A THEORETICAL COMPARISON OF RESOLUTION PROOF SYSTEMS FOR CSP ALGORITHMS

by

Cho Yee Joey Hwang

B.Sc., Simon Fraser University, 2001

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Fall 2004

APPROVAL

Name: Cho Yee Joey Hwang

Degree: Master of Science

Title of thesis: A Theoretical Comparison of Resolution Proof Systems for CSP Algorithms

Examining Committee: Dr. Ramesh Krishnamurti
Chair

Dr. David G. Mitchell
Senior Supervisor

Dr. Arthur L. Liestman
Supervisor

Dr. Andrei A. Bulatov
SFU Examiner

Date Approved: December 2, 2004

Abstract

Many problems from a variety of applications such as graph coloring and circuit design can be modelled as constraint satisfaction problems (CSPs). This provides strong motivation to develop effective algorithms for CSPs. In this thesis, we study two resolution-based proof systems, *NG-RES* and *C-RES*, for finite-domain CSPs which have a close connection to common CSP algorithms. We give an almost complete characterization of the relative power among the systems and their restricted tree-like variants. We demonstrate an exponential separation between *NG-RES* and *C-RES*, improving on the previous super-polynomial separation, and present other new separations and simulations. We also show that most of the separations are nearly optimal. One immediate consequence of our results is that simple backtracking with 2-way branching is exponentially more powerful than simple backtracking with d-way branching.

To my parents

Acknowledgments

I would like to take this opportunity to extend my gratitude to my senior supervisor, Dr. David Mitchell, for introducing me to the field of constraint satisfaction and providing tremendous guidance, support, and feedback at every stage of this thesis work. I would also like to thank my supervisor, Dr. Art Liestman, and my examiner, Dr. Andrei Bulatov, for their valuable comments which greatly improved the quality of this thesis. I am grateful to Dr. Michael Monagan, Dr. Bob Russell, and Dr. Wo-shun Luk for encouraging me to pursue graduate studies. Last but not least, I want to thank my family and friends for their continuous support and encouragement.

Contents

List of Figures

Chapter 1

Introduction

Constraint satisfaction problems (CSPs) involve finding values for a finite set of variables satisfying all of a given set of constraints between the variables. They are widely used to encode problems such as planning, scheduling, graph coloring, and circuit design. The satisfiability problem (SAT) for propositional formulas in conjunctive normal form (CNF) can also be viewed as a CSP in which variables can take values from the domain $\{0, 1\}$. The importance of these applications provides strong motivation to develop efficient algorithms to solve CSPs. A natural approach to search for a solution in practice is backtracking. Indeed, most studies on CSP algorithms and most commonly used CSP solvers are based on backtracking. A considerable amount of work has been done on the study of enhanced versions of backtracking and their empirical effectiveness. Our work was motivated by improving our understanding of the relative efficiency and limitations of standard backtracking-based CSP algorithms.

We compare the relative power of such algorithms in terms of how efficiently they can refute an unsatisfiable CSP instance in the optimal case. When running a backtracking algorithm on an unsatisfiable instance, a trace of an execution is a "proof" which may convince observers of the unsatisfiability of the instance. This establishes a close connection between backtracking algorithms and proof systems (a.k.a. refutation systems). We will consider two resolution-based proof systems for CSPs and state how they are related to standard algorithms. We then examine the relative power of the systems.

Since we are going to study proof systems and refutations, we will restrict our attention to unsatisfiable CSP instances. Note that every complete backtracking algorithm has to handle unsatisfiable instances and, as we will state, lower bounds on refutation size for unsatisfiable instances give lower bounds on the execution time of those algorithms on the instances.

1.1 Constraint Satisfaction and CSP Algorithms

A CSP instance consists of a set of variables and a set of constraints. Each variable has a finite domain and each constraint limits the values that can be taken simultaneously by some specified subsets of the variables. The problem is to find an assignment of values to all the variables such that all the constraints are satisfied, or to determine that there is no such assignment.

A straightforward approach to solve CSPs is backtracking. There are two main schemes for backtracking algorithms: backtracking with d-way branching and backtracking with 2-way branching. A backtracking algorithm with d-way branching works as follows. For a CSP instance $\mathcal{I}$, the algorithm picks a variable x and for each domain value a of x, a recursive call is made to solve $\mathcal{I}$ with x set to a. If the domain size of x is d and all d recursive calls fail, then $\mathcal{I}$ is unsatisfiable. Backtracking algorithms with 2-way branching, on the other hand, follow a different procedure. For $\mathcal{I}$ a CSP instance, a 2-way branching algorithm selects a variable x and a value a from x's current domain. Then, two recursive calls are made: one with x set to a and the other with a removed from the domain of x. $\mathcal{I}$ is unsatisfiable if both of these two recursive calls fail. Figures 1.1 and 1.2 illustrate the ideas graphically.

It is not hard to see that any d-way branching strategy can be simulated by a 2-way branching strategy. If, at some point of the search, a d-way branching algorithm chooses to branch on variable x, then the corresponding 2-way branching algorithm will just keep branching on x until the domain of x becomes empty. For example, if a d-way branching backtracking algorithm constructs a search tree as shown on the left of Figure 1.3, then the corresponding search tree generated by a 2-way branching algorithm simulating the d-way branching algorithm will look like the one on the right

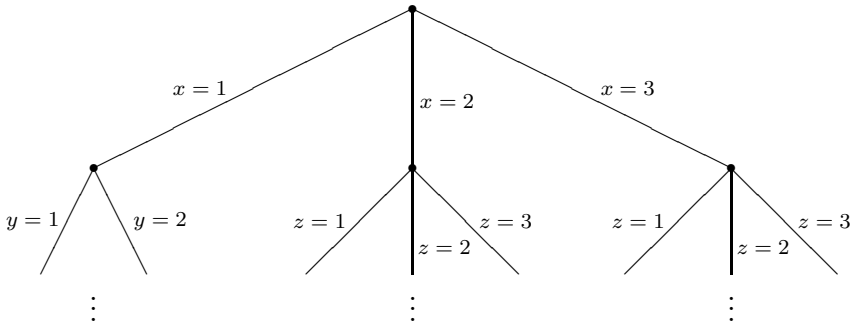

Figure 1.1: A d-way branching search tree

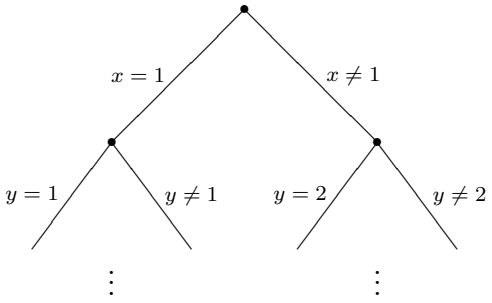

Figure 1.2: A 2-way branching search tree

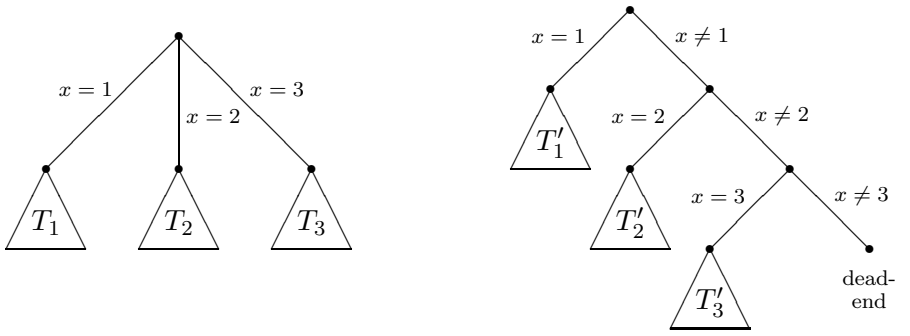

Figure 1.3: Simulation of d-way branching by 2-way branching

of the figure, where T'_i is the search tree simulating T_i. Therefore, if a CSP instance $\mathcal{I}$ can be solved by a d-way branching algorithm in t steps, then there is a 2-way branching algorithm that can solve $\mathcal{I}$ in $O(t)$ steps. However, the converse does not hold. We will show that there are unsatisfiable CSP instances for which every search tree formed by a d-way branching algorithm is exponentially larger than the smallest search tree constructed by a 2-way branching algorithm.

Although we do not expect to find a general backtracking algorithm that can solve all CSP instances in polynomial time, as CSPs are in the class of NP-complete problems, refining backtracking algorithms can improve their performance greatly. One technique to improve backtracking is learning. When an algorithm backtracks from some dead-end during search, it can record some explicit information and re-use it to prune duplicate searches later. For example, somewhere during the search, it may become explicit that x and y cannot both take value 1, although this is not explicitly restricted by the constraint set. If the algorithm chooses to cache this information, then next time x and y are set to 1, it can immediately conclude that this is a dead-end and backtrack. Studies on learning strategies usually focus on what to learn and how much information to store. Our work here was partially motivated by the question of how much power an algorithm can gain when enhanced with learning.

We are also interested in the relative power of 2-way branching with learning and

d-way branching with learning. The structure of 2-way branching search trees allows 2-way branching algorithms to learn more specific information than that d-way branching algorithms can learn. This makes 2-way branching more powerful than d-way branching when learning is involved. Mitchell [27] has shown that there is an infinite family of unsatisfiable CSP instances MPH_n such that any d-way branching algorithm, even with optimal variable ordering and optimal use of learning strategies, cannot solve MPH_n in less than $n^{\Omega(\log n)}$ time. But, there is a 2-way branching algorithm, with specific variable ordering and learning strategies, that can solve MPH_n in $O(n^3)$ time. Therefore, 2-way branching with learning is strictly more powerful than d-way branching with learning and there is a super-polynomial separation between them. However, is super-polynomial an upper bound for the separation? Or, does there exist an exponential separation between them? Although we do not answer this question in this thesis, we make a contribution toward finding the answer by characterizing the power of proof systems which are closely connected to the reasoning of the branching schemes.

1.2 Resolution

Since we are studying resolution-based proof systems here, we recall the resolution proof system first. The SAT problem involves finding (the existence of) a truth assignment α for the variables in a CNF formula ϕ such that α satisfies all clauses in ϕ. Propositional resolution, or simply *resolution*, is a proof system for SAT. The resolution rule allows us to derive the new clause $A \vee B$ if we already have the clauses $x \vee A$ and $\overline{x} \vee B$. The derivation step is sound because if a truth assignment α satisfies both $x \vee A$ and $\overline{x} \vee B$, then at least one of A and B must be satisfied by α since x is either true or false. Thus, $A \vee B$ must be satisfied by α as well. A resolution derivation from a CNF formula ϕ is a sequence of clauses in which each clause is either in ϕ or derived from previous clauses in the sequence. If we can derive the empty clause from a set of clauses ϕ, then the derivation is a proof that ϕ is unsatisfiable because any assignment that satisfies ϕ must satisfy all clauses derived from ϕ, but the empty clause is tautologically false. We call such a resolution derivation a refutation. In

fact, the resolution proof system is sound and complete. That is, a CNF formula ϕ is unsatisfiable if and only if there exists a resolution refutation of ϕ.

We say that a resolution derivation π is tree-like if every derived clause in it is used at most once to derive another clause. That is, to use a derived clause C a second time, C must be derived again from the initial clauses. The size of π is the number of clauses in π. It is well-known that running a backtracking algorithm for SAT (a.k.a. a DLL algorithm) on an unsatisfiable CNF formula implicitly constructs a tree-like resolution refutation of ϕ. Given a backtracking search tree of an unsatisfiable CNF formula ϕ, if we label each leaf with a clause in ϕ which is falsified by the assignment defined on the path leading to that leaf and label each internal node with the clause derived by resolving the clauses labelling its children, then the root of the search tree will be labelled with the empty clause. Moreover, given a tree-like resolution refutation of a formula, if a backtracking algorithm follows a variable branching ordering corresponding to the refutation (e.g., if x and $\overline{x}$ are resolved together to derive the empty clause in the refutation, then the first variable picked by the algorithm to branch on will be x), then the search tree will have the same size as the refutation. Hence, the smallest tree-like resolution refutation of a formula ϕ is of the same size as the smallest search tree constructed by a DLL algorithm on ϕ. Thus, as systems to refute unsatisfiable CNF formulas, DLL algorithms and tree-like resolution have the same power. This provides us an approach to analyze the efficiency of DLL algorithms. For example, there are formulas for which every tree-like resolution refutation is of exponential size. So, no DLL algorithm can refute them in less than exponential time even with an optimal branching strategy.

Unrestricted resolution, unlike tree-like resolution, allows derived clauses to be used arbitrarily many times to derive other clauses. It is known that unrestricted resolution is exponentially stronger than tree-like resolution [9, 7]. The most effective current complete SAT solvers enhance DLL algorithms with clause learning which helps avoid redundant search with the use of learned clauses. This makes DLL algorithms more powerful than tree-like resolution. Several researchers, e.g., Moskewicz et al. [29] and Zhang et al. [38], showed that clause learning, with efficient implementation, can handle problems that are hard for other standard techniques. The idea

of learning can be viewed as the re-use of derived clauses in unrestricted resolution refutations. Beame et al. [6] have already shown that DLL algorithms with clause learning and unlimited *restarts*, which allow the algorithms to restart their searching process anytime, is equivalent to unrestricted resolution. However, it is still unclear if clause learning with no or limited restarts is also as powerful as unrestricted resolution or not.

Our main interest here is to study resolution-based proof systems for CSPs and their relative complexity, with the belief that the results will provide useful insight into CSP algorithms.

1.3 Resolution-based CSP Proof Systems

Baker [4] extended resolution to a more general resolution-based proof system for CSPs. The expressions used in this system are *nogoods*, instead of clauses. Here, we call this system *nogood resolution* (*NG-RES*). Given a CSP instance $\mathcal{I}$, for each forbidden value combination of a set of variables $x_1, x_2, \cdots, x_t$, we have a nogood

$$\neg(x_1 = a_1 \wedge x_2 = a_2 \wedge \cdots \wedge x_t = a_t)$$

which intuitively disallows any corresponding partial assignment. (Later, we will simplify the notation by writing nogoods in the form $\eta(x_1 = a_1, \cdots, x_t = a_t)$.) If the domain of a variable x is $\{1, \cdots, d\}$ and we already have the nogoods

$$\begin{gathered}\neg(x = 1 \wedge X_1)\\ \neg(x = 2 \wedge X_2)\\ \vdots\\ \neg(x = d \wedge X_d)\end{gathered}$$

then the nogood resolution rule allows us to resolve them together and soundly derive

$$\neg(X_1 \wedge X_2 \wedge \cdots \wedge X_d).$$

An assignment that satisfies all the d antecedent nogoods (i.e., is not an extension of a partial assignment forbidden by one of those nogoods), also satisfies the derived nogood.

An *NG-RES* refutation of a CSP instance $\mathcal{I}$ is an *NG-RES* derivation of the empty nogood from the set of nogoods corresponding to the partial assignments forbidden by the constraints of $\mathcal{I}$. There is an *NG-RES* refutation of $\mathcal{I}$ if and only if $\mathcal{I}$ is unsatisfiable. The size of an *NG-RES* refutation is the number of nogoods in it. Applying arguments similar to those in the previous section, we can show that for an unsatisfiable CSP instance $\mathcal{I}$, the size of the smallest tree-like *NG-RES* refutation of $\mathcal{I}$ is exactly the same as the minimum number of steps that a d-way branching backtracking algorithm requires to refute $\mathcal{I}$. Baker showed some correspondences between *NG-RES* and d-way branching backtracking algorithms enhanced with backjumping [31] and dynamic backtracking [20], and used *NG-RES* as a tool to analyze those algorithms.

Later, Mitchell [25] extended Baker's work and introduced another CSP proof system, *constraint resolution* (or *C-RES* for short), corresponding to 2-way branching algorithms. A *C-RES* refutation for a CSP instance $\mathcal{I}$ is essentially a resolution refutation of the CNF encoding of $\mathcal{I}$. The CNF encoding used is a natural transformation of CSPs to CNF formulas suggested by de Kleer in [14]. With this encoding, *C-RES* can model the reasoning of 2-way branching. Furthermore, tree-like *C-RES* and simple 2-way branching algorithms have equivalent power, in the sense that the smallest tree-like *C-RES* refutation of any unsatisfiable instance $\mathcal{I}$ is of the same size as the smallest search tree constructed by a 2-way branching backtracking algorithm on $\mathcal{I}$.

We say that a proof system $\boldsymbol{A}$ *efficiently simulates* a proof system $\boldsymbol{B}$ if any $\boldsymbol{B}$ refutation of a CSP instance $\mathcal{I}$ can be transformed into an $\boldsymbol{A}$ refutation of $\mathcal{I}$ with only a polynomial blowup in size. There is an *exponential separation* of system $\boldsymbol{B}$ from system $\boldsymbol{A}$ if there is an infinite set of instances $\{\mathcal{I}_1, \mathcal{I}_2, \cdots\}$ such that the smallest $\boldsymbol{B}$ refutation of $\mathcal{I}_n$ is of size exponential in n, but the smallest $\boldsymbol{A}$ refutation of $\mathcal{I}_n$ is of size polynomial in n. If $\boldsymbol{A}$ efficiently simulates $\boldsymbol{B}$ and there is an exponential separation of $\boldsymbol{B}$ from $\boldsymbol{A}$, then $\boldsymbol{A}$ is exponentially more powerful than $\boldsymbol{B}$. Obviously, *NG-RES* efficiently simulates tree-like *NG-RES* and *C-RES* efficiently simulates tree-like *C-RES*.

Mitchell has already proven that *C-RES* is strictly more powerful than *NG-RES*. In particular, he showed that there are *C-RES* refutations of size $O(n^3)$ of the CSP

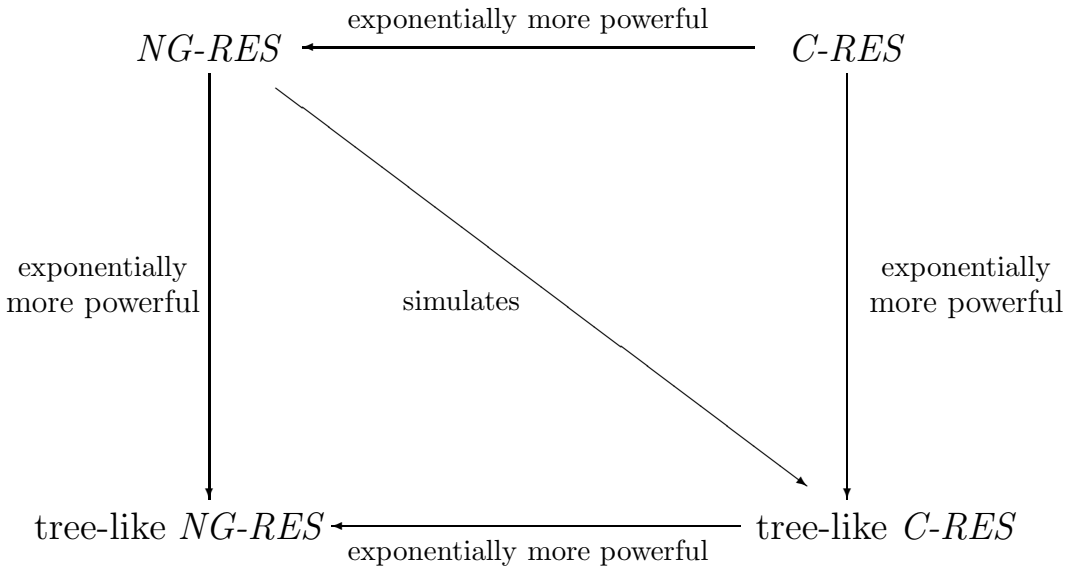

Figure 1.4: Relative Efficiency of *NG-RES*, *C-RES* and their tree-like variants

instance MPH_n but every *NG-RES* refutation of MPH_n must be of size $n^{\Omega(\log n)}$. The instance is based on the one that Goerdt [21] used to obtain an $n^{\Omega(\log n)}$ separation between resolution and negative resolution. To the best of our knowledge, no better separation has been shown in the literature.

Our work here mainly focuses on examining the relative power of *NG-RES*, *C-RES* and their tree-like versions. We present simulations and new separations between the systems. Moreover, we show upper bounds of the separations as well. All these together constitute an almost complete picture of the relationships between the systems. Figure 1.4 illustrates the relative power of the proof systems and summarizes the separation and simulation results we present.

1.4 Proof Systems and CSP Algorithms

We already claimed that tree-like *NG-RES* and d-way branching have equivalent power, and similarly, tree-like *C-RES* and 2-way branching have the same power. Now we introduce a term, *bounded*, for comparing the relative power of an algorithm and a proof system. Given an algorithm $\boldsymbol{A}$ and a proof system $\boldsymbol{P}$, if an execution trace of $\boldsymbol{A}$ on any unsatisfiable instance is at least as large as the size of the smallest $\boldsymbol{P}$ refutation of the instance, then $\boldsymbol{A}$ is $\boldsymbol{P}$ bounded. For example, a DLL algorithm for SAT is tree-like resolution bounded.

In addition to learning strategies [35, 18] we mentioned earlier, other standard techniques used in common CSP algorithms include: variable ordering heuristics [3, 19], backjumping [15, 16, 31], forward checking [34, 17], arc-consistency filtering [32, 8], k-consistency enforcement [23], and their variants.

In [27], Mitchell showed that

1. d-way branching backtracking algorithms with the use of any combination of variable ordering heuristics, backjumping and forward checking are tree-like *NG-RES* bounded.

2. d-way branching backtracking algorithms with the use of any combination of variable ordering heuristics, backjumping, forward checking, arc-consistency filtering, k-consistency enforcement and learning are *NG-RES* bounded.
   (Tree-like *NG-RES* bounded algorithms are also *NG-RES* bounded.)

