Relative NAND-clause growth in Neg refutations

Wim Van den Broeck

Master's Thesis
Department of Informatics
University of Bergen

January 18, 2022

Acknowledgements

A great debt of gratitude goes to my supervisor, Michał Walicki, for helpful feedback and outstanding guidance.

Abstract

Any given propositional discourse can be expressed as a NAND-OR theory – a set of OR-clauses (disjunctions of literals) and NAND-clauses (negated conjunctions e.g. $\neg(x \wedge y)$) – making NAND-OR theories similar to traditional normal forms such as CNF and DNF. This thesis will discuss the inference system Neg; a non-explosive, sound and refutationally complete resolution system over NAND-OR theories. Of particular interest will be inconsistent theories which can be refuted in Neg by deriving the empty clause. The main result in this thesis will be a counterexample to the conjecture that the NAND-clauses of inconsistent NAND-OR theories need not grow greater than the size of its OR-clauses in the derivation of the empty clause. Further properties of such inconsistency derivations are explored with the aim of analysing the creation NAND-clauses in Neg.

Contents

Chapter 1

Introduction

1.1 Proof complexity

Proof complexity concerns itself with the lengths of proofs in various proof systems. Much work has been devoted to finding proof length lower and upper bounds in standard proof systems with various interrelated goals in mind.

Of particular interest to the field are the lengths of the shortest proofs of propositional tautologies. This area of study has been shown to be inherently connected to open questions in computational complexity theory such as whether $NP = coNP$. It was shown in [1] [2] by Cook and Reckhow that $NP = coNP$ if and only if there exists a super proof system – a propositional proof system that admits polynomial size proofs of tautologies $\tau \in TAUT$ in the length of the tautology –. In particular it was shown that $\overline{TAUT}$ is NP-complete (hence $TAUT$ is coNP-complete). Thus the existence of a super proof system would give rise to a procedure in the likes of guessing non-deterministically a proof of τ to then verifying it deterministically with the super proof system. This would put $TAUT$ in NP and so $NP = coNP$. With that, finding super-polynomial lower bounds for standard propositional proof systems has been regarded as evidence toward separating NP and $coNP$.

Further, studying proof complexity of propositional proof systems is interesting in the context of automated theorem proving. Automated theorem provers are usually based on some propositional proof system. This is true also for many algorithms and other mathematical methods used in various fields of science. Finding proof complexity lower and upper bounds in general implies (computational) complexity lower and upper bounds of these automated theorem provers. In this way the study of proof complexity is intimately connected to practical work in for example SAT solving.

In addition, powerful insights have been developed on the parallels between proof complexity and areas such as circuit complexity and bounded arithmetic.

In this text proofs for propositional logic in the resolution system Neg from [3] shall be looked at. Neg reasons over NAND-OR theories. These are theories comprising of a set of OR-clauses (disjunctions of literals, $x_1 \vee x_2 \vee \ldots \vee x_i$) and NAND-clauses (negated conjunctions, $\neg(x_1 \wedge x_2 \wedge \ldots \wedge x_i)$). Any propositional theory can be written in this form. Neg's only rule, Rneg, is a resolution style rule that takes in its premise a set of NAND-clauses and an OR-clause and concludes in a NAND-clause that is a union of the premise NAND-clauses with atoms from the OR-clause omitted. Neg is, sound, refutationally complete and non-explosive over NAND-OR theories. A theory is

considered proven in Neg if we can derive the empty clause ($\vdash_{Neg} \{\}$) from its negation. A main goal of this thesis is to analyse the NAND-clause growth in such derivations. We content ourselves in only analysing NAND-clause growth as these are the only clauses that the Rneg rule can produce. In keeping with the program of finding lower bounds for propositional proof systems, we see the study of the growth of these NAND-clauses as a step toward finding a lower bound for proofs in the Neg system. Neg's resolution-like reasoning, making it potentially suitable in automated theorem proving, also motivates the study of its proof complexity.

