe a recursive procedure, called DT, that in top-down fashion maps a proof tree T for $\mathbb{F}$ into a decision tree $DT(T)$ for $\mathbb{F}$ that has not more nodes than T: If

T consists of one leaf node (labeled by an initial clause) then $\mathrm{DT}(T) = T$. Otherwise let D denote the clause labeling the root of T and consider three cases:

1. If $D = A \vee B$ is obtained by a k-cut from the clauses $A \vee \bigwedge_{l \in L} l$ and $B \vee \bigvee_{l \in L} \neg l$ labeling the roots of the two direct subtrees T_1 and T_2 (respectively) of T, then the root of $\mathrm{DT}(T)$ is labeled by the k-term $C = \bigwedge_{l \in L} l$ and $\mathrm{DT}(T)$ consists of the two direct subtrees $\mathrm{DT}(T_1)$, $\mathrm{DT}(T_2)$, such that any assignment satisfying (falsifying) C is led into $\mathrm{DT}(T_2)$ (resp., $\mathrm{DT}(T_1)$).

2. If D is obtained by $\wedge$-introduction, involving the k-terms C_1, C_2 such that $C_1 \wedge C_2$ is in D, then label the root of $\mathrm{DT}(T)$ by C_1 and branch to $\mathrm{DT}(T_1)$ (resp. $\mathrm{DT}(T_2)$) if C_1 is falsified (satisfied).

3. If D is obtained by weakening and T' is the direct subtree of T then let $\mathrm{DT}(T) = \mathrm{DT}(T')$.

The correctness of the transformation is proved by observing that the following invariant is maintained: any complete assignment α that is led to $\mathrm{DT}(T)$ through the yet partially constructed decision tree, is falsifying the clause D labeling the root of T. Q.E.D.

For $k = 1$ also the reverse inequality holds, see [16]. Since for any contradiction $\mathbb{F}$ in k-CNF there is a trivial k-decision tree of linear size: just check for each clause whether it is falsified; we obtain the following separation between the size of k-decision trees and the size of $\mathrm{R}^*(k)$ refutations.

Proposition 66 *There is a family $(\mathbb{F}_n)$ of contradictions such that $\mathbb{F}_n$ has a 3-decision tree of size $O(n)$ but any $\mathrm{R}^*(k)$ refutation of $\mathbb{F}_n$ has size $2^{\Omega(n)}$.*

Proof. Since by Theorem 77 $\mathrm{R}^*(k)$ is simulated by $\mathrm{R}(1)$, the lower bound is given by the known lower bounds for 3-CNF contradictions, see [17]. Q.E.D.

Proposition 67 *If $\mathbb{F}$ has a k-decision tree of size S, then the Prover has a strategy for $G_k(\mathbb{F})$ such that the Delayer scores at most $\lfloor \log S \rfloor$ points.*

Proof. Let T be a k-decision tree of size S. The Prover's strategy will maintain the following invariant: if the Delayer has scored p points, then the currently constructed partial assignment α will lead to a node in T such that the subtree T_α rooted at this node is of size at most $S/2^p$.

At the beginning the invariant holds since T is by assumption of size at most S. Now assume that the partial assignment α constructed so far is such that T_α is of size at most $S/2^p$. Let C be the k-term labeling the root of T_α.

In the next round the Prover asks for the set of those literals L in C that are yet unassigned by α. Now α is extended in this round to an assignment α' that will assign a value to the conjunction of L and therefore also the same value to C. Hence, α' will lead to a subtree $T_{\alpha'}$ of T_α. If the delayer scores a point the Prover is able to guarantee that $T_{\alpha'}$ is of at most half the size of T_α: Since the assignments chosen by the Delayer left C unassigned, the Prover is able to choose α' such that it leads into the smaller one of the both direct subtrees of T_α. Hence $T_{\alpha'}$ has a size less than half of the size of T_α, in this case. This shows that the invariant can be maintained. Q.E.D.

As a consequence we obtain the following corollary.

Corollary 68 *If the Delayer in $G_k(\mathbb{F})$ has a strategy that yields at least p points, then any k-decision tree for F, as well as any $\mathrm{R}^*(k)$ refutation for $\mathbb{F}$, is of size at least 2^p.*

Notice however that this method will not allow us to prove directly lower bounds for $\mathrm{R}^*(k)$ refutations of formulas in k-CNF.

The Delayer's strategy

We show a strategy for the Delayer that gives a high score which will be translated into $\mathrm{R}^*(k)$ size lower bounds.

Let us in the following fix a dag $G = (V, E)$ where each nonsource node has indegree 2, fix further constants l, k with $l \geq k \geq 1$. We will describe a strategy for the delayer that yields at least $(\mathcal{P}(G) - 3)/k$ points in the game $\mathrm{G}_k(\mathbb{PEB}^l_{k+1}(G))$.

For sets $S, T \subseteq V$ let us denote by $\mathcal{P}(S, T)$ the pebbling number of the graph $G' = (V, E')$

where $E' = E \setminus ((V \times S) \cup (T \times V))$. In other words we obtain G' from G by additionally making each node in S to a source node, and each node in T to a target node.

To describe the strategy of the Delayer we will need Lemma 70. It is a generalization of the following lemma from [16].

Lemma 69 *[16] For any node v in G and any subsets $S, T \subseteq V$*

$$\mathcal{P}(S,T) \leq \max\{\mathcal{P}(S, T \cup \{v\}), \mathcal{P}(S \cup \{v\}, T) + 1\}\,.$$

Lemma 70 *For any disjoint sets $W, S, T \subseteq V$, there exists a partition X, Y of W ($X \cup Y = W$ and $X \cap Y = \emptyset$) such that: $\mathcal{P}(S,T) \leq |X| + \mathcal{P}(S \cup X, T \cup Y)$.*

Proof. We proceed by induction on $|W|$. If $|W| = 1$, the claim follows by Lemma 69 For the inductive step consider a partition of W into two nonempty sets W' and W''. By applying the inductive hypothesis to W', there is a partition X', Y' of W' such that $\mathcal{P}(S,T) \leq |X'| + \mathcal{P}(S \cup X', T \cup Y')$.

Let now $S' = S \cup X'$ and $T' = T \cup Y'$. By the inductive hypothesis applied to W'', there is a partition X'', Y'' of W'' such that $\mathcal{P}(S', T') \leq |Y''| + \mathcal{P}(S' \cup X'', T' \cup Y'')$. Define $X = X' \cup X''$ and $Y = Y' \cup Y''$. All together we have

$$\begin{aligned} \mathcal{P}(S,T) &\leq |X'| + \mathcal{P}(S \cup X', T' \cup Y') \\ &\leq |X'| + |X''| + \mathcal{P}(S \cup X' \cup X'', T \cup Y' \cup Y'') \\ &= |X| + \mathcal{P}(S \cup X, T \cup Y)\,. \end{aligned}$$

Q.E.D.

Now we are ready to describe the strategy of the Delayer for the game $\mathrm{G}_k(\mathbb{PEB}^l_{k+1}(G))$. She keeps two sets of source and target nodes that she (eventually) modifies at each round. At the beginning $S_0 = T_0 = \emptyset$. Let S_r and T_r be the sets built after round r. Assume that at round $r+1$ the Prover asks for a term C of at most k literals. Let us denote by W the set of nodes associated with the variables in C. W is divided into the four sets $W \cap S_r$, $W \cap T_r$, $W_=$, and $W_> = W \setminus (S_r \cup T_r \cup W_=)$, where $W_= \subseteq W \setminus (S_r \cup T_r)$ is a maximal set with the property that $\mathcal{P}(S_r, T_r \cup W_=) = \mathcal{P}(S_r, T_r)$. Now the Delayer assigns 1 to every unassigned

variable in C that is associated with a node in $W \cap S_r$, and she assigns 0 to every unassigned variable in C associated with a node in $(W \cap T_r) \cup W_=$. If now C is either satisfied or falsified by the constructed assignment, the round is over, and the Delayer sets $T_{r+1} = T_r \cup W_=$, and $S_{r+1} = S_r$, in this case the pebbling number remains the same $\mathcal{P}(S_r, T_r) = \mathcal{P}(S_{r+1}, T_{r+1})$. Otherwise the Prover assigns a value to the remaining unassigned variables in C, the Delayer scores one point and defines S_{r+1} and T_{r+1} as follows: by Lemma 70, she chooses a partition X, Y of $W_>$ s.t.

$$\mathcal{P}(S_r, T_r \cup W_=) \leq \mathcal{P}(S_r \cup X, T_r \cup W_= \cup Y) + |X|.$$

Now $S_{r+1} = S_r \cup X$, and $T_{r+1} = T_r \cup W_= \cup Y$ (in this case the pebbling number decreases by at most $|X| \leq k$).

Assuming that the Delayer follows this strategy, she maintains the following invariants: (*I1*) If a variable $v_{i,j}$ is assigned a value in round r or before then the associated node v is in $S_r \cup T_r$. (*I2*) If $v \in S_r$ then there are at most k associated variables $v_{i,j}$ that are assigned to 0. (*I3*) If $v \in T_r$ then there are at most $k-1$ associated variables $v_{i,j}$ that are assigned to 1. (*I4*) $\mathcal{P}(G) \leq \mathcal{P}(S_r, T_r) + |S_r|$. (*I5*) At the end of round r the Delayer achieved at least $\lceil |S_r|/k \rceil$ points.

To see that (*I2*) holds, notice that for any node v the Prover is allowed to assign at most k of its associated variables. (*I3*) follows by a similar argument, by observing that if the Prover was allowed to assign a variable in round $r+1$ then $W_>$ was not empty in that round, and therefore at least one node in $W_>$ has been added to S_r which follows by the maximality of $W_=$. To see Invariant (*I4*), observe that in each round $r+1$ the pebbling number decreases at most by the number of nodes we add to S_r. (*I5*) follows since in case the Delayer scores no point in round $r+1$ then $S_{r+1} = S_r$, and otherwise if she scores a point, $|S_{r+1}| \leq |S_r| + k$.

Now observe that at the end of the game $\mathrm{G}_k(\mathbb{PEB}^l_{k+1}(G))$, say at round e, the pebbling number is considerably reduced. Namely we have:

Lemma 71 $\mathcal{P}(S_e, T_e) \leq 3$.

Proof. Let $G' = (V, E')$ where $E' = E \setminus ((V \times S_e) \cup (T_e \times V))$. Remember that $\mathcal{P}(S_e, T_e)$ was defined to be the pebbling number of G'. The game ends when the constructed partial

assignment falsifies a clause of $\mathbb{PEB}^l_{k+1}(G)$. If a source clause $s_{i,1} \vee \cdots \vee s_{i,k+1}$ associated to a source s in G is falsified then $s \in T_e$ due to (*I1*) and (*I2*). Hence s is both a source and a target node in G', which shows that one pebble suffices for a pebbling of G'. Similarly, when a target clause $\bar{t}_{1,j_1} \vee \cdots \vee \bar{t}_{l,j_l}$ is falsified then $t \in S_e$ by (*I1*) and (*I3*) (since $l \geq k$) and the pebbling number of G' is one. Finally assume that a pebbling clause associated to a node w with predecessors u and v is falsified. Similar to the previous considerations we obtain that $u, v \in S_e$, and $w \in T_e$. Hence, for a pebbling of G' it suffices to use three pebbles. Q.E.D.

Due to Invariant (*I4*) this implies that $|S_e| \geq \mathcal{P}(G) - 3$. Moreover we have

Lemma 72 *The Delayer scores at least $|S_e|/k$ points.*

Proof. In any round at most k nodes are added, and in case a node has been added to S_r in round r, the Delayer has scored a point in round r. Q.E.D.

Hence the Delayer will score at least $(\mathcal{P}(G) - 3)/k$ points.

Theorem 73 *If G is a dag where any nonsource node has indegree 2, and $l \geq k \geq 1$, then the Delayer can score at least $(\mathcal{P}(G) - 3)/k$ points in the game $G_k(\mathbb{PEB}^l_{k+1}(G))$.*

Almost exponential separations for $\mathrm{R}^*(k)$

It is shown in [53] that there is an infinite family of graphs G, where each nonsource node in G has indegree 2, such that $\mathcal{P}(G) = \Omega(n/\log n)$, where n is the number of nodes in G. Combining Theorem 73 with Corollary 68 this shows that for such a graph G, any $\mathrm{R}^*(k)$ refutation for $\mathbb{PEB}^l_{k+1}(G)$ has size $2^{\Omega(n/k \log n)}$. On the other hand $\mathbb{PEB}^l_{k+1}(G)$ consists of at most $O(n)$ clauses. Hence, by Theorem 63 there is $\mathrm{R}^*(k+1)$ refutation of $\mathbb{PEB}^l_{k+1}(G)$ of size $O(n)$. This yields an almost exponential separation between $\mathrm{R}^*(k)$ and $\mathrm{R}^*(k+1)$.

Corollary 74 *There is a family of graphs G such that any $\mathrm{R}^*(k)$ refutation for $\mathbb{PEB}^l_{k+1}(G)$ has size $2^{\Omega(n/\log n)}$ whereas there is a $\mathrm{R}^*(k+1)$ refutation for $\mathbb{PEB}^l_{k+1}(G)$ of size $O(n)$.*

Corollary 75 *Let $k > 0$. There is a family of CNF formulas $\mathbb{F}$ with a $\mathrm{R}^*(k+1)$ refutation of size s such that any $\mathrm{R}^*(k)$ refutation has size $2^{s/\log s}$.*

Corollary 76 $\mathrm{R}^*(k+1)$ *dominates* $\mathrm{R}^*(k)$.

Besides these separation between successive levels of the $\mathrm{R}^*(k)$ hierarchy, there are a few known separations between R(2) and R(1). We have shown a superpolynomial separation in Section 2.2. Later this separation was improved in [5] to slightly exponential. In [63] they proved a separation between successive levels of the $\mathrm{R}(k)$ hierarchy, and extended some lower bounds of [6] that hold only for R(2) to $\mathrm{R}(k)$.

2.4.3 R(1) dominates $\mathrm{R}^*(k)$

Moreover, R(1) simulates $\mathrm{R}^*(k)$, see [45, 5]. In fact we are able to improve the simulations from [45, 5], by showing that an increment by factor 2, independent of k, suffices.

Theorem 77 *If* $\mathbb{F}$ *has a* $\mathrm{R}^*(k)$ *refutation of size* S *then* $\mathbb{F}$ *has a* R(1) *refutation of size* $2S$.

Proof. For a $\mathrm{R}^*(k)$ derivation P let $s(P)$ denote the number of k-clauses in P that are not obtained by the weakening rule, and $a(P)$ denote the number of k-clauses in P that are obtained by $\wedge$-introduction. Below we will prove the following statement by induction on $a(T)$: for all formulas $\mathbb{F}$ in CNF, and for all clauses C, if T is a $\mathrm{R}^*(k)$ derivation of C from $\mathbb{F}$ then there is a R(1) derivation P of C from $\mathbb{F}$ with $s(P) = s(T) + a(T)$. Since weakenings can be removed in R(1) refutations the theorem follows.