3. 2-way branching backtracking algorithms with the use of any combination of variable ordering heuristics, backjumping and forward checking are tree-like *C-RES* bounded.

4. 2-way branching backtracking algorithms with the use of any combination of variable ordering heuristics, backjumping, forward checking, arc-consistency filtering, k-consistency enforcement and learning are *C-RES* bounded.
   (Tree-like *C-RES* bounded algorithms are also *C-RES* bounded.)

Hence, the power of a proof system provides significant insight into the efficiency of the algorithms bounded by the system. The size of the smallest $\boldsymbol{P}$ refutation of a CSP instance $\mathcal{I}$ gives a lower bound on the running time of any implementation of the algorithms bounded by $\boldsymbol{P}$ on $\mathcal{I}$.

Our results show that there are CSP instances for which every *NG-RES* refutation is of exponential size. By the second point above, no *NG-RES* bounded algorithm, including d-way branching algorithms enhanced with standard techniques, can solve the instances in less than exponential time. Since tree-like *NG-RES* bounded algorithms are also *NG-RES* bounded and *NG-RES* can efficiently simulate tree-like *C-RES*, the instances are also exponentially hard for tree-like *NG-RES* bounded algorithms and tree-like *C-RES* bounded algorithms.

From the exponential separation we obtained between tree-like *NG-RES* and tree-like *C-RES* and the fact that tree-like *C-RES* and 2-way branching have the same power, we know that 2-way branching is exponentially more powerful than d-way branching. The instances that separate tree-like *NG-RES* from tree-like *C-RES* cannot be solved by any tree-like *NG-RES* bounded algorithm, including d-way branching with backjumping and forward checking, in less than exponential time, but a 2-way branching algorithm can solve the instances in polynomial time with optimal branching choices. We do not provide here a poly-time computable branching strategy under which 2-way branching solves the instances we use in polynomial time, but we believe that such a strategy exists.

We do not have enough information to make any claim yet about the relative power of enhanced versions of backtracking algorithms from the other exponential separations we obtain. For example, although we know that the instances exponentially separating *NG-RES* from *C-RES* must be exponentially hard for *NG-RES* bounded algorithms, we are still not sure if there exists a 2-way branching algorithm, possibly enhanced with learning and other standard techniques, that can solve the hard instances in polynomial time. The problem here is that there may exist a short refutation but no polynomial time strategy which finds such a refutation. Our results, however, suggest possible instances that may be useful in future studies in separating the algorithms.

1.5 Summary of Results

Below is a summary of the results we present in this thesis.

1. Exponential separation of tree-like *NG-RES* from *NG-RES*.
   We modify the CNF formulas used by Ben-Sasson in [7] to separate tree-like resolution from resolution and adapt his proof method to obtain the separation between tree-like *NG-RES* and *NG-RES*.

2. Exponential separation of tree-like *C-RES* from *C-RES*.
   This separation follows from the separation between tree-like resolution and resolution [7], and some properties of *C-RES*.

3. Exponential separation of tree-like *NG-RES* from tree-like *C-RES*.
   We use the same family of CSP instances that we used to separate tree-like *NG-RES* from *NG-RES* to separate tree-like *NG-RES* from tree-like *C-RES* by explicitly constructing poly-size tree-like *C-RES* refutations of the instances.

4. Exponential separation of *NG-RES* from *C-RES*.
   We construct an infinite family of CSP instances and show that the instances have poly-size *C-RES* refutations but any *NG-RES* refutation of them is of exponential size. This improves the previous bound of $n^{\Omega(\log n)}$ from [25]. The proof technique was inspired by [11].

5. *NG-RES* simulation of tree-like *C-RES*.
   We prove this by showing how we can transform a tree-like *C-RES* refutation into an *NG-RES* refutation with only a polynomial blowup in size.

6. Separation upper bounds.
   Applying the same technique used to obtain an upper bound of the separation between tree-like resolution and resolution in [7], we can show an upper limit of the separation between tree-like *NG-RES* and *NG-RES*. Then, with these two bounds, plus simulations among the systems, we can prove the other separation upper bounds.

1.6 Related Work

Our work here concentrates on finding CSP instances that are hard for *NG-RES* but have poly-size *C-RES* refutations. There exist, however, many hard instances for *C-RES* (which are also hard for CSP algorithms). A number of hard instances are studied in [24]. Moreover, resolution complexity on random CSPs has been studied in [26, 28, 2, 5].

We compare the relative power of CSP backtracking algorithms by examining the relative efficiency of the resolution-based proof systems modelling the reasoning of the algorithms. In propositional logic, resolution is useful in studying the efficiency of backtracking-based SAT solvers. Beame et al. [6] analyzed the power of clause learning by characterizing the technique as the resolution proof system. They compared the relative power of DLL algorithms and their variants enhanced with clause learning by analyzing the power of their corresponding restricted resolution systems.

There have been only a few empirical studies on 2-way and d-way branching strategies. Park [30] showed that in most cases, with the variable and value ordering heuristics used in his work, 2-way branching ends up simulating d-way branching.

Barbara and Sturdy [36] investigated the effect of changing the value ordering in 2-way branching. They also compared 2-way branching with d-way branching and the experimental results indicated that 2-way branching, even with the worst value ordering, is not worse than d-way branching.

1.7 Thesis Organization

The remainder of this thesis is organized as follows. We formally define constraint satisfaction problems in Chapter 2. In Chapter 3, we introduce two resolution-based proof systems for CSPs and also explore the relative power between them and their restricted versions. We prove exponential separations between the two proof systems in Chapter 4. Chapter 5 contains conclusions and some potential future work.

Chapter 2

Constraint Satisfaction Problems

An instance of constraint satisfaction problem (CSP) consists of a set of variables and a set of constraints. Each variable has a finite domain and each constraint restricts the values that can be assigned simultaneously to some specific subset of the variables. An *assignment* for a CSP instance $\mathcal{I}$ is a function that assigns domain values to some variables in $\mathcal{I}$. A *total* assignment is an assignment that assigns values to all the variables. Given a CSP instance $\mathcal{I}$, a *solution* is a total assignment for $\mathcal{I}$ such that all the constraints in $\mathcal{I}$ are satisfied. A CSP instance is *unsatisfiable* if there is no such solution.

Example 2.1. Let $\mathcal{I}$ be a CSP instance containing three variables $\{x, y, z\}$, each with domain $\{1, 2, 3\}$, and constraints expressing that x, y and z must be assigned distinct values and they cannot take value 3. Obviously, $\mathcal{I}$ is unsatisfiable.

Conventionally, constraints are represented as relations, each indicating the allowed value combinations of certain variables. But for technical convenience, we represent constraints as the set of combinations of values that are disallowed. Each forbidden combination is essentially a partial assignment that cannot be extended to a solution. We can write such partial assignments as *nogoods*, which by their name denotes the meaning "not a good assignment". Here we define nogoods formally.

Definition 2.2 (literal, nogood, subnogood). A *literal* is an expression of the form $x = a$, where x is a variable and a is a domain value of x, asserting that x takes

value a. A *nogood* is a set of literals in which no variable can appear in more than one literal. We write a nogood as $\eta(x_1 = a_1, x_2 = a_2, \cdots, x_t = a_t)$. A nogood N_s is a *subnogood* of a nogood N if every literal in N_s also appears in N. The empty nogood is denoted by $\square$, which is tautologically false.

In the CSP literature, nogoods are sometimes written as $\neg(x_1 = a_1 \ \wedge \ x_2 = a_2 \ \wedge \ \cdots \ \wedge \ x_t = a_t)$ which expresses the semantics directly.

Now we can represent the constraints of a CSP instance as a set of nogoods.

Definition 2.3 (CSP instance). A CSP instance $\mathcal{I}$ is a triple $\langle X, \mathcal{D}, \Gamma \rangle$ where X is a finite set of variables, $\mathcal{D}(x)$ is the domain of a variable $x \in X$, and Γ is a set of nogoods indicating which value combinations of variables are disallowed.

Example 2.4. The CSP instance in Example 2.1 can be defined as $\mathcal{I} = \langle X, \mathcal{D}, \Gamma \rangle$ where $X = \{x, y, z\}$, $\mathcal{D}(v) = \{1, 2, 3\}$ for all $v \in X$, and $\Gamma = \{\eta(x = 1, y = 1), \eta(x = 2, y = 2), \eta(x = 3, y = 3), \eta(y = 1, z = 1), \eta(y = 2, z = 2), \eta(y = 3, z = 3), \eta(x = 1, z = 1), \eta(x = 2, z = 2), \eta(x = 3, z = 3), \eta(x = 3), \eta(y = 3), \eta(z = 3)\}$.

Next, we describe in what conditions a nogood is satisfied by an assignment.

Definition 2.5 (satisfies). Let $\mathcal{I} = \langle X, \mathcal{D}, \Gamma \rangle$ be a CSP instance and α be an assignment for $\mathcal{I}$. α *satisfies* a nogood N if and only if there is some literal $(x = a) \in N$ such that α assigns b to x, for some $b \neq a$. That is, to satisfy a nogood $\eta(x_1 = a_1, \cdots, x_t = a_t)$, α cannot simultaneously assign a_i to x_i for all $i \in \{1, \cdots, t\}$. α satisfies Γ if and only if it satisfies all nogoods in Γ. $\mathcal{I}$ is *satisfiable* if and only if there is a total assignment for $\mathcal{I}$ which satisfies Γ. If $\mathcal{I}$ is not satisfiable, then it is *unsatisfiable*.

Definition 2.6 (width). The *width* of a nogood N, $w(N)$, is the number of literals in it. The width of a CSP instance $\mathcal{I} = \langle X, \mathcal{D}, \Gamma \rangle$ is the width of the widest nogood in Γ.

Definition 2.7 (*vars*). Let $\mathcal{I}$ be a CSP instance, Γ be a set of nogoods and N be a nogood. We define *vars*($\mathcal{I}$), *vars*(Γ) and *vars*(N) be the sets of variables occurring in $\mathcal{I}$, Γ and N, respectively.

From now on, we will write $\mathcal{I} = \langle X, \mathcal{D}, \Gamma \rangle$ as $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$. We will use $vars(\mathcal{I})$ when we want to refer to the variables in $\mathcal{I}$. When all the variables in $\mathcal{I}$ have the same domain D, we will simply write $\mathcal{I} = \langle D, \Gamma \rangle$. Moreover, most of the time, we will use $[d] \stackrel{def}{=} \{1, \cdots, d\}$ to denote domains with size d.

Chapter 3

Resolution-based Proof Systems for CSP

In this chapter, we define propositional resolution, and two resolution-based proof systems, *NG-RES* and *C-RES*, for constraint satisfaction. We also introduce the concept of resolution complexity and for each of the systems *NG-RES* and *C-RES*, we examine the relative power of the system and its restricted tree-like version.

3.1 Preliminaries

Before defining proof systems for CSPs, we first consider a simple well-known proof system, propositional resolution, for propositional logic. We will state some basic facts about propositional resolution without giving proofs for them. When we move on to *NG-RES* for CSPs, we will prove theorems analogous to these facts.

A *propositional variable* is a boolean variable and a *literal* is either a propositional variable (denoted as x or x^1) or its negation (denoted as $\overline{x}$ or x^0). A *clause* is a set of literals and is viewed as a disjunction of its literals. We write a clause as $(l_1 \ \ l_2 \ \cdots \ l_t)$ where $l_1, l_2, \cdots, l_t$ are the literals in it. A CNF formula is a conjunction of clauses. We say that a CNF formula ϕ is *satisfiable* if there exists a truth assignment to the variables in ϕ that sets ϕ to 1. If there is no such truth assignment, then ϕ is *unsatisfiable*.

Definition 3.1 (propositional resolution). *Propositional resolution*, or simply *resolution*, is a proof system for CNF formulas in which one can derive new clauses by applying the *resolution rule*

$$\frac{(C \;\; x) \quad (D \;\; \overline{x})}{(C \;\; D)}$$

where C and D are arbitrary clauses and x is a variable. The rule allows us to derive $(C \;\; D)$ by *resolving* $(C \;\; x)$ and $(D \;\; \overline{x})$ *on* x. $(C \;\; D)$ is called the *resolvent* of $(C \;\; x)$ and $(D \;\; \overline{x})$, the *premises*, on x.

A *resolution derivation* of a clause C from a CNF formula ϕ is a sequence of clauses $C_1, C_2, \cdots, C_m$ in which each C_i is either a clause in ϕ or is derived from previous clauses in the sequence by the resolution rule and $C_m = C$. A *resolution refutation* of ϕ is a resolution derivation of the empty clause, denoted $\square$, from ϕ. We denote this system by *RES*.

The resolution rule is sound because a variable x must take either 0 or 1. So, if a truth assignment α satisfies both $(C \;\; x)$ and $(D \;\; \overline{x})$, then α must satisfy C if it sets x to 0 and satisfy D if it sets x to 1. Thus, α must also satisfy $(C \;\; D)$. In fact, the refutation system *RES* is sound and complete. That is, for any CNF formula ϕ, there is a *RES* refutation of ϕ if and only if ϕ is unsatisfiable.

Definition 3.2 (tree-like resolution derivation, *tree-RES*). A *tree-like resolution* (denoted *tree-RES*) derivation is a *RES* derivation in which every derived clause is used at most once as a premise to derive other clauses.

Definition 3.3 (size, width). The *size* of a *RES* derivation π, $|\pi|$, is the number of clauses in π. The *width* of a clause C, $w(C)$, is the number of literals appearing in C and the width of a *RES* derivation π, $w(\pi)$, is the width of the widest clause in π.

Definition 3.4 (resolution complexity, *RES*(ϕ) and *tree-RES*(ϕ)). For any unsatisfiable CNF formula ϕ,

$$RES(\phi) \stackrel{\text{def}}{=} \min\{|\pi| : \pi \text{ is a } RES \text{ refutation of } \phi\}$$

and

$$\textit{tree-RES}(\phi) \stackrel{\text{def}}{=} \min\{|\pi| : \pi \text{ is a } \textit{tree-RES} \text{ refutation of } \phi\}.$$

To help make it simpler in proving refutation related theorems, we allow ourselves to use the *weakening rule*

$$\frac{(C)}{(C \quad D)}$$

in addition to the resolution rule in a resolution derivation. The weakening rule does not strengthen the system and we can always eliminate the use of the weakening rule in a resolution refutation without increasing the size or width of the refutation.

Proposition 3.5. *For any CNF formula ϕ, if π is a RES (tree-RES resp.) refutation of ϕ using the resolution rule and the weakening rule, then π can be transformed into a RES (tree-RES resp.) refutation π' of ϕ such that $|\pi'| \leq |\pi|$, $w(\pi') \leq w(\pi)$, and π' makes use of the resolution rule only.*

Definition 3.6 (unit assignment). A *unit assignment* for a CNF formula ϕ sets a variable x in ϕ to a truth value $a \in \{0, 1\}$. Let ρ be a unit assignment setting x to a. For C a clause, the result of applying ρ to C is denoted $C\lceil_\rho$ and is defined to be

$$C\lceil_\rho \equiv C\lceil_{x=a} \stackrel{def}{=} \begin{cases} 1 & \text{if the literal } x^a \text{ appears in } C \\ C \setminus \{x^{1-a}\} & \text{otherwise} \end{cases}$$

For a CNF formula ϕ, $\phi\lceil_\rho \stackrel{def}{=} \{C\lceil_\rho : C \in \phi\}$. If $\pi = (C_1, C_2, \cdots, C_n)$ is a *RES* derivation, define $\pi\lceil_\rho$ to be $(C_1\lceil_\rho, C_2\lceil_\rho, \cdots, C_n\lceil_\rho)$, but with any $C_i\lceil_\rho$ that is identical to 1 removed.

Proposition 3.7. *For any CNF formula ϕ, if π is a RES (tree-RES resp.) derivation of C from ϕ and ρ is a unit assignment for ϕ, then $\pi\lceil_\rho$ is a RES (tree-RES resp.) derivation of $C\lceil_\rho$ from $\phi\lceil_\rho$ using the resolution rule and possibly also the weakening rule.*

Proposition 3.8. *For ϕ a CNF formula, x a variable, and $a \in \{0, 1\}$, if π is a RES (tree-RES resp.) refutation of $\phi\lceil_{x=a}$, then there is a RES (tree-RES resp.) derivation π' of either (x^{1-a}) or $\square$ from ϕ with $|\pi'| = |\pi|$.*

Sketch of proof. Suppose $\pi = (C_1, C_2, \cdots, C_S)$ is a *RES* (*tree-RES* resp.) refutation of $\phi\lceil_{x=a}$. Inductively transform π to $\pi' = (C'_1 \;\; C'_2 \;\; \cdots \;\; C'_S)$ as follows:

$$C'_i = \begin{cases} C_i & \text{if } C_i \in \phi \\ (C_i \;\; x^{1-a}) & \text{if } C_i \in \phi\lceil_{x=a} \text{ but } C_i \notin \phi \\ & \text{(Note that } (C_i \;\; x^{1-a}) \in \phi \text{ in this case)} \\ \text{the resolvent of } C'_j \text{ and } C'_k & \text{if } C_i \text{ is the resolvent of } C_j \text{ and } C_k \end{cases}$$

Then π' is a *RES* (*tree-RES* resp.) derivation of either (x^{1-a}) or $\square$ from ϕ and π' is of the same size as π. $\square$

A clause is *positive* if it contains only positive literals. A clause is *negative* if it contains only negative literals.

Definition 3.9 (negative resolution, *N-RES*). A *negative resolution* (*N-RES*) derivation is a resolution derivation in which the resolution rule is restricted to be negative: one of the two premises must be negative.

3.2 Nogood Resolution (*NG-RES*)

We have seen that the resolution rule is based on the fact that a propositional variable can take values only from $\{0, 1\}$. The same idea of "exhausting the domain" can be extended to handle domains with size larger than two. For example, if the domain of a variable x is $\{1, 2, 3\}$, then we can resolve the nogoods

$$\eta(x = 1, y = 1)$$
$$\eta(x = 2, w = 2)$$
$$\eta(x = 3, y = 1, z = 1)$$

together to soundly infer

$$\eta(w = 2, y = 1, z = 1).$$

This generalization yields a resolution-based refutation system for CSPs.

Definition 3.10 (nogood resolution, *NG-RES*). Given that the domain of a variable x is $\{1, 2, \cdots, d\}$, the *nogood resolution rule* allows one to infer a nogood, called the *resolvent*, from a set of nogoods, the *premises*, by resolving on x:

$$\frac{\begin{array}{c}\eta(x = 1,\ N_1)\\ \eta(x = 2,\ N_2)\\ \vdots\\ \eta(x = d,\ N_d)\end{array}}{\eta(N_1, N_2, \ldots N_d)} \quad x \in \{1, \ldots, d\}$$

Let $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ be a CSP instance. A *nogood resolution derivation* of a nogood N from Γ is a sequence of nogoods $N_1, N_2, \cdots, N_m$ in which each nogood N_i is either in Γ or is derived from a set of previous nogoods in the sequence by the nogood resolution rule, and $N_m = N$. A *nogood resolution refutation* of $\mathcal{I}$ is a nogood resolution derivation of the empty nogood $\eta() = \square$ from Γ. We use *NG-RES* to denote this system.

Note that if we resolve the nogoods $\eta(x = 1,\ y = 1)$, $\eta(x = 2,\ z = 1)$ and $\eta(x = 3,\ y = 2)$ on x, where the domain of x is $\{1, 2, 3\}$, we will get the nogood $\eta(y = 1,\ y = 2,\ z = 1)$ which is a tautology.

The refutation system *NG-RES* is sound and complete.

Proposition 3.11 (Soundness and Completeness of *NG-RES*). *For any CSP instance $\mathcal{I}$, there is an NG-RES refutation of $\mathcal{I}$ if and only if $\mathcal{I}$ is unsatisfiable.*

We will prove the soundness and completeness of *NG-RES* after we state all necessary definitions.

Example 3.12. The instance in Example 2.4 is unsatisfiable. Figure 3.1 shows an *NG-RES* derivation of the empty nogood from the nogoods in the instance.

Definition 3.13 (tree-like nogood resolution derivation, *tree-NG-RES*). A *tree-like nogood resolution* (denoted *tree-NG-RES*) derivation is an *NG-RES* derivation in which every derived nogood is used at most once to derive other nogoods.

The *NG-RES* refutation in Example 3.12 is tree-like. It is sometimes useful to represent an *NG-RES* derivation as a directed acyclic graph (DAG).

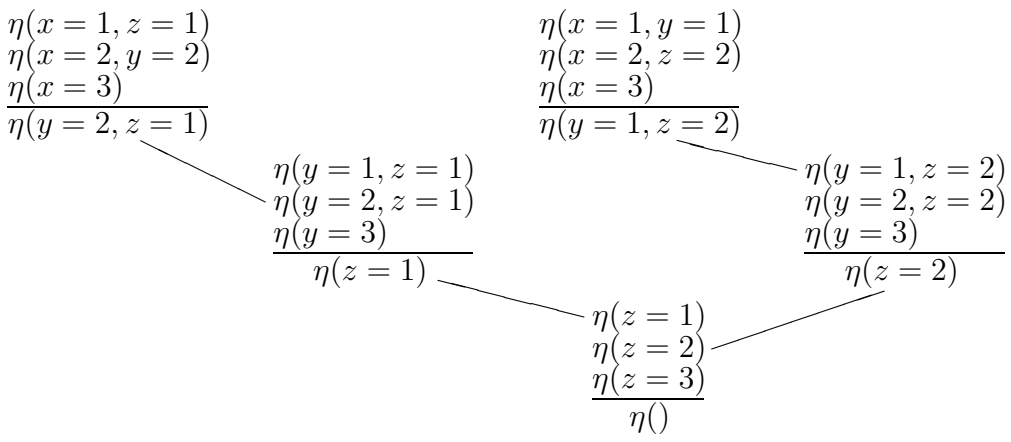

Figure 3.1: An *NG-RES* refutation

Definition 3.14 (G_π, graph of a derivation). For any *NG-RES* derivation π, we define G_π, the *graph of* π, to be the directed acyclic graph in which vertices are nogoods in π and there is an edge from vertex v to vertex u if and only if v is used as a premise in an *NG-RES* derivation step to derive u in π.