A main result in this thesis will be the proof of a conjecture claiming that a certain family of NAND-OR theories representing the pigeonhole principle has proofs in Neg where it's NAND-clauses must grow to the size of it's OR-clauses. We also give counterexamples to some conjectures claiming NAND-clause bounds for arbitrary NAND-OR theories and some relevant restrictions of NAND-OR theories.

Chapter 2

Background

Before we present the Neg system we describe the form of the theories over which it reasons. This in part motivates the work on the Neg system as well as these theories being interesting in their own right.

2.1 GNF and NAND-OR

A propositional theory over some alphabet Σ is in graph normal form (from here on abbreviated to GNF) if it consists of formulae in the form:

$$x \leftrightarrow \bigwedge \{\neg x_i | i \in I\} \tag{2.1}$$

where I is some index set and $x \in \Sigma$, all $x_i \in \Sigma$ and each $x \in \Sigma$ occurs exactly once on the left of such an equivalence [4].

Every theory in propositional logic can be represented as a directed graph such that the models of this theory correspond bijectively to the kernels –an independent set of vertices of the graph such that every vertex not in the set has an edge leading to a vertex in the set– in its representative digraph. Finding a theory's representative digraph can be done especially easily if the theory is in GNF. Further, every theory in propositional logic not in GNF can be translated to GNF (see appendix A). Additionally theories represented as a GNF theory have the interesting property of being inconsistent if and only if they are paradoxical. By paradoxical we mean that we can use substitutions to arrive at the classical logical form of paradox, $x \leftrightarrow \neg x$. This relation with paradoxicality is in contrast to the general case where a theory may be inconsistent but not necessarily paradoxical. Consider the following propositional theory (A similar example is given in [4]):

$$\begin{array}{lcl} a & \leftrightarrow & \neg b \\ b & \leftrightarrow & \neg c \wedge \neg d \\ c & \leftrightarrow & \neg a \\ d & \leftrightarrow & \neg a \end{array} \tag{2.2}$$

Here we see that no satisfying truth value assignment exists. Since every formulae in this theory is in GNF we may go on to conclude that it is paradoxical. We can see this more clearly with a series of substitutions:

$$a \leftrightarrow \neg b \Leftrightarrow a \leftrightarrow \neg(\neg c \wedge \neg d) \Leftrightarrow a \leftrightarrow c \vee d \Leftrightarrow a \leftrightarrow \neg a \vee \neg a \Leftrightarrow a \leftrightarrow \neg a \tag{2.3}$$

GNF theories, their correspondence with digraphs and their relation with paradoxicality has been studied in [3] [5] [6] [7].

This paper will mainly be handling NAND-OR theories into which GNF theories easily can be translated. Separating the bi-implication in 2.1 we get the following two formulae:

$$x \leftarrow \bigwedge\{\neg x_i | i \in I\} \tag{2.4}$$

$$x \rightarrow \bigwedge\{\neg x_i | i \in I\} \tag{2.5}$$

We can reformulate 2.4 as follows:

$$x \leftarrow \bigwedge\{\neg x_i | i \in I\} \Leftrightarrow x \vee \neg(\bigwedge\{\neg x_i | i \in I\}) \Leftrightarrow x \vee (\bigvee\{x_i | i \in I\}) \tag{2.6}$$

Further, we can also reformulate 2.5 as follows:

$$x \rightarrow \bigwedge\{\neg x_i | i \in I\} \Leftrightarrow \neg x \vee (\bigwedge\{\neg x_i | i \in I\}) \Leftrightarrow \bigwedge\{\neg(x \wedge \neg x_i) | i \in I\} \tag{2.7}$$

We see now that for each variable x of a GNF theory we have two types of clauses:

$$\text{OR-clauses: } x \vee \bigvee_{i \in I} x_i \tag{2.8}$$

$$\text{NAND-clauses: } \neg(x \wedge x_i), \text{ for every } i \in I \tag{2.9}$$