If $a(T) = 0$ then T is already a R(1) derivation. Now assume $a(T) > 0$, and consider the last k-cut in T where a k-term $\bigwedge_{l\in L} l$ with $|L| \geq 2$ is involved, say

$$\frac{A \vee \bigwedge_{l\in L} l \qquad B \vee \bigvee_{l\in L} \neg l}{A \vee B}$$

Since this was a last cut, $A \vee B$, and $B \vee \bigvee_{l\in L} \neg l$ are clauses. Let T_1, T_2 denote subtrees deriving $A \vee \bigwedge_{l\in L} l$ and $B \vee \bigvee_{l\in L} \neg l$, respectively. Since T_1 must contain some $\wedge$-introduction to produce the term $\bigwedge_{l\in L} l$ we have that $a(T_2) < a(T)$ and we conclude by the inductive hypotheses that there is a R(1) derivation P_2 of $B \vee \bigvee_{l\in L} \neg l$ from $\mathbb{F}$ of size $s(P_2) = s(T_2) + a(T_2)$. Consider also the rest of the derivation $T' = T \setminus (T_1 \cup T_2)$. T' derives C from $\mathbb{F} \wedge (A \vee B)$.

By the inductive hypothesis we obtain a R(1) derivation P' of C from $\mathbb{F} \wedge (A \vee B)$ with $s(P') = s(T') + a(T') = s(T) + a(T) - \sum_{i=1,2} s(T_i) + a(T_i)$.

Now we add $B \vee \bigvee_{l \in L} \neg l$ to the initial clauses and show how to transform T_1 to a derivation tree T_1' of $A \vee B$ from $\mathbb{F} \wedge (B \vee \bigvee_{l \in L} \neg l)$ with $s(T_1') = s(T_1) + r$, and $a(T_1') = a(T_1) - r$ for some $r \geq 1$. Note that $\bigwedge_{l \in L} l$ can arrive to $A \vee \bigwedge_{l \in L} l$ through several paths, say r. Now, trace in T_1 the occurrence of the term $\bigwedge_{l \in L} l$ towards the leaves until one encounters a k-clause in which this term is introduced by $\wedge$-introduction. Denote these k-clauses by $C_i \vee \bigwedge_{l \in L} l$ for $i = 1, \ldots, r$, and denote the clauses from which they're derived by $A_i \vee \bigwedge_{l \in L_i} l$ and $B_i \vee \bigwedge_{l \in L_i'} l$ with $L = L_i \cup L_i'$, and $C_i = A_i \vee B_i$. Now replace for $i = 1, \ldots, r$ the $\wedge$-introduction

$$\frac{A_i \vee \bigwedge_{l \in L_i} \neg l \qquad B_i \vee \bigwedge_{l \in L_i'} l}{C_i \vee \bigwedge_{l \in L} l}$$

by two k-cuts (and eventually one weakening)

$$\dfrac{\dfrac{\dfrac{B \vee \bigvee_{l \in L} \neg l \qquad A_i \vee \bigwedge_{l \in L_i} l}{A_i \vee B \vee \bigvee_{l \in L \setminus L_i} \neg l}}{A_i \vee B \vee \bigvee_{l \in L_i'} \neg l} \qquad B_i \vee \bigwedge_{l \in L_i'} l}{C_i \vee B}$$

Further replace on the path towards the root of T_1 the term $\bigwedge_{l \in L} l$ by B. To obtain the derivation tree T_1' one may again need to add some weakenings on this path.

Applying the inductive hypothesis to T_1' we obtain a R(1) derivation P_1 of $A \vee B$ from $\mathbb{F} \wedge (B \vee \bigvee_{l \in L} \neg l)$ with $s(P_1) = s(T_1') + a(T_1') = (s(T_1) + r) + (a(T_1) - r) = s(T_1) + a(T_1)$.

Now combine the R(1) derivations P_2, P', and P_1 to obtain the R(1) derivation $P = P_2, P_1, P'$ of C from $\mathbb{F}$ with size $s(P) = S(P_2) + s(P_1) + s(P') = s(T) + a(T)$. Q.E.D.

An immediate corollary of previous Theorem and lower bounds for each level of the R$^*(k)$ hierarchy is the following Theorem.

Corollary 78 R(1) *dominates* R$^*(k)$ *for* $k \geq 1$.

Chapter 3

Space and width

In this chapter we include the results concerning mainly to space. In Section 3.1 we deal with R(1) and R*(1) space, the results come from [30, 9]. In Section 3.2 we show a combinatorial characterization of R*(1) space. A very simple Player-Adversary game from [55] played over any CNF formula $\mathbb{F}$, can be used to find out $\mathcal{S}_1^*(\mathbb{F})$. This characterization however does not hold for R(1) space. In [7] there is a combinatorial characterization of width. In Section 3.3 we show that for $\mathbb{PEB}_2^1(T_n)$ where T_n is the complete binary tree of n levels requires less space in R(1) than in R*(1). This is the first separation between R*(1) space and R(1) space. This leaves two interesting open problems about R(1) space. First, how much is $\mathcal{S}_1(\mathbb{PEB}_2^1(G))$ for any G or concretely for T_n. Second, is it possible to give an easy combinatorial characterization of R(1) space? The results in Section 3.2 and Section 3.3 come from [31]. The rest of this chapter is devoted to results about R(k). In Section 3.4 it is proven that R*(k) forms an strict hierarchy concerning space. Remember that in Section 2.4 it was proven that R*(k) formed also a strict hierarchy concerning size. Last, there has been some work done for proving space lower bound for R(1). We can cite [65, 2, 15]. In Section 3.5 give an unified way for proving all known space lower bounds which holds not only for R(1), it also holds for R(k).

3.1 Space for R(1)

In this section we give upper and lower bounds for R(1) space. After some general results, we will show two examples of families of unsatisfiable formulas that can be refuted within less space than its number of clauses. The first example are unsatisfiable 2-CNF formulas, Theorem 83. The second example are the formulas whose clauses are all possible combinations of literals in such a way that every variable appears once in every clause. We will see that the space needed to refute these formulas is bounded by the number of different variables in it. In fact we will prove a more general result about the space needed in R*(1) refutations.

Definition 79 *We say that a graph G_1 is* EMBEDDED *in a graph G_2 if a graph isomorphic to G_2 can be obtained from G_1 by adding nodes and edges or inserting nodes in the middle of edges of G_1.*

The following claim is straightforward:

Claim 80 *If G_1 is embedded in G_2 then the number of pebbles needed to pebble G_1 is less or equal that the number of pebbles needed to pebble G_2.*

This is so because any pebbling strategy for the G_2 can be easily adapted to pebble G_1.

We restate here with more detail what a restriction is, recall Definition 5. Let $\mathbb{F}$ a CNF formula, and α a (partial) truth assignment to the variables in $\mathbb{F}$. $\mathbb{F}_\alpha$ is a modification of $\mathbb{F}$ according to α. For every variable x in α if its truth value is 1, all the clauses in $\mathbb{F}$ containing the positive literal x are deleted and all occurrences of $\bar{x}$ are deleted. If the truth value of x is 0, then all clauses in $\mathbb{F}$ containing $\bar{x}$ are deleted and all occurrences of the literal x are deleted.

The next lemma, an easy adaptation of [54, Theorem 1], states the well known fact that for a R(1) refutation of a formula $\mathbb{F}$, for any partial truth assignment α to the variables, we can get a R(1) refutation of $\mathbb{F}_\alpha$, the formula after applying the partial assignment, embedded in the initial R(1).

Lemma 81 *Let $\mathcal{R}$ be a* R(1) *refutation of the CNF formula $\mathbb{F}$, let α be a partial truth assignment and the $\mathbb{F}_\alpha$ formula after applying the partial assignment. There is a* R(1) *refutation of $\mathbb{F}_\alpha$ whose* R(1) *graph is embedded in $\mathcal{R}$.*

Proof. We construct a new refutation $\mathcal{R}'$ transforming the clauses of $\mathcal{R}$. Every original clause is either eliminated or transformed into a new one. The new graph of clauses, after maybe contracting some adjacent nodes representing the same clause, is also a refutation graph, and by construction, the new refutation graph is embedded in the original one.

To build the new refutation we start transforming the initial clauses going downward following the original refutation. If an original clause contains a literal that has been assigned value 1 by α, then the whole clause is deleted. If it contains a literal with value 0, then the literal is deleted from the clause. Otherwise the clause remains unchanged.

If a clause in the original refutation is the resolvent of two previous ones, there are two cases depending on whether the resolved variable has been given a value by α or not. Suppose that clause C is the resolvent of $A \vee x$ and $B \vee \overline{x}$.

- variable x has been assigned by α. If $A \vee x$ (resp. $B \vee \bar{x}$) has been replaced by A' (resp. B') then C is replaced by A' (resp. B') if $\alpha(x) = 0$ (resp. $\alpha(x) = 1$).

- variable x has not been assigned by α. If $A \vee x$ (resp. $B \vee \bar{x}$) has been replaced by A' (resp. B') then C is replaced by the resolvent of A' and B' if both contain variable x, and otherwise C is replaced by any of A' or B' that does not contain variable x.

Consider the part of the new graph connected to λ. Contracting nodes of indegree one, we obtain a refutation graph that is embedded in the original one. Q.E.D.

We now give a relation between treelike size and treelike space.

Theorem 82 *If $\mathcal{L}_1^*(\mathbb{F}) \leq s$ then $\mathcal{S}_1^*(\mathbb{F}) \leq \lceil \log s \rceil + 1$.*

Proof. We will show that the refutation tree for $\mathbb{F}$ can be pebbled with $d+1$ pebbles, where d is the depth of the biggest complete binary tree embedded in the refutation graph. As the biggest possible complete binary tree embedded in a tree of size s has depth $\lceil \log s \rceil$, the theorem holds. It is a well known fact, see for example [62], that $d+1$ pebbles suffice

to pebble a complete binary tree of depth d with the directed edges pointing to the root. In fact $d+1$ pebbles suffice to pebble any binary tree whose biggest embedded complete binary tree has depth d. In order to see this we use induction on the size of the tree. The base case is obvious. Let $\mathcal{T}$ be refutation tree, and $\mathcal{T}_1$ and $\mathcal{T}_2$ be the two subtrees from the root. Let us call $d_c(\mathcal{T})$ the depth of the biggest embedded subtree in $\mathcal{T}$. So

$$d_c(\mathcal{T}) = \begin{cases} \max(d_c(\mathcal{T}_1), d_c(\mathcal{T}_2)) & \text{if } d_c(\mathcal{T}_1) \neq d_c(\mathcal{T}_2) \\ d_c(\mathcal{T}_1) + 1 & \text{if } d_c(\mathcal{T}_1) = d_c(\mathcal{T}_2) \end{cases}$$

By induction hypothesis one can pebble $\mathcal{T}_1$ with $d_c(\mathcal{T}_1)+1$ pebbles and $\mathcal{T}_2$ with $d_c(\mathcal{T}_2)+1$ pebbles. Let us suppose that $d_c(\mathcal{T}_1) < d_c(\mathcal{T}_2)$, then $d_c(\mathcal{T}) = d_c(\mathcal{T}_2)$ and one can pebble first $\mathcal{T}_2$ with $d_c(\mathcal{T}_2)+1$ pebbles, leave a pebble in the root of $\mathcal{T}_2$ and then pebble $\mathcal{T}_1$ with $d_c(\mathcal{T}_1)+1$. For this second part of the pebbling one needs $d_c(\mathcal{T}_1) + 2 \leq d_c(\mathcal{T}_2) + 1$. The other case is similar. Q.E.D.

As a first example, consider the class of unsatisfiable formulas in CNF with at most two literals per clause.

Theorem 83 *Any unsatisfiable CNF formula with at most two literals in each clause can be refuted in* $\mathrm{R}^*(1)$ *within constant space.*

Proof. The first part of the proof is similar to the one for showing that the set of 2-CNF unsatisfiable formulas can be recognized in nondeterministic logarithmic space. In fact it is not hard to see that this result can also be derived from this Theorem. Given a 2-CNF formula $\mathbb{F}$ one can construct a directed graph $G_{\mathbb{F}}$ related to it. This graph will be useful to know whether the formula is unsatisfiable or not, and in the former case, will provide us with a strategy to find a refutation that can be pebbled with constant space.

The set V of vertices of $G_{\mathbb{F}}$ is the set of literals in $\mathbb{F}$. For any clause $(x_1 \vee x_2)$, that can be viewed as the implication $\bar{x}_1 \to x_2$ or also $\bar{x}_2 \to x_1$, we include in E a directed edge from $\bar{x}_1$ to x_2 and another one from $\bar{x}_2$ to x_1. If the clause has only one literal x_1 we consider it as $(x_1 \vee x_1)$ and include in E and edge from $\bar{x}_1$ to x_1. No other edge is included in E.

The formula is unsatisfiable if and only if there is a cycle in the graph that contains a literal, say x_1, and its negation. We can use this cycle to get a R(1) refutation. Starting from

node x_1, let us call the clauses related to the edges in the cycle $C_1, C_2, \ldots, C_k$. All these are initial clauses, and suppose that $C_1, \ldots C_l$ are the clauses corresponding to the edges from x_1 to $\bar{x}_1$ in the cycle, and $C_{l+1} \ldots C_k$ correspond to the edges from $\bar{x}_1$ to x_1. One can resolve C_1 with C_2 getting a new clause which will be resolved with C_3 and so on. When resolving with C_l one gets the clause x_1. For this only 2 pebbles are needed. Analogously, starting from literal $\bar{x}_1$ one can resolve C_{l+1} with C_{l+2} and so on, until resolving with C_k and thus getting the clause $\bar{x}_1$. Resolving finally both clauses x_1 and $\bar{x}_1$ λ is obtained. This shows that at most 3 pebbles are needed to pebble such a refutation. Q.E.D.

We can apply Theorem 82 to compute the space needed in the refutation of our second example, the formula $\mathbb{CT}_n$ defined as follows.

Definition 84 *The formula* COMPLETE TREE, $\mathbb{CT}_n$ *for short, on* n *variables,* $\{x_1, \ldots, x_n\}$ *is the set of clauses with all possible combinations of literals with the restriction that each variable appears once in each clause.*

$$\mathbb{CT}_n = \{x_1 x_2 \ldots x_n, \bar{x}_1 x_2 \ldots x_n, \ldots, \bar{x}_1 \bar{x}_2 \ldots \bar{x}_n\}.$$

Observe that this formula has 2^n clauses. It is not hard to see that $\mathbb{CT}_n$ can be refuted using space $n+1$. This is so since a straightforward R*(1) refutation that resolves the variables in different stages, has size $2^{n+1} - 1$. Theorem 82 assures that this refutation can be pebbled with $n+1$ pebbles. In Corollary 92 it is shown that this amount of space is also necessary.

For some of the following results this concept will be very useful.

Definition 85 *We say that a CNF unsatisfiable formula is* MINIMALLY UNSATISFIABLE *if removing any clause the formula becomes satisfiable.*

The following result attributed to M. Tarsi can be found in [1].

Lemma 86 *Any minimally unsatisfiable CNF formula must have more clauses than variables.*

In [9] we give a new, simpler proof of the fact that any minimally unsatisfiable CNF formula must have more clauses than variables. To prove this, we only use elementary properties of regular R*(1).