So, if π is a *tree-NG-RES* derivation, then every vertex in G_π must have out-degree 0 or 1.

Given a CSP instance $\mathcal{I}$ and a CSP refutation system, our main interest is the size of the smallest refutation of $\mathcal{I}$ in the system. This is what we focus on when comparing the relative power of different CSP refutation systems. Formally, we define the *NG-RES* complexity (*tree-NG-RES* complexity resp.) of an instance $\mathcal{I}$ to be the size of the minimal *NG-RES* (*tree-NG-RES* resp.) refutation of $\mathcal{I}$.

Definition 3.15 (size, width). Let π be an *NG-RES* derivation. The *size* of π, $|\pi|$, is the number of nogoods in π. The *width* of π, $w(\pi)$, is the width of the widest nogood in π.

Definition 3.16 (*NG-RES* complexity, *NG-RES*($\mathcal{I}$) and *tree-NG-RES*($\mathcal{I}$)). For any unsatisfiable CSP instance $\mathcal{I}$,

$$\textit{NG-RES}(\mathcal{I}) \stackrel{\text{def}}{=} \min\{|\pi| : \pi \text{ is an } \textit{NG-RES} \text{ refutation of } \mathcal{I}\}$$

and

$$\textit{tree-NG-RES}(\mathcal{I}) \stackrel{\text{def}}{=} \min\{|\pi| : \pi \text{ is a } \textit{tree-NG-RES} \text{ refutation of } \mathcal{I}\}.$$

To simplify proofs, it is sometimes convenient to allow an *NG-RES* derivation to contain nogoods that are derived by the *nogood weakening rule*, in addition to the nogood resolution rule. What we need to confirm is that adding the nogood weakening rule does not increase the power of the refutation system in terms of the minimal size and width of refutations.

Definition 3.17 (nogood resolution with weakening, *NG-RES*$^{+weak}$ and *tree-NG-RES*$^{+weak}$). The *nogood weakening rule* allows a nogood N_2 to be inferred from N_1 if N_1 is a subnogood of N_2. *Nogood resolution with weakening* (*NG-RES*$^{+weak}$) is a variant of the *NG-RES* refutation system in which nogoods can be derived by the nogood resolution rule and the nogood weakening rule. A *tree-NG-RES*$^{+weak}$ derivation is an *NG-RES*$^{+weak}$ derivation in which every nogood is used at most once to derive other nogoods.

Proposition 3.18. *For $\mathcal{I}$ a CSP instance, if π is an NG-RES^{+weak} (tree-NG-RES^{+weak} resp.) refutation of $\mathcal{I}$, then π can be transformed into an NG-RES (tree-NG-RES resp.) refutation of $\mathcal{I}$ of at most the same size and width.*

Proof. Let $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ be a CSP instance. Let $\pi = (N_1, N_2, \cdots, N_S)$ be an *NG-RES*$^{+weak}$ (*tree-NG-RES*$^{+weak}$ resp.) refutation of $\mathcal{I}$.

We inductively transform π into $\pi' = (N'_1, N'_2, \cdots, N'_S)$ as follows:

$$N'_i = \begin{cases} N_i & \text{if } N_i \in \Gamma \\ N'_j & \text{if } N_i \text{ is derived from } N_j \text{ by weakening} \\ \text{the resolvent of } N'_{i_1}, \cdots, N'_{i_d} & \text{if } N_i \text{ is the resolvent of } N_{i_1}, \cdots, N_{i_d} \text{ on } x \text{ in } \pi \\ & \text{and } x \text{ appears in all } N'_{i_1}, \cdots, N'_{i_d} \\ N'_{i_j} & \text{if } N_i \text{ is the resolvent of } N_{i_1}, \cdots, N_{i_d} \text{ on } x \text{ in } \pi \\ & \text{but } x \text{ does not appear in some } N'_{i_j}, i \leq j \leq d \end{cases}$$

It is not hard to see that π' is an *NG-RES*$^{+weak}$ (*tree-NG-RES*$^{+weak}$ resp.) derivation in which all weakening steps are of the form $\frac{N}{N}$ (i.e., deriving the same nogood) and for all $i \in \{1, 2, \cdots, S\}$, N'_i is a subnogood of N_i. Since $N_S = \square$, we have $N'_S = \square$ and thus π' is an *NG-RES*$^{+weak}$ (*tree-NG-RES*$^{+weak}$ resp.) refutation.

Now we eliminate the use of weakening rule in π'. We look at the nogoods in π' one by one. If a nogood N_i is derived from N_j by weakening, we remove N_i and for all derivation steps that use N_i as a premise, make them use N_j instead. In a *tree-NG-RES*$^{+weak}$ refutation, every nogood is used at most once to derive other nogoods. Hence, the elimination step maintains the tree-like property of π' if π' is tree-like.

The resulting refutation is an *NG-RES* (*tree-NG-RES* resp.) refutation. Moreover, the size and width can only decrease during the transformation.

□

We have already mentioned assignments for CSP instances in Chapter 2. Now we define them formally.

Definition 3.19 (assignment). An *assignment* for a CSP instance $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ assigns domain values to some variables in $\mathcal{I}$. An assignment ρ can be written as a set of literals. For example, the assignment $\rho = \{x_1 = a_1, x_2 = a_2, \cdots, x_t = a_t\}$ assigns a_i to x_i, $i \in \{1, \cdots, t\}$. Let ρ be an assignment. We write $N\lceil_\rho$ as the result of applying ρ to nogood N, and

$$N\lceil_\rho \stackrel{def}{=} \begin{cases} 1 & \text{if there is some } x \text{ and some } a, b \in \mathcal{D}(x) \text{ s.t.} \\ & a \neq b, (x = a) \in \rho \text{ and } (x = b) \in N \\ N \setminus \{(x = a) : (x = a) \in \rho\} & \text{otherwise} \end{cases}$$

Define $\mathcal{I}\lceil_\rho \stackrel{def}{=} \langle \mathcal{D}\lceil_\rho, \Gamma\lceil_\rho \rangle$, where

$$\begin{aligned} \Gamma\lceil_\rho &= \{N\lceil_\rho : N \in \Gamma \text{ and } N\lceil_\rho \neq 1\} \\ \mathit{vars}(\mathcal{I}\lceil_\rho) &= \mathit{vars}(\mathcal{I}) \setminus \{x : (x = a) \in \rho \text{ for some } a\} \\ \mathcal{D}\lceil_\rho(x) &= \mathcal{D}(x) \qquad \text{for all } x \in \mathit{vars}(\mathcal{I}\lceil_\rho). \end{aligned}$$

If $\pi = (N_1, N_2, \cdots, N_S)$ is an *NG-RES* derivation, define $\pi\lceil_\rho$ to be $(N_1\lceil_\rho, \cdots, N_S\lceil_\rho)$, but with any $N_i\lceil_\rho$ that is identical to 1 removed.

Proposition 3.20. *For $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ a CSP instance, N a nogood and ρ an assignment, if π is an NG-RES*$^{+weak}$ *(tree-NG-RES*$^{+weak}$ *resp.) derivation of N from Γ, then $\pi\lceil_\rho$ is an NG-RES*$^{+weak}$ *(tree-NG-RES*$^{+weak}$ *resp.) derivation of $N\lceil_\rho$ from $\Gamma\lceil_\rho$.*

Lemma 3.21. *For $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ a CSP instance, $x \in vars(\mathcal{I})$ and $a \in \mathcal{D}(x)$, if there is an NG-RES (tree-NG-RES resp.) refutation π of $\mathcal{I}\lceil_{x=a}$ of size S, then there is an NG-RES (tree-NG-RES resp.) derivation π' of either $\eta(x = a)$ or $\square$ of $\mathcal{I}$ of size S.*

Proof. Let $\pi = (N_1, N_2, \cdots, N_S)$ be an *NG-RES* refutation of $\mathcal{I}\lceil_{x=a}$ of size S. Construct $\pi' = (N'_1, N'_2, \cdots, N'_S)$ as follows:

$$N'_i = \begin{cases} N_i & \text{if } N_i \in \Gamma \\ \eta(N_i, x = a) & \text{if } N_i \in \Gamma\lceil_{x=a} \text{ but } N_i \notin \Gamma \\ & (\eta(N_i, x = a) \in \Gamma \text{ in this case}) \\ \text{the resolvent of } N'_{i_1}, \cdots, N'_{i_d} & \text{if } N_i \text{ is the resolvent of } N_{i_1}, \cdots, N_{i_d} \text{ in } \pi \end{cases}$$

Then π' is an *NG-RES* derivation of $\mathcal{I}$ since every N'_i is either in Γ or is a resolvent of some previous nogoods in π'. Moreover, for every nogood N_i in π, N'_i is either N_i or $\eta(N_i, x = a)$. Hence, the last nogood in π' must be either $\square$ or $\eta(x = a)$. The size of π' is the same as π.

The case when π is tree-like is the same, as our construction of π' preserves the tree-like property. $\square$

Now we can prove the soundness and completeness of *NG-RES*.

Proof. [Proposition 3.11]

(Soundness) Suppose $\pi = (N_1, N_2, \cdots, N_S)$ is an *NG-RES* refutation of a CSP instance $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$. Towards a contradiction, suppose $\mathcal{I}$ is satisfiable. Then, there exists a total assignment α such that α satisfies every nogood in Γ. It is obvious that the nogood resolution rule is sound. That is, if an assignment satisfies all premises, then it must also satisfy the resolvent. Therefore, by induction, α must satisfy every nogood in π. However, since π is a refutation, $N_S = \square$ and no assignment would satisfy N_S. Hence, there is a contradiction.

(Completeness) Suppose $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ is an unsatisfiable CSP instance. We will show, by induction on the number of variables in $\mathcal{I}$, that there is an *NG-RES* refutation of $\mathcal{I}$. If $|vars(\mathcal{I})| = 0$, then Γ contains the empty nogood which is an *NG-RES* refutation. Assume the claim is true for all instances with less than n variables. Consider the case where $\mathcal{I}$ has n variables. Let x be a variable in $\mathcal{I}$. For each

$a \in \mathcal{D}(x)$, $\mathcal{I}\lceil_{x=a}$ is an instance with $n-1$ variables and is unsatisfiable. By I.H., there is an *NG-RES* refutation of $\mathcal{I}\lceil_{x=a}$. Then, by Lemma 3.21, there exists an *NG-RES* derivation of either $\eta(x = a)$ or $\square$ (done in this case) of $\mathcal{I}$. Resolving the nogoods $\eta(x = a)$, $a \in \mathcal{D}(x)$, together gives the empty nogood.

□

Exponential separations between tree-like and general resolution have been known for some time [9]. Recently, a nearly optimal separation between the two resolution systems was obtained by Ben-Sasson [7]. Ben-Sasson showed that there exists an infinite family of CNF formulas with $O(n)$-size resolution refutations for which every tree-like resolution refutation is of size $2^{\Omega(n/\log n)}$. He also proved that for every unsatisfiable CNF formula ϕ, if S is the size of the smallest resolution refutation of ϕ, then the smallest tree-like resolution refutation of ϕ must be of size at most $2^{O(S \log\log S/\log S)}$ (upper bound). Therefore, the gap is almost tight.

In the next two subsections, we adapt the techniques used in [7] to show corresponding separations and upper bounds for *tree-NG-RES* and *NG-RES*.

3.2.1 Separation of *tree-NG-RES* from *NG-RES*

In this section, we explore the relative power of the *tree-NG-RES* and *NG-RES* proof systems. Specifically, we will show, for any integer $d \geq 3$, that there is an infinite family of CSP instances with domain size d which have poly-size *NG-RES* refutations, but for which any *tree-NG-RES* refutation has exponential size. The instances are based on directed acyclic graphs. They were first introduced as CNF formulas, called implication graph formulas, by Raz and Mckenzie [33]. Since then, a generalized form of the formulas has been used in separating several restricted versions of resolution from resolution [1, 11, 9]. In [7], Ben-Sasson exposed a direct connection between the tree-like resolution complexity of the generalized formulas and the pebbling numbers (to be defined later) of the formulas' underlying graphs. He used this relationship to accomplish a nearly optimal separation between tree-like resolution and resolution. Here, we will formulate a CSP version of the implication graph formulas and refer

closely to Ben-Sasson's approach to achieve a separation between *tree-NG-RES* and *NG-RES*. We start with some basic graph definitions.

Definition 3.22. Let $G = (V, E)$ be a directed acyclic graph. If $(v_i, v_j) \in E$, then we say that v_i is a *predecessor* of v_j and v_j is a *successor* of v_i. The *in-degree* (*out-degree* resp.) value of a vertex is the number of predecessors (successors resp.) of it. A vertex with in-degree 0 is a *source* and a vertex with out-degree 0 is a *target*. An *internal vertex* is a non-source vertex.

Definition 3.23 (topological ordering of a DAG). A *topological ordering* of a directed acyclic graph $G = (V, E)$ is a linear ordering of all the vertices in V such that if there is an edge $(v_i, v_j) \in E$, then v_i precedes v_j in the ordering.

Definition 3.24. A *circuit* is a DAG in which every vertex has in-degree 2 or 0.

Our CSP instances, implication graph contradictions, are based on circuits.

Definition 3.25 (Implication Graph Contradictions). Let $G = (V, E)$ be a circuit with n vertices. Let $d \geq 3$ be an integer. Let S and T be the sets of sources and targets in G respectively. For each vertex $v_i \in V$, there is a variable x_i associated with it. The implication graph contradiction of G, $IMP_{G,S,T,d}$, is a CSP instance with n variables, $x_1, \cdots, x_n$, domain $A = [d]$, and the following nogoods:

Source axioms: $\eta(x_i = 1)$ for every $v_i \in S$

Target axioms: $\eta(x_i = a)$ for every $v_i \in T$, and for all $a \in [d] \setminus \{1\}$

Pebbling axioms: $\eta(x_i = a,\ x_j = b,\ x_k = 1)$ for every v_k with predecessors v_i and v_j, and for all $a, b \in [d] \setminus \{1\}$

Intuitively, $IMP_{G,S,T,d}$ expresses the following contradiction: Every vertex of G can be labelled with a number from 1 to d. The sources are not labelled with 1, and the targets are labelled with 1. If both of the predecessors of a vertex are not labelled with 1, neither is the vertex itself.

We first show that $IMP_{G,S,T,d}$ has short *NG-RES* refutations.

Lemma 3.26. *For any integer $d \geq 3$, if G is a circuit with n vertices and S and T are the sets of sources and targets in G, then NG-RES$(IMP_{G,S,T,d}) = O(d^2 n)$.*

Proof. Let $d \geq 3$ be an integer and G be a circuit with n vertices. Let S and T be the sets of sources and targets in G respectively. Pick a topological ordering $\mathcal{O}$ on the vertices of G. We can derive $\eta(x_k = 1)$ for each vertex v_k one by one according to the ordering $\mathcal{O}$. If $v_k \in S$, then we already have $\eta(x_k = 1)$ since it is a Source axiom. Otherwise, suppose v_i and v_j are the predecessors of v_k. We must have derived $\eta(x_i = 1)$ and $\eta(x_j = 1)$ as v_i and v_j precede v_k in $\mathcal{O}$. Resolve the $(d-1)^2$ Pebbling axioms of v_k with $\eta(x_i = 1)$ to get $\eta(x_j = b,\ x_k = 1)$, for all $b \in [d] \backslash \{1\}$, and then resolve these with $\eta(x_j = 1)$ to derive $\eta(x_k = 1)$. This requires d derivation steps. Once $\eta(x_t = 1)$ is derived for a vertex $t \in T$, we can resolve it with the Target axioms of v_t to obtain $\eta()$. Hence, there is an *NG-RES* refutation with at most $O(dn)$ derivation steps where each derivation step requires d premises. So, *NG-RES*$(IMP_{G,S,T,d}) = O(d^2 n)$. □

While $IMP_{G,S,T,d}$ has poly-size *NG-RES* refutation, the *tree-NG-RES* complexity of $IMP_{G,S,T,d}$ depends on the pebbling number of G. Roughly speaking, *tree-NG-RES* refutations of $IMP_{G,S,T,d}$ will be long if G has large pebbling number. We will examine this precisely after defining the pebbling number of a directed acyclic graph.

Definition 3.27 (Pebbling number). Let $G = (V, E)$ be a directed acyclic graph. Let $S, T \subseteq V$. To *pebble* a vertex $v \in V$ from S, one has to follow the rules below until a pebble is placed on v.

1. A pebble can be placed on a vertex in S.
2. A pebble can be removed from any vertex.
3. If a vertex is not in S, then it can only be pebbled if all its immediate predecessors have a pebble on them.

The *pebbling number* of T on G from S, denoted $P_G(S, T)$, is the minimal number of pebbles needed to pebble some vertex in T from S.

The following lemma from [7] states a straightforward property of $P_G(S, T)$.

Lemma 3.28 (Ben-Sasson [7]). *Let $G = (V, E)$ be a DAG. For any $v \in V$ and any sets $S, T \subseteq V$, $P_G(S, T) \leq \max\{P_G(S, T \cup \{v\}), P_G(S \cup \{v\}, T) + 1\}$.*

Proof. To pebble T from S, we can first pebble $T \cup \{v\}$ from S with $P_G(S, T \cup \{v\})$ pebbles. If some vertex in T is pebbled, then we are done. Otherwise, only v is pebbled. Leave the pebble on v and try to pebble T from $S \cup \{v\}$. This requires $P_G(S \cup \{v\}, T) + 1$ pebbles. □

Note that for a DAG G with n vertices, $P_G(S, T) = O(n)$ since we can always use n pebbles and thus do not need to remove pebbles from vertices. What we are interested is a lower bound on the number of pebbles needed. Cook [13] showed that the pebbling number of the target from the sources on a pyramid graph with n vertices is $\Omega(\sqrt{n})$. In a pyramid graph with $n = m + (m-1) + \cdots + 1$ vertices, there are m layers of vertices. The i-th layer has i vertices labelled $v_{i,1}, v_{i,2}, \cdots, v_{i,i}$. The vertex at layer 1 is a target and the vertices at layer m are sources. Each non-source vertex $v_{i,j}$ has predecessors $v_{i+1,j}$ and $v_{i+1,j+1}$. Figure 3.2 shows what the pyramid graph looks like when $n = 10$. One can try to pebble $v_{1,1}$ from the sources and get an intuition of the minimum number of pebbles required in order to pebble the vertex. Celoni et al. [12] presented an infinite family of graphs G_n with n vertices, each has in-degree 2 or 0, for which $P_{G_n}(S, T) = \Omega(n/\log n)$ where S and T are the sets of sources and targets in G_n respectively. We will leave out the description of G_n as it is too complicated to illustrate here but a reader can refer to their paper if interested. The significant thing for us is that the implication graph contradiction based on G_n is hard for *tree-NG-RES*. In particular, every *tree-NG-RES* refutation of $IMP_{G_n,S,T,d}$ must be of size $(d-1)^{\Omega(n/\log n)}$. We will prove this by showing the general lower bound on the size of *tree-NG-RES* refutations of $IMP_{G,S,T,d}$ using a modification of the game approach from [7].

Our modified game is as follows. Let $\mathcal{I} = \langle [d], \Gamma \rangle$ be an unsatisfiable CSP instance. The game involves two players: Prover and Delayer. In each round, Prover picks a variable from $vars(\Gamma)$. Then, Delayer can choose 1 or *. If 1 is chosen, the variable is set to 1. Otherwise, Prover can pick a value from $\{2, \cdots, d\}$ and assign it to the variable. Delayer scores one point if he chooses *. The game ends when the current

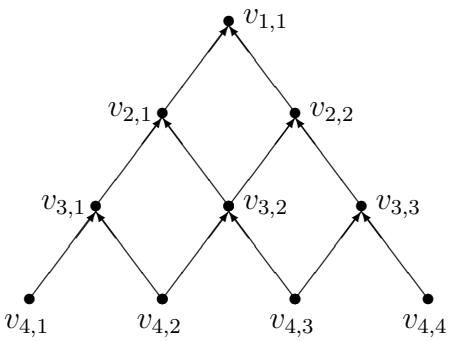

Figure 3.2: Pyramid graph with 10 vertices

assignment falsifies at least one of the nogoods in Γ. Remark: When the game ends, there must be some nogood in Γ that is made false since $\mathcal{I}$ is unsatisfiable.

Here is the rough idea of the proof. We first show that a *tree-NG-RES* refutation of $IMP_{G,S,T,d}$ gives a strategy for Prover to limit the number of points Delayer can win. This in turn will imply that any *tree-NG-RES* refutation of $IMP_{G,S,T,d}$ is of size exponential in the number of points Delayer can score. Then, we prove that there is a good strategy for Delayer to win at least $\Omega(P_G(S,T))$ points. So, every *tree-NG-RES* refutation of $IMP_{G,S,T,d}$ must be of size exponential to $\Omega(P_G(S,T))$. We call the above Delayer's strategy *superstrategy*.