From this point, OR-clauses may be written as xx_1x_2 instead of, $x \vee x_1 \vee x_2 \vee x_3$ and NAND-clauses may be written as $\overline{xx_1}$ rather than $\neg(x \wedge x_1)$. Both types of clauses will be treated as sets of atoms where over-bars denote only that a set is a NAND-clause. Hence we may write, $\overline{ab} \subseteq \overline{abcd}$. Further, $A \subseteq \Sigma$ will denote an OR-clause and $\overline{A} = \{\overline{a} | a \in A\}$ will denote a NAND-clause.

Since every theory in propositional logic can be expressed in GNF, it follows that every theory in propositional logic can be expressed as a NAND-OR theory. The theory of 2.2 will have the following NAND-OR form:

$$\begin{aligned} \text{OR:=}&\{ab, bcd, ac, ad\} \\ \text{NAND:=}&\{\overline{ab}, \overline{bc}, \overline{bd}, \overline{ac}, \overline{ad}\} \end{aligned} \tag{2.10}$$

2.2 The RIP system

The following inference system was introduced in Michał Walicki's paper, "Resloving infinitary paradoxes" [3]. It is sound for arbitrary NAND-OR theories and refutationaly complete for theories with a countable set of OR-clauses and where each NAND-clause is finite.

$$(\mathsf{Ax})\ \Gamma \vdash C,\ \text{for } C \in \Gamma$$

$$(\mathsf{Rneg})\ \frac{\{\Gamma \vdash \overline{a_i A_i} | i \in I\} \quad \{\Gamma \vdash a_i | i \in I\}}{\Gamma \vdash \overline{\bigcup_{i \in I} A_i}}$$

$$(\mathsf{Rpos})\ \frac{\Gamma \vdash A\ \{\Gamma \vdash B_i K_i | i \in I\} \quad \{\Gamma \vdash \overline{a_i k} | i \in I, k \in K_i\}}{\Gamma \vdash (A \backslash \{a_i | i \in I\}) \cup \bigcup_{i \in I} B_i}$$

With the Rneg rule a NAND-clause can be created using NAND-clauses in the premise and an OR-clause as a side condition. With the Rpos rule an OR-clause can be created using OR-clauses in the premise and a NAND-clause as a side condition. Premise clause $\overline{a_i A_i}$ in Rneg represents the NAND-clause $\overline{\{a_i\} \cup A_i}$, where $\overline{A_i}$ may be empty.

The focus in this paper will be on proofs in Neg; a restriction of the RIP system which uses only the axioms and the Rneg rule.

2.2.1 Neg

As shown in [3], Neg is sound for arbitrary theories and refutationally complete for theories with countable number of OR-clauses. In this paper only finite theories are considered.

For the purpose of easing readability, when the theory is clear from the context, we omit '$\Gamma \vdash$' in notation. For similar reasons, and to highlight its use as a side condition, the OR-clause in the premise of a the Rneg rule application is moved to the side. The figure below illustrates these changes:

$$(1)\ \frac{\Gamma \vdash \overline{abx} \quad \Gamma \vdash \overline{cdy} \quad \Gamma \vdash xy}{\Gamma \vdash \overline{abcd}} \qquad (1')\ \frac{\overline{abx} \quad \overline{cdy}}{\overline{abcd}}\, xy \tag{2.11}$$

In an Rneg rule application the premise is a set of NAND-clauses as well as a single OR-clause. There must be a bijective relationship between the atoms in the OR-clause and the set of NAND-clauses. The corresponding NAND-Clause to eeach OR-clause atom must contain that atom. In the conclusion will be a single NAND-clause formed by the union of the NAND-clauses in the premise but omitting their corresponding atom from the OR-clause. Below (2) and (3) are examples of incorrect Rneg rule aplications:

$$(2)\ \frac{\overline{abx} \quad \overline{cdy}}{\overline{abcd}}\, xyz \qquad (3)\ \frac{\overline{axz} \quad \overline{by} \quad \overline{cy}}{\overline{abc}}\, xyz \tag{2.12}$$