Proof. (of Lemma 86) Let $\mathbb{F}$ be a minimally unsatisfiable formula over a set of variables $x_1, \ldots, x_n$. We consider a regular R*(1) refutation of the formula. This must exist since regular R(1) is refutationally complete. Observe that since the formula is minimally unsatisfiable, every variable in $\mathbb{F}$ is resolved at least once in the refutation. Let $\mathcal{T}$ be the tree associated to the R*(1) refutation of $\mathbb{F}$ and consider a postorder transversal of $\mathcal{T}$, the root comes after the nodes of its subtrees in the transversal. For every variable x_i we mark with v_i the first node in the transversal of the tree that is a resolvent of variable x_i. There are as many such nodes as variables.

Let us call *outer nodes of type 1* to the marked nodes of $\mathcal{T}$ that do not have any marked nodes in one of the two subtrees hanging from them, and *outer nodes of type 2* to those marked nodes in $\mathcal{T}$ that do not have any marked nodes in neither one of the subtrees hanging from them.

We claim that we can associate to each outer node x_i one or two initial clauses (depending on the type of the node) containing variable x_i that are not associated to any other outer node. For doing so, we order the outer nodes in the order given by the postorder transversal of $\mathcal{T}$. Let v_i be such an outer node. We consider first the case in which v_i is of type 2. It results from resolving variable x_i and therefore it must have in its left and right subtrees two initial clauses, one containing the literal x_i, and the other one containing the negated literal. Moreover, these clauses cannot be in the subtrees of any of the other previous outer nodes in the postorder. To prove this, let us suppose that there is an outer node v_j previous to v_i containing variable x_i in one of the initial clauses in its subtrees, and let v be the deepest common ancestor of v_i and v_j in $\mathcal{T}$. The occurrence of x_i in the subtree of v_j has to be resolved at some point. If it is resolved in the subtree of v containing v_j then we contradict the fact that v_i is the first place in the postorder traversal in which x_i is resolved. Otherwise the clause in node v must contain the variable x_i, but this contradicts the fact that $\mathcal{T}$ is a regular R*(1) refutation, since in the path from λ to v_i through v, variable x_i is resolved more than once. If v_i is of type 1, then its subtree that does not contain any other outer

node must contain some initial clause with the variable x_i. By the same argument as above this clause cannot appear in the subtrees from any of the other previous outer nodes in the postorder.

Starting from the last outer node in the postorder, we can then always associate to each outer node v_i one or two initial clauses, containing variable x_i belonging to the subtree rooted at v_i and have not been associated yet.

We consider now a binary tree $\mathcal{T}'$ embedded in $\mathcal{T}$ containing all the marked nodes in $\mathcal{T}$, each one corresponding to one of the variables. $\mathcal{T}'$ can have some other inner nodes of $\mathcal{T}$ that are not marked, but have two marked nodes as descendants. The leaves in $\mathcal{T}'$ are the initial clauses associated to the outer nodes in $\mathcal{T}$ as explained in the claim above. All the leaves are different clauses. $\mathcal{T}'$ is a binary tree with at least n inner nodes, one for each variable, and therefore at least $n+1$ leaves. From this follows that the number of clauses in $\mathbb{F}$ is at least $n+1$. Q.E.D.

It is obvious that the result does not apply for non minimally unsatisfiable CNF formulas in general. As an easy counterexample just consider the formula $\{x_1, \bar{x}_1, x_1x_2x_3\}$. But the result can be extended to some nonminimally unsatisfiable formulas provided that all the variables are used in some R(1) refutation. Let us formalize the concept of used subset of variables of a formula.

Definition 87 *Let $\mathbb{F}$ an unsatisfiable CNF formula over the set of variables V. We say that a subset of the variables $V' \subseteq V$ is* USED *in* R(1) *if there exists a* R(1) *refutation of $\mathbb{F}$, possibly deriving λ more than once, in which all the variables in V' are resolved at least once, that is there exists a* R(1) *refutation of $\mathbb{F}$ with associated dag G such that:*

- *The target (or targets) of G correspond to λ,*
- *the leafs are labeled with clauses in $\mathbb{F}$ and*
- *every variable in V' is resolved at least once.*

Lemma 86 can be generalized in the following way:

Theorem 88 *If $\mathbb{F}$ is an unsatisfiable formula with a* R(1) *refutation in which all its variables are used, then $\mathbb{F}$ has more clauses than variables.*

Proof. In proof in fact a more general result from which the Theorem directly follows. We consider a extension of R(1), that we call UNION R(1) in which besides the resolution rule we allow to infer from two clauses C_1 and C_2 with the property that no literal in C_1 appears negated in C_2, the clause $C_1 \vee C_2$. Clearly the new inference rule is sound. We define an unsatisfiable set of clauses to be used in union R(1) in the same way as above but allowing also the new rule in the refutation of the set of clauses.

We prove by induction of the number $|V|$ of variables in $\mathbb{F}$ that if V is used in union R(1) then $\mathbb{F}$ has more clauses than variables. Observe that the result for R(1) follows directly from this fact, since R(1) is a special case of union R(1).

The base case is straightforward, if V has one just variable, then $\mathbb{F}$ must have two clauses.

Let us suppose now that V contains n clauses. If $\mathbb{F}$ is minimally unsatisfiable we are done. Otherwise let us consider a minimally unsatisfiable subset of clauses $\mathbb{F}_m \subseteq F$. Let us call V_1 the variables in $\mathbb{F}_m$ and V_2 to the rest of variables in $\mathbb{F}$. If $V_2 = \emptyset$ then since the number of clauses is $\mathbb{F}$ is greater than the number of clauses in $\mathbb{F}_m$ and this number is greater than $|V_1|$ (S_m is minimally unsatisfiable) the result is also proved.

If $V_2 \neq \emptyset$, then we transform $\mathbb{F}$ into $\mathbb{F}'$ by deleting from the clauses in $\mathbb{F}$ all the variables in V_1. We claim that $\mathbb{F}'$ is unsatisfiable and all its variables are used in union R(1). As the variables in $\mathbb{F}$ are used, there is a union R(1) graph G that fulfills the conditions of Definition 87. We remove from the refutation graph all the variables in V_1. The nodes in G corresponding to clauses that only have variables from V_1 have no label. Also, if a node in G resulted from the R(1) of a variable in V_1, this node contains now the union of the corresponding clauses, without the variables in V_1. All the nodes in G containing some variable in V_2 appear also in G'. We obtain in this way a union refutation for $\mathbb{F}'$ with a graph G' embedded in G. λ might be derived in G' more times than in G. All the variables in V_2 appear in some leaf of G' and all the leaves are used in the new union R(1). Because of this, V_2 is a set of variables used in union R(1) of cardinality smaller than n, and by the induction hypothesis $\mathbb{F}'$ has more clauses than variables. The number of clauses in $\mathbb{F}'$ is greater than $|V_2|$ and the number of clauses in $\mathbb{F}_m$ grater than $|V_1|$. Putting both parts together, the total number of clauses in $\mathbb{F}$ is greater than the number of its variables. Q.E.D.

The concept of a set of clauses used in a proof, can be defined in an analogous way as that of used variable. In a used set of clauses, all the variables must be used. From this observation and the above result we obtain the following consequence.

Corollary 89 *Any subset of clauses S of an unsatisfiable CNF formula $\mathbb{F}$ used in a* R(1) *refutation has more clauses than variables.*

We may wonder whether it is of any use having nonminimally unsatisfiable formulas. The answer is yes. Buss and Pitassi have proved in [24] that $\mathbb{PHP}_n^m$, where $m = 2^{\sqrt{n \log n}}$ is faster to refute than its minimally unsatisfiable subformula, which is of course, $\mathbb{PHP}_n^{n+1}$.

Now we give space bounds with respect to the number of variables.

Theorem 90 *Every unsatisfiable formula with n variables can be resolved using* R(1) *in space at most $n+1$.*

Proof. As mentioned in the proof of Theorem 82, for pebbling a tree of depth d, $d+1$ pebbles suffice. If we consider regular R*(1), which is complete, we have refutation trees whose depth is at most the number of variables in the formula being refuted. Q.E.D.

There is a matching lower bound, since there are formulas of n variables whose refutation graphs can only be pebbled with $n+1$ pebbles. This is a consequence of the following result:

Theorem 91 *Let $\mathbb{F}$ an unsatisfiable CNF formula and k the smallest number of literals of a clause of $\mathbb{F}$. Any* R(1) *refutation of $\mathbb{F}$ needs at least space $k+1$.*

Proof. For any pebbling strategy, there is a first step, let us call it s, in which the set of pebbled clauses becomes unsatisfiable. This step must exist because the first pebbling step consists of pebbling an initial clause, which is always satisfiable, and the last step pebbles λ. In step s, an initial clause has to be pebbled since according to the pebbling rules the only other possibility would be to pebble a clause with both parents pebbled, and this step would not transform the set of pebbled clauses into an unsatisfiable set. Therefore the set of pebbled clauses at step s contains at least k variables,the ones of the initial clause.

Let us suppose than the set of pebbled clauses at step s is minimally unsatisfiable, then, by Lemma 86, it has at least $k+1$ clauses because it has at least k variables. On the

other hand, if this set is not minimally unsatisfiable, we can throw aside clauses until the remaining set becomes minimally unsatisfiable. Notice that we cannot delete the initial clause last added to the set, otherwise the set of clauses would be a subset of the clauses at stage $s-1$ and becomes therefore satisfiable. So, $k+1$ clauses are still needed because the initial clause is contained in the set and has at least k variables . Q.E.D.

Theorem 91 can be used to give a R(1) space lower bound for $\mathbb{CT}_n$, Definition 84. All the clauses of $\mathbb{CT}_n$ have exactly n variables, hence:

Corollary 92 $\mathcal{S}_1(\mathbb{CT}_n) \geq n+1$.

Theorem 91 can be strengthened to allow to prove lower bounds for the space needed in the refutation of a more general class of formulas.

Theorem 93 *Let $\mathbb{F}$ be a unsatisfiable CNF formula, and let k be the maximum over all partial assignments α of the minimum number of literals of a clause in $\mathbb{F}_\alpha$. The space needed in a* R(1) *refutation of $\mathbb{F}$ is at least k.*

Proof. Let α be any partial assignment to the variables in $\mathbb{F}$, and $\mathcal{R}$ a refutation of $\mathbb{F}$ that needs the smallest amount of space. From Lemma 81 we know that there exists a refutation $\mathcal{R}'$ for $\mathbb{F}_\alpha$ embedded in the structure of $\mathcal{R}$. Theorem 91 guarantees that to pebble $\mathbb{F}_\alpha$ one needs at least a number of pebbles equal to the length of the shortest clause in $\mathbb{F}_\alpha$. But as $\mathcal{R}'$ is embedded in $\mathcal{R}$, one cannot pebble $\mathcal{R}$ with fewer pebbles than $\mathcal{R}'$. To finish the proof we just need to consider an assignment α which produces a shortest clause of maximal length. Q.E.D.

We give an upper bound on the size of R(1) refutations of a formula in terms of the space and the depth needed in a refutation. We say that the DEPTH of a R(1) refutation is the size of the longest path from λ to an initial clause in the graph of the refutation.

Theorem 94 *If a* R(1) *refutation $\mathcal{R}$ of $\mathbb{F}$ has depth d, $\mathcal{S}_1(\mathcal{R}) = s$, then $\mathcal{L}_1(\mathbb{F}) \leq \binom{d+s}{s}$.*

Proof. Let $\mathcal{R}$ be the R(1) refutation proof that can be pebbled with s pebbles. The depth of a clause C in $\mathcal{R}$ is the length of the longest path from C to λ.

We associate a set A of at most s clauses in $\mathcal{R}$ with an array $\text{depth}(A) = a_1 \dots a_s$ of s numbers between 1 and $d+1$ in the following way: Sort the clauses in A by depth in $\mathcal{R}$ and for $1 \leq j \leq s$ let a_j be the depth of the clause of j-th smallest depth. If there are less than j clauses in A then let $a_j = d+1$. In this way the array $\text{depth}(A)$ has always s positions. We can compare these arrays as base $d+1$ numbers in the usual way.

$\mathcal{R}$ can be pebbled with s pebbles. W.l.o.g. we can suppose that in the pebbling strategy pebbles are removed from clauses in the first moment they are not needed anymore, that is, pebbles can only be removed from a clause only immediately after one of its successors has been pebbled.

In the pebbling strategy pebbles are placed and removed. We consider the stages right before the pebbles are placed. Let $\mathbb{F}_i$ be the set of clauses containing pebbles at the stage right before the i-th time a pebble is set or shifted. $\mathbb{F}_1$ is the empty set. Observe that, by the special form of pebbling strategy we are considering, $\mathbb{F}_{i+1}$ is obtained from $\mathbb{F}_i$ by pebbling one clause, and eventually removing one or the two predecessors of this clause.

We claim that if $\mathbb{F}_{i+1}$ and $\mathbb{F}_i$ are two consecutive pebbling stages as described, then $\text{depth}(\mathbb{F}_i) > \text{depth}(\mathbb{F}_{i+1})$. If in stage $\mathbb{F}_{i+1}$ no clauses are deleted, then the result is clear, since either one of the non-used pebbles at stage i (with depth $d+1$) is placed at depth $\leq d$, or some pebble is shifted to a position with smaller depth. In the other case one or two pebbles are deleted in stage $i+1$, but this can only happen if at stage $i+1$ a clause C resolvent of the clauses with the removed pebbles is pebbled. $\mathbb{F}_{i+1}$ differs from $\mathbb{F}_i$ since it contains C and does not contain one or the two predecessors of C. Since the depth of C is smaller than the depth of its predecessors the inequality holds.

In each stage i in the pebbling strategy at most a new clause is considered and it holds $\text{depth}(\mathbb{F}_i) > \text{depth}(\mathbb{F}_{i+1})$. Because of this the number of clauses in the refutation is bounded by the set of possible values of the function $\text{depth}(A)$ for sets A of size at most s. $\text{depth}(A)$ is encoded by an ordered sequence of s numbers ranging from 1 to $d+1$. Since there are $\binom{d+s}{s}$ possible values for these sequences, the size of the refutation is bounded by $\binom{d+s}{s}$. Q.E.D.

We get several consequences from this result:

Corollary 95 *Any family of unsatisfiable CNF formulas with* R(1) *refutations of polyno-*

mial depth and constant space, have R(1) *refutations of polynomial size.*

In some types of R(1), the depth of the proof is automatically bounded. An example is regular R(1). For this type of R(1) it is required that in every path from λ to an initial clause in the refutation graph, every variable is solved at most once. Clearly in this case the number of variables is a bound on the depth of the proof.

Corollary 96 *If an unsatisfiable CNF formula on n variables has a regular* R(1) *refutation of space s, then the size of this refutation is bounded by* $\binom{n+s}{s}$.

An interesting question is whether the depth of the refutation can be taken out of the bound given by Theorem 94. A way to do this would be by showing that a refutation of a formula can be transformed into another one that uses the same amount of space, but has bounded depth. It is not clear that this result holds, but as we see in the next section, it does hold for the case of R*(1).