Lemma 3.29. *For $\mathcal{I}$ an unsatisfiable CSP instance, if $\mathcal{I}$ has a tree-NG-RES refutation of size S, then Prover has a strategy where Delayer can win at most $\lceil \log_{d-1} S \rceil$ points.*

Proof. Suppose $\mathcal{I}$ has a *tree-NG-RES* refutation π of size S. We will give a strategy which allows Prover to bound the number of points Delayer can win and show that as long as Prover follows the strategy, the following invariant will be maintained after each round: If p is the current points Delayer has scored, then there is a nogood N in π such that N is falsified by the current partial assignment and the sub-tree rooted at N in G_π is of size at most $S/(d-1)^p$.

At the beginning, Delayer has no points and the only nogood that is falsified is the emply nogood. So, the invariant holds. Consider the i-th round. Let p_{i-1} be the number of points Delayer has scored after the previous round and N_{i-1} be the nogood satisfying the invariant at the previous round. If N_{i-1} is a leaf in G_π, then N_{i-1} is

a nogood in Γ that is falsified by the current partial assignment and hence the game ends. Otherwise, Prover picks the variable x which is resolved on to derive N_{i-1} from nogoods $N_1, N_2, \cdots, N_d$ in π. W.L.O.G., suppose $(x = 1) \in N_1, (x = 2) \in N_2$, and so on. If Delayer assigns 1 to x, then N_1 is falsified and it becomes the new nogood for the invariant. In this case, Delayer does not score any points and the sub-tree rooted at N_1 is obviously smaller than the one rooted at N_{i-1}. Thus, the invariant holds. If the Delayer chooses *, then Prover assigns x the value $j \in \{2, \cdots, d\}$ which will falsify the nogood N_j, among $N_2, \cdots, N_d$, with the smallest sub-tree. The size of this sub-tree is at most $1/(d-1)$ of $S/(d-1)^{p_{i-1}}$ which is the size of the sub-tree rooted at N_{i-1}. So, the sub-tree rooted at N_j is of size at most $1/(d-1)^{p_{i-1}+1}$. Since Delayer chooses *, he can score a point and the number of points he has scored after this round is $p_{i-1} + 1$. Therefore, the invariant is maintained.

When the game halts, the size of the sub-tree is 1. If Delayer scores p points at the end of the game, then $1 \leq S/(d-1)^p$. This implies $p \leq \log_{d-1} S \leq \lceil \log_{d-1} S \rceil$. Hence, Delayer can win at most $\lceil \log_{d-1} S \rceil$ points if Prover follows the above strategy. □

Corollary 3.30. *For $\mathcal{I}$ an unsatisfiable CSP instance, if the Delayer has a strategy which always scores r points on $\mathcal{I}$, then tree-NG-RES($\mathcal{I}$) $\geq (d-1)^{r-1}$.*

Proof. Suppose the Delayer has a strategy which always scores r points on $\mathcal{I}$. Towards a contradiction, suppose *tree-NG-RES*$(\mathcal{I}) < (d-1)^{r-1}$. Then, by Lemma 3.29, the Prover has a strategy where the Delayer can win at most $\lceil \log_{d-1}(d-1)^{r-1} \rceil = r-1 < r$ points. This contradicts that the Delayer can always scores r points. □

The superstrategy for Delayer is simple. Before each game, Delayer sets $S' = S$ and $T' = T$. Then, in each round, if Prover asks about variable $x_i, i \in [n]$, Delayer responds as follows:

1. If $v_i \in T'$, assign 1 to the variable.
2. If $v_i \in S'$, respond *.
3. If $v_i \notin S' \cup T'$ and $P_G(S', T' \cup \{i\}) = P_G(S', T')$, assign the variable 1 and add v_i to T'.
4. If $v_i \notin S' \cup T'$ and $P_G(S', T' \cup \{i\}) < P_G(S', T')$, respond * and add v_i to S'.

We will prove that $P_G(S', T')$ can only decrease by at most the number of points Delayer scores and it is at most 3 at the end of the game. This implies the superstrategy guarantees Delayer to earn at least $P_G(S, T) - 3$ points.

Lemma 3.31. *After each round, if Delayer has scored p points, then $P_G(S', T') \geq P_G(S, T) - p$.*

Proof. Let S'_i and T'_i be the sets S' and T' respectively in Delayers superstrategy after round i. Let p_i be the number of points Delayer has scored after round i. We show that the invariant $P_G(S'_i, T'_i) \geq P_G(S, T) - p_i$ will be maintained after each round. At the beginning, $p_0 = 0$, $S'_0 = S$ and $T'_0 = T$. So, $P_G(S'_0, T'_0) = P_G(S, T) - 0$ and the invariant holds. Now consider round i. For case 1, 2, and 3, $P_G(S'_{i-1}, T'_{i-1}) = P_G(S'_i, T'_i)$ and $p_i \geq p_{i-1}$. So, $P_G(S'_i, T'_i) = P_G(S'_{i-1}, T'_{i-1}) \geq P_G(S, T) - p_{i-1} \geq P_G(S, T) - p_i$. For case 4, $P_G(S'_{i-1}, T'_{i-1} \cup \{v\}) < P_G(S'_{i-1}, T'_{i-1})$, $p_i = p_{i-1} + 1$, $S'_i = S'_{i-1} \cup \{v\}$, and $T'_i = T'_{i-1}$. By Lemma 3.28, we have $P_G(S'_{i-1} \cup \{v\}, T'_{i-1}) \geq P_G(S'_{i-1}, T'_{i-1}) - 1$. Hence,

$$\begin{aligned} P_G(S'_i, T'_i) &= P_G(S'_{i-1} \cup \{v\}, T'_{i-1}) \\ &\geq P_G(S'_{i-1}, T'_{i-1}) - 1 \\ &\geq P_G(S, T) - p_{i-1} - 1 \\ &= P_G(S, T) - p_i \end{aligned}$$

Therefore, the invariant is maintained after each round. □

Lemma 3.32. *At the end of the game, $P_G(S', T') \leq 3$.*

Proof. When the game ends, some nogood N must be falsified since $IMP_{G,S,T,d}$ is unsatisfiable. N cannot be a Source axiom for some source v_i because $v_i \in S \subseteq S'$ and thus it can only be assigned values from $\{2, \cdots, d\}$ through case 2. This assignment does not violate the Source axiom. Similarly, N cannot be a Target axiom either. Hence, N must be a Pebbling axiom for some vertex v_k with predecessors v_i and v_j. To falsify N, x_k must be set to 1 and both x_i and x_j must be set to some values from $\{2, \cdots, d\}$. So, $v_k \in T'$ (via case 1 or case 3) and $v_i, v_j \in S'$ (via case 2 or case 4). Therefore, to pebble T' from S', we can first pebble v_i and v_j, then v_k. This only requires three pebbles. □

Corollary 3.33. *Following the superstrategy described, Delayer can score at least* $P_G(S,T) - 3$ *points at the end of the game.*

Proof. This is an immediate consequence from Lemma 3.31 and Lemma 3.32. □

Theorem 3.34. *tree-NG-RES*$(IMP_{G,S,T,d}) = (d-1)^{\Omega(P_G(S,T))}$.

Proof. Corollary 3.33 shows that Delayer has a superstrategy to score $P_G(S,T) - 3$ points on $IMP_{G,S,T,d}$. Therefore, by Corollary 3.30, we have *tree-NG-RES*$(IMP_{G,S,T,d})$ $\geq (d-1)^{P_G(S,T)-4}$. Hence, *tree-NG-RES*$(IMP_{G,S,T,d}) = (d-1)^{\Omega(P_G(S,T))}$. □

Combining everything discussed in this section, we obtain an exponential separation of *tree-NG-RES* from *NG-RES*:

Theorem 3.35 (Separation). *For every integer* $d \geq 3$*, there exists an infinite family of CSP instances* $\{\mathcal{I}_n\}$*, with domain size* d*, such that*

1. *NG-RES*$(\mathcal{I}_n) = O(d^2 n)$,
2. *tree-NG-RES*$(\mathcal{I}_n) = (d-1)^{\Omega(n/\log n)}$.

Proof. Let $d \geq 3$ be an integer. [12] provides an infinite family of circuits $\{G_n\}$, with $|V(G_n)| = n$, for which $P_{G_n}(S,T) = \Omega(n/\log n)$ where S and T are the sets of sources and targets in G. From Lemma 3.26, we have *NG-RES*$(IMP_{G_n,S,T,d}) = O(d^2 n)$. And, by Theorem 3.34, *tree-NG-RES*$(IMP_{G_n,S,T,d}) = (d-1)^{\Omega(P_G(S,T))} = (d-1)^{\Omega(n/\log n)}$. □

3.2.2 Separation Upper Bound

In the previous section, we showed that there are families of CSP instances such that they have linear-size *NG-RES* refutations but no *tree-NG-RES* refutation of size smaller than $exp(\Omega(n/\log n))$. The next question one may ask is: How big can this separation be? Or, more specifically, if we know that the smallest *NG-RES* refutation of a CSP instance $\mathcal{I}$ is of size S, then what is the upper limit for the size of the smallest *tree-NG-RES* refutation of $\mathcal{I}$ in terms of S? Ben-Sasson presented an upper bound for the separation between *tree-RES* and *RES* in [7]. We will take his proof and modify it for *tree-NG-RES*.

Before proceeding to show the upper bound, we first define decision trees for CSP search problems which are closely related to *tree-NG-RES*. We will also describe a different parameter to measure the size of the underlying graph of a minimal refutation.

Let $\mathcal{I} = \langle [d], \Gamma \rangle$ be a CSP instance. Given an assignment α to all the variables in Γ, a *search problem* for $\mathcal{I}$ is to find a nogood N in Γ such that α falsifies N. If such a nogood does not exist, then output 1.

Definition 3.36 (Decision Trees for CSP Search Problems). A *d-ary Decision Tree* is a d-ary tree, in which internal vertices, edges, and leaves are labelled with variables, elements in $[d]$, and possible outputs, respectively. Given a d-ary decision tree D, each assignment α to the variables corresponds to a unique path in D in the natural way, and the label at the end of the path is the output of D on α. D is a d-ary decision tree for the search problem for $\mathcal{I}$ if for every input assignment α, D outputs a nogood in Γ which is falsified by α or outputs 1 if α satisfies all nogoods in Γ. Define $S_D(\mathcal{I})$ to be the minimal size of a d-ary decision tree for the search problem for $\mathcal{I}$.

Decision trees for CSP search problems are closely connected to *tree-NG-RES* in such a way that the size of the smallest *tree-NG-RES* refutation of an unsatisfiable CSP instance $\mathcal{I}$ is equal to the size of the minimal decision tree for $\mathcal{I}$.

Lemma 3.37. *If $\mathcal{I}$ is an unsatisfiable CSP instance, then tree-NG-RES$(\mathcal{I}) = S_D(\mathcal{I})$.*

Proof. Let $\mathcal{I}$ be an unsatisfiable CSP instance. Let π be a *tree-NG-RES* refutation of $\mathcal{I}$. For each internal vertex in G_π, if we label it with the variable resolved on to obtain the nogood on the vertex and label each edge pointing to the vertex with the corresponding domain value of the variable, then the resulting graph is a d-ary decision tree for the search problem for $\mathcal{I}$. So, *tree-NG-RES*$(\mathcal{I}) \geq S_D(\mathcal{I})$.

To show *tree-NG-RES*$(\mathcal{I}) \leq S_D(\mathcal{I})$, suppose we are given a d-ary decision tree D for $\mathcal{I}$. We can construct a *tree-NG-RES* refutation of $\mathcal{I}$ with at most the same size of D. Since $\mathcal{I}$ is unsatisfiable, all leaves must be labelled with nogoods. Working from leaves towards the root, for each internal vertex v with label x, we do the following. If x does not appear in some nogood N labelled at some child of v, then we label

v with N and remove its children. Otherwise, for each $i \in [d]$, x must appear as $(x = i)$ in the nogood labelled at the child along the edge labelled with i. Label v with the resolvent of the labels on v's children. Eventually, every internal vertex v will be labelled with a nogood falsified by the partial assignment defined on the path leading to v. Hence, the root must be labelled with the empty nogood and we get a *tree-NG-RES* refutation. □

The next lemma will be needed in the proof of the upper bound.

Lemma 3.38. *If $\mathcal{I} = \langle [d], \Gamma \rangle$ is an n-variable CSP instance where $|\Gamma| = m$, then there is a d-ary decision tree for the search problem for $\mathcal{I}$ with at most $\sum_{i=0}^{m} \binom{n}{i}(d-1)^i$ leaves.*

Proof. Let $\mathcal{I} = \langle [d], \Gamma \rangle$ be an n-variable CSP instance with $|\Gamma| = m$. We can build a d-ary decision tree recursively as follows. We start with an empty truth assignment and a tree with one vertex. In each step, we have a partial assignment α and we are at some vertex v. If α falsifies some nogood N in Γ, we label v with N. If α satisfies all nogoods in Γ, we label v with 1. Otherwise, we pick the first nogood N such that $\alpha(N)$ is undefined and pick the first variable x appearing in N for which $\alpha(x)$ is undefined. Then, we label v with x and create d children for v, where the d edges leading to the children are labelled with $1, 2, \cdots, d$ respectively. After that, for each $i \in [d]$, we recursively set the child along edge i to be the new current vertex and set $\alpha \cup (x = i)$ to be the new partial assignment for the next step.

Consider the last case we just described. Among the d values that we can assign to x, $d-1$ of them will satisfy N. We call the $d-1$ values "satisfying values". Any path of the d-ary decision tree constructed as above can contain at most m satisfying values since there are only m nogoods in Γ. Hence, a path can be described by a sequence of n numbers with at most m of those being satisfying values. Therefore, the number of paths is at most

$$\binom{n}{0} + \binom{n}{1}(d-1) + \binom{n}{2}(d-1)^2 + \cdots + \binom{n}{m}(d-1)^m = \sum_{i=0}^{m} \binom{n}{i}(d-1)^i. \qquad \square$$

For some technical reasons, we have to limit ourselves to refutations π for which G_π has maximal out-degree 2. The next lemma shows that any *NG-RES* refutation

π can be transformed into an *NG-RES*$^{+weak}$ refutation π', with only a small blowup in size, such that $G_{\pi'}$ has maximal out-degree 2.

Lemma 3.39. *Let $\mathcal{I} = \langle [d], \Gamma \rangle$ be a CSP instance. If there is an NG-RES refutation π of $\mathcal{I}$, then there exists an NG-RES^{+weak} refutation π' of $\mathcal{I}$ such that $G_{\pi'}$ has maximal out-degree 2 and π' is of size at most $d \cdot |\pi|$.*

Proof. Given an *NG-RES* refutation π of $\mathcal{I}$, we can transform it into an *NG-RES*$^{+weak}$ refutation π' of $\mathcal{I}$ as follows. If a nogood N in π is used to derive k other nogoods, $N_1, N_2, \cdots, N_k$, we create $k-1$ copies of N and call them $N_{c_1}, N_{c_2}, \cdots, N_{c_{k-1}}$. For each $i \in \{1, 2, \cdots, k-2\}$, make N_{c_i} be the premise to derive N_i and $N_{c_{i+1}}$ (by weakening). Make $N_{c_{k-1}}$ be used to derive N_{k-1} and N_k. Then, $G_{\pi'}$ has maximal out-degree 2 and $|\pi'| \leq d \cdot |\pi|$ since the transformation only increases the number of nogoods by at most $\sum_{v \in V(G_\pi)} [\textit{out-deg}(v) - 1] = \sum_{v \in V(G_\pi)} \textit{out-deg}(v) - |\pi| = \sum_{v \in V(G_\pi)} \textit{in-deg}(v) - |\pi| \leq d \cdot |\pi| - |\pi|$. (Note that every vertex in G_π has in-degree d or 0 as π is an *NG-RES* refutation.) □

To prove the upper bound, we need to define a different parameter to measure the *NG-RES* complexity. The parameter counts the number of edges in the graph of a minimal *NG-RES* refutation.

Definition 3.40 (magnitude). For a DAG $G = (V, E)$, define $e(G) \stackrel{def}{=} |E(G)|$. For an unsatisfiable CSP instance $\mathcal{I}$, define the *magnitude* of refuting $\mathcal{I}$, $e(\mathcal{I})$, to be:

$$e(\mathcal{I}) \stackrel{def}{=} \min\{e(G_\pi) : \pi \text{ is an } \textit{NG-RES}^{+weak} \text{ refutation of } \mathcal{I} \text{ and } G_\pi \text{ has max out-deg } 2\}$$

We define $f_d(k)$ to be the maximal *tree-NG-RES*$^{+weak}$ complexity of refuting CSP instances with domain size d and magnitude at most k. That is, for any $d, k \in \mathbb{N}, d \geq 2$,

$$f_d(k) \stackrel{def}{=} \max\{\textit{tree-NG-RES}^{+weak}(\mathcal{I}) : \mathcal{I} \text{ has domain size } d \text{ and } e(\mathcal{I}) \leq k\}.$$

So, if the magnitude of refuting $\mathcal{I} = \langle [d], \Gamma \rangle$ is at most k, then *tree-NG-RES*$^{+weak}(\mathcal{I}) \leq f_d(k)$. $\mathcal{I}$ is k-maximal if $e(\mathcal{I}) = k$, *tree-NG-RES*$^{+weak}(\mathcal{I}) = f_d(k)$ and removing any nogood from $\mathcal{I}$ enlarges $e(\mathcal{I})$ or makes $\mathcal{I}$ satisfiable.

Note that f is monotonically increasing and for all $k \in \mathbb{N}$, a k-maximal CSP instance always exists.

Lemma 3.41. *For all $d, k \in \mathbb{N}, d \geq 2$, $f_d(k+d) \leq d \cdot f_d(k)$.*

Proof. Let $\mathcal{I} = \langle [d], \Gamma \rangle$ be a $(k+d)$-maximal unsatisfiable CSP instance. Then, $e(\mathcal{I}) = k + d$ and *tree-NG-RES*$^{+weak}(\mathcal{I}) = f_d(k+d)$. Let π be an *NG-RES*$^{+weak}$ refutation of $\mathcal{I}$ such that G_π has $k+d$ edges. Let N be a nogood in π which is a resolvent of d nogoods $N_1, N_2, \cdots, N_d \in \Gamma$. (There exists at least one such N since otherwise, Γ contains the empty nogood and $e(\mathcal{I}) = 0$.)

Let $\mathcal{I}' = \langle [d], \Gamma \cup \{N\} \rangle$. Then $e(\mathcal{I}') \leq k$ since deleting the derivation of N from π gives an *NG-RES*$^{+weak}$ refutation π' of $\mathcal{I}'$ with at most k edges in $G_{\pi'}$. Therefore,

$$\textit{tree-NG-RES}^{+weak}(\mathcal{I}') \leq f_d(k). \tag{3.1}$$

Let T' be a *tree-NG-RES*$^{+weak}$ refutation of $\mathcal{I}'$ of size *tree-NG-RES*$^{+weak}(\mathcal{I}')$. Let T be constructed from T' as follows. Whenever N appears as an axiom in T', substitute it with the *NG-RES* derivation $\frac{N_1 \ N_2 \ \cdots \ N_d}{N}$. Then, T is a *tree-NG-RES*$^{+weak}$ refutation of $\mathcal{I}$ and the size of T is at most d times of the size of T'. Hence,

$$\textit{tree-NG-RES}^{+weak}(\mathcal{I}) \leq d \cdot \textit{tree-NG-RES}^{+weak}(\mathcal{I}'). \tag{3.2}$$

Since $\mathcal{I}$ is $(k+d)$-maximal, we have $f_d(k+d) = \textit{tree-NG-RES}^{+weak}(\mathcal{I})$. With (3.1) and (3.2), we get $f_d(k+d) \leq d \cdot f_d(k)$. □

We now show that for any directed acyclic graph G with in-degree d and out-degree 2, G can be partitioned into two subgraphs such that the number of edges in them are roughly the same.

Definition 3.42 (topological partition). Let $G = (V, E)$ be a DAG. Let $v_1, \cdots, v_s$ be a topological ordering of the vertices in V. For $0 \leq i \leq S$, let $V_0(i) = \{v_1, \cdots, v_i\}$ and $V_1(i) = \{v_{i+1}, \cdots, v_S\}$. Let $G_0(i)$ ($G_1(i)$ resp.) be the subgraph of G induced by $V_0(i)$ ($V_1(i)$ resp.). Let $e_0(i)$ ($e_1(i)$ resp.) be the number of edges in $G_0(i)$ ($G_1(i)$ resp.). Let M_i be the set of vertices in $V_1(i)$ which are connected to vertices in $V_0(i)$ and define $m_i = |M_i|$. A *topological partition* of G is an ordered pair $\langle V_0(i), V_1(i) \rangle$, for some i, $0 \leq i \leq S$.

Note that vertices in M_i are internal vertices in G. Also notice that for $\mathcal{I} = \langle [d], \Gamma \rangle$ an unsatisfiable CSP instance and $\pi = (N_1, \cdots, N_S)$ an *NG-RES*$^{+weak}$ refutation of $\mathcal{I}$, π is a topological ordering of G_π. And if $\langle V_0(i), V_1(i) \rangle$ is a topological partition of G_π, then $V_0(i)$ is an *NG-RES*$^{+weak}$ derivation from Γ and $V_1(i)$ is an *NG-RES*$^{+weak}$ refutation from nogoods in $\Gamma \cup M_i$.