(2) is not correct both in that there are more atoms in the OR-cause than there are NAND-clauses in the premise and also since z from the OR-clause does not appear in any NAND-clauses. In (3), though every atom in the OR-clause is represented in some NAND-clause in the premise there is no bijective relation between them so it fails also. Using instead xy as the OR-clause would fail also as y cannot 'cancel' its negated literal from both $\overline{by}$ and $\overline{cy}$, so again there is no bijective relation between the

atoms in the OR-clause and the NAND-clauses in the premise. Now follows correct Rneg rule applications:

$$(4)\ \frac{\overline{abx} \quad \overline{cy} \quad \overline{cz}}{\overline{abc}}\, xyz \qquad (5)\ \frac{\overline{x} \quad \overline{y} \quad \overline{z}}{\{\}}\, xyz \tag{2.13}$$

The atoms in each of the OR-clauses above are matched to precisely one NAND-clause in their respective premises and all NAND-clauses have some match.

We can now show that the NAND-OR theory in 2.10, and thus also the theory in 2.2, is inconsistent in Neg. We do this by deriving the empty clause:

$$bcd\ \frac{\dfrac{\overline{ab} \quad \overline{ac} \quad \overline{ad}}{ac\ \overline{a}} \qquad \dfrac{\overline{ac}\ \overline{bc}}{\overline{c}}}{\{\}}\, ab \tag{2.14}$$

Note that OR-clauses in Neg are fixed by the theory and only NAND-clauses are generated. Consequently, our proof complexity analysis will focus on the size of the NAND-clauses occurring in proofs.

2.2.2 Inconsistency of $\neg PHP_4^3$

The pigeonhole principle is a counting principle which states that for p pigeons being housed in h holes, where $p > h$, then at least one hole must house two pigeons. In the study of propositional proof complexity, the pigeonhole principle is a canonical example of a specific tautology for which much work has been devoted to proving lower bounds under various proof systems. This is typically with the goal of determining that the proof system in question is not efficient (does not admit polynomial lower bounds for all tautologies) [8]. In its essence, the pigeonhole principle disallows bijective functions from p to h for $p > h$. Consider the following common example showing an application of the pigeonhole principle: in a room of 367 people, the pigeonhole principle requires that at least two people share a birthday. The pigeonhole principle will be a recurring example throughout this text through which we shall seek to analyse the NAND-clause growth of inconsistency proofs in the Neg system.

PHP_p^h refers to a family of theories representing instances of the pigeonhole principle where p is the number of pigeons and h is the number of holes (we will only consider instances where $p \geq h+1$ and $h > 1$). We can prove any particular PHP_p^h theory in Neg by proving the inconsistency of its negation $\neg PHP_p^h$. An instance of $\neg PHP_p^h$ amounts to a theory where each of the p pigeons is contained in a hole and each hole contains only one pigeon. Letting the atom x_i represent pigeon i occupying hole x, a $\neg PHP_p^h$ theory can be generally formalized as a NAND-OR theory with the following axioms:

$$\begin{aligned} OR: &= \left\{1_i 2_i \ldots h_i | i \leq p\right\} \\ NAND: &= \left\{\overline{x_i x_j} | 0 < i < j \leq p\ , 0 < x \leq h\right\} \end{aligned} \tag{2.15}$$

Each OR-clause is of size h and represents the fact that pigeon i can either occupy the first hole, or the second hole, or the third and so on up until and including the h'th hole. Each NAND-clause is binary and represents the fact that no hole can be occupied by both pigeon i and pigeon j ($i \neq j$), that is to say no hole can contain more than one pigeon.