We consider now the question of measuring the space for R*(1) refutations. Recall that in this case all the nodes in the underlying graph have fanout one, and that the same clause may appear more than once in this graph. Since in Definition 8 does not refer to the structure of the underlying graph, we measure initially the treelike space needed for the refutation of an unsatisfiable formula as the minimum number of pebbles needed to play the game on a refutation tree of the formula. Later on we will show that it is also possible to give a characterization of treelike space in terms of list of clauses kept in memory, in a similar way as in Definition 8. We start showing that a R*(1) refutation can be made regular without increasing the space. Tseitin [65] showed that the same result holds also if the size of the R*(1) refutation, instead of the space, is considered.

Theorem 97 *If* $\mathbb{F}$ *is a CNF unsatisfiable formula, such that* $\mathcal{S}_1^*(\mathbb{F}) < s$ *then* $\mathbb{F}$ *has a regular* R*(1) *refutation that can be pebbled with s pebbles.*

Proof. Let $\mathbb{F}$ be any formula and $\mathcal{T}$ any R*(1) refutation of $\mathbb{F}$ and for any clause C let $\mathcal{T}_C$ be the subtree in the refutation tree that derives C from initial clauses. Suppose that the last R*(1) step in the refutation, the one having λ as resolvent, resolves the variable x, and

that this variable is resolved more than once in $\mathcal{T}$. Applying Lemma 81 to $\mathcal{T}_x$ (resp. $\mathcal{T}_{\bar{x}}$) with the partial truth assignment $\alpha(x) = 0$ (resp. $\alpha(x) = 1$) and then adding again the literal x (resp. $\bar{x}$) to the clauses that had it deleted, one derives x (resp. $\bar{x}$) or directly λ. Putting both refutation trees together, the resulting R*(1) refutation is embedded in $\mathcal{T}$ and resolves variable x at most once. One can continue in this way with the parent clauses of x and $\bar{x}$ modifying the refutation until the initial clauses are reached. The way in which the new refutation is constructed assures that on every path from λ to an initial one, every variable is resolved at most once, and moreover the new refutation in embedded in the former one, and therefore it does not need more space. Q.E.D.

We can give now a definition of space in R*(1) considering list of clauses kept in memory, with the particularity that when a clause is used to derive other clauses, it is removed from the memory.

Definition 98 *Let $k \in \mathbb{N}$, we say that an unsatisfiable CNF formula $\mathbb{F}$ has a* R*(1) *refutation bounded by space k if there is a series of CNF formulas (without having repeated clauses) $\mathbb{F}_1, \ldots, \mathbb{F}_s$, such that $\mathbb{F}_1 \subseteq \mathbb{F}$, $\lambda \in \mathbb{F}_n$, in any $\mathbb{F}_i$ there are at most k clauses, and for each $i < s$, $\mathbb{F}_{i+1}$ is obtained from $\mathbb{F}_i$ by*

- *deleting (if wished) some of its clauses,*
- *adding the resolvent of two clauses of $\mathbb{F}_i$ and deleting the parent clauses.*
- *adding (if wished) some of the clauses of $\mathbb{F}$ (initial clauses).*

We show the equivalence of this definition and Definition 11 applied to trees. Clearly if a formula can be refuted in space k according to Definition 98, then there is a refutation tree than can be pebbled with k pebbles.

For the other direction, the successive lists $\mathbb{F}_i$ will be formed by the pebbled clauses in the tree. A problem can happen in case there are repetitions in the set of pebbled clauses, because in the list there can be only one copy of each clause. When deleting one instance of this clause we are deleting the only occurrence of the clause in the list. We show that one can always have a R*(1) refutation using the same space and in which two occurrences of the same clause are never pebbled simultaneously.

Lemma 99 *Let s be the minimum number of pebbles needed in any* $\mathrm{R}^*(1)$ *refutation of* $\mathbb{F}$. *There is a regular* $\mathrm{R}^*(1)$ *refutation of* $\mathbb{F}$ *that can be pebbled with s pebbles in such a way that two nodes corresponding to the same clause are not pebbled simultaneously.*

Proof. By Theorem 97 we can suppose that there is a regular $\mathrm{R}^*(1)$ refutation of $\mathbb{F}$ using s pebbles. Since every clause in the tree has at most one successor clause, when the successor clause is pebbled, in any sensible strategy, the parent clause can be deleted immediately. In Theorem 82 it is proved that the space needed to pebble a tree is the depth of its biggest embedded subtree. An optimal strategy is then: starting from the root, pebble first the subtree with the biggest embedded complete subtree and then the other subtree. Apply this rule recursively to both subtrees. If we follow this strategy when a clause, A is pebbled then we pebble the subtree that derives its mating clause A'. Since we are dealing with a regular refutation, A cannot be in the tree deriving A'. Otherwise, there would a path going from the copy of A deriving A' to the resolvent of A and A' and then to λ, in which a variable has to be resolved twice, contradicting the fact that we are dealing with regular $\mathrm{R}^*(1)$. Q.E.D.

Using Theorem 97 and the fact that in the proof of Theorem 94, applies to any kind of R(1), we get:

Lemma 100 *If an unsatisfiable formula* $\mathbb{F}$ *with n variables has a* $\mathrm{R}^*(1)$ *refutation of space s, then it has a* $\mathrm{R}^*(1)$ *refutation of size* $\binom{n+s}{s}$.

In the case of $\mathrm{R}^*(1)$ we can show a connection between the concepts of size and width. For any unsatisfiable formula $\mathbb{F}$, the difference between the width in a refutation of $\mathbb{F}$ minus the initial width of the formula, is bounded by the space in any $\mathrm{R}^*(1)$ refutation of the formula. The proof of this fact relies on the following lemma from Ben-Sasson and Wigderson:

Lemma 101 *[17] Let* $\mathbb{F}$ *be a CNF unsatisfiable formula, and for a literal a, let* $\mathbb{F}_0$ *and* $\mathbb{F}_1$ *be the formulas resulting from assigning a the truth values 0 and 1 respectively. If for some value k,* $\mathcal{W}(\mathbb{F}_0 \vdash \lambda) \leq k-1$ *and* $\mathcal{W}(\mathbb{F}_1) \leq k$ *then* $\mathcal{W}(\mathbb{F} \vdash \lambda) \leq \max\{k, \mathcal{W}(\mathbb{F})\}$

Theorem 102 $\mathcal{S}_1^*(\mathbb{F}) - 1 \geq \mathcal{W}(\mathbb{F} \vdash \lambda) - \mathcal{W}(\mathbb{F})$.

Proof. Let $\mathbb{F}$ be an unsatisfiable CNF formula, and s the minimum number of pebbles needed in any R*(1) refutation of $\mathbb{F}$, $\mathcal{T}$. We prove by induction on the depth of $\mathcal{T}$, d, that $\mathcal{W}(\mathbb{F} \vdash \lambda) \leq \mathcal{W}(\mathbb{F}) + s - 1$. For $d = 0$, we have that λ is an initial clause, and the results holds trivially. For $d > 0$, let $\mathcal{T}$ be a R*(1) refutation of $\mathbb{F}$ of depth d and let x be the last variable being resolved. Let $\mathcal{T}_0$ and $\mathcal{T}_1$ be the subtrees in the refutation deriving the literals x and $\bar{x}$ from initial clauses, and let s_0 and s_1 be the number of pebbles needed to pebble these subtrees reaching the literals x and $\bar{x}$.

Since we are dealing with a R*(1) refutation, by the proof of Theorem 82, either s_0 or s_1 must be smaller than s. W.l.o.g. let us consider $s_0 < s$. Also, $\mathcal{T}_0$ and $\mathcal{T}_1$ have depth smaller than d.

Applying the partial assignment $x = 0$ to all the clauses in $\mathcal{T}_0$ (respectively the partial truth assignment $x = 1$ to the clauses in $\mathcal{T}_1$), we obtain two refutation trees deriving λ from two sets of clauses $\mathbb{F}_0$, $\mathbb{F}_1$. By induction, $\mathcal{W}(\mathbb{F}_0 \vdash \lambda) \leq \mathcal{W}(\mathbb{F}_0) + s_0 - 1 \leq \mathcal{W}(\mathbb{F}) + s - 2$, and $\mathcal{W}(\mathbb{F}_1 \vdash \lambda) \leq \mathcal{W}(\mathbb{F}_1) + s_1 - 1 \leq \mathcal{W}(\mathbb{F}) + s - 1$. Applying Lemma 101 we obtain $s - 1 \geq \mathcal{W}(\mathbb{F} \vdash \lambda) - \mathcal{W}(\mathbb{F})$ Q.E.D.

This result shows that width lower bounds can be used to obtain space lower bounds for R*(1). Consider for example, for the case of a Tseitin formulas related to an undirected graph G with odd marking. Ben-Sasson and Wigderson have proved a width lower bound of the expansion of G [17]. By Corollary 102, this can be translated into a space lower bound for R*(1) refutations of this formulas of at least the expansion of G minus the maximal degree of the graph. In [7] it is proven that Theorem 102 also holds for R(1), solving an open problem from [30].

3.2 Combinatorial characterization of R*(1) space

We show that the Player-Adversary game from [55] played over CNF propositional formulas gives an exact characterization R*(1) space. This characterization is purely combinatorial. This game was used for proving size lower bounds for R*(1), see [55, 16]. Let us call $\mathbb{F}$ to a generic CNF formula. A generalization of the game, $\mathrm{G}_k(\mathbb{F})$, was presented in Subsection 2.4.2. Here we recall the original game as in [55, 16] that here is called $\mathrm{G}_1(\mathbb{F})$.

The combinatorial game

The game is played in rounds on an unsatisfiable CNF formula $\mathbb{F}$ by two players: Prover and Delayer. Prover wants to falsify some initial clause and Delayer tries to retard this as much as possible. In each round Prover chooses a variable in $\mathbb{F}$ and asks Delayer for a value for this variable. Delayer can answer either 0,1 or $*$. In this last case Prover can choose the truth value (0 or 1) for the variable and Delayer scores one point. The variable is set to the selected value and the next round begins. The game ends when a clause in $\mathbb{F}$ is falsified (all its literals are set to 0) by the partial assignment constructed this way. The goal of Delayer is to score as many points as possible and Prover tries to prevent this. The outcome of the game is the number of points scored by Delayer.

Definition 103 *Let $\mathbb{F}$ be an unsatisfiable formula in CNF. We denote by $g(\mathbb{F})$ the maximum number of points that Delayer can score while playing the game on $\mathbb{F}$ with an optimal strategy of Prover.*

Our result shows that for an unsatisfiable CNF formula $\mathbb{F}$, the space needed in R*(1) refutation of $\mathbb{F}$ is exactly $g(\mathbb{F}) + 1$. Observe that the outcome of the combinatorial game depends only on the structure of $\mathbb{F}$. This characterization of R*(1) space is therefore completely independent of the notion of R(1). We use the characterization and the relations from space and size in R*(1) to slightly improve a lower bound for R*(1) size in terms of the points scored in the combinatorial game from [55].

Atserias and Dalmau have given recently [7] a combinatorial characterization of R(1) width that also depends only on the structure of the formula being considered. These two results point out the naturalness of R(1) and its space and width complexity measures.

We show that for an unsatisfiable CNF formula $\mathbb{F}$, the number of points that Delayer can score while playing the game on $\mathbb{F}$ provides both an upper and a lower bound on the R*(1) space of $\mathbb{F}$.

We show first that $g(\mathbb{F}) + 1$ is an upper bound for the R*(1) space.

Theorem 104 *If for a CNF formula $\mathbb{F}$, $\mathcal{S}^*_1(\mathbb{F}) = S$, then Delayer has a strategy in which she can score at least $S - 1$ points, that is, $S - 1 \leq g(\mathbb{F})$.*

Proof. Let be S the minimum space needed in any R*(1) refutation of $\mathbb{F}$. We give a strategy for Delayer for playing the combinatorial game on $\mathbb{F}$ that scores at least $S-1$ points with any strategy of Prover. We prove the result by induction on S.

For the base case $S=2$, Delayer just needs to answer $*$ to the first variable asked by Prover.

For $S>2$, let x be the first variable asked by Prover and let $\mathbb{F}_{x=1}$ and $\mathbb{F}_{x=0}$ the CNF formulas obtained after given value 1 and 0 respectively to variable x in $\mathbb{F}$. Any R*(1) refutation of $\mathbb{F}$ requires S pebbles and therefore either

i) the R*(1) space for refuting each of $\mathbb{F}_{x=1}$ and $\mathbb{F}_{x=0}$ is at least $S-1$ or

ii) for one of the formulas, say $\mathbb{F}_{x=1}$, the R*(1) space is at least S.

Any other possibility would imply that $\mathbb{F}$ could be refuted in space less than S.

In the first case Delayer can answer $*$ and scores one point. By induction hypothesis Delayer can score $S-2$ more points playing the game in any of the formulas $\mathbb{F}_{x=1}$ or $\mathbb{F}_{x=0}$. In the second case Delayer answers the value leading to the formula that requires treelike resolution space S ($x=1$ in this case) and the game is played on $\mathbb{F}_{x=1}$ in the next round. Q.E.D.

On the other hand $g(\mathbb{F})$ is also a lower bound for the R*(1) space. Let us consider a R*(1) refutation of $\mathbb{F}$, $\mathcal{T}$. Prover and Delayer play a modification of the game $\mathrm{G}_1(\mathbb{F})$ over $\mathcal{T}$ that is called $\mathrm{G}'_1(\mathcal{T})$. This new game can be seen as several $\mathrm{G}_1(\mathbb{F})$ games played in parallel where Prover chooses the variables to ask in an order induced by the refutation $\mathcal{T}$. Delayer will still use the strategy that scores at least $g(\mathbb{F})$ points. Prover starts at the empty clause in $\mathcal{T}$ and in general, when placed in a clause C, Prover chooses the resolved variable x from the two parent clauses of C, and at the end of the round moves to one of the parent clauses of C or both of them according to the answer of Delayer. In this later case the current game forks into to games. If Prover is placed in a clause C is because the assignment built so far falsifies all clauses in the path from C to λ. If Delayer assigns to x a value 0 or 1 then Prover moves to the parent clause that is falsified by the partial assignment and the new round starts. If Delayer assigns x value $*$ then Prover mark the clause with $*$ and the game

forks into two games. In one of them x is set to 0 and Prover moves to the parent clause falsified, in the other game x is set 1 and Prover moves to the other parent clause and a new round start for each of the two new games. Each of the parallel games ends when Prover can move to an initial clause. The game $\mathrm{G}'_1(\mathcal{T})$ ends when all parallel games end.

For a refutation $\mathcal{T}$ let us denote by game($\mathcal{T}$) the subgraph of $\mathcal{T}$ formed by all the clauses that are visited by Prover and the edges joining them, in the game $\mathrm{G}'_1(\mathcal{T})$, with a strategy from Delayer scoring at least $g(\mathbb{F})$ points. We show that the pebble game played on game($\mathcal{T}$) needs at least $g(\mathbb{F})+1$ pebbles. Since game($\mathcal{T}$) is a subgraph of $\mathcal{T}$, by Claim 80 this implies that $\mathrm{R}^*(1)$ space for $\mathbb{F}$ is at least $g(\mathbb{F})+1$.