Lemma 3.43 (equal partition). *Let $v_1, \cdots, v_S$ be a topological ordering of the vertices of a single-target DAG $G = (V, E)$ with maximal in-degree d and maximal out-degree 2. There exists a topological partition $\langle V_0(i), V_1(i) \rangle$ of G such that $|e_0(i) - e_1(i)| \leq \frac{d+2}{2}$. Such a partition is called* equal*, and for an equal partition, $e_j(i) < \frac{e(G)-m_i}{2} + d, j \in \{0, 1\}$.*

Proof. Consider any partition $\langle V_0(i), V_1(i) \rangle$ of G, $0 \leq i \leq S$. Since every vertex in G has maximal in-degree d, $e_0(i+1) \leq e_0(i) + d$. And, since G has maximal out-degree 2, $e_1(i+1) \geq e_1(i) - 2$. It is obvious that $e_0(0) = 0 = e_1(S), e_0(S) = e(G) = e_1(0)$. Moreover, e_0 is monotonically increasing and e_1 is monotonically decreasing. Therefore, there exists an i such that $|e_0(i) - e_1(i)| \leq \frac{d+2}{2}$.

For an equal partition $\langle V_0(i), V_1(i) \rangle$, $|e_0(i) - e_1(i)| \leq \frac{d+2}{2}$. This implies $e_0(i) - e_1(i) \leq \frac{d+2}{2}$. Moreover, $e(G) \geq e_0(i) + e_1(i) + m_i$ because vertices in M_i have non-zero in-degree. Hence, for $j \in \{0, 1\}$, $e_j(i) \leq \frac{e(G)-m_i+\frac{d+2}{2}}{2} \leq \frac{e(G)-m_i}{2} + d$. □

Next, we prove that if the magnitude of refuting a CSP instance $\mathcal{I}$ with domain size d is at most $k2^k$, then we can construct a *tree-NG-RES*$^{+weak}$ refutation of $\mathcal{I}$ of size at most $d^{c\,2^k \log k}$ for some constant c. Therefore, $f_d(k2^k) \leq d^{c\,2^k \log k}$.

Theorem 3.44. *For any integer $d \geq 2$, there exists a constant $c > 0$, such that for all integers $k \geq 4$, $f_d(k2^k) \leq d^{c\,2^k \log k}$.*

Proof. (By induction on k)

Let $d \geq 2$ be some fixed integer.

Basis: Let $c \geq 4d$ be large enough so that $f_d(k2^k) \leq d^{c\,2^k \log k}$ is true for $k = 4$.

I.H.: Assume it is true for all values smaller than k, for some fixed $k \geq 4$.

Induction step: Suppose $\mathcal{I} = \langle [d], \Gamma \rangle$ is an unsatisfiable CSP instance with $e(\mathcal{I}) \leq k2^k$. We will show how to construct a *tree-NG-RES*$^{+weak}$ refutation of $\mathcal{I}$ with size at

most $d^{c\,2^k \log k}$ and this would imply $f_d(k2^k) \leq d^{c\,2^k \log k}$, by the definition of f. Let $\pi = (N_1, N_2, \cdots, N_S)$ be an *NG-RES*$^{+weak}$ refutation of $\mathcal{I}$ such that G_π has $e \stackrel{def}{=} e(\mathcal{I}) \leq k2^k$ edges and maximal out-degree 2. Let n be the number of variables appearing in π. Then, we must have $n \leq e$ due to the minimality of e. Let $\langle \pi_0, \pi_1 \rangle$ be an equal partition of G_π at some i and set $e_0 = e_0(i), e_1 = e_1(i)$ and $m = |M_0(i)|$.

Case 1: $m \geq 2^k$. Consider a nogood N in π_0. N is derived by an *NG-RES*$^{+weak}$ derivation π' with at most $e_0 \leq \frac{e-m}{2} + d \leq \frac{k2^k - 2^k}{2} + d = (k-1)2^{k-1} + d$ edges in $G_{\pi'}$ (by Lemma 3.43). Let α be a partial assignment on the variables in N such that $(x = a) \in \alpha$ if and only if $(x = a) \in N$. Then $\pi' \restriction_\alpha$ is an *NG-RES*$^{+weak}$ refutation of $\Gamma \restriction_\alpha$ (Proposition 3.20). By the definition of f and Lemma 3.41, there exists a *tree-NG-RES*$^{+weak}$ refutation from $\Gamma \restriction_\alpha$ of size at most $f_d((k-1)2^{k-1} + d) \leq d \cdot f_d((k-1)2^{k-1})$. Without increasing the size, this can be converted to a *tree-NG-RES*$^{+weak}$ derivation of some subnogood N_1 of N from Γ (Lemma 3.21). So, N has a *tree-NG-RES*$^{+weak}$ derivation of size at most $d \cdot f_d((k-1)2^{k-1}) + 1$ (we may need one weakening step to get N if $N_1 \neq N$).

Similarly, since π_1 is an *NG-RES*$^{+weak}$ refutation with $e_1 \leq (k-1)2^{k-1} + d$ edges in G_{π_1}, it can be replaced by a *tree-NG-RES*$^{+weak}$ refutation T from $\Gamma \cup M_i$ of size at most $d \cdot f_d((k-1)2^{k-1})$. For every axiom nogood N in T, if $N \notin \Gamma$, then $N \in M_i$ and the derivation step to derive N in π involves at most d nogoods in π_0. For each of those nogoods in π_0, plug a *tree-NG-RES*$^{+weak}$ derivation as described above. This yields a *tree-NG-RES*$^{+weak}$ refutation of $\mathcal{I}$. By inductive hypothesis, the size of this *tree-NG-RES*$^{+weak}$ refutation is at most:

$$[d \cdot f_d((k-1)2^{k-1})] \cdot [d \cdot (d \cdot f_d((k-1)2^{k-1}) + 1)] \leq 2d^3 \cdot [f_d((k-1)2^{k-1})]^2 \leq d^4 \cdot d^{2 \cdot c2^{k-1} \log (k-1)} = d^{4 + c2^k \log (k-1)} \leq d^{c2^k \log k}$$

whenever $c \geq 4d$ and $k \geq 4$.

Case 2: $m < 2^k$. By Lemma 3.38, there is a d-ary decision tree D with at most $\sum_{i=0}^{m} \binom{n}{i}(d-1)^i$ leaves which can solve the CSP search problem for $\mathcal{I}_M = \langle [d], M_i \rangle$. Each leaf v of D must be labelled with 1 or a nogood in M_i. If v is labelled with $N \in M_i$, then N has a *tree-NG-RES*$^{+weak}$ derivation from Γ of size

at most $d\cdot[d\cdot f_d((k-1)2^{k-1})+1]$ (see case 1). If v is labelled 1, then we know that the partial assignment ρ defined by the path leading to v in D satisfies every nogood in M_i. Since π_1 is an *NG-RES*$^{+weak}$ refutation from $\Gamma \cup M_i$, $\pi_1\lceil_\rho$ is an *NG-RES*$^{+weak}$ refutation of $\Gamma\lceil_\rho$ with at most $\frac{e-m}{2}+d$ edges in $G_{\pi_1\lceil_\rho}$ (Proposition 3.20). By Lemma 3.41, there is a *tree-NG-RES*$^{+weak}$ refutation of $\Gamma\lceil_\rho$ of size at most $d\cdot f_d(\frac{e-m}{2})$. This can be converted to a *tree-NG-RES*$^{+weak}$ derivation of N_ρ from from Γ of size at most $d\cdot f_d(\frac{e-m}{2})+1$, where $(x=a)\in N_\rho$ iff $(x=a)\in\rho$. (Note that N_ρ is falsified by ρ.) Substitute the *tree-NG-RES*$^{+weak}$ derivation of N_ρ into v. We now have a tree in which every leaf is labelled by a nogood in Γ that is falsified by the assignment defined on the path leading to it. This tree can then be transformed into a *tree-NG-RES*$^{+weak}$ refutation of Γ (details can be found in the proof of Lemma 3.37.) The size of this tree is bounded by:

$$
\begin{aligned}
&\leq \left[\sum_{i=0}^{m}\binom{n}{i}(d-1)^i\right]\cdot\left[d\cdot(d\cdot f_d(\frac{e-m}{2})+1)\right]+\sum_{i=0}^{m}\binom{n}{i}(d-1)^i\\
&\leq \left[\sum_{i=0}^{m}\binom{k2^k}{i}(d-1)^i\right]\cdot\left[d^2 f_d(\frac{k2^k-m}{2})+d+1\right] \qquad \because n\leq e \text{ and } e\leq k2^k\\
&\leq \left[\sum_{i=0}^{m}\binom{k2^k}{i}(d-1)^i\right]\cdot\left[d^3 f_d(\frac{k2^k-m}{2})\right]\\
&\leq \left[2^k(d-1)^{2^k}\binom{k2^k}{m}\right]\cdot\left[d^3 f_d(\frac{k2^k-2^k}{2}+\frac{2^k-m}{2})\right] \qquad \because m<2^k\\
&\leq d^{k+2^k+3}\binom{k2^k}{m}\cdot f_d(\frac{k2^k-2^k}{2}+\frac{2^k-m}{2})\\
&\leq d^{k+2^k+3}\binom{k2^k}{m}\cdot d^{\frac{2^k-m}{2}} f_d(\frac{k2^k-2^k}{2}) \qquad \because \text{Lemma 3.41}\\
&\leq d^{k+2^k+3+2^{k-1}}\binom{k2^k}{2}\cdot f_d((k-1)2^{k-1}) \qquad \because m<2^k\\
&\leq d^{2^{k+1}}\binom{k2^k}{2}\cdot f_d((k-1)2^{k-1}) \qquad \because k\geq 4\\
&\leq d^{d+2^{k+1}}\binom{k2^k}{2}\cdot d^{c\,2^{k-1}\log(k-1)} \qquad \because I.H.\\
&\leq d^{2^{k+1}}(4k)^{2^k}\cdot d^{c\,2^{k-1}\log(k-1)} \qquad \because \binom{k2^k}{2}\leq(4k)^{2^k} \quad \text{(see Appendix A)}
\end{aligned}
$$

$$\leq d^{2^{k+1}+2\cdot 2^k+2^k \log k+\frac{1}{2}c\,2^k \log k}$$
$$\leq d^{c\,2^k \log k}$$

The last inequality is true since for $d \geq 2$, $k \geq 4$, and $c \geq 4d$, we have $d + 2^{k+2} + 2^k \log k \leq \frac{1}{2}c\,2^k \log k$.

□

We are now ready to prove the upper bound for the separation of *tree-NG-RES* and *NG-RES*.

Theorem 3.45 (Separation Upper Bound). *For any unsatisfiable CSP instance $\mathcal{I}$ with domain size $d \geq 2$, tree-NG-RES($\mathcal{I}$) $= d^{O(\frac{d^2 S \log\log S}{\log S})}$ where $S =$ NG-RES($\mathcal{I}$).*

Proof. Let $\mathcal{I}$ be an unsatisfiable CSP instance, and π be a minimal *NG-RES* refutation of $\mathcal{I}$. Let $S = |\pi| \equiv$ *NG-RES*($\mathcal{I}$). Then, by Lemma 3.39, there exists an *NG-RES*$^{+weak}$ refutation π' such that $|\pi'| \leq dS$ and $G_{\pi'}$ has maximal out-degree 2. Hence, $e \stackrel{def}{=} e(G_{\pi'}) \leq d^2 S$, since there are at most d edges entering a vertex in $G_{\pi'}$. Set $k \stackrel{def}{=} \lceil \log \frac{2e}{\log e} \rceil$. Then, $e \leq k2^k$. Therefore,

$$\begin{aligned}
\textit{tree-NG-RES}(\mathcal{I}) &\leq \textit{tree-NG-RES}^{+weak}(\mathcal{I}) && \because \text{Proposition 3.18}\\
&\leq f(e) && \because \text{definition of } f\\
&\leq f(k2^k) && \because f \text{ is monotonically increasing}\\
&= d^{O(2^k \log k)} && \because \text{Lemma 3.44}\\
&= d^{O(\frac{2e}{\log e} \log\log \frac{2e}{\log e})}\\
&= d^{O(\frac{d^2 S \log\log S}{\log S})}.
\end{aligned}$$

□

3.3 Constraint Resolution (*C-RES*)

Next, we define another refutation system, constraint resolution, for CSPs which is based on one of the common CNF encodings [14] of CSP instances.

Let $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ be a CSP instance. We encode $\mathcal{I}$ as a CNF formula, CNF($\mathcal{I}$), as follows. For each variable $x \in \mathit{vars}(\mathcal{I})$ and each possible value $a \in \mathcal{D}(x)$ that x can take, we introduce a propositional variable $x{:}a$ asserting that x takes value a when $x{:}a$ is true.

First we need a set of clauses ensuring that every variable in $\mathcal{I}$ must be given some value. We call them *domain clauses*. Then, there are *unique value clauses* stating that no variable can take more than one value. Finally, for each nogood in Γ, we have a *constraint clause* which rules out the forbidden assignment corresponding to the nogood. Hence, the CNF encoding of $\mathcal{I}$ is:

$$\begin{aligned} \mathrm{CNF}(\mathcal{I}) &= \mathit{domainCls} \cup \mathit{uniqueValueCls} \cup \mathit{constraintCls} \\ \text{where} \quad \mathit{domainCls} &= \{(x{:}a_1 \ \cdots \ x{:}a_d) : x \in \mathit{vars}(\mathcal{I}), \mathcal{D}(x) = \{a_1, \cdots, a_d\}\} \\ \mathit{uniqueValueCls} &= \{(\overline{x{:}a} \ \ \overline{x{:}c}) : x \in \mathit{vars}(\mathcal{I}), a, c \in \mathcal{D}(x), a \neq c\} \\ \mathit{constraintCls} &= \{(\overline{x_1{:}a_1} \ \cdots \ \overline{x_k{:}a_k}) : \eta(x_1 = a_1, \cdots, x_k = a_k) \in \Gamma\}. \end{aligned}$$

CNF($\mathcal{I}$) encodes all the requirements an assignment needs to meet in order to satisfy $\mathcal{I}$. Thus, every solution of $\mathcal{I}$ corresponds to a truth assignment which satisfies all clauses in CNF($\mathcal{I}$), and vice versa. Therefore, CNF($\mathcal{I}$) is satisfiable if and only if $\mathcal{I}$ is satisfiable.

Example 3.46. The CNF encoding of the CSP instance in Example 2.4 consists of the following clauses.

Domain clauses: $(x{:}1 \ \ x{:}2 \ \ x{:}3)$
$(y{:}1 \ \ y{:}2 \ \ y{:}3)$
$(z{:}1 \ \ z{:}2 \ \ z{:}3)$

Unique value clauses: $(\overline{x{:}1} \ \ \overline{x{:}2})$ $(\overline{x{:}1} \ \ \overline{x{:}3})$ $(\overline{x{:}2} \ \ \overline{x{:}3})$
$(\overline{y{:}1} \ \ \overline{y{:}2})$ $(\overline{y{:}1} \ \ \overline{y{:}3})$ $(\overline{y{:}2} \ \ \overline{y{:}3})$
$(\overline{z{:}1} \ \ \overline{z{:}2})$ $(\overline{z{:}1} \ \ \overline{z{:}3})$ $(\overline{z{:}2} \ \ \overline{z{:}3})$

Constraint clauses: $(\overline{x{:}1} \ \ \overline{y{:}1})$ $(\overline{x{:}2} \ \ \overline{y{:}2})$ $(\overline{x{:}3} \ \ \overline{y{:}3})$
$(\overline{y{:}1} \ \ \overline{z{:}1})$ $(\overline{y{:}2} \ \ \overline{z{:}2})$ $(\overline{y{:}3} \ \ \overline{z{:}3})$
$(\overline{x{:}1} \ \ \overline{z{:}1})$ $(\overline{x{:}2} \ \ \overline{z{:}2})$ $(\overline{x{:}3} \ \ \overline{z{:}3})$
$(\overline{x{:}3})$ $(\overline{y{:}3})$ $(\overline{z{:}3})$

Definition 3.47 (constraint resolution, *C-RES* and *tree-C-RES*). A *constraint resolution* (*C-RES*) refutation of a CSP instance $\mathcal{I}$ is a *RES* refutation of CNF($\mathcal{I}$). A *tree-C-RES* refutation of $\mathcal{I}$ is a *tree-RES* refutation of CNF($\mathcal{I}$).

Theorem 3.48 (Soundness and Completeness of *C-RES*). *For any CSP instance $\mathcal{I}$, there is a C-RES refutation of $\mathcal{I}$ if and only if $\mathcal{I}$ is unsatisfiable.*

Proof. This follows from the soundness and completeness of *RES* and the fact that CNF($\mathcal{I}$) is satisfiable if and only if $\mathcal{I}$ is satisfiable. □

Definition 3.49 (*C-RES* complexity, *C-RES*($\mathcal{I}$) and *tree-C-RES*($\mathcal{I}$)). For any unsatisfiable CSP instance $\mathcal{I}$,

$$\textit{C-RES}(\mathcal{I}) \stackrel{\text{def}}{=} \min\{|\pi| : \pi \text{ is a } \textit{C-RES} \text{ refutation of } \mathcal{I}\}$$

and

$$\textit{tree-C-RES}(\mathcal{I}) \stackrel{\text{def}}{=} \min\{|\pi| : \pi \text{ is a } \textit{tree-C-RES} \text{ refutation of } \mathcal{I}\}.$$

3.3.1 Direct Translation of SAT to CSP

As CSPs with domain size 2 can be viewed as a generalization of SAT problems, if an unsatisfiable CNF formula ϕ is transformed directly into a CSP instance $\mathcal{I}$, one might expect that the *C-RES* complexity of CNF($\mathcal{I}$) should not be worse, at least within a constant factor, than the *RES* complexity of ϕ. We are going to show that this presumption is correct.

Definition 3.50 (direct translation of SAT to CSP). A CNF formula ϕ can be transformed into a CSP instance $\mathcal{I} = \langle\{0,1\}, \Gamma\rangle$ as follows. The variables in $\mathcal{I}$ would be the variables in ϕ. For each clause C in ϕ, there is a nogood $\eta(\alpha)$ in Γ if and only if α is a minimal size truth assignment that makes C false.

Example 3.51. The CNF formula $\phi = (x\ \overline{y}\ \overline{z}) \wedge (y\ \overline{z}) \wedge (\overline{x}) \wedge (z)$ can be transformed into a CSP instance $\mathcal{I} = \langle\{0,1\}, \Gamma\rangle$, where $\Gamma = \{\eta(x=0, y=1, z=1), \eta(y=0, z=1), \eta(x=1), \eta(z=0)\}$.

Proposition 3.52. *Let ϕ be a CNF formula. Let $\mathcal{I}$ be the CSP instance which is the direct transformation of ϕ to CSP. Then,*

1. $RES(\phi) \leq C\text{-}RES(\mathcal{I}) \leq 3 \cdot RES(\phi)$,
2. $tree\text{-}RES(\phi) \leq tree\text{-}C\text{-}RES(\mathcal{I}) \leq 3 \cdot tree\text{-}RES(\phi)$.

Proof. Let ϕ be a CNF formula. Let $\mathcal{I}$ be the CSP instance that is the direct transformation of ϕ to CSP. For $RES(\phi) < C\text{-}RES(\mathcal{I})$, suppose π is a *C-RES* refutation of $\mathcal{I}$, i.e., a *RES* refutation of CNF($\mathcal{I}$). We will construct a *RES* refutation of ϕ of size at most $|\pi|$. When ϕ is transformed to $\mathcal{I}$, each clause C in ϕ is mapped to a nogood N_C such that

$$x \in C \;\Leftrightarrow\; (x = 0) \in N_C \quad \text{and} \quad \overline{x} \in C \;\Leftrightarrow\; (x = 1) \in N_C.$$

Then, when $\mathcal{I}$ is transformed to CNF($\mathcal{I}$), each N_C is encoded as a constraint clause $\widehat{C}$ where

$$(x = 0) \in N_C \;\Leftrightarrow\; \overline{x\!:\!0} \in \widehat{C} \quad \text{and} \quad (x = 1) \in N_C \;\Leftrightarrow\; \overline{x\!:\!1} \in \widehat{C}.$$

Now, construct $\widehat{\pi}$ by replacing all occurrences of $\overline{x\!:\!0}$ and $x\!:\!1$ in π with x, and all $\overline{x\!:\!1}$ and $x\!:\!0$ with $\overline{x}$. The constraint clauses in π will be converted back to clauses in ϕ. Domain and unique value clauses will become $(x \;\; \overline{x})$ which is a tautology. So, $\widehat{\pi}$ is a *RES* refutation of the formula $\phi \wedge (x \;\; \overline{x})$. To eliminate $(x \;\; \overline{x})$ in $\widehat{\pi}$, we modify each derivation step involving $(x \;\; \overline{x})$ according to one of the following:

$$\frac{(x \;\; \overline{x}) \quad (x \;\; X_1)}{(x \;\; X_1)} \quad \Rightarrow \quad (x \;\; X_1)$$

$$\frac{(x \;\; \overline{x}) \quad (\overline{x} \;\; X_2)}{(\overline{x} \;\; X_2)} \quad \Rightarrow \quad (\overline{x} \;\; X_2)$$

What we obtain is a *RES* refutation of ϕ of size at most $|\pi|$.

For $C\text{-}RES(\mathcal{I}) \leq 3 \cdot RES(\phi)$, suppose π is a *RES* refutation of ϕ. Let $\widehat{\;}$ be a mapping from clauses in ϕ to clauses in CNF($\mathcal{I}$) such that for C a clause in ϕ,

$$x \in C \;\Leftrightarrow\; \overline{x\!:\!0} \in \widehat{C} \quad \text{and} \quad \overline{x} \in C \;\Leftrightarrow\; \overline{x\!:\!1} \in \widehat{C}.$$

Then, for each derivation step

$$\frac{(x \;\; X_1) \quad (\overline{x} \;\; X_2)}{(X_1 \;\; X_2)}$$

in π, we change it to

$$\cfrac{\cfrac{(x{:}0 \;\; x{:}1) \qquad (\overline{x{:}0} \;\; \widehat{X_1})}{(x{:}1 \;\; \widehat{X_1}) \qquad (\overline{x{:}1} \;\; \widehat{X_2})}}{(\widehat{X_1} \;\; \widehat{X_2})} \; .$$

The resulting derivation is a *RES* refutation of CNF($\mathcal{I}$) and is of size at most three times of π.