The specific instance $\neg PHP_4^3$ has been proven inconsistent in Neg by Kjetil Golid in [5]. The set of axiomatic OR-clauses and NAND clauses for $\neg PHP_4^3$ are given below:

$$\begin{aligned} OR: &= \left\{1_1 2_1 3_1, 1_2 2_2 3_2, 1_3 2_3 3_3, 1_4 2_4 3_4\right\} \\ NAND: &= \left\{\begin{array}{l} \overline{1_1 1_2}, \overline{1_1 1_3}, \overline{1_1 1_4}, \overline{1_2 1_3}, \overline{1_2 1_4}, \overline{1_3 1_4}, \\ \overline{2_1 2_2}, \overline{2_1 2_3}, \overline{2_1 2_4}, \overline{2_2 2_3}, \overline{2_2 2_4}, \overline{2_3 2_4}, \\ \overline{3_1 3_2}, \overline{3_1 3_3}, \overline{3_1 3_4}, \overline{3_2 3_3}, \overline{3_2 3_4}, \overline{3_3 3_4} \end{array}\right\} \end{aligned} \tag{2.16}$$

The proof showing that the empty clause can be derived is repeated here as it is a good example of a Neg proof and will be referenced multiple times.

Proposition 2.17. *$\neg PHP_4^3$ is inconsistent in Neg.*

Proof.

$$\dfrac{\dfrac{\overline{1_2 1_4}\ \ \overline{2_2 2_3} \qquad \dfrac{\overline{1_1 1_4}\ \ \overline{2_1 2_3}\ \ \overline{3_1 3_2}}{\overline{3_2 3_1 1_4}}\, 1_1 2_1 3_1 \qquad \overline{1_2 1_4} \qquad \dfrac{\overline{1_1 1_4}\ \ \overline{2_1 2_4}\ \ \overline{3_1 3_3}}{\overline{2_3 3_1 1_4}}\, 1_1 2_1 3_1 \qquad \overline{3_2 3_3}}{\overline{2_3 1_4}}\, 1_2 2_2 3_2 \qquad \overline{3_1 1_4}}{\overline{1_4}}\, 1_3 2_3 3_3$$

$$\dfrac{\dfrac{\overline{1_2 1_3}\ \ \overline{2_2 2_4} \qquad \dfrac{\overline{1_1 1_3}\ \ \overline{2_1 2_4}\ \ \overline{3_1 3_2}}{\overline{3_2 1_3 2_4}}\, 1_1 2_1 3_1 \qquad \overline{2_2 2_4} \qquad \dfrac{\overline{1_1 1_2}\ \ \overline{2_1 2_4}\ \ \overline{3_1 3_3}}{\overline{1_2 3_3 2_4}}\, 1_1 2_1 3_1 \qquad \overline{3_2 3_3}}{\overline{1_3 2_4}}\, 1_2 2_2 3_2 \qquad \overline{3_2 2_4}}{\overline{2_4}}\, 1_3 2_3 3_3$$

$$\dfrac{\dfrac{\overline{1_2 1_3}\ \ \overline{3_2 3_4} \qquad \dfrac{\overline{1_1 1_3}\ \ \overline{2_1 2_2}\ \ \overline{3_1 3_4}}{\overline{2_1 1_3 3_4}}\, 1_1 2_1 3_1 \qquad \overline{2_2 2_3} \qquad \dfrac{\overline{1_1 1_2}\ \ \overline{2_1 2_3}\ \ \overline{3_1 3_4}}{\overline{1_2 3_3 3_4}}\, 1_1 2_1 3_1 \qquad \overline{3_2 3_4}}{\overline{1_3 3_4}}\, 1_2 2_2 3_2 \qquad \overline{2_3 3_4}}{\overline{3_4}}\, 1_3 2_3 3_3$$

$$\dfrac{\overline{1_4} \quad \overline{2_4} \quad \overline{3_4}}{\{\}}\, 1_4 2_4 3_4$$