Theorem 105 *For any CNF* $\mathbb{F}$, $\mathcal{S}^*_1(\mathbb{F}) \geq g(\mathbb{F})+1$.

Proof. Let $\mathcal{T}$ be a $\mathrm{R}^*(1)$ refutation of $\mathbb{F}$. game($\mathcal{T}$) is also a tree and in any path from λ to an initial clause in game($\mathcal{T}$) there are at least $g(\mathbb{F})$ nodes marked with $*$ (branching nodes). We will show that game($\mathcal{T}$) requires at least $g(\mathcal{T})+1$ pebbles. This implies the result since game($\mathcal{T}$) is a subgraph of $\mathcal{T}$.

Consider any strategy for pebbling the tree game($\mathcal{T}$), and consider the first moment s in which all the paths going from an initial clause to the empty clause contain a pebble. After moment $s-1$ a pebble has to be placed on an initial clause C_i, and before that, the path going from C_i to λ is the only path without pebbles. This path contains at least $g(\mathbb{F})$ nodes marked with $*$. In each one of these nodes starts a path going to an initial clause. All these paths are disjoint and they all contain a pebble at instant $s-1$ (otherwise there would be at moment s a path from the empty clause to some initial clause without any pebble). Together with the pebble at moment s, this makes at least $g(\mathbb{F})+1$ pebbles. Q.E.D.

This combinatorial game was defined in [55] as a tool for proving lower bound for the size of $\mathrm{R}^*(1)$ refutations. Impagliazzo and Pudlák proved the following result:

Theorem 106 *[55] If Delayer has a strategy on a formula* $\mathbb{F}$ *which scores* r *points then* $\mathcal{L}^*_1(\mathbb{F}) \geq 2^r$.

Based on the relations between size and space in $\mathrm{R}^*(1)$ refutations and the above characterization, we can slightly improve this result by a factor of two. For this we rephrase Theorem 82:

Theorem 107 *If for a CNF formula* $\mathbb{F}$ $\mathcal{S}_1^*(\mathbb{F}) \geq s$ *then* $\mathcal{L}_1^*(\mathbb{F}) \geq 2^s - 1$.

Together with the combinatorial characterization of R*(1) space this implies:

Corollary 108 *For any unsatisfiable CNF formula* $\mathbb{F}$*, if Delayer has a strategy on* $\mathbb{F}$ *which scores* r *points then,* $\mathcal{L}_1^*(\mathbb{F}) \geq 2^{r+1} - 1$.

We have given an exact characterization of the R*(1) space for refutations of a *CNF* formula based on a purely combinatorial game and independent of the R(1) method. In Section 3.3 we show that this characterization cannot be used for R(1) space. It would be interesting to find a characterization for R(1) space. This could help to answer the question of whether there are families of formulas that have R(1) refutations of small width but require a large amount of space, a question proposed by Ben-Sasson in [15]. We conjecture that the Pebbling Contradictions, $\mathbb{PEB}_2^1(G)$ for a suitable G, defined in [17] are an example of a family with this property. These formulas have R(1) refutations with small size and width [17] and require a large amount of space in R*(1). This last result follows from our characterization and the fact that Delayer has always a strategy scoring many points [16] when playing the combinatorial game on these formulas.

3.3 Separation between R(1) space and R*(1) space

In this section we give a R(1) space upper bound that separates R(1) space from R*(1) space. Recall the definition of the Pebbling Contradictions from Subsection 1.3.4. We are using $\mathbb{PEB}_2^1(G)$ for a suitable graph G. It is convenient here to write the formula $\mathbb{PEB}_2^1(G)$.

Definition 109 *For a dag* G *with indegree 2, the clauses of* $\mathbb{PEB}_2^1(G)$ *are as follows.*

1. *The source clause for a source node* s *is* $s_1 s_2$.
2. *The target clauses for a target node* t *are* $\bar{t}_1$ *and* $\bar{t}_2$.
3. *The pebbling clauses for any nonsource node* w *with parent nodes* u *and* v *are* $\bar{u}_1\bar{v}_1w_1w_2$, $\bar{u}_1\bar{v}_2w_1w_2$, $\bar{u}_2\bar{v}_1w_1w_2$ *and* $\bar{u}_2\bar{v}_2w_1w_2$.

Let T_n be the complete tree with n levels and let $\mathbb{F} \vdash^s C$ mean that clause C can be derived in space s from $\mathbb{F}$ or a subset of $\mathbb{F}$.

Lemma 110 *For $n \geq 5$, if $\mathbb{PEB}_2^1(T_{n-3}) \vdash^{s-2} \lambda$, $\mathbb{PEB}_2^1(T_{n-2}) \vdash^{s-1} \lambda$ and $\mathbb{PEB}_2^1(T_{n-1}) \vdash^s \lambda$ then $\mathbb{PEB}_2^1(T_n) \vdash^s \lambda$.*

Proof. We give a R(1) strategy for refuting $\mathbb{PEB}_2^1(T_n)$ measuring the space needed. The variables names follows the representation of T_n in Figure 3.3. Since $\mathbb{PEB}_2^1(T_{n-1}) \vdash^s \lambda$ it follows that $\mathbb{PEB}_2^1(T_n) \vdash^s b_1 b_2$. This is because all the clauses in $\mathbb{PEB}_2^1(T_{n-1})$ occurs in $\mathbb{PEB}_2^1(T_n)$ except for clauses $\bar{b}_1$ and $\bar{b}_2$. Similarly, since $\mathbb{PEB}_2^1(T_{n-2}) \vdash^{s-1} \lambda$ it is also clear that $\mathbb{PEB}_2^1(T_n) \vdash^{s-1} d_1 d_2$. So we can derive the two clauses $b_1 b_2$ and $d_1 d_2$ using space s by first deriving $b_1 b_2$ in space s, keeping it, and then deriving $d_1 d_2$. The maximum amount of space used until this point is s.

From clauses $\bar{a}_1$, $\bar{a}_2$, the pebbling clauses for a (which are initial clauses) and clause $b_1 b_2$, we can derive using constant space 3 $\bar{c}_1$ and $\bar{c}_2$. This means that from the stage with the clauses $d_1 d_2$ and $b_1 b1$ we can derive $d_1 d_2$ $\bar{c}_1$ and $\bar{c}_2$ using space 4, see Table 3.3.

Now from $d_1 d_2$, $\bar{c}_1$, $\bar{c}_2$ and the pebbling clauses for c we get in space 5 $\bar{e}_1$ and $\bar{e}_2$. The derivation is very similar to that in Table 3.3, but now clauses $\bar{c}_1$ and $\bar{c}_2$ must be kept in memory as they are not initial clauses. The detailed derivation is in Table 3.3.

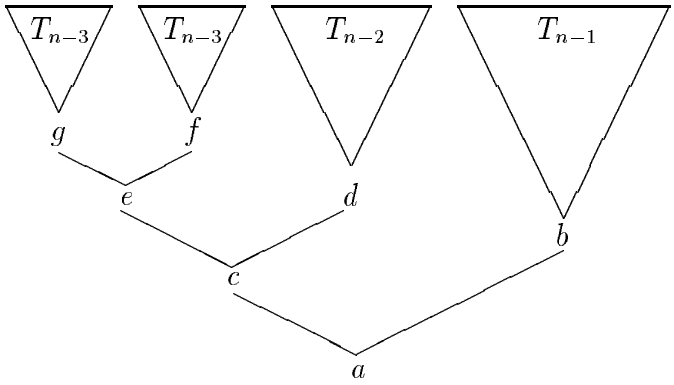
Since $\mathbb{PEB}_2^1(T_{n-3}) \vdash^{s-2} \lambda$ it follows that $\mathbb{PEB}_2^1(T_n) \vdash^{s-2} f_1 f_2$. During this derivation we have to keep $\bar{e}_1$ and $\bar{e}_2$, so the maximum amount of space used is s. From $f_1 f_2$, $\bar{e}_1$, $\bar{e}_2$ and the pebbling clauses for e we get $\bar{g}_1$ and $\bar{g}_2$ in space 5 as in Table 3.3. Again as $\mathbb{PEB}_2^1(T_{n-3}) \vdash^{s-2} \lambda$ it follows clear that $\mathbb{PEB}_2^1(T_n) \vdash^{s-2} g_1 g_2$. From $g_1 g_2$, $\bar{g}_1$ and $\bar{g}_2$ we derive λ in space 3. Q.E.D.

It is easy to check that $\mathbb{PEB}_2^1(T_2)$ can be pebbled with 3 pebbles, see Table 3.3. That means that $\mathbb{PEB}_2^1(T_3)$ needs at most 4 pebbles and $\mathbb{PEB}_2^1(T_4)$ 5 pebbles. Using Theorem 110, $\mathbb{PEB}_2^1(T_5)$ can be pebbled also with 5 pebbles, thus saving one pebble. From this follows the upper bound for the resolution space of $\mathbb{PEB}_2^1(T_n)$.

Corollary 111 *For every n, $\mathcal{S}_1(\mathbb{PEB}_2^1(T_n)) \leq 2n/3 + 3$.*

Proof. The result follows from the fact that for $n = 2 \mod 3$, $\mathbb{PEB}_2^1(T_n)$ has a R(1) refutation with space at most $2(n+1)/3 + 1$. We prove this by induction on n. The base

Figure 3.1: Drawing of T_n



case $n = 2$ is clear since in Table 3.3 there is a R(1) refutation of $\mathbb{PEB}_2^1(T_2)$ with space 3. It also holds that for any n, $\mathbb{PEB}_2^1(T_{n+1})$ requires space at most $s+1$ if $\mathbb{PEB}_2^1(T_n)$ can be refuted using space s. For the induction step, let us suppose that $n = 2 \mod 3$. By induction hypothesis the space needed for $\mathbb{PEB}_2^1(T_{n-3})$ is at most $2(n-2)/3+1$. Using the above property we get that the space needed for $\mathbb{PEB}_2^1(T_{n-2})$ and for $\mathbb{PEB}_2^1(T_{n-1})$ respectively at most $2(n-2)/3+2$ and $2(n-2)/3+3 = 2(n+1)/3+1$. By Lemma 110 $\mathbb{PEB}_2^1(T_n)$ requires also at most space $2(n+1)/3+1$. Q.E.D.

On the other hand in the case of R*(1), the space needed in a refutation of $\mathbb{PEB}_2^1(T_n)$ is at least $n-2$. This follows our characterization of R*(1) space together with the lower bound obtained in [16] on the number on points obtained by Delayer's when playing the combinatorial game on $\mathbb{PEB}_1^2(G)$. We just need the particular case of this result when G is T_n.

Theorem 112 *[16] For every n Delayer has a strategy in which at least $n-3$ points can be scored, when playing the combinatorial game on $\mathbb{PEB}_2^1(T_n)$.*

Corollary 113 *For every n, $\mathcal{S}_1^*(\mathbb{PEB}_2^1(T_n)) \geq n-2$.*

Table 3.1: R(1) derivation of $\bar{c}_2$ and $\bar{c}_1$

$d_1 d_2$	$b_1 b_2$		
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1 \bar{b}_1 a_1 a_2$	
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1 b_2 a_1 a_2$	
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1 b_2 a_1 a_2$	$\bar{c}_1 \bar{b}_2 a_1 a_2$
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1 a_1 a_2$	
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1 a_1 a_2$	$\bar{a}_1$
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1 a_2$	
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1 a_2$	$\bar{a}_2$
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1$	
$d_1 d_2$	$b_1 b_2$	$\bar{c}_1$	$\bar{c}_2 \bar{b}_1 a_1 a_2$
$d_1 d_2$	$\bar{c}_2 b_2 a_1 a_2$	$\bar{c}_1$	
$d_1 d_2$	$\bar{c}_2 b_2 a_1 a_2$	$\bar{c}_1$	$\bar{c}_2 \bar{b}_2 a_1 a_2$
$d_1 d_2$	$\bar{c}_2 a_1 a_2$	$\bar{c}_1$	
$d_1 d_2$	$\bar{c}_2 a_1 a_2$	$\bar{c}_1$	$\bar{a}_1$
$d_1 d_2$	$\bar{c}_2 a_2$	$\bar{c}_1$	
$d_1 d_2$	$\bar{c}_2 a_2$	$\bar{c}_1$	$\bar{a}_2$
$d_1 d_2$	$\bar{c}_2$	$\bar{c}_1$	

Table 3.2: R(1) derivation of $\bar{e}_2$ and $\bar{e}_1$

$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$		
$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$	$\bar{e}_1 \bar{d}_1 c_1 c_2$	
$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$	$\bar{e}_1 d_2 c_1 c_2$	
$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$	$\bar{e}_1 d_2 c_1 c_2$	$\bar{e}_1 \bar{d}_2 c_1 c_2$
$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$	$\bar{e}_1 c_1 c_2$	
$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$	$\bar{e}_1 c_1$	
$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$	$\bar{e}_1$	
$\bar{c}_2$	$\bar{c}_1$	$d_1 d_2$	$\bar{e}_1$	$\bar{e}_2 \bar{d}_1 c_1 c_2$
$\bar{c}_2$	$\bar{c}_1$	$\bar{e}_2 d_2 c_1 c_2$	$\bar{e}_1$	
$\bar{c}_2$	$\bar{c}_1$	$\bar{e}_2 d_2 c_1 c_2$	$\bar{e}_1$	$\bar{e}_2 \bar{d}_2 c_1 c_2$
$\bar{c}_2$	$\bar{c}_1$	$\bar{e}_2 c_1 c_2$	$\bar{e}_1$	
$\bar{c}_2$	$\bar{e}_2 c_2$		$\bar{e}_1$	
$\bar{e}_2$			$\bar{e}_1$	

Table 3.3: R(1) refutation of $\mathbb{PEB}_2^1(T_2)$

$a_1 a_2$		
$a_1 a_2$	$\bar{a}_1 \bar{b}_1 c_1 c_2$	
$a_2 \bar{b}_1 c_1 c_2$		
$a_2 \bar{b}_1 c_1 c_2$	$\bar{a}_2 \bar{b}_1 c_1 c_2$	
$\bar{b}_1 c_1 c_2$		
$\bar{b}_1 c_1 c_2$	$b_1 b_2$	
$b_2 c_1 c_2$		
$b_2 c_1 c_2$	$a_1 a_2$	
$b_2 c_1 c_2$	$a_1 a_2$	$\bar{a}_1 \bar{b}_2 c_1 c_2$
$b_2 c_1 c_2$	$a_2 \bar{b}_2 c_1 c_2$	
$b_2 c_1 c_2$	$a_2 \bar{b}_2 c_1 c_2$	$\bar{a}_2 \bar{b}_2 c_1 c_2$
$b_2 c_1 c_2$	$\bar{b}_2 c_1 c_2$	
$c_1 c_2$		
$c_1 c_2$	$\bar{c}_1$	
c_2		
c_2	$\bar{c}_2$	
λ		

3.4 Space separations for $\mathrm{R}^*(k)$

We proof that $\mathrm{R}^*(k)$ forms a hierarchy with respect to space. Remember that in Section 2.4 a similar result was proven for size.