The tree-like case is similar. □

3.3.2 Separation of *tree-C-RES* from *C-RES*

Employing the result from the previous subsection, together with Ben-Sasson's [7] exponential separation of *tree-RES* from *RES*, we get an exponential separation of *tree-C-RES* from *C-RES*.

Theorem 3.53 (Separation). *There exists an infinite family of CSP instances* $\{\mathcal{I}_n\}$ *such that*

1. *C-RES*$(\mathcal{I}_n) = O(n)$,
2. *tree-C-RES*$(\mathcal{I}_n) = 2^{\Omega(n/\log n)}$.

Proof. [7] presents an infinite family of unsatisfiable CNF formulas $\{\phi_n\}$ such that $|\phi_n| = O(n)$, *RES*$(\phi_n) = O(n)$ and *tree-RES*$(\phi_n) = 2^{\Omega(n/\log n)}$. Let $\mathcal{I}_n$ be the direct transformation of ϕ_n to CSP. By Proposition 3.52, we have

$$\textit{C-RES}(\mathcal{I}_n) \leq 3 \cdot \textit{RES}(\phi_n) = O(n) \qquad \text{and}$$

$$\textit{tree-C-RES}(\mathcal{I}_n) \geq \textit{tree-RES}(\phi_n) = 2^{\Omega(n/\log n)}.$$

□

3.3.3 Separation Upper Bound

Another immediate result from Ben-Sasson's paper is an upper bound for the separation between *tree-C-RES* and *C-RES*.

Theorem 3.54. *For any unsatisfiable CSP instance $\mathcal{I}$, if $S = C\text{-}RES(\mathcal{I})$, then $\text{tree-}C\text{-}RES(\mathcal{I}) = 2^{O(S \log\log S/\log S)}$.*

Proof. [7] proves that for any unsatisfiable CNF formula ϕ, if $S = RES(\phi)$, then $\text{tree-}RES(\phi) = 2^{O(S \log\log S/\log S)}$. Therefore, for every unsatisfiable CSP instance $\mathcal{I}$, if $S = C\text{-}RES(\mathcal{I}) = RES(\text{CNF}(\mathcal{I}))$, then $\text{tree-}C\text{-}RES(\mathcal{I}) = \text{tree-}RES(\text{CNF}(\mathcal{I})) = 2^{O(S \log\log S/\log S)}$. □

Chapter 4

Relative Efficiency of Resolution Systems

In this chapter, we compare the relative power of the CSP refutation systems *NG-RES*, *C-RES*, and their restricted tree-like versions. We first show that *NG-RES* can simulate *tree-C-RES* efficiently. Then we prove exponential separations between *tree-NG-RES* and *tree-C-RES*, and also between *NG-RES* and *C-RES*.

4.1 *tree-C-RES* vs *NG-RES*

We show that *NG-RES* can efficiently simulate *tree-C-RES* in this section. Throughout the proof, one can observe a close relationship between *NG-RES* and negative resolution (*N-RES*). An *NG-RES* refutation of a CSP instance $\mathcal{I}$ is essentially an *N-RES* refutation of CNF($\mathcal{I}$). For our convenience, we use *N-C-RES* to denote negative resolution for CNF($\mathcal{I}$).

Definition 4.1 (*N-C-RES*). An *N-C-RES* refutation of a CSP instance $\mathcal{I}$ is a negative resolution refutation of CNF($\mathcal{I}$).

4.1.1 Simulations

To prove the simulation, we start by showing that *NG-RES* and *N-C-RES* can simulate each other. Then, together with the fact that *N-RES* can simulate *tree-RES*, we can prove that *NG-RES* simulates *tree-C-RES*.

Proposition 4.2 (*NG-RES* and *N-C-RES* efficiently simulate each other). *For any n-variable CSP instance $\mathcal{I}$, with domain size d,*

1. *if there is an N-C-RES refutation of $\mathcal{I}$ of size S, then there is an NG-RES refutation of $\mathcal{I}$ of size at most S, and*

2. *if there is an NG-RES refutation of $\mathcal{I}$ of size S, then there is an N-C-RES refutation of $\mathcal{I}$ of size at most $dS + n$.*

Proof. Let $\mathcal{I}$ be an n-variable CSP instance with domain size d.

Suppose there is an *N-C-RES* refutation π of $\mathcal{I}$ of size S. Note that π is a negative resolution refutation of CNF($\mathcal{I}$). The only clauses in CNF($\mathcal{I}$) containing positive literals are the domain clauses. Since every resolution step in a negative refutation must involve a negative clause and the domain clauses are all positive, once we resolve a negative clause with a domain clause on some variable $x{:}a_i$, we have to keep resolving on the rest of the variables in the domain clause until we get a negative clause. Therefore, every negative clause in π must be either an element of CNF($\mathcal{I}$) or derived as follows:

$$\dfrac{\dfrac{\dfrac{(x{:}a_1 \;\; x{:}a_2 \;\; \ldots \;\; x{:}a_d) \qquad (\overline{x{:}a_1} \;\; X_1)}{(x{:}a_2 \;\; \ldots \;\; x{:}a_d \;\; X_1)} \qquad (\overline{x{:}a_2} \; X_2)}{(x{:}a_3 \;\; \ldots x{:}a_d \;\; X_1 \; X_2)}}{\ddots}$$

$$\dfrac{(x{:}a_d \;\; X_1 \;\; \ldots \;\; X_{d-1}) \qquad (\overline{x{:}a_d} \;\; X_d)}{(X_1 \; X_2 \; \ldots \; X_d)}$$

where each X_i contains negative literals only. We can construct an *NG-RES* refutation

of $\mathcal{I}$ by replacing each of the derivation steps which looks like the above with

$$\frac{\begin{array}{c}\eta(x = a_1,\ N_1)\\ \eta(x = a_2,\ N_2)\\ \vdots\\ \eta(x = a_d,\ N_d)\end{array}}{\eta(N_1, N_2, \ldots, N_d)}\quad x \in \{1, \ldots, d\}$$

where $\overline{v{:}a} \in X_i$ if and only if $(v = a) \in N_i,\ 1 \leq i \leq d$. So, there is an *NG-RES* refutation of $\mathcal{I}$ of size at most S.

For the second part, suppose π is an *NG-RES* refutation of $\mathcal{I}$ of size S. We construct an *N-C-RES* refutation π' of $\mathcal{I}$ by replacing every *NG-RES* derivation step in π with a corresponding resolution derivation that makes use of a domain clause (as shown in the *RES* derivation above). It is clear that π' is a negative resolution refutation. Since $(x{:}a_1\ \ x{:}a_2\ \ \ldots\ \ x{:}a_d)$ is a domain clause in CNF($\mathcal{I}$) and each leaf in π' is a constraint clause in CNF($\mathcal{I}$), π' is indeed an *N-C-RES* refutation of $\mathcal{I}$. There are at most n domain clauses and each nogood derived in π corresponds to d derived clauses in π'. Therefore, π' is of size at most $dS + n$. □

Proposition 4.3 (*N-RES* efficiently simulates *tree-RES*). *For any CNF formula ϕ with n variables, if there is a tree-RES refutation T of ϕ of size S, then there is an N-RES refutation π of ϕ of size at most nS.*

Proof. (By induction on n) The base step is trivial. Assume the claim is true for CNF formulas with less than n variables. Let ϕ be a CNF formula with n variables. Suppose there is a *tree-RES* refutation T of ϕ of size S. Let x be the variable which is resolved on to derive the empty clause in T. Let $T_{\overline{x}}$ be the *tree-RES* derivation of $\overline{x}$ and T_x be the *tree-RES* derivation of x. Then, $T_{\overline{x}}\lceil_{x=1}$ is a *tree-RES* refutation of $\phi\lceil_{x=1}$ using the resolution rule and possibly also the weakening rule (Proposition 3.7). By inductive hypothesis and Proposition 3.5, $\phi\lceil_{x=1}$ has an *N-RES* refutation $\pi_{\overline{x}}$ of size at most $(n-1)|\,T_{\overline{x}}|$. "Plugging" $\overline{x}$ back into $\pi_{\overline{x}}$ yields either an *N-RES* refutation of ϕ (we are done in this case) or an *N-RES* derivation of $\overline{x}$ from ϕ of size at most $(n-1)|\,T_{\overline{x}}|$ (see Proposition 3.8 for details).

Similarly, $T_x\lceil_{x=0}$ is a *tree-RES* refutation of $\phi\lceil_{x=0}$. By inductive hypothesis, $\phi\lceil_{x=0}$ has an *N-RES* refutation π_x of size at most $(n-1)|\,T_x|$. For each leaf C of π_x, if $C \notin \phi$, then $(C \quad x)$ must be in ϕ and C can be derived from $(C \quad x)$ and $\overline{x}$ (where $\overline{x}$ can be derived in at most $(n-1)|\,T_{\overline{x}}|$ steps). The number of such C's is at most $|\,T_x|$. Hence, we can construct an *N-RES* refutation of ϕ of size at most $(n-1)|\,T_{\overline{x}}| + (n-1)|\,T_x| + |\,T_x| \leq n(|\,T_{\overline{x}}| + |\,T_x| + 1) = nS$. □

Proposition 4.4 (*N-C-RES* efficiently simulates *tree-C-RES*). *For every n-variable CSP instance $\mathcal{I}$, with domain size d, if there is a tree-C-RES refutation T of $\mathcal{I}$ of size S, then there is an N-C-RES refutation π of $\mathcal{I}$ of size at most ndS.*

Proof. Let $\mathcal{I}$ be an n-variable CSP instance with domain size d. Suppose there is a *tree-C-RES* refutation T of $\mathcal{I}$ of size S. Then T is a tree-like resolution refutation of CNF($\mathcal{I}$) which has nd variables. By Proposition 4.3, there exists an *N-RES* refutation π of CNF($\mathcal{I}$) of size at most ndS. By definition, π is an *N-C-RES* refutation of $\mathcal{I}$. Therefore, there is an *N-C-RES* refutation of $\mathcal{I}$ of size at most ndS. □

Corollary 4.5 (*NG-RES* efficiently simulates *tree-C-RES*). *For every n-variable CSP instance $\mathcal{I}$, with domain size d, if there is a tree-C-RES refutation T of $\mathcal{I}$ of size S, then there is an NG-RES refutation π of $\mathcal{I}$ of size at most ndS.*

4.2 *tree-NG-RES* vs *tree-C-RES*

In this section, we compare the relative power of *tree-NG-RES* and *tree-C-RES*. We show that *tree-C-RES* efficiently simulates *tree-NG-RES*, but the converse does not hold. There is an exponential separation of *tree-NG-RES* from *tree-C-RES*.

4.2.1 Simulation

Any *tree-NG-RES* refutation can be efficiently transformed into a *tree-C-RES* refutation with only a small blowup in size.

Proposition 4.6 (*tree-C-RES* efficiently simulates *tree-NG-RES*). *For any CSP instance $\mathcal{I}$, if there is a tree-NG-RES refutation of $\mathcal{I}$ of size S, then there is a tree-C-RES refutation of $\mathcal{I}$ of size at most $2S$.*

Proof. Let $\mathcal{I}$ be a CSP instance and d be the domain size of $\mathcal{I}$. Let π be a *tree-NG-RES* refutation of $\mathcal{I}$. We can transform π into a *tree-C-RES* refutation by replacing every *NG-RES* derivation step with a *tree-C-RES* derivation (as in the proof for Proposition 4.2). Note that in a tree-like derivation, every nogood (or clause) is used at most once to derive other nogoods (or clauses). For each *NG-RES* derivation step with d input nogoods, the corresponding *tree-C-RES* derivation has $2d$ input and intermediate clauses. Hence, the resulting *tree-C-RES* refutation is of size at most $2S$. □

4.2.2 Separation of *tree-NG-RES* from *tree-C-RES*

The implication graph contradictions defined in Chapter 3 have poly-size *tree-C-RES* refutations. Hence, they also separate *tree-NG-RES* from *tree-C-RES*.

Lemma 4.7. *For any circuit G with n vertices, tree-C-RES($IMP_{G,S,T,d}$) $= O(d^2 n)$.*

Proof. Let $G = (V, E)$ be a circuit with n vertices. Let S and T be the sets of sources and targets in G. CNF($IMP_{G,S,T,d}$) consists of the following clauses:

Source axioms: $(\overline{x_i : 1})$ for every $v_i \in S$

Target axioms: $(\overline{x_i : a})$ for every $v_i \in T$, and for all $a \in [d] \setminus \{1\}$

Pebbling axioms: $(\overline{x_i : a}\ \overline{x_j : b}\ \overline{x_k : 1})$ for every v_k with predecessors v_i and v_j, and for all $a, b \in [d] \setminus \{1\}$

Domain clauses: $(x_i : 1\ x_i : 2\ \cdots\ x_i : d)$ for every $v_i \in V$

Resolving the domain clauses with the Target axioms, one can derive $(x_t : 1)$ for each target v_t. Then, one can derive $(x_i : 1\ x_j : 1\ \overline{x_k : 1})$ for each internal vertex v_k with predecessors v_i and v_j by resolving the Pebbling axioms with domain clauses. This derivation is of size $O(d^2 n)$.

Pick a target v_t in G and infer $(x_i{:}1\ x_j{:}1)$ by resolving $(x_t{:}1)$ and $(x_i{:}1\ x_j{:}1\ \overline{x_t{:}1})$ together, where v_i and v_j are predecessors of v_t. Let $L = \{v_i, v_j\}$ and $C = (x_i{:}1\ x_j{:}1)$. We then recursively do the following until L becomes empty. Take a vertex v_i from L. If v_i has predecessors v_1 and v_2 (i.e., v_i is not a source), then let C be the resolvent of C and $(x_1{:}1\ x_2{:}1\ \overline{x_i{:}1})$, and add v_1 and v_2 to L. Remove v_i from L.

Eventually, we will have a clause $(x_{i_1}{:}1\ x_{i_2}{:}1\ \cdots\ x_{i_m}{:}1)$ such that $v_{i_1}, v_{i_2}, \cdots,$ and v_{i_m} are all sources. Together with the Source axioms $(\overline{x_{i_j}{:}1})$, $1 \le j \le m$, we can derive the empty clause. This requires at most n derivation steps (one for each vertex). Since each derived clause is used at most once to derive other clauses, it is a tree-like derivation. Therefore, *tree-C-RES*$(IMP_{G,S,T,d}) = O(d^2 n)$. □

Theorem 4.8 (Separation). *For every integer $d \ge 3$, there exists an infinite family of CSP instances $\{\mathcal{I}_n\}$, with domain size d, such that*

1. *tree-C-RES*$(\mathcal{I}_n) = O(d^2 n)$,
2. *tree-NG-RES*$(\mathcal{I}_n) = (d-1)^{\Omega(n/\log n)}$.

Proof. Let $d \ge 3$ be some fixed integer. [12] gives an infinite family of circuits $\{G_n\}$, with $|V(G_n)| = n$, for which $P_{G_n}(S,T) = \Omega(n/\log n)$ where S and T are the sets of sources and targets in G. Lemma 4.7 implies that *tree-C-RES*$(IMP_{G_n,S,T,d}) = O(d^2 n)$. Moreover, from Theorem 3.34, we know that *tree-NG-RES*$(IMP_{G_n,S,T,d}) = \Omega((d-1)^{P_{G_n}(S,T)}) = (d-1)^{\Omega(n/\log n)}$. □

4.2.3 Separation Upper Bound

The separation upper bound presented below follows immediately from our earlier results. Most likely, there exists a slightly better bound which may be obtained by a direct proof. We will leave this as a possible future work.

Theorem 4.9. *For any n-variable unsatisfiable CSP instance $\mathcal{I}$ with domain size d, if $S =$ tree-C-RES$(\mathcal{I})$, then tree-NG-RES$(\mathcal{I}) = d^{O(nd^3 S \log\log S/\log S)}$.*

Proof. Let $S =$ *tree-C-RES*$(\mathcal{I})$. By Corollary 4.5, *NG-RES*$(\mathcal{I}) \le ndS$. So, by the upper bound on the separation between *NG-RES* and *tree-NG-RES*(Theorem 3.45), we have *tree-NG-RES*$(\mathcal{I}) = d^{O(nd^3 S \log\log S/\log S)}$. □

4.3 *NG-RES* vs *C-RES*

We now turn to the major result we obtained. We present a new exponential separation between *NG-RES* and *C-RES* by constructing an infinite family of CSP instances such that there exist poly-size *C-RES* refutations for the instances but any *NG-RES* refutation of them is of exponential size.

4.3.1 Simulation

Before proceeding to show the separation, we give a corollary stating that *C-RES* simulates *NG-RES* efficiently.

Proposition 4.10 (*C-RES* efficiently simulates *NG-RES*). *For every n-variable CSP instance $\mathcal{I}$, with domain size d, if there is an NG-RES refutation of $\mathcal{I}$ of size S, then there is a C-RES refutation of $\mathcal{I}$ of size at most $dS + n$.*

Proof. Since an *N-C-RES* refutation is a *C-RES* refutation and an *NG-RES* refutation of size S can be transformed into an *N-C-RES* refutation of size at most $dS + n$ (Proposition 4.2), the claim is true. □

4.3.2 Separation of *NG-RES* from *C-RES*

The best obviously known separation between *C-RES* and *NG-RES* is a super-polynomial one.

Theorem 4.11 (Mitchell [27]). *There is an infinite family of CSP instances MPH_n such that*

1. *C-RES(MPH_n) = $O(n^3)$,*
2. *NG-RES(MPH_n) = $n^{\Omega(\log n)}$.*

We improve the separation from super-polynomial to exponential by showing that there is an infinite family of CSP instances MGT'_n such that *C-RES*$(MGT'_n) = O(n^3)$ but *NG-RES*$(MGT'_n) = 2^{\Omega(n)}$.

Our family of CSP instances, MGT'_n, is based on the unsatisfiable CNF formula GT_n introduced by Krishnamurthy [22]. For each $n \in \mathbb{N}$, GT_n encodes the negation of the fact that every loopless transitive directed graph with n vertices and with no 2-cycles must have a source. The contradictory statement can be stated as a CNF formula containing the following clauses:

$$\begin{array}{lll} (1) & \overline{x_{j,j}} & j \in [n] \\ (2) & x_{i,j} \wedge x_{j,k} \rightarrow x_{i,k} & i,j,k \in [n],\ i \neq j \neq k \\ (3) & x_{i,j} \rightarrow \overline{x_{j,i}} & i,j \in [n],\ i \neq j \\ (4) & \bigvee_{i \in [n]} x_{i,j} & j \in [n] \end{array}$$

where $x_{i,j}$ takes value 1 if and only if there is an edge from i to j.

The first three sets of clauses ensure that the graph is loopless, transitive, and free of 2-cycles, respectively. The clauses in (4) assure that for each vertex j, there exists some vertex i such that there is an edge from i to j, i.e., there is no source. It has been proven by Stålmarck [37] that there is an $O(n^3)$-size resolution refutation of GT_n.

Bonet and Galesi gave a modified version of GT_n, MGT_n, in [10]. For each $j \in [n]$, they introduced $n+1$ new variables $y_{0,j}, \ldots, y_{n,j}$ and replaced the clauses in (4) by:

$$(4^*) \quad \overline{y_{0,j}} \ \wedge \bigwedge_{i \in [n]} (y_{i-1,j} \vee x_{i,j} \vee \overline{y_{i,j}}) \ \wedge \ y_{n,j} \qquad j \in [n]$$

The total number of variables is still $O(n^2)$ but MGT_n has constant width clauses. It is easy to see that we can derive the clauses in (4) from those in (4*) by resolving on the y variables and this takes $O(n^2)$ steps. Then, by applying Stålmarck's $O(n^3)$-size refutation of GT_n, we can obtain an $O(n^3)$-size resolution refutation for MGT_n. (From this refutation, we will later construct an analogous $O(n^3)$-size *C-RES* refutation for MGT'_n.) To show that our instance MGT'_n is hard for *NG-RES*, we will need the following important property of MGT_n (proof in Appendix B):

Theorem 4.12 (Bonet and Galesi [10]). *Any resolution refutation of MGT_n has width $\Omega(n)$.*

We are now ready to define our CSP instances that achieve the lower bound. MGT'_n has the same set of variables as MGT_n and the domain for each variable is

$D = \{1, 2, 3, 4\}$. If α is an assignment for MGT'_n, then

$$\alpha(x_{i,j}) = \begin{cases} 1 \text{ or } 2 & \text{means there exists an edge from } i \text{ to } j \\ 3 \text{ or } 4 & \text{means there is no edge from } i \text{ to } j. \end{cases}$$

So, every total assignment for the variables in MGT'_n corresponds to a directed graph with n vertices. To encode the contradictory statement, MGT'_n consists of the following nogoods:

$(1')$	$\eta(x_{j,j} = 1), \eta(x_{j,j} = 2)$	$j \in [n]$
$(2')$	$\eta(x_{i,j} = a,\ x_{j,k} = b,\ x_{i,k} = c)$	$i, j, k \in [n],\ i \neq j \neq k,$ $a, b \in \{1, 2\},\ c \in \{3, 4\}$
$(3')$	$\eta(x_{i,j} = a,\ x_{j,i} = b)$	$i, j \in [n],\ i \neq j,\ a, b \in \{1, 2\}$
$(4')$	for each $i \in [n]$, $\eta(y_{0,j} = 1), \eta(y_{0,j} = 2)$ $\eta(y_{i-1,j} = c,\ x_{i,j} = a,\ y_{i,j} = b)$ $\eta(y_{n,j} = 3), \eta(y_{n,j} = 4)$	$j \in [n],\ a, b \in \{1, 2\},\ c \in \{3, 4\}$

We first show that there exists an $O(n^3)$-size *C-RES* refutation of MGT'_n.