□

2.3 Problem Statements and Thesis outline

This thesis will explore the following conjectures:

Conjecture 1 (First thesis statement)**.** *For any Neg proof of the pigeonhole principle represented by 2.15, NAND-clauses grow to the size of the OR-clauses.*

Note here again that proving a claim X in Neg amounts to deriving the empty clause from a clausal theory, $C_{\neg X}$, representing $\neg X$, the negation of X (i.e. $C_{\neg X} \vdash_{Neg} \{\}$). This is how proving the pigeonhole principle is to be understood. We may also say that clausal theory $C_{\neg X}$ has a refutation (or is refutable) in Neg to mean the same thing.

Conjecture 2 (Second thesis statement)**.** *Given any NAND-OR theory refutable in Neg, there is always a Neg refutation in which no NAND-clause grows beyond the size of the largest OR-clause.*

Said differently, for conjecture 2 to hold for a specific theory we require that no NAND-clause greater than the largest OR-axiom to be involved in its refutation. Throughout the text we may say that a particular theory T has a narrow refutation if conjecture 2 holds for that specific theory. In other words conjecture 2 states that any theory refutable in Neg has a narrow refutation. Is it not the case that conjecture 2 holds for T (i.e. NAND-clauses must grow beyond the size of OR-clauses in any refutation of T), we may say that T has a wide refutation.

Upon deeper investigation it was found that conjecture 2 was too imprecisely formulated to give interesting results. Discovery of a NAND-OR theory with NAND-axioms greater than OR-axioms gave a trivial counterexample to the conjecture. [1] This observation motivated the investigation of some specialisations of conjecture 2:

Conjecture 3. *Given any NAND-binary NAND-OR theory refutable in Neg, there is always a Neg refutation in which no NAND-clause grows beyond the size of the largest OR-clause.*

By NAND-binary we refer to theories for which all NAND-axioms are binary. This assures that we do not consider theories where NAND-axioms are greater than OR-axioms.

Lastsly, we consider GNF theories:

Conjecture 4. *Given any GNF theory refutable in Neg, there is always a Neg refutation in which no NAND-clause grows beyond the size of the largest OR-clause.*

GNF theories are themselves NAND-binary and NAND-binary NAND-OR theories are of course just restrictions of arbitrary NAND-OR theories. These inclusions are illustrated in figure 2.1.

[1] We must mention here also that we will disregard theories with unary NAND-clauses as they also may give trivial counterexamlpes to our conjectures leaving little insight.

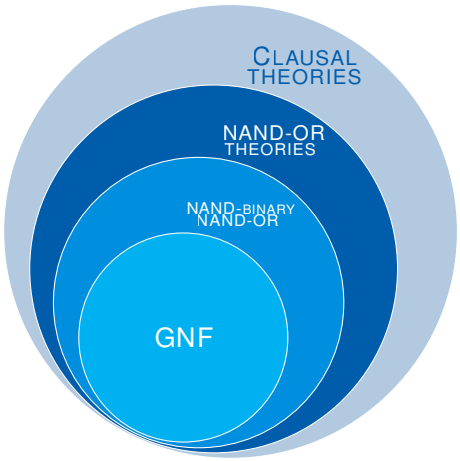

Figure 2.1: Restrictions of NAND-OR theories

With that, finding a counterexample of conjecture 4 (which we were not able to do) would mean finding a counterexample of conjectures 2 and 3 as well. Meanwhile, had we been able to prove conjecture 2, then conjectures 3 and 4 would also be proven.

In 2.2.2 it was shown that conjecture 1 holds for a specific instance of the pigeonhole principle. In Chapter 3 it will be shown that the conjecture holds in general. In Chapter 4 it will first be shown that conjecture 2 holds in general for the pigeonhole principle but not for arbitrary NAND-OR theories. A variation of the the pigeonhole principle with increased hole capacity will be presented as a counter example disproving the conjecture. Aslo in chapter 4 we present a counterexample of conjecture 3 and present some comments on the search for a counterexample of conjecture 4.