Consider the following definition from [5]. Given a *CNF* $\mathbb{F}$ over variables in X, and a $k \in \mathbb{N}$, define a new formula $\mathbb{F}_k$ this way: for any set of literals $l_1, \ldots, l_s$ over X, with $s \leq k$, consider a new literal $z_{l_1,\ldots,l_s}$ meaning $l_1 \wedge \ldots \wedge l_s$. Let $\mathbb{E}(X,k)$ be the set of clauses $\neg z_{l_1,\ldots,l_s} \vee l_i$, for $i \in [s]$ and $\neg l_1 \vee \cdots \vee \neg l_s \vee \neg z_{l_1,\ldots,l_s}$. Then $\mathbb{F}_k$ is $\mathbb{F} \cup \mathbb{E}(X,k)$.

The following two Lemmas were proved in [5].

Lemma 114 *[5] For any CNF* $\mathbb{F}$ *and* $k \in \mathbb{N}$*, if* $\mathbb{F}$ *has* $\mathrm{R}(k)$ *(resp.* $\mathrm{R}^*(k)$*) refutations of size* S*, then* $\mathbb{F}_k$ *has* $\mathrm{R}(1)$ *(resp.* $\mathrm{R}^*(1)$*) refutations of size* $O(kS)$*.*

Lemma 115 *[5] For any CNF* $\mathbb{F}$ *and* $k \in \mathbb{N}$*, if* $\mathbb{F}_k$ *has* $\mathrm{R}(1)$ *(resp.* $\mathrm{R}^*(1)$*) refutations of size* S*, then* $\mathbb{F}$ *has* $\mathrm{R}(k)$ *(resp.* $\mathrm{R}^*(k)$*) refutations of size* $O(kS)$*.*

It is not difficult to see that similar relations holds for the space.

Lemma 116 *For any CNF* $\mathbb{F}$ *and* $k \in \mathbb{N}$*, if there are* $\mathrm{R}(k)$ *(resp.* $\mathrm{R}^*(k)$*) refutations of* $\mathbb{F}$ *using space* S*, then there are* $\mathrm{R}(1)$ *(resp.* $\mathrm{R}^*(1)$*) refutations of* $\mathbb{F}_k$ *using space at most* $S+2$*.*

Lemma 117 *For any CNF* $\mathbb{F}$ *and* $k \in \mathbb{N}$*, if there are* $\mathrm{R}(1)$ *(resp.* $\mathrm{R}^*(1)$*) refutation of* $\mathbb{F}_k$ *using space* S*, then there are* $\mathrm{R}(k)$ *(resp.* $\mathrm{R}^*(k)$*) refutations using space at most* $S+2$*.*

We will extend Lemma 100 in to give exponential lower bounds for $\mathrm{R}^*(k)$ space.

Lemma 118 *For any CNF* $\mathbb{F}$ *over* n *variables and* $k \in \mathbb{N}$*, if* $\mathcal{L}_k^*(\mathbb{F}) \geq S$*, then* $\mathcal{S}_k^*(\mathbb{F}) \geq \Omega(\frac{\log S}{\log n})$*.*

Proof. Let $\mathbb{F}$ be a *CNF* contradiction over n variables such that $\mathcal{L}_k^*(\mathbb{F}) \geq S$. Lemma 115 implies that $\mathcal{L}_1^*(\mathbb{F}_k) \geq \frac{S}{k}$. Since the space in $\mathrm{R}(1)$ is always upper bounded by the number of variables, it is easy to see that Lemma 100 in turn implies that $\mathcal{S}_1^*(\mathbb{F}_k) \geq \Omega(\log S/\log n)$, which implies the claim by Lemma 116. Q.E.D.

As a corollary of the previous lemma and the size lower bound of Corollary 74, we obtain a space lower bound for $\mathbb{PEB}_{k+1}^k(G)$ for any constant k.

Corollary 119 $\mathcal{S}_k^*(\mathbb{PEB}_{k+1}^k(G)) \geq \Omega(n/\log n)$.

On the other hand we can obtain constant space $\mathrm{R}^*(k+1)$ refutations of $\mathbb{PEB}_{k+1}^k(G)$.

Lemma 120 $\mathcal{S}_{k+1}^*(\mathbb{PEB}_{k+1}^k(G)) = O(1)$.

Proof. Notice that the refutation presented in Theorem 63 consists of an underlying linear treelike refutation where the leaves are replaced by complete binary trees of constant size, because the number of leaves in these binary trees is at most $(2k)^{k+1}$. It is obvious that only a constant number of pebbles is needed to pebble such a proof. Q.E.D.

Therefore the $\mathrm{R}^*(k)$ space hierarchy, for k constant, is strict.

Corollary 121 *Let $k > 0$. There is a family of CNF formulas $\mathbb{F}$ over n variables such that $\mathcal{S}_{k+1}^*(\mathbb{F}) = O(1)$, but $\mathcal{S}_k^*(\mathbb{F}) = \Omega(n/\log n)$.*

3.5 Space lower bounds for $\mathrm{R}(k)$

We present the concept of μ-dynamical satisfiability for a CNF formula $\mathbb{F}$. This concept provides an unified way for proving space lower bounds for $\mathrm{R}(k)$. There are similarities between the concept of dynamical satisfiability and the combinatorial characterization of width in [7], but our method was found independently as the result of a detailed inspection of the common points to all space lower bounds proofs already known ([30, 2, 15]).

In [7] it is proved that $\mathcal{S}_1(\mathbb{F}) \geq \mathcal{W}(\mathbb{F} \vdash \lambda) - \mathcal{W}(\mathbb{F})$, so width lower bounds for $\mathbb{F}$ can be translated into space lower bounds for $\mathbb{F}$ when the width of $\mathbb{F}$ is small. In the case of $\mathbb{PHP}$ the width characterization in [7] cannot be used to derive meaningful space lower bounds as $\mathbb{PHP}$ has large width, whereas the concept of μ-dynamical satisfiability can provide meaningful lower bounds for $\mathbb{PHP}$. Besides width lower bounds cannot be used effectively to derive size lower bounds when the initial width is large. To overcome this difficulty also in [7] they transform any formula $\mathbb{F}$ with large clauses into the standard non-deterministic extension of $\mathbb{F}$, called $E\mathbb{F}$, see [2]. The formula $E\mathbb{F}$ has small initial width, so the width lower bounds for this formula can be translated into width lower bounds for the original formula $\mathbb{F}$ and

consequently to space lower bounds. So this method from [7] can be used instead the μ-dynamical satisfiability to get space lower bounds. Nevertheless neither the authors of [7] nor the authors of [29] noticed this possibility and the concept of μ-dynamical satisfiability is simpler.

Definition 122 *Let $\mathbb{F}$ be a CNF over n variables and let $1 \leq \mu \leq n$. $\mathbb{F}$ is* μ-dynamically satisfiable *if there is a class $\Delta_{\mathbb{F}}$ of partial assignments such that the following properties hold:*

1. CLOSURE *under inclusion: if $\alpha \in \Delta_{\mathbb{F}}$ and $\beta \sqsubseteq \alpha$, then $\beta \in \Delta_{\mathbb{F}}$;*
2. EXTENDIBILITY: *if $\alpha \in \Delta_{\mathbb{F}}$ and $|\alpha| < \mu$ and C is a clause in $\mathbb{F}$, then there is a partial assignment $\beta \in \Delta_{\mathbb{F}}$ such that $\beta \sqsupseteq \alpha$, $\beta(C) = 1$.*

We show that dynamical satisfiability implies space lower bounds for R(k).

Theorem 123 *Let $\mathbb{F}$ be an unsatisfiable CNF formula, which is μ-dynamically satisfiable. Then $\mathcal{S}_k(\mathbb{F}) > \frac{\mu}{k}$.*

Proof. Let $\Delta_{\mathbb{F}}$ be the class of partial assignments that makes $\mathbb{F}$ μ-dynamically satisfiable. Let $\mathcal{C}_0, \ldots, \mathcal{C}_s$ be a set of configurations expressing a refutation of $\mathbb{F}$ in R(k). Assuming by contradiction that $\mathcal{S}_k(\mathbb{F}) \leq \frac{\mu}{k}$, we build a sequence of partial assignments to the variables of $\mathbb{F}$, α_i, where $i = 0, \ldots, s$. These assignments have the following three properties: $\alpha_i \in \Delta_{\mathbb{F}}$, $\mathcal{C}_i|_{\alpha_i} \equiv 1$ and $|\alpha_i| \leq k|\mathcal{C}_i|$. The contradiction is reached since no partial assignment can satisfy $\mathcal{C}_s$ which includes the empty clause, so $\mathcal{S}_k(\mathbb{F}) > \frac{\mu}{k}$.

Since $\mathcal{C}_0 = \emptyset$, α_0 can be set as the empty assignment. Given α_i, we build α_{i+1} according to the rule used to produce $\mathcal{C}_{i+1}$ from $\mathcal{C}_i$.

- *Axiom Download:* Let C be the downloaded clause of $\mathbb{F}$. If a clause can be downloaded, then $|\mathcal{C}_i| \leq \mu/k - 1$, hence $|\alpha_i| \leq \mu - k \leq \mu - 1$, since $k \geq 1$. Since $\mathbb{F}$ is μ-dynamically unsatisfiable and $|\alpha_i| < \mu$, by the extendibility of $\Delta_{\mathbb{F}}$, there is a $\beta \in \Delta_{\mathbb{F}}$ such that $\beta \sqsupseteq \alpha$ and $C|_\beta \equiv 1$. Notice that by the closure property of $\Delta_{\mathbb{F}}$ and the fact that C is a clause, we can assume that β is setting to 1 at most on literal in C. Setting α_{i+1} to β it follows that $\alpha_{i+1} \in \Delta_{\mathbb{F}}$ and $\mathcal{C}_{i+1}|_{\alpha_{i+1}} \equiv 1$. As $|\beta| \leq |\alpha| + 1$ and $|\mathcal{C}_{i+1}| = |\mathcal{C}_i| + 1$, then $|\alpha_{i+1}| \leq k|\mathcal{C}_{i+1}|$.

- *Inference Adding:* Set $\alpha_{i+1} = \alpha_i$. The derived k-clause is satisfied from soundness of $\mathrm{R}(k)$ and $\alpha_{i+1} \in \Delta_{\mathbb{F}}$ because $\alpha_i \in \Delta_{\mathbb{F}}$.

- *Memory Erasing:* Let C be the k-clause deleted from $\mathcal{C}_i$ to get $\mathcal{C}_{i+1}$. Clearly $\mathcal{C}_{i+1}|_{\alpha_i} \equiv 1$. For every k-clause C_j in $\mathcal{C}_{i+1}$ let $\beta_j \sqsubseteq \alpha_i$ be minimal (with respect to $\sqsubseteq$) such that $C_j|_{\beta_j} \equiv 1$. Define $\alpha_{i+1} = \bigsqcup_j \beta_j$. As $\alpha_{i+1} \sqsubseteq \alpha_i$ and $\alpha_i \in \Delta_{\mathbb{F}}$ then by the closure property $\alpha_{i+1} \in \Delta_{\mathbb{F}}$. By construction $\mathcal{C}_{i+1}|_{\alpha_{i+1}} \equiv 1$. Finally, as at most k variables are needed to satisfy a k-clause, $|\alpha_{i+1}| \leq k|\mathcal{C}_{i+1}|$

Q.E.D.

It is easy to prove size lower bounds for $\mathrm{R}^*(k)$ from Theorem 123 and Theorem 107. Since $\mathcal{S}^*_k(\mathbb{F}) \geq \mathcal{S}_k(\mathbb{F})$, a space lower bound for $\mathrm{R}(1)$ also yields a size lower bound for $\mathrm{R}^*(1)$.

Corollary 124 *If $\mathbb{F}$ is μ-dynamically unsatisfiable, then $\mathcal{L}^*_k(\mathbb{F}) \geq 2^{\Omega(\mu/k)}$.*

The rest of this section will be devoted to prove space lower bounds for $\mathrm{R}(k)$ using μ-dynamical satisfiability.

3.5.1 Semiwide formulas

We show that the concept of semiwideness, introduced in [2], implies dynamical satisfiability.

Definition 125 *[2] A partial assignment α for a satisfiable CNF $\mathbb{F}$ is* $\mathbb{F}$-CONSISTENT *if α does not falsify $\mathbb{F}$ and can be extended to an assignment satisfying $\mathbb{F}$.*

The notion of consistency is used to define semiwideness for a CNF $\mathbb{F}$.

Definition 126 *[2] A CNF $\mathbb{F}$ is* μ-SEMIWIDE *if and only if there exists a partition $\mathbb{F}', \mathbb{F}''$ of $\mathbb{F}$ such that $\mathbb{F}'$ is satisfiable and for any clause C in $\mathbb{F}''$, any $\mathbb{F}'$-consistent assignment α, with $|\alpha| < \mu$, can be extended to an $\mathbb{F}'$-consistent assignment satisfying C.*

Now we prove that semiwideness is a particular case of dynamical satisfiability.

Lemma 127 *Let $\mathbb{F}$ be an unsatisfiable CNF over n variables. If $\mathbb{F}$ is μ-semiwide, then $\mathbb{F}$ is μ-dynamically satisfiable.*

Proof. Let $\mathbb{F}', \mathbb{F}''$ be the partition of $\mathbb{F}$ guaranteed by μ-semiwideness of $\mathbb{F}$. Fix

$$\Delta_{\mathbb{F}} = \{\alpha \mid \ \alpha \text{ is } \mathbb{F}'\text{-consistent}\}$$

If α is $\mathbb{F}'$-consistent, any β such that $\beta \sqsubseteq \alpha$ is $\mathbb{F}'$-consistent, so $\Delta_{\mathbb{F}}$ has the closure property. Finally to show that $\Delta_{\mathbb{F}}$ preserves extendibility, we prove that for any clause C in $\mathbb{F}$ and any $\alpha \in \Delta_{\mathbb{F}}$, such that $|\alpha| < \mu$, there is an extension β of α in $\Delta_{\mathbb{F}}$ that satisfies C. If $C \in \mathbb{F}'$, by $\mathbb{F}'$-consistency of α, there is a β extending α satisfying α and $\mathbb{F}'$-consitent. Hence $\beta \in \Delta_{\mathbb{F}}$.

If $C \in \mathbb{F}''$, since $|\alpha| < \mu$, then by semiwideness of $\mathbb{F}$, there is a β extending α satisfying C and $\mathbb{F}'$-consistent. Hence $\beta \in \Delta_{\mathbb{F}}$. Q.E.D.

We will consider now two semiwide formulas, namely Graph Tautologies and Pigeonhole Principle. [2] proved that the class of contradictions $\mathbb{GT}_n$ is $\frac{n}{2}$-semiwide. Hence by Lemma 127 and Theorem 123:

Corollary 128 *$\mathbb{GT}_n$ is $\frac{n}{2}$-dynamically satisfiable and $\mathcal{S}_k(\mathbb{GT}_n) > \frac{n}{2k}$.*

Besides, these formulas provide another example that separates R(1) from R$^*(k)$. In [64] it is proved that $\mathbb{GT}_n$ has polynomial size R(1) refutations, hence also polynomial size R(k) refutations. This along with Corollaries 128 and 124 gives another proof for Corollary 78.