Theorem 4.13. *C-RES*$(MGT'_n) = O(n^3)$.

Proof. The CNF encoding of MGT'_n contains the following clauses:

$(1'')$	$(\overline{x_{j,j} : 1}), (\overline{x_{j,j} : 2})$	$j \in [n]$
$(2'')$	$(\overline{x_{i,j} : a}\ \ \overline{x_{j,k} : b}\ \ \overline{x_{i,k} : c})$	$i, j, k \in [n],\ i \neq j \neq k,$ $a, b \in \{1, 2\},\ c \in \{3, 4\}$
$(3'')$	$(\overline{x_{i,j} : a}\ \ \overline{x_{j,i} : b})$	$i, j \in [n],\ i \neq j,\ a, b \in \{1, 2\}$
$(4'')$	for each $i \in [n]$, $(\overline{y_{0,j} : 1}), (\overline{y_{0,j} : 2})$ $(\overline{y_{i-1,j} : c}\ \ \overline{x_{i,j} : a}\ \ \overline{y_{i,j} : b})$ $(\overline{y_{n,j} : 3}), (\overline{y_{n,j} : 4})$	$j \in [n],\ a, b \in \{1, 2\},\ c \in \{3, 4\}$

$$\text{Domain clauses:} \quad \begin{array}{l}(x_{i,j}{:}1 \;\; x_{i,j}{:}2 \;\; x_{i,j}{:}3 \;\; x_{i,j}{:}4)\\ (y_{i,j}{:}1 \;\; y_{i,j}{:}2 \;\; y_{i,j}{:}3 \;\; y_{i,j}{:}4)\end{array}$$

With the clauses in $(2'')$ and the domain clauses of the x variables, we can derive the clauses $(\overline{x_{i,j}{:}a} \;\; \overline{x_{j,k}{:}b} \;\; x_{i,k}{:}1 \;\; x_{i,k}{:}2)$, $i,j,k \in [n]$, $i \neq j \neq k$, $a,b \in \{1,2\}$, in $O(n^3)$ steps since there are $O(n^3)$ of them. Define

$$A(i,j,k) \stackrel{def}{=} \bigwedge_{a,b\in\{1,2\}} (\overline{x_{i,j}{:}a} \;\; \overline{x_{j,k}{:}b} \;\; x_{i,k}{:}1 \;\; x_{i,k}{:}2).$$

Then, by resolving the clauses in $(4'')$ together with the domain clauses of the y and x variables, we obtain the clauses $(X(1,j) \;\; X(2,j) \;\; \cdots \;\; X(n,j))$ for each $j \in [n]$, where $X(i,j) \stackrel{def}{=} (x_{i,j}{:}1 \;\; x_{i,j}{:}2)$. After that, by using the unit clauses in $(1'')$, we get the clauses $P_n(j)$ for each $j \in [n]$, where $P_m(j)$ is defined as

$$P_m(j) \stackrel{def}{=} \bigvee_{\substack{i\in[n]\\ i\neq j}} X(i,j).$$

All these take $O(n^2)$ steps. We also define $B(m,j)$ as

$$B(m,j) \stackrel{def}{=} \bigwedge_{a,b\in\{1,2\}} (\overline{x_{m,j}{:}a} \;\; \overline{x_{j,m}{:}b})$$

which are just the clauses in $(3'')$.

Now, for each $m < n$ and $j \leq m$, we can derive $P_m(j)$ from $P_{m+1}(j), A(i,m+1,j)$, and $B(m+1,j)$ as shown in Figure 4.1. Once we get $P_2(1)$ and $P_2(2)$, the empty clause can be derived in six steps (Figure 4.2). The *C-RES* derivation of $P_m(j)$ is of size $O(n)$. Therefore, we need $O(n^3)$ steps in total to derive the empty clause.

□

We complete the separation by proving an exponential lower bound for *NG-RES* on MGT'_n. The proof approach is inspired by [11]. We show that if there is a short *NG-RES* refutation of MGT'_n, then we can construct a narrow resolution refutation of MGT_n. Then, as we already knew that every refutation of MGT_n is wide, we can conclude that every *NG-RES* refutation must be exponentially long.

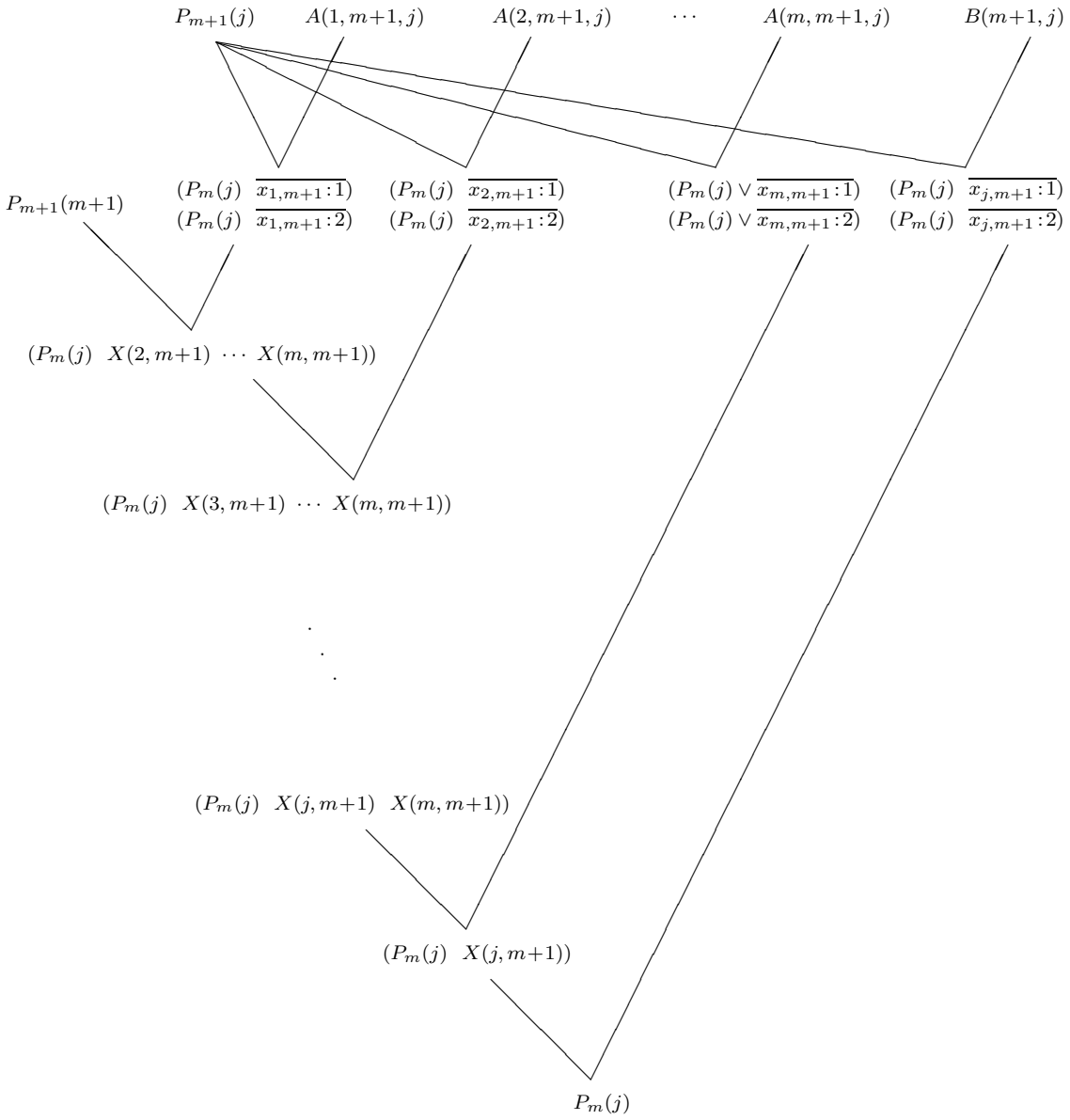

Figure 4.1: *C-RES* derivation of $P_m(j)$

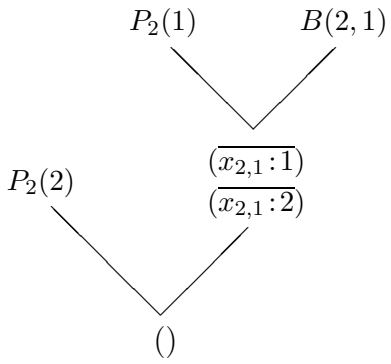

Figure 4.2: *C-RES* refutation from $P_2(1), P_2(2)$, and $B(2,1)$

Definition 4.14 (restriction). A *restriction* for a CSP instance $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ forbids some variables to take some domain values. A restriction ρ is written as a set of variables with the forbidden values. For example, the restriction $\rho = \{x \neq 2, x \neq 3, y \neq 1\}$ disallows x to take 2 and 3, and y to take 1.

Let $\rho = \{x_1 \neq a_1, x_2 \neq a_2, \cdots, x_k \neq a_k\}$ be a restriction. Define $N\lceil_\rho$ as the result of applying ρ to a nogood N where

$$N\lceil_\rho \stackrel{def}{=} (\cdots (N\lceil_{x_a \neq a_1})\lceil_{x_2 \neq a_2}) \cdots \lceil_{x_k \neq a_k}),$$

and for x a variable, $a \in \mathcal{D}(x)$,

$$N\lceil_{x \neq a} \stackrel{def}{=} \begin{cases} 1 & \text{if } (x = a) \in N \\ N & \text{otherwise.} \end{cases}$$

We define $\mathcal{I}\lceil_\rho \stackrel{def}{=} \langle \mathcal{D}\lceil_\rho, \Gamma\lceil_\rho \rangle$, where

$$\begin{aligned} \Gamma\lceil_\rho &= \{N : N \in \Gamma \text{ and } N\lceil_\rho \neq 1\} \\ \mathit{vars}(\mathcal{I}\lceil_\rho) &= \mathit{vars}(\mathcal{I}) \\ \mathcal{D}\lceil_\rho(x) &= \mathcal{D}(x) \setminus \{a : (x \neq a) \in \rho\} \qquad \text{for all } x \in \mathit{vars}(\mathcal{I}\lceil_\rho). \end{aligned}$$

If $\pi = (N_1, N_2, \cdots, N_S)$ is an *NG-RES* derivation, define $\pi\lceil_\rho$ to be $(N_1\lceil_\rho, \cdots, N_S\lceil_\rho)$, but with any $N_i\lceil_\rho$ that is identical to 1 removed. Note that $\pi\lceil_\rho$ is actually a subsequence of π.

Lemma 4.15. *If π is an NG-RES refutation of a CSP instance $\mathcal{I}$ and ρ is a restriction, then there is an NG-RES refutation of $\mathcal{I}\lceil_\rho$ of width at most $w(\pi\lceil_\rho)$.*

Proof. It is enough to show the claim for just the unit case $\rho = \{x \neq a\}$ and the general case will follow from it.

Let $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ be a CSP instance and $\rho = \{x \neq a\}$ be a restriction. Let π be an *NG-RES* refutation of $\mathcal{I}$. Transform $\pi\lceil_\rho$ inductively to an *NG-RES*$^{+weak}$ refutation π' as follows. Consider a nogood N_i in $\pi\lceil_\rho$. $(x = a)$ must not appear in N_i since $N_i\lceil_{x\neq a} \neq 1$. If $N_i \in \Gamma$, then $N_i \in \Gamma\lceil_\rho$. (Note that $\pi\lceil_\rho$ is a subsequence of π.) Otherwise, N_i must be derived, in π, by resolving some previous nogoods $N_{i_1}, \cdots, N_{i_d}$ on some variable v. If $v \neq x$, then $(x = a)$ does not appear in any of $N_{i_1}, \cdots, N_{i_d}$ because $(x = a) \notin N_i$. So, $N_{i_1}, \cdots, N_{i_d}$ must be in $\pi\lceil_\rho$ and they can be resolved to derive N_i in $\pi\lceil_\rho$. If $v = x$, then there is a nogood $N_{i_a} \in \{N_{i_1}, \cdots, N_{i_d}\}$ such that $N_{i_a} = \eta(x = a, N_a)$ and thus $N_{i_a}\lceil_\rho$ is not in $\pi\lceil_\rho$ since $N_{i_a}\lceil_\rho = 1$. But, all the nogoods in $\{N_{i_1}, \cdots, N_{i_d}\} \setminus \{N_{i_a}\}$ are in $\pi\lceil_\rho$. So, we can resolve them on x, over the new domain of x, to get a subnogood of N_i. Then, we can derive N_i using the nogood weakening rule.

Therefore, π' is an *NG-RES*$^{+weak}$ refutation of $\mathcal{I}\lceil_\rho$ of width $w(\pi\lceil_\rho)$. By Proposition 3.18, there is an *NG-RES* refutation of $\mathcal{I}\lceil_\rho$ of width at most $w(\pi\lceil_\rho)$. □

Lemma 4.16. *If there is an NG-RES refutation of MGT'_n of size at most S, then there is a resolution refutation of MGT_n of width at most w, for any $w > \log S$.*

Proof. Let π be an *NG-RES* refutation of MGT'_n of size at most S. Let $w > \log S$. Define that a nogood is wide if its width is greater than w.

Define a random restriction ρ as follows. For each variable $v_{i,j}$, $v \in \{x, y\}$, ρ randomly picks a value a from $\{1, 2\}$ and a value c from $\{3, 4\}$, and restricts that $v_{i,j} \neq a$ and $v_{i,j} \neq c$. So, for every variable, a domain value is prohibited by ρ with probability $1/2$.

We say that a restriction is bad if not all wide nogoods in π are set to 1 by ρ. A wide nogood would be set to 1 by ρ if there is some literal, $(x = a)$, in it such that $(x \neq a) \in \rho$. The probability that this is not the case is at most $1/2^w$. Since there is at most S nogoods in π, the probability that ρ is bad is at most $S/2^w$ which is less

than 1 as we have $w > \log S$. Therefore, there must exist at least one good restriction which would set all wide nogoods in π to 1.

Apply a good restriction ρ to π. By Lemma 4.15, there is an *NG-RES* refutation π' of $MGT'_n\lceil_\rho$ of width at most w. After we apply ρ to MGT'_n, some initial nogoods disappears. For example, for each j, two of the nogoods in $(1')$ are set to 1 by ρ and thus not included in $MGT'_n\lceil_\rho$. Moreover, the domain size of each variable becomes 2.

Therefore, the CNF encoding of $MGT'_n\lceil_\rho$ consists of the following clauses:

$$
\begin{array}{lll}
(1'') & (\overline{x_{j,j}:a_{j,j}}) & j \in [n] \\
(2'') & (\overline{x_{i,j}:a_{i,j}} \;\; \overline{x_{j,k}:a_{j,k}} \;\; \overline{x_{i,k}:c_{i,k}}) & i,j,k \in [n],\ i \neq j \neq k \\
(3'') & (\overline{x_{i,j}:a_{i,j}} \;\; \overline{x_{j,i}:a_{j,i}}) & i,j \in [n],\ i \neq j \\
(4'') & (\overline{y_{0,j}:b_{0,j}}) & \\
 & \bigwedge_{i\in[n]} (\; \overline{y_{i-1,j}:c_{i-1,j}} \;\; \overline{x_{i,j}:a_{i,j}} \;\; \overline{y_{i,j}:b_{i,j}} \;) & j \in [n] \\
 & (\overline{y_{n,j}:d_{n,j}}) &
\end{array}
$$

Domain clauses: $(x_{i,j}:a_{i,j} \;\; x_{i,j}:c_{i,j})$
$(y_{i,j}:b_{i,j} \;\; y_{i,j}:d_{i,j})$

where each of $a_{i,j}$'s and $b_{i,j}$'s is equal to either 1 or 2 and each of $c_{i,j}$'s and $d_{i,j}$'s is equal to either 3 or 4.

Rename the variables $\overline{x_{i,j}:a_{i,j}}$, $\overline{x_{i,j}:c_{i,j}}$, $\overline{y_{i,j}:b_{i,j}}$, and $\overline{y_{i,j}:d_{i,j}}$ as $\overline{x_{i,j}}$, $x_{i,j}$, $\overline{y_{i,j}}$, and $y_{i,j}$, respectively. Now the constraint clauses of $\mathrm{CNF}(MGT'_n\lceil_\rho)$ are exactly the clauses in MGT_n and the *NG-RES* derivation steps

$$
\frac{\eta(x_{i,j} = a_{i,j},\ N_1) \quad \eta(x_{i,j} = c_{i,j},\ N_2)}{\eta(N_1, N_2)} \;\; x_{i,j} \in \{a_{i,j}, c_{i,j}\} \qquad \text{and} \qquad \frac{\eta(y_{i,j} = b_{i,j},\ N_1) \quad \eta(y_{i,j} = d_{i,j},\ N_2)}{\eta(N_1, N_2)} \;\; y_{i,j} \in \{b_{i,j}, d_{i,j}\}
$$

in π' can be transformed into the following resolution derivation steps

$$
\frac{(\overline{x_{i,j}} \;\; X_1) \quad (x_{i,j} \;\; X_2)}{(X_1 \;\; X_2)} \qquad \text{and} \qquad \frac{(\overline{y_{i,j}} \;\; X_1) \quad (y_{i,j} \;\; X_2)}{(X_1 \;\; X_2)}\ .
$$

The resulting resolution refutation has the same width as π'. Hence, there is a resolution refutation of MGT_n of width at most w. □

Theorem 4.17. *Any NG-RES refutation of MGT'_n must have size $2^{\Omega(n)}$.*

Proof. Let π be an *NG-RES* refutation of MGT'_n. Let S be the size of π. Pick $w = \log S + \epsilon,\ \epsilon > 0$. It follows from Lemma 4.16 that MGT_n has a resolution refutation π' of width at most $\log S + \epsilon$. We know that any resolution refutation of MGT_n must have width $\Omega(n)$ (Theorem 4.12). Therefore, $\log S + \epsilon \geq \Omega(n)$, and thus $S \geq 2^{\Omega(n)}$. Hence, any *NG-RES* refutation of MGT'_n must have size $2^{\Omega(n)}$. □

Combining Theorem 4.13 and 4.17, we get an exponential separation between *NG-RES* and *C-RES*.

Theorem 4.18 (Separation). *There is an infinite family of CSP instances MGT'_n such that*

1. *C-RES(MGT'_n) = $O(n^3)$,*
2. *NG-RES(MGT'_n) = $2^{\Omega(n)}$.*

Proof. It follows immediately from Theorem 4.13 and 4.17. □

4.3.3 Separation Upper Bound

We conclude our discussion with an upper bound for the separation between *NG-RES* and *C-RES*.

Theorem 4.19. *For any n-variable unsatisfiable CSP instance $\mathcal{I}$ with domain size d, if S = C-RES($\mathcal{I}$), then NG-RES($\mathcal{I}$) = $2^{O(S \log\log S/\log S)}$.*

Proof. Let S = *C-RES*($\mathcal{I}$). By Theorem 3.54, *tree-C-RES*($\mathcal{I}$) = $2^{O(S \log\log S/\log S)}$. Then, by Corollary 4.5, *NG-RES*($\mathcal{I}$) $\leq nd2^{O(S \log\log S/\log S)} = 2^{O(S \log\log S/\log S)}$. □

Chapter 5

Conclusion and Future Work

5.1 Summary

In this thesis, we studied the proof systems *NG-RES* and *C-RES* which correspond to d-way branching and 2-way branching respectively. We showed exponential separations among the systems and their restricted tree-like variants. The relative efficiency of the proof systems are summarized in Figure 5.1. A "$\rightarrow$" arrow from A to B means that a B refutation can be transformed into an A refutation with only a small blowup in size (blowup factor labelled on the arrow). A "$\nrightarrow$" arrow from A to B means that B is strictly more powerful than A. For each separation presented, we also proved an upper limit on the separation. These upper bounds show that most of the separations are nearly optimal.

The exponential separation obtained between *tree-NG-RES* and *tree-C-RES* provides valuable information on the power of d-way and 2-way branching backtracking algorithms. We presented an infinite family of CSP instances $IMP_{G_n,S,T,d}$ and proved that there is an $O(d^2n)$-size *tree-C-RES* refutation of $IMP_{G_n,S,T,d}$, but any *tree-NG-RES* refutation of $IMP_{G_n,S,T,d}$ must be of size $(d-1)^{\Omega(n/\log n)}$. Hence, there exists a 2-way branching algorithm which can solve $IMP_{G_n,S,T,d}$ in polynomail time (assuming optimal branching choices can be made in polynomial time), but any *tree-NG-RES* bounded algorithm, including d-way branching enhanced with backjumping and forward checking, requires $(d-1)^{\Omega(n/\log n)}$ time to refute $IMP_{G_n,S,T,d}$.