Chapter 3

Pigeonhole principle in Neg

Showing that conjecture 1 holds would be significant as it would give a preliminary lower bound on the growth of the NAND-clauses for our family of tautologies representing the pigeonhole principle and in turn give some insight into the complexity and other properties of the Neg proof system. We first highlight some properties of the Rneg rule that will be used in the proof of conjecture 1 and proofs later in the thesis.

3.1 Properties of Rneg

In [3] it was observed that if from some theory an empty clause can be derived in Neg then unary NAND-clauses equal to the union of some OR-clause must be derivable (or present as an axiom):

$$\Gamma \vdash_{Neg} \{\} \Leftrightarrow \exists K \in OR : (\forall k \in K : \Gamma \vdash \overline{k}) \tag{3.1}$$

With that, the last step of any Neg refutation will always look the same:

$$\frac{\overline{k_1} \quad \overline{k_2} \quad \overline{k_3} \quad \dots}{\{\}} \, k_1 k_2 k_3 \dots \tag{3.2}$$

Recall that the final step in the refutation of $\neg PHP_4^3$ (proposition 2.17) was of the form of 3.2 but also that several new non-axiomatic NAND-clauses were created in the steps leading up to the final step. In particular, the first steps in creating each unary NAND-clause created NAND-clauses of size 3 (equal to the size of the OR-Clauses). It was shown in [5] that this was unavoidable for this case and it will be shown shortly that this is unavoidable for $\neg PHP_p^h$ theories in general.

First we can make further observations on how Rneg constructs NAND-clauses. The size of the concluding NAND-clause in any Rneg rule application will depend on how many distinct atoms there are in the premise NAND-clauses. Consider again the Rneg rule from section 2.2 where we had $\{a_i | i \in I\}$ be the atoms in some OR-clause and $\{\overline{a_i A_i} | i \in I\}$, be the premise NAND-clauses. By elementary set theory we can express

the cardinality of the concluding NAND-clause with the following equation:

$$|\overline{\bigcup_{i\in I} A_i}| = \sum_{i\in I} |A_i \backslash (A_i \cap \bigcup_{j>i} A_j)|) \tag{3.3}$$

The above equality is simply a generalisation of the cardinality of the union of two sets. From this we see that in general a concluding NAND-clause is at its greatest size when all A_i are distinct.

Remark 3.4. *For $\{a_i | i \in I\}$ atoms in some OR-clause and $\{\overline{a_i A_i} | i \in I\}$ premise NAND-clauses, when each A_i is distinct we have $A_i \cap \bigcup_{j>i} A_j = \emptyset$. Then by equation 3.3 the size of the concluding NAND-clause can be expressed as follows:*

$$|\overline{\bigcup_{i\in I} A_i}| = \sum_{i\in I} |A_i| = \sum_{i\in I} |a_i A_i| - |a_i|$$

In other words, the size of the concluding NAND-clause will be the differnce between the sum of the cardinalities of the premise NAND-clauses and the cardinality of the OR-clause. This represents an upper bound on the possible size of the concluding NAND-clause that can be created from these premises.

On the other hand, a concluding NAND-clause is at its smallest size when each A_i is contained in some A_m, a premise NAND-clause of maximum size.

Remark 3.5. *Let $\{a_i | i \in I\}$ be atoms in some OR-clause, $\{\overline{a_i A_i} | i \in I\}$ premise NAND-clauses, and $a_m A_m \in \{a_i | i \in I\}$ where $a_m A_m$ is the largest premise NAND-clause, $|a_m A_m| = max(|a_i A_i|)$. When each A_i is contained in $a_m A_m$ then by equation 3.3 the size of the concluding NAND-clause can be expressed as follows:*

$$|\overline{\bigcup_{i\in I} A_i}| = |A_m|$$

Letting m be the largest index in our index set we have $A_i \cap \bigcup_{j>i} A_j = A_i$ up until $i = m$. This is a lower bound on the possible size of the concluding NAND-clause that can be created from these premises.