Alekhnovich *et al.* prove in [2] that for $m > n$, $\mathbb{PHP}_n^m$ is n-semiwide, we have by Lemma 127 and Theorem 123:

Corollary 129 *For any $m > n$, $\mathbb{PHP}_n^m$ is n-dynamically satisfiable and $\mathcal{S}_k(\mathbb{PHP}_n^m) > \frac{n}{k}$.*

3.5.2 Random formulas

Recall the definition of Random Formulas from Definition 1.3.5.. A random 3-CNF formula $\mathbb{F}$ is a formula $\mathbb{F} \sim \mathbf{F}_m^n$. In this subsection we prove that random 3-CNF with clause/variable ratio $\Delta > 4.6$ requires $\Omega(n/k\Delta^{\frac{1+\epsilon}{1-\epsilon}})$ space in R(k). Our result can be extended to any l-CNF.

We need some preliminary definitions from [15].

The MATCHING GAME is a two-player game defined on bipartite graphs $G = (U, V, E)$. For a node $u \in U$, let $N(u) = \{v \in V \mid (u, v) \in E\}$.

The first player, Pete, is looking for a subset $U' \subseteq U$ unmatchable into V, downloading vertices of U into U' or removing vertices from U', one at time. The second Player, Dana, tries to delay as long as she can Pete, forcing a matching of U' into V. During the game the players will build a set of edges $m \subseteq E$ and the set U' as follows:

Initially $m = \emptyset = U'$. At each round only one the following occurs:

1. Pete downloads a $u \in U$ into U', and Dana, if possible, answers by $v_u \in N(u)$ such that v is not a vertex of any edge in m. Then (u, v_u) is added to m;

2. Pete removes a u from U'. Then (u, v_u) is also removed form m, releasing v_u for a future use by Dana.

Pete wins when no answer is possible for Dana in case 1. Dana wins the game when she can force a matching of the whole U into V. The set m defines a partial matching in G. The complexity of the game, $\mathcal{M}(G)$, is the cardinality of the smallest U' Pete has to produce in any strategy to win. Notice that when $|U| > |V|$ Pete can always win and $\mathcal{M}(G) \leq |V| + 1$. Moreover, if $\mathcal{M}(G) > k$, then there is strategy for Dana such that for any $U' \subseteq U$, $|U| \leq k$, and for any $u \in U \setminus U'$ she can always find a v_u to match u.

Given a CNF $\mathbb{F}$, the bipartite graph $G_{\mathbb{F}} = (U, V, E)$ associated to $\mathbb{F}$ is defined this way: U is the set of clauses of $\mathbb{F}$, V is the set of variables of $\mathbb{F}$ and $(C, x) \in E$ iff the variable x appears (negated or not) in C. It is the clear that any partial matching m in $G_{\mathbb{F}}$, defines an assignment α_m that satisfies all clauses mentioned in m and such that $|\alpha_m| = |m|$.

Lemma 130 *Let $\mathbb{F}$ be a CNF. If $\mathcal{M}(G_{\mathbb{F}}) > \mu$, then $\mathbb{F}$ is μ-dynamically satisfiable.*

Proof. Let $\mathbb{F}$ be formed by the clauses $C_1, \ldots, C_t$. Since $\mathcal{M}(G_{\mathbb{F}}) > \mu$, there is a strategy $\mathcal{S}$ for Dana such that as long as $|U'| < \kappa$, she can always extends the matching m built so far, to any other possible clause still not in U'.

Let $I = \{i_1, \ldots, i_l\} \subseteq [t]$ be a set of indices. We need the order of the indices in I to be meaningful. Therefore any set J obtained permuting the elements of I will be considered different from I. For $I \subseteq [t]$, let $P_I = \{J \mid J \text{ is a permutation of } I\}$. Given an ordered set $I \subseteq [t]$, let $\mathbb{F}_I = \{C_i \in \mathbb{F} \mid i \in I\}$, where the order of I is inherited in $\mathbb{F}_I$. Let moreover m_I

the matching built by Dana following the strategy $\mathcal{S}$ when the clauses in $\mathbb{F}_I$ are put by Pete into U' in the order inherited from I. Let α_I be the assignment associated to the matching m_I. We define $\Delta_{\mathbb{F}}$ as follows:

$$\Delta_{\mathbb{F}} = \bigcup_{I \subseteq [t], |I| \leq \mu} \bigcup_{J \in P_I} \alpha_J$$

$\Delta_{\mathbb{F}}$ is clearly closed under inclusion by definition. Let $\alpha \in \Delta_{\mathbb{F}}$, with $|\alpha| < \mu$ and let C_l be a clause in $\mathbb{F}$. There is a $I \subseteq [t]$, and a $J \in P_I$, such that $\alpha = \alpha_J$. Since there is a 1-1 correspondence between m_J and the domain of α_J, then $|I| < \mu$. If $l \in I$, then C_l is satisfied by α_I and we have nothing to prove. Otherwise let $J' = J \cup \{l\}$ and l is the last element in the order of J'. $|J'| \leq \mu$ and hence $\alpha_{J'} \in \Delta_{\mathbb{F}}$. Moreover $\alpha_{J'}$ clearly satisfies C_l, $\alpha_{J'} \sqsupseteq \alpha_J$ since l is defined as last element in the order of J'. Hence $\Delta_{\mathbb{F}}$ verifies extendibility. Q.E.D.

When $\mathbb{F}$ is a random k-CNF, Ben-Sasson and Galesi in [15] proved the following result

Lemma 131 *[15] Let $\mathbb{F} \sim \mathbf{F}^n_{\Delta \cdot n}$, $\Delta > 4.6$. For any $\epsilon < 1$, $\mathcal{M}(G_{\mathbb{F}}) \geq \frac{n}{\Delta^{1+\epsilon}}$.*

which, by Lemma 130, implies

Corollary 132 *If $\mathbb{F} \sim \mathbf{F}^n_{\Delta \cdot n}$, $\Delta > 4.6$, then $\mathbb{F}$ is $\frac{n}{\Delta^{1+\epsilon}}$-dynamically satisfiable.*

Which by Theorem 123 and Corollary 124 in turns implies:

Corollary 133 *If $\mathbb{F} \sim \mathbf{F}^n_{\Delta \cdot n}$, $\Delta > 4.6$, then for each $k \geq 1$, $\mathcal{S}_k(\mathbb{F}) \geq \Omega(n/k \cdot \Delta^{\frac{1+\epsilon}{1-\epsilon}})$ and $\mathcal{L}^*_k(\mathbb{F}) \geq 2^{\Omega(n/k \cdot \Delta^{\frac{1+\epsilon}{1-\epsilon}})}$.*

3.5.3 Tseitin Contradictions

Recall the definition of Tseitin Contradictions from Section 1.3.2. To prove the R(k) space lower bound we follow [2].

Definition 134 *Let G be a connected graph over n nodes. The* CONNECTIVITY EXPANSION *$c(G)$ of a connected graph G is the the minimal number of edges to remove from G to obtain a graph in which the largest connected component is of size at most $n/2$.*

Let $G = (V, E)$ be a constant degree connected graph and consider the CNF $\mathbb{T}(G)$. Let α be a partial assignment on variables of $\mathbb{T}(G)$. Let $E(\alpha)$ be the subset of E corresponding to the variables assigned by α, and let $G_{max}(\alpha) = (V_{max}(\alpha), E_{max}(\alpha))$ be the maximal connected component in $(V, E - E(\alpha))$.

Definition 135 *We say that an assignment α with $|\alpha| < c(G)$ is* ADMISSIBLE *for $\mathbb{T}(G)$ if there exists an assignment α' such that (1) $\alpha \subseteq \alpha'$, and (2) for all $v \notin V_{max}(\alpha)$, α' satisfies $\mathbb{PAR}_v$.*

Note that in order admissible assignments to exist it must happen that all remaining connected components outside V_{max} must have even weight, otherwise one small connected component could not be satisfied.

The following lemma was proved in [2].

Lemma 136 *Assume that α is admissible for $\mathbb{T}(G)$. Then for any $v_0 \in V_{max}(\alpha)$ there exists an assignment α' such that $\alpha \subseteq \alpha'$ and for each vertex $v \neq v_0$, α' satisfies $\mathbb{PAR}_v$.*

We will prove that Tseitin Contradictions associated to a graph G of bounded degree d are $(c(G) - d)$-dynamically satisfiable.

Theorem 137 *Let G be a connected graph, then $\mathbb{T}(G)$ is $(c(G) - d(G))$-dynamically satisfiable.*

Proof. We define the class of partial assignments $\Delta_{\mathbb{T}(G)}$ as:

$$\Delta_{\mathbb{T}(G)} = \{\alpha \mid |\alpha| < c(G) - d(G) \text{ and } \alpha \text{ is admissible}\}$$

We need to show that $\Delta_{\mathbb{T}(G)}$ fulfills the properties of closure and extendibility.

For closure, if $\alpha \in \Delta_{\mathbb{T}(G)}$, any $\beta \sqsubseteq \alpha$ is also admissible. For extendibility, let $\alpha \in \Delta_{\mathbb{T}(G)}$ such that $|\alpha| < c(G) - d(G)$. Now consider any clause C from $\mathbb{T}(G)$. Let v be such that $C \in \mathbb{PAR}_v$. We will build a β that preserves extendibility for C. Now we split the proof in to cases:

- $v \notin V_{max}$: As α is admissible we can satisfy C by setting one free variable of C. Let β be α plus the set variable. Clearly $|\beta| \leq c(G) - d(G)$ and β is admissible, so $\beta \in \Delta_{\mathbb{T}(G)}$.

- $v \in V_{max}$: Our goal is to set a variable in C in such a way that the new biggest connected component has odd weight and all the remaining connected components have even weight. Let $e_1, \ldots, e_i$ all the edges incident to v, clearly $i \leq d(G)$. Let V^j_{max} be the biggest connected component after assigning truth values to the variables $e_1, \ldots, e_j$. Any e_j can be always assigned is such a way that V^j_{max} has odd weight. It can happen than a new connected component is detached from V^j_{max} because it was linked to V^j_{max} only by edge e_j through v. This new connected component must have even weight. We will set edges until one of them, say e, satisfies C. This must always happen. Let us suppose that none of the variables $e_1, \ldots, e_{i-1}$ satisfies C. Then $\mathbb{PAR}_v$ after applying the assignment is either e_j or $\bar{e}_j$ depending on the actual weight of v. We can set variable e_i to satisfy C and V^i_{max} must have odd weight, otherwise we can satisfy $T(V^i_{max})$ and so $\mathbb{T}(G)$ which is unsatisfiable. Note that $|V^i_{max}| > n/2$. The assignment β will be α plus e and its truth value. β is admissible because all connected components outside V_{max} have even weight and as we are adding only one variable to α clearly $\beta \leq c(G) - d(G)$.

Q.E.D.

Linear lower bounds for Tseitin contradictions are a consequence of the following Lemma which uses expander graphs.

Lemma 138 *[66, 2] There exists a family of constant degree connected graphs $G = (V, E)$ with connectivity expansion $\Omega(|V|)$.*

Theorem 139 *Let G be connected graph over n vertices provided by Lemma 138. Then for any $k \geq 1$, $\mathcal{S}_k(\mathbb{T}(G)) > \Omega(n)$.*

Chapter 4

Recapitulation

This work has dealed mainly with Proof Complexity. Our aim was to prove lower and upper bounds for complexity measures such as size and space, related to refutational Proof Systems as R(1), R(k) and CP.

In order to prove some of the results, for example the separation between R*(1) and R(1) and CP* and CP, Section 2.1, or the separation between R(1) and R(2), Section 2.2, we needed to use results from Circuit Complexity, see Theorem 24, or extend a result from [57] for monotone boolean functions to monotone real function as we did in Section 2.1.

Separation of Proof Systems regarding different complexity measures is one of the main aims of Proof Complexity. Section 2.1 is an intermediate step in separating treelike version of proof systems from the daglike version. Our separation of CP* from CP, in fact from regular R(1), represents an improvement of previous results, see [41]. Later our separation of R(1) from R(1), was improved in [16].

We were among the first researchers interested in a recent Proof System, R(k), proposed by Krajíček in [48]. We gave some of the first results about the size and space complexity of this Proof System.

In Section 2.2 we solve an open problem posed by Krajíček also in [48]. We show that R(2) does not have the feasible monotone interpolation property. That means that R(2) refutations of certain CNF formulas cannot be transformed into monotone boolean circuits of similar size, computing a function related to the CNF formula. To do so, we proved a

polynomial size upper bound for R(2) refutations of the CNF formula based on a Clique-Coclique principle. As it is known that the monotone boolean circuit computing a related function need superpolynomial size we conclude that R(2) does not have the Interpolation property. Besides, as R(1) has this property we get a superpolynomial separation between R(1) and R(2), which was the first separation between both systems.

In Section 2.3 we present an unpublished result that shows that R(2) lower bounds for $\mathbb{PHP}_n^{n^{1.5}}$ provides R(1) lower bounds for $\mathbb{PHP}_n^{n^2}$. This was a new attempt of solving a long standing open problem, the R(1) size for $\mathbb{PHP}_n^{n^2}$. Of course we do not know whether this approach would have made the proof easier, but as the problem was solved while we were working a it, see [56, 58], we abandoned this approach.

In Section 2.4, we study the size complexity of R$^*(k)$ It was known that R(2) was more powerful that R(1) and R*(2) more powerful than R*(1), so a natural question was to find out whether we can separate succesive levels of R(k) or R$^*(k)$. We show exponential separations between successive levels of what we can call now the R$^*(k)$ hierarchy and Segerlind *et al.* [63] showed separations for the R(k) hierarchy. We also prove that R(1) simulates R$^*(k)$ which is a particular case of a theorem by Krajíček, but we can make the simulation shorter than the general simulation.

In [30] we introduced the space complexity measure for R(1). This new measure has been studied in several papers as for example. [2, 15, 29, 31]. In Section 3.1 we give general results for R(1) and R*(1) space that appeared mainly in [30]. In Section 3.2 a combinatorial characterization of R*(1) space is proved. This characterization makes easier the task of proving bounds for R*(1) space. As in the case of the width characterization in [7] it is also via a Player-Adversary game over CNF formulas. It would be interesting to find a combinatorial characterization for R(1) space.

In Section 3.3 we give the first space separation from R(1) to R*(1). We show that $\mathbb{PEB}_2^1(G)$ requires less space for R(1) than for R*(1), at least one third less. In Section 3.4 we show that, as happened with respect to size, R$^*(k)$ forms a hierachy respect to space. So, there are formulas that require nearly linear space for R$^*(k)$ whereas they have constant space R$^*(k+1)$ refutations. In Section 3.5 all known R(1) space lower bounds from [30, 2, 15] have been extended to R(k) in an simpler and unified way, that also holds for R(1), using

the concept of dynamical satisfiability from [29].