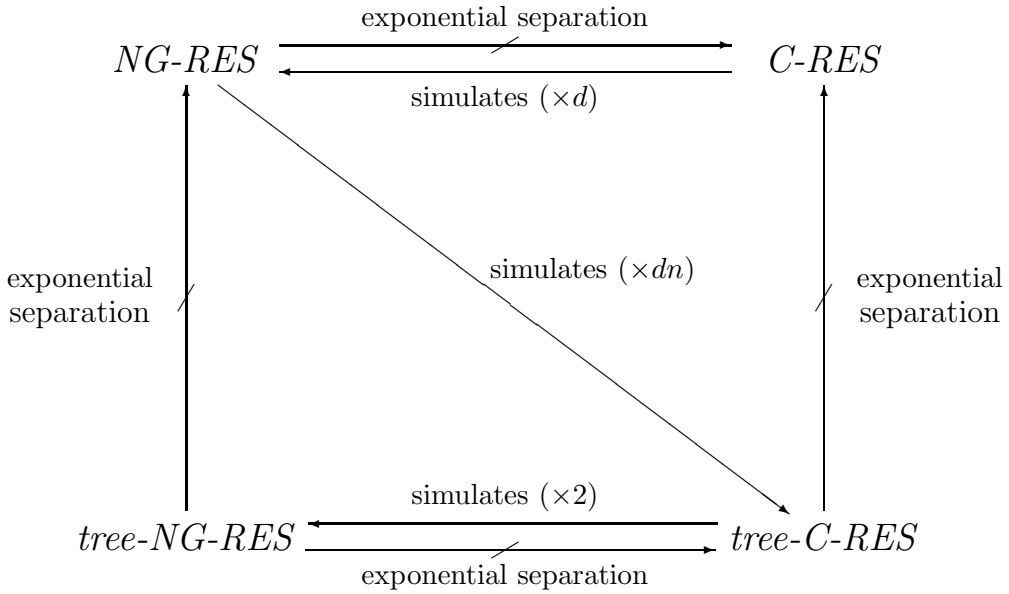

Figure 5.1: Relative power of *NG-RES*, *C-RES* and their tree-like variants

Our results also show that there are instances which are exponentially hard for *NG-RES*, *tree-NG-RES* and *tree-C-RES*, and thus they are also exponentially hard for *NG-RES* bounded, *tree-NG-RES* bounded and *tree-C-RES* bounded algorithms respectively in the sense that the algorithms cannot solve the corresponding instances in less than exponential time.

Current commercial CSP solvers, such as ILOG Solver and ECL$^{\text{i}}$PS$^{\text{e}}$, use 2-way branching by default. However, the option of branching on another variable rather than trying other values of the same variable after backtracking is not implemented in these solvers. According to the poly-size tree-like *C-RES* refutations of the instances that separate d-way branching from 2-way branching, the advantage of trying any variable after backtracking in 2-way branching is one of the main factors that makes 2-way branching much stronger than d-way branching. Further studies on variable and value ordering heuristics for 2-way branching should be done to find out how to obtain benefits from 2-way branching. We also expect that learning will become an essential technique in effective CSP algorithms in the future.

5.2 Future Work

Several suggested directions for future research are listed:

1. Perform an experimental comparison of d-way branching and 2-way branching on our hard CSP instances. The instances used to compare d-way branching algorithms and 2-way branching algorithms in previous work did not yield significant difference between the power of the algorithms. It is worth to run the same algorithms on our instances to see how they perform.

2. Study variable and value ordering heuristics for 2-way branching. One possible approach is to investigate our CSP instances that separate (tree-like) *NG-RES* from (tree-like) *C-RES* and find out how to gain advantages from 2-way branching.

3. Further exploration of other restricted versions of *NG-RES*. This may lead to a better understanding of the power of different techniques used in common CSP

algorithms. For example, in an *ordered NG-RES* refutation, every sequence of variables labelled on a path from the empty nogood to a nogood in the refutation must respect to the same variable ordering. The relative efficiency of this system and general *NG-RES* will thus be useful in analyzing static variable ordering heuristics in CSP algorithms.

4. Find a 2-way branching strategy to solve MGT'_n in $O(n^3)$ time. That is, establish a variable branching ordering and a learning strategy to build into a 2-way branching algorithm so that it will always solve MGT'_n in $O(n^3)$ time. If such a strategy exists, then the current super-polynomial separation between d-way branching algorithms with learning and 2-way branching algorithms with learning can be improved to exponential.

5. Investigate the relative power between *NG-RES* and d-way branching with learning, and also between *C-RES* and 2-way branching with learning, and try to achieve analogous relationships as between resolution and clause learning algorithms.

6. Improve the gap between the lower bound and upper bound of the separation of *NG-RES* from *C-RES* to make it tight.

Appendix A

Proof of the Inequality $\binom{k2^k}{2^k} \leq (4k)^{2^k}$

This appendix contains a proof of the inequality used in Theorem 3.44.

Lemma A.1. $\binom{k2^k}{2^k} \leq (4k)^{2^k}$ *for all integers* $k \geq 4$.

Proof. (By induction on k)

Basis: For $k = 4$, $\binom{4\cdot 2^4}{2^4} = 4.88 \times 10^{14} \leq 1.84 \times 10^{19} = (4 \cdot 4)^{2^4}$.

I.H.: Assume the claim holds for all values smaller than some fixed k.

Induction step:

$$
\begin{aligned}
\binom{k2^k}{2^k} &= \frac{k2^k!}{2^k!(k2^k - 2^k)!} \\
&= \frac{(k2^k) \cdot (k2^k - 1) \cdots (k2^{k-1} + 1) \cdot (k2^{k-1}) \cdots ((k-1)2^{k-1} + 1) \cdot ((k-1)2^{k-1})!}{2^k \cdot (2^k - 1) \cdots (2^{k-1} + 1) \cdot 2^{k-1}! \cdot (k2^k - 2^k) \cdots (k2^{k-1} - 2^k + 1) \cdot (k2^{k-1} - 2^k)!} \\
&= \binom{(k-1)2^{k-1}}{2^{k-1}} \frac{(k2^k) \cdot (k2^k - 1) \cdots (k2^{k-1} + 1)}{(k2^k - 2^k) \cdots (k2^{k-1} - 2^k + 1)} \cdot \frac{k2^{k-1} \cdots ((k-1)2^{k-1} + 1)}{2^k \cdot (2^k - 1) \cdots (2^{k-1} + 1)} \\
&\leq (4(k-1))^{2^{k-1}} \cdot \frac{(k2^k) \cdot (k2^k - 1) \cdots (k2^{k-1} + 1)}{(k2^k - 2^k) \cdots (k2^{k-1} - 2^k + 1)} \cdot \frac{k2^{k-1} \cdots ((k-1)2^{k-1} + 1)}{2^k \cdot (2^k - 1) \cdots (2^{k-1} + 1)} \quad \because I.H. \\
&\leq (4(k-1))^{2^{k-1}} \cdot \frac{(k2^k) \cdot (k2^k - 1) \cdots (k2^{k-1} + 1)}{(k2^k - 2^k) \cdots (k2^{k-1} - 2^k + 1)} \cdot \left[\frac{(k-1)2^{k-1} + 1}{2^{k-1} + 1}\right]^{2^{k-1}} \\
&\qquad \because \frac{i}{j} \leq \frac{i-1}{j-1} \text{ for } i \geq j \\
&\leq (4(k-1))^{2^{k-1}} \cdot \frac{(k2^k) \cdot (k2^k - 1) \cdots (k2^{k-1} + 1)}{(k2^k - 2^k) \cdots (k2^{k-1} - 2^k + 1)} \cdot (k-1)^{2^{k-1}} \quad \because \frac{i}{j} \leq \frac{i-1}{j-1} \text{ for } i \geq j
\end{aligned}
$$

It is enough to show that $\frac{(4k)^{2^k}}{(4(k-1))^{2^{k-1}}} \geq \frac{(k2^k) \cdot (k2^k - 1) \cdots (k2^{k-1} + 1)}{(k2^k - 2^k) \cdots (k2^{k-1} - 2^k + 1)} \cdot (k-1)^{2^{k-1}}$.

$$
\begin{aligned}
L.H.S. &= \frac{(4k)^{2^k}}{(4(k-1))^{2^{k-1}}} = \left[\frac{4^2k^2}{4(k-1)}\right]^{2^{k-1}} = \left[\frac{4k^2}{k-1}\right]^{2^{k-1}} \\
R.H.S. &= \frac{(k2^k)\cdot(k2^k-1)\cdots(k2^{k-1}+1)}{(k2^k-2^k)\cdots(k2^{k-1}-2^k+1)}\cdot(k-1)^{2^{k-1}} \\
&= \left[\frac{(k2^k)\cdots(k2^k-2^{k-1}+1)}{(2^k(k-1))\cdots(2^k(k-1)-2^{k-1}+1)}\cdot\frac{(k2^k-2^{k-1})\cdots(k2^k-2\cdot 2^{k-1}+1)}{(2^k(k-1)-2^{k-1})\cdots(2^k(k-1)-2\cdot 2^{k-1}+1)}\cdots\right. \\
&\quad \left.\cdots\frac{(k2^k-(k-1)2^{k-1})\cdots(k2^{k-1}+1)}{(2^k(k-1)-(k-1)\cdot 2^{k-1})\cdots(k2^{k-1}-2^k+1)}\right]\cdot(k-1)^{2^{k-1}} \\
&\leq \left[\left(\frac{(2k-1)\cdot 2^{k-1}+1}{(2k-3)\cdot 2^{k-1}+1}\right)^{2^{k-1}}\left(\frac{(2k-2)\cdot 2^{k-1}+1}{(2k-4)\cdot 2^{k-1}+1}\right)^{2^{k-1}}\cdots\right. \\
&\quad \left.\cdots\left(\frac{(2k-k)\cdot 2^{k-1}+1}{(2k-k-2)\cdot 2^{k-1}+1}\right)^{2^{k-1}}\right]\cdot(k-1)^{2^{k-1}} \qquad \because \frac{i}{j} \leq \frac{i-1}{j-1} \text{ for } i \geq j \\
&\leq \left[\frac{2k-1}{2k-3}\cdot\frac{2k-2}{2k-4}\cdots\frac{k}{k-2}\right]^{2^{k-1}}\cdot(k-1)^{2^{k-1}} \qquad \because \frac{i}{j} \leq \frac{i-1}{j-1} \text{ for } i \geq j \\
&= \left[\frac{(2k-1)\cdot(2k-2)\cdot(2k-3)\cdot(2k-4)\cdots(k+1)\cdot k}{(2k-3)\cdot(2k-4)\cdots k\cdot(k-1)\cdot(k-2)}\cdot(k-1)\right]^{2^{k-1}} \\
&= \left[\frac{(2k-1)(2k-2)}{k-2}\right]^{2^{k-1}}
\end{aligned}
$$

We want to show $L.H.S. \geq R.H.S.$, i.e.,

$$\frac{4k^2}{k-1} \geq \frac{(2k-1)(2k-2)}{k-2}.$$

This is true since the inequality

$$
\begin{aligned}
\frac{4k^2}{k-1} &\geq \frac{(2k-1)(2k-2)}{k-2} \\
4k^3-8k^2 &\geq 4k^3-6k^2+2k-4k^2+6k-2 \\
2k^2-8k+2 &\geq 0 \\
k^2-4k+1 &\geq 0
\end{aligned}
$$

holds whenever $k \geq 4$.

□

Appendix B

Width Lower Bound for MGT_n

Theorem 4.12 (Bonet and Galesi [10]). Any resolution refutation of MGT_n has width $\Omega(n)$.

Recall that the CNF formula GT_n consists of the following clauses:

(1)	$\overline{x_{j,j}}$	$j \in [n]$
(2)	$x_{i,j} \wedge x_{j,k} \rightarrow x_{i,k}$	$i,j,k \in [n],\ i \neq j \neq k$
(3)	$x_{i,j} \rightarrow \overline{x_{j,i}}$	$i,j \in [n],\ i \neq j$
(4)	$\bigvee_{i \in [n]} x_{i,j}$	$j \in [n]$

And, MGT_n is the conjunction of the clauses in (1), (2), (3), and (4*) where

(4*) $\overline{y_{0,j}} \ \wedge \ \bigwedge_{i \in [n]} (y_{i-1,j} \vee x_{i,j} \vee \overline{y_{i,j}}) \ \wedge \ y_{n,j}$ $\qquad j \in [n]$

Definition B.1. A *critical* truth assignment of the variables in GT_n is a total assignment generated by the following algorithm:

Input: An undefined truth assignment α.
Output: A critical truth assignment α.
$S \leftarrow [n]$; $P \leftarrow \{\}$;
while S is not empty **do**
    Choose i in S;
    Set $\alpha(x_{i,i})$ to 0;
    Set $\alpha(x_{i,j})$ to 1 for all $j \in S \backslash \{i\}$;
    Set $\alpha(x_{j,i})$ to 0 for all $j \in P$;

$S \leftarrow S \backslash \{i\}$; $P \leftarrow P \cup \{i\}$;
end (while)
Output α;

A critial truth assignment corresponds to a linear directed acyclic graph over n vertices and closed under transitivity. In such a graph, only the first vertex j in the line does not have a predecessor. Therefore, the critical assignment satisfies all clauses in GT_n except the one for vertex j in (4). We call such an assignment a j-critical assignment.

Now, let B_j be the formula for j in (4*). A j-critical assignment for MGT_n is obtained as follows. We first find a j-critical assignment for the x variables. Then, we assign values to the y variables so that all the B_k's for $k \neq j$ are made true and B_j is made false.

Let A_j be the conjunction of the clauses $(\overline{x_{i,j}} \ \ \overline{x_{j,i}})$ for all $i \in [n], i \neq j$. Let C_j be $A_j \wedge B_j$ and let $VARS(j)$ be the set of variables appearing in C_j. Notice that $VARS(j)$ contains all variables that mention j.

Proof. [Theorem 4.12] Let π be a resolution refutation of MGT_n. For each set $I \subseteq [n]$, define C_I as $\bigwedge_{i \in I} C_i$. For any clause C, define $\mu(C)$ to be the size of the minimal nonempty set $I \subseteq [n]$ such that every critical assignment satisfying C_I also satisfies C. Obviously, μ is subadditive with respect to the resolution rule. That is, if $\frac{A \quad B}{C}$, then $\mu(A) + \mu(B) \geq \mu(C)$. Moreover, $\mu(\{\}) = n$ and $\mu(C) = 1$ for all $C \in C_i, i \in [n]$. Therefore, there must exist a clause C in π for which $\frac{n}{3} \leq \mu(C) \leq \frac{2n}{3}$. We will show that C contains at least $\frac{n}{6}$ literals and this implies $w(MGT_n) = \Omega(n)$.

Let $I \subseteq [n]$ be the minimal set such that all critical assignments satisfying C_I also satisfy C. Then, $\frac{n}{3} \leq |I| \leq \frac{2n}{3}$. Towards a contradiction, suppose $|C| < \frac{n}{6}$. We claim that if $S \subseteq [n]$ and $|S| \geq \frac{n}{3}$, then there exists some $i \in S$ such that no variable from $VARS(i)$ appears in C. Hence, there is an $l \in I$ such that no variable in $VARS(l)$ appears in C. By the minimality of I, there exists an l-critical assignment α such that $\alpha(C_l) = 0, \alpha(C) = 0$ and for all $i \in I \backslash \{l\}, \alpha(C_i) = 1$. Let $J = [n] \backslash I$. Then, $|J| > \frac{n}{3}$ since $|I| \leq \frac{2n}{3}$. By the above claim again, there exists a $j \in J$ such that no variable from $VARS(j)$ appears in C. We will construct a j-critical assignment β from α such

that $\beta(C_i) = 1$ for all $i \in I$ but $\beta(C) = 0$. This will contradict the definition of μ. β is built as follows. We first set $\beta(x_{i,j}) = \alpha(x_{i,j})$ for all $i, j \in [n]$. Then, for all $i \neq j$, if $\beta(x_{i,j}) = 1$, we change it to 0. And, if $\beta(x_{j,i}) = 0$, we change it to 1. Intuitively, we take the linear ordering of the vertices corresponding to α and move the vertex j to the beginning of the ordering. The change does not affect the value of C since no variable in $VARS(j)$ appears in C. After the change, $\beta(x_{j,l}) = 1$ because $\alpha(x_{i,l}) = 0$ for all $i \in [n]$ (note that α is an l-critical assignment). What remains is to change the values of $y_{i,l}, i \in \{0, \cdots, n\}$ in order to make β satisfy C_l. This change does not affect the value of C either since no variable in $VARS(l)$ appears in C. Furthermore, all these changes do not alter the values of the C_i's, $i \in I \setminus \{l\}$. So, there exists a critical assignment β such that $\beta(C_i) = 1$ for all $i \in I$ but $\beta(C) = 0$.

It remains to prove the claim. The indices of each variable $x_{i,j}, i, j \in [n]$, mention at most 2 elements in $[n]$. Since we assumed $|C| < \frac{n}{6}$, the variables in C mention less than $\frac{2n}{6}$ different $VARS()$ sets. That is, the variables in C mention less than $\frac{2n}{6} = \frac{n}{3}$ elements in $[n]$. Therefore, if $S \subseteq [n]$ and $|S| \geq \frac{n}{3}$, there must exist some i in S such that no variable in $VARS(i)$ appears in C.

□

Bibliography

[1] M. Alekhnovich, J. Johannsen, T. Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(056), 2001.

[2] A. Atserias. On sufficient conditions for unsatisfiability of random formulas. *J. ACM*, 51(2):281–311, 2004.

[3] F. Bacchus and P. van Run. Dynamic Variable Ordering in CSPs. In Ugo Montanari and Francesca Rossi, editors, *Proceedings First International Conference on Constraint Programming*, pages 258–275. Springer-Verlag, 1995.

[4] Andrew B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. PhD thesis, University of Oregon, 1995.

[5] P. Beame, J. Culberson, and D. Mitchell. The resolution complexity of random graph k-colourability. In preparation.

[6] P. Beame, H. Kautz, and A. Sabharwal. Understanding the power of clause learning, 2003.

[7] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near-optimal separation of treelike and general resolution. Technical Report TR01-005, Electronic Colloquium on Computational Complexity (ECCC), 2000.

[8] C. Bessière and J.C. Régin. Arc-consistency for general constraint networks: Preliminary results. In *Proc. of IJCAI*, pages 398–404, 1997.

[9] M. L. Bonet, J. L. Esteban, N. Galesi, and J. Johansen. Exponential separations between restricted resolution and cutting planes proof systems. In *Proc. of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS'98)*, pages 638–647. IEEE Press, 1998.

[10] M. L. Bonet and N. Galesi. A study of proof search algorithms for resolution and polynomial calculus. In *Proc. 40th Symposium on Foundations of Computer Science*, pages 422–432, 1999.

[11] J. Buresh-Oppenheim, D. Mitchell, and T. Pitassi. Linear and negative resolution are weaker than resolution. Technical Report TR01-074, Electronic Colloquium on Computational Complexity (ECCC), 2001.

[12] J.R. Celoni, W.J. Paul, and R.E. Tarjan. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[13] Stephen A. Cook. An observation on time-storage trade off. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 29–33. ACM Press, 1973.

[14] J. De Kleer. A comparison of ATMS and CSP techniques. In *Proc. of the 11th Int'l. Joint Conf. on A. I. (IJCAI-89)*, pages 290–296, 1989.

[15] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[16] R. Dechter and D. Frost. Backjump-based backtracking for constraint satisfaction problems. *Artificial Intelligence*, 136(2):147–188, 2002.

[17] M. Dent and R. Mercer. Minimal forward checking. In *Proceedings of the 6th Int'l. Conf. on Tools with Artificial Intelligence.*, pages 432–438, 1994.

[18] D. Frost and R. Dechter. Dead-end driven learning. In *Proc., Twelfth Nat. Conf. on Artificial Intelligence (AAAI-94)*, pages 294–300, 1994.

[19] D. Frost and R. Dechter. Look-ahead value ordering for constraint satisfaction problems. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'95*, pages 572–578, Montreal, Canada, 1995.

[20] Matthew L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[21] A. Goerdt. Unrestricted resolution versus N-resolution. *Theoretical Computer Science*, 93:159–167, 1992.

[22] B. Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–274, 1985.

[23] A. K. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–73, 1985.

[24] D. G. Mitchell. Hard problems for CSP algorithms. In *Proc., 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*, pages 398–405, 1998.

[25] David G. Mitchell. *The Resolution Complexity of Constraint Satisfaction*. PhD thesis, University of Toronto, 2002.

[26] David G. Mitchell. Resolution complexity of random constraints. In *Lecture Notes in Computer Science, LNCS 2470*, pages 295–309. Springer, 2002.

[27] David G. Mitchell. Resolution and constraint satisfaction. In *Lecture Notes in Computer Science, LNCS 2833*, pages 555–569. Springer, 2003.

[28] M. Molloy and M. Salavatipour. The resolution complexity of random constraint satisfaction problems. In *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 330–339. IEEE Press, 2003.

[29] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, 2001.

[30] V. Park. An empirical study of different branching strategies for constraint satisfaction problems. Master's thesis, University of Waterloo, 2004.

[31] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3):268–299, August 1993.

[32] P. Prosser. MAC-CBJ: maintaining arc consistency with conflict-directed backjumping. Technical Report Research Report/95/177, Dept. of Comp. Sci., U. of Strathclyde, Dept. of Computer Science, University of Strathclyde, 1995.

[33] R. Raz and P. McKenzie. Separation of the monotone NC hierarchy. In *IEEE Symposium on Foundations of Computer Science*, pages 234–243, 1997.

[34] D. Sabin and E. C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proc. of the 2nd Int'l. Conference on the Principles and Practice of Constraint Programming (CP'94).*, pages 10–19, 1994. (Published as Springer LNCS-874).

[35] T. Schiex and G. Verfaillie. Nogood recording for static and dynamic constraint satisfaction problem. *International Journal on Artificial Intelligence Tools (IJAIT)*, 3(2):187–207, 1994.

[36] B. M. Smith and P. Sturdy. An empirical investigation of value ordering for finding all solutions. Presented at the ECAI 2004 workshop on Modelling and Solving Problems with Constraints.

[37] G. Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33:277–280, 1996.

[38] Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient conflict driven learning in boolean satisfiability solver. In *ICCAD*, pages 279–285, 2001.