In the next table we list the bounds we have proved in this work among several related bounds. Note that this list is not intended to be complete, for example only R(k) and CP is mentioned. Its only purpose is to help to put in context this work. Citations in boldface appear in this work. To interpret the table note that not all bounds follows the O and Ω notation. For the bounds not following this notation, if nothing is said in the corresponding cell, we understand that we are referring to a lower bound. For example, the R(1) size bound for $\mathbb{PHP}_n^{n+1}$ is a lower bound and appeared in [38].

Last, we must recall some open problems related to this work. An interesting open problem for us, and also for Ben-Sasson [14] is the exact R(1) space complexity of $\mathbb{PEB}_2^1(G)$. We gave a nontrivial space upper bound [31] but we could not find a matching lower bound or prove a lower upper bound matching trivial space lower bounds for $\mathbb{PEB}_2^1(G)$. Our upper bound is the first space separation between R(1) and R*(1). In [31] a combinatorial characterization of R*(1) space is proved, similar to the width characterization in [7]. It will interesting to find a combinatorial characterization for R(1) space, which may help to solve the space complexity of $\mathbb{PEB}_2^1(G)$. The space separation in [31] shows that the characterization of R*(1) space is not valid for R(1) space. As $\mathbb{PEB}_2^1(G)$ has constant space R*(2) refutations using the dynamical satisfiability concept only a constant space lower bound can be proved, so if it happens that $\mathbb{PEB}_2^1(G)$ requires nonconstant space R(1) refutations, the dynamical satisfiability concept will not be a tight characterization of R(1) space.

Formulas		Proof System	Size	Space
$\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{COL}(\vec{p},\vec{r})$		CP*	$2^{\Omega(n^\epsilon)}$ **[18]**	
		R(1)	$n^{O(1)}$ **[18]**	
$\mathbb{GEN}(\vec{p},\vec{q}) \cup \mathbb{RCOL}(\vec{p},\vec{r})$		CP*	$2^{\Omega(n^\epsilon)}$ **[18]**	
		Regular R(1)	$n^{O(1)}$ **[18]**	
$\mathbb{PHP}$	$\mathbb{PHP}_n^{n+1}$	R(1)	$(1.49^{0.01})^n$ [38]	
		CP	$n^{O(1)}$ [28]	
	$\mathbb{PHP}_n^{cn}$	R(1)	$\frac{1}{2}(\frac{3}{2})^{n/50c}$ [25]	
		R(2)	$e^{n/\log^{14} n}$ [6]	
		R($\sqrt{\frac{\log n}{\log\log n}}$)	2^{n^ϵ} [63]	
	$\mathbb{PHP}_n^m$	R(1)	2^{n^ϵ} [56, 58]	$n+1$ [30, 2, 50] 1)
		R(k)		n/k **[29]**
$\mathbb{CLIQUE}_{k,k'}^m$		R(1)	$e^{\Omega(\log^2 m/\sqrt{\log\log m})}$ **[6]**	
		R(2)	$m^{O(1)}$ **[6]**	
$\mathbb{GT}_n$		R(1)	$n^{O(1)}$ [64]	$n/2$ [2]
		R(k)		$n/2k$ **[29]**
$\mathbb{PEB}(G)$	$\mathbb{PEB}_2^1(G)$	R*(1)	$2^{\Omega(n/\log n)}$ [16],**[31]**	$n/\log n - 2$ **[31]**
		R(1)	$O(n)$ [16]	
	$\mathbb{PEB}_2^1(T_n)$	R*(k)		$n-2$ [16], **[31]**
		R(k)		$2n/3+3$ **[31]** 2)
	$\mathbb{PEB}_{k+1}^k(G)$	R*(k)	$2^{\Omega(n/\log n)}$ **[29]**	$n/\log^2 n$ **[29]**
		R*($k+1$)	$O(n)$ **[29]**	$O(1)$ **[29]**
$\mathbb{T}(G)$		R(1)	$2^{\Omega(\lvert G\rvert)}$ [66]	$\lvert G\rvert$ [30, 2]
		R(k)		$\lvert G\rvert/k$ **[29]**
Random Formulas	$\mathbf{F}_{\Delta\cdot n}^{3,n}$	R(1)	$(1+\epsilon)^n$ [26]	$n\cdot\Delta^{\frac{1+\epsilon}{1-\epsilon}}$ [15]
	$\mathbf{F}_{5n}^{3,n}$	R(2)	$2^{\Omega(n^{1/3}/\log^2 n)}$ [6]	
	$\mathbf{F}_{\Delta\cdot n}^{3,n}$	R(k)		$n/k\cdot\Delta^{\frac{1+\epsilon}{1-\epsilon}}$ **[29]**

1) Exact bound 2) Upper bound

Bibliography

[1] R. Aharoni and N. Linial. Minimal unsatisfiable formulas and minimal non two-colorable hypergraphs. *Journal of Combinatorial Theory, Series A*, 43:196–204, 1986.

[2] M. Alekhnovich, E. Ben-Sasson, A.A. Razborov, and A. Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002.

[3] M. Alekhnovich, J. Johannsen, T. Pitassi, and A. Urquhart. An Exponential Separation between Regular and General Resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 448–456, New York, May 19–21 2002. ACM Press.

[4] A. Atserias. *The Complexity of Resource-Bounded Propositional Proofs*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 2002. Advisors: J.L Balcázar and M.L. Bonet.

[5] A. Atserias and M.L. Bonet. On the Automatizability of Resolution and Related Propositional Proof Systems. In *Computer Scince Logic, CSL 2002*, volume 2471 of *Lecture Notes in Computer Science*, pages 569–583, 2002.

[6] A. Atserias, M.L. Bonet, and J.L. Esteban. Lower Bounds for the Weak Pigeonhole Principle and Random Formulas beyond Resolution. *Information and Computation*, 176(2):136–152, 2002.

[7] A. Atserias and V. Dalmau. A Combinatorial Characterization of Resolution Width. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity (CCC-03)*, pages 239–247. IEEE Computer Society, 2003.

[8] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, New York, N.Y., 1988.

[9] S. Baumer, J.L. Esteban, and J. Torán. Minimally Unsatisfiable CNF Formulas. *Bulletin of the European Association for Theoretical Computer Science*, 74:190–, June 2001. Technical Contributions.

[10] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlák, and A. Woods. Exponential lower bounds for the Pigeonhole Principle. In ACM, editor, *Proceedings of the twenty-fourth annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 4–6, 1992*, pages 200–220, New York, NY, USA, 1992. ACM Press.

[11] P. Beame, R. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability proofs for random k-CNF formulas. In ACM, editor, *Proceedings of the thirtieth annual ACM Symposium on Theory of Computing: Dallas, Texas, May 23–26, 1998*, pages 561–571, New York, NY, USA, 1998. ACM Press.

[12] P. Beame, R. Karp, T. Pitassi, and M. Saks. The efficiency of Resolution and Davis-Putnam procedures. *SICOMP: SIAM Journal on Computing*, 31(4):1048–1075, 2002.

[13] P. Beame and T. Pitassi. Simplified and Improved Resolution Lower Bounds. In *37th Annual Symposium on Foundations of Computer Science*, pages 274–282, Burlington, Vermont, 14–16 October 1996. IEEE.

[14] E. Ben-Sasson. Size Space Tradeoffs for Resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 457–464, New York, May 19–21 2002. ACM Press.

[15] E. Ben-Sasson and N. Galesi. Space Complexity of Random Formulae in Resolution. In Frances M. Titsworth, editor, *Proceedings of the Sithteenth Annual Conference on Computational Complexity (CCC-01)*, pages 42–51, Los Alamitos, CA, June 18–21 2000. IEEE Computer Society. To appear in Random Structures and Algorithms.

[16] E. Ben-Sasson, R. Impagliazzo, and A. Wigderson. Near-Optimal Separation of Treelike and General Resolution. In *Electronic Colloquium on Computational Complexity, Report TR02-005*, 2000. To appear in Combinatorica.

[17] E. Ben-Sasson and A. Wigderson. Short proofs are narrow — Resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001.

[18] M.L. Bonet, J.L. Esteban, N. Galesi, and J. Johannsen. On the Relative Complexity of Resolution Refinements and Cutting Planes Proof Systems. *SIAM Journal on Computing*, 30(5):1462–1484, October 2001.

[19] M.L. Bonet and N. Galesi. Optimality of Size-Width Tradeoffs for Resolution. *Computational Complexity*, 10(4):261–276, 2001.

[20] M.L. Bonet, T. Pitassi, and R. Raz. Lower Bounds for Cutting Planes Proofs with Small Coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, September 1997.

[21] R.B. Boppana and N. Alon. The monotone Circuit Complexity of Boolean Functions. *Combinatorica*, 7(1):1–22, 1987.

[22] J. Buresh-Oppenheim, D. Mitchell, and T. Pitassi. Linear and Negative Resolution are Weaker than Resolution. Electronic Colloquium on Computational Complexity, Report TR01-074, 2001.

[23] S.R. Buss. Polynomial sized proofs of the Pigeonhole Principle. *Journal of Symbolic Logic*, 52(4):916–927, 1987.

[24] S.R. Buss and T. Pitassi. Resolution and the Weak Pigeonhole Principle. In *Computer Science Logic, 11th International Workshop, CSL'97*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156. Springer Verlag, Berlin, Heidelberg, New York., 1998.

[25] S.R. Buss and G. Turán. Resolution Proofs of Generalized Pigeonhole Principles. *Theoretical Computer Science*, 62(3):311–317, December 1988.

[26] V. Chvátal and E. Szemerédi. Many hard examples for Resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[27] S.A. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36ff, 1979.

[28] W. Cook, C.R. Coullard, and G. Turán. On the complexity of Cutting Planes proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.

[29] J.L. Esteban, N. Galesi, and J. Messner. On the Complexity of Resolution with Bounded Conjunctions. In *29th International Colloquium, ICALP 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 220–231, 2002.

[30] J.L. Esteban and J. Torán. Space Bounds for Resolution. *Information and Computation*, 171(1):84–97, 2001.

[31] J.L. Esteban and J. Torán. Combinatorial characterization of treelike Resolution space. *Information Processing Letters*, 87(6):295–300, 2003.

[32] X. Fu. Lower bounds on sizes of cutting planes proofs for modular coloring principles. In P. Beame and S.R. Buss, editors, *Proof Complexity and Feasible Arithmetics*, DIMACS Ser. Discrete Math. Theoret. Comput. Sc., pages 135–148. AMS, 99.

[33] N. Galesi. *On the Complexity of Propositional Proof Systems.* PhD thesis, Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 2000. Advisor M.L. Bonet.

[34] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to NP-Completeness.* W.H. Freeman and Company, San Francisco, California, 1979.

[35] A. Goerdt. Davis-Putnam Resolution versus unrestricted Resolution. *Annals of Mathematics and Artificial Intelligence*, 6:169–184, 1992.

[36] A. Goerdt. Unrestricted Resolution versus N-resolution. *Theoretical Computer Science*, 93(1):159–167, February 1992.

[37] A. Goerdt. Regular Resolution versus Unrestricted Resolution. *SIAM Journal on Computing*, 22(4):661–683, August 1993.

[38] A. Haken. The intractability of Resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[39] A. Haken and S.A. Cook. An Exponential Lower Bound for the Size of Monotone Real Circuits. *JCSS: Journal of Computer and System Sciences*, 58, 1999.

[40] R. Impagliazzo, T. Pitassi, and A. Urquhart. Upper and Lower Bounds for Tree-Like Cutting Planes Proofs. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 220–228, Paris, France, 4–7 July 1994. IEEE Computer Society Press.

[41] J. Johannsen. Lower bounds for monotone real circuit depth and formula size and tree-like Cutting Planes. *Information Processing Letters*, 67(1):37–41, 16 July 1998.

[42] S. Jukna. Combinatorics of monotone computations. *Combinatorica*, 19(1):65–85, 1999.

[43] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, May 1990.

[44] H. Kleine-Büning and Th. Lettmann. *Aussagenlogik: Deduktion und Algorithmen.* B.G. Teubner, Stuttgart, 1994.

[45] J. Krajíček. Lower Bounds to the Size of Constant-Depth Propositional Proofs. *The Journal of Symbolic Logic*, 59(1):73–86, March 1994.

[46] J. Krajíček. Interpolation Theorems, Lower Bounds for Proof Systems, and Independence Results for Bounded Arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.

[47] J. Krajíček. Interpolation by a Game. *Mathematical Logic Quarterly (formerly Zeitschrift für Mathematische Logik und Grundlagen der Mathematik)*, 44:450–458, 1998.

[48] J. Krajíček. On the Weak Pigeonhole Principle. *Fundamenta Mathematicae*, 170(1–3):123–140, 2001.

[49] A. Maciel, T. Pitassi, and A.R. Woods. A new proof of the Weak Pigeonhole Principle. In ACM, editor, *Proceedings of the thirty second annual ACM Symposium on Theory of Computing: Portland, Oregon, May 21–23, [2000]*, pages 368–377, New York, NY, USA, 2000. ACM Press.

[50] J. Messner. Space upper bound for the Pigeonhole Principle. Unpublished, 1999.

[51] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, New York, 1994.

[52] J.B Paris, A.J. Wilkie, and A.R. Woods. Provability of the Pigeonhole Principle and the Existence of Infinitely Many Primes. *JSL: Journal of Symbolic Logic*, 53, 1988.

[53] W.J. Paul, R.E. Tarjan, and J.R. Celoni. Space Bounds for a Game on Graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[54] P. Pudlák. Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations. *The Journal of Symbolic Logic*, 62(3):981–998, September 1997.

[55] P. Pudlák and R. Impagliazzo. A lower bound for DLL algorithms for k-SAT. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 128–136, N.Y., January 9–11 2000. ACM Press.

[56] R. Raz. Resolution Lower Bounds for the Weak Pigeonhole Principle. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 553–562, New York, May 19–21 2002. ACM Press.

[57] R. Raz and P. McKenzie. Separation of the Monotone NC Hierarchy. *Combinatorica*, 19(3):403–435, 1999.

[58] A.A. Razborov. Improved Resolution Lower Bounds for the Weak Pigeonhole Principle. In *Electronic Colloquium on Computational Complexity, TR01-055*, 2001.

[59] A.A. Razborov. Proof Complexity of Pigeonhole Principles. In *5th International Conference on Developments in Language Theory*, 2001.

[60] J.A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[61] A. Rosenbloom. Monotone real circuits are more powerful than monotone Boolean circuits. *Information Processing Letters*, 61(3):161–164, March 1997.

[62] J. Savage. *Models of Computation.* Addison-Wesley, 1998.

[63] N. Segerlind, S. Buss, and R. Impagliazzo. A Switching Lemma for Small Restrictions and Lower Bounds for k-DNF Resolution (extended abstract). In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[64] G. Stålmarck. Short Resolution Proofs for a Sequence of Tricky Formulas. *Acta Informatica*, 33(3):277–280, 1996.

[65] G.S. Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part 2.*, pages 115–125. Consultants Bureau, 1968.

[66] A. Urquhart. Hard Examples for Resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

[67] A. Urquhart. The Complexity of Propositional Proofs. *Bulletin of Symbolic Logic*, 1(4):425–467, 1995.

[68] P. van Emde-Boas and J. van Leeuwen. Move rules and trade-offs in the pebble game. *Lecture Notes in Computer Science*, 67:101–112, 1979.