In other words, a concluding NAND-clause will never be smaller than the largest NAND-clause minus 1.

Finally, combining remarks 3.4 and 3.5 we make a corollary observation on Rneg when all premise NAND-clauses are binary.

Remark 3.6. *For OR-clause $\{a_i | i \in I\}$, when all premise NAND-clauses $\{\overline{a_i A_i} | i \in I\}$ are binary, the concluding NAND-clause can have size at most, $|\overline{\bigcup_{i\in I} A_i}| = |\{a_i | i \in I\}|$ (the size of the OR-clause) and can have smallest size 1. This follows from remarks 3.4 and 3.5.*

When all premise NAND-clauses are binary, the concluding NAND-clause can never be larger than the premise OR-clause.

3.2 NAND-clause growth in refutation of $\neg PHP_p^h$

We repeat for convenience the axioms of a $\neg PHP_p^h$ theory from 2.15:

$$\begin{aligned} OR: &= \left\{1_i 2_i \dots h_i | i \leq p\right\} \\ NAND: &= \left\{\overline{x_i x_j} | 0 < i < j \leq p\ , 0 < x \leq h\right\} \end{aligned} \tag{3.7}$$

Consider again the refutation of $\neg PHP_4^3$ in proposition 2.17. The first step in the derivation of each unary NAND-clause created a NAND-clause of size 3. The proposition below shows that this is the case for all $\neg PHP_p^h$.

Proposition 3.8. *For any Neg proof of the pigeonhole principle represented by 3.7 NAND-clauses must grow to the size of the OR-clauses.*

Proof. By 3.1, the final step of a derivation of an refutation requires a set of unary NAND-clauses in the premise. In the absence of any such unary NAND-clauses among the axioms, it is not possible to derive a refutation from the axioms alone and a new clause must be created. Any derivation of a new NAND-clause must start with some OR-clause as the side condition in a rule application . Choosing an arbitrary OR-clause as side condition, any first step must look in general as follows:

$$\frac{\overline{1_i 1_{j_1}} \quad \overline{2_i 2_{j_2}} \quad \dots \quad \overline{h_i h_{j_h}}}{\overline{1_{j_1} 2_{j_2} \dots h_{j_h}}}\, 1_i 2_i \dots h_i$$

Each NAND-axiom in the premise is comprised of two atoms for the same hole. Hence, the NAND-clauses in the premise are mutually disjoint on account of all OR-clause being comprised of atoms for distinct holes. By remark 3.4 we take the difference between the sum of the lengths of the premise NAND-clauses and the length of the OR-clause. Thus the concluding NAND-clause must be of size h as shown.

□

This result is in no way surprising and is really just a generalisation of what was observed in proposition 2.17. With that we found a relative lower bound, h, on the size of the longest NAND-clause necessary to refute a $\neg PHP_p^h$ theory. In the next chapter we investigate if NAND-clauses of this size are the largest necessary in a refutation.

Chapter 4

Arbitrary NAND-OR theories in Neg

Conjectures 2, 3 and 4 would have arguably more far reaching consequences than conjecture 1. It would allow us to find a bound for the length of a line in a Neg proof relative to the size of the OR-clauses in the theory. This would be meaningful for a more complete analysis of the complexity of proofs in Neg.

Here, it will be shown that conjecture 2 holds for the pigeonhole principle in, section 4.1. In sections 4.2 and 4.3 counterexamples of conjectures 2 and 3 in the form of a variation of the $\neg PHP_p^h$ theories. In section 4.4 we discuss the search for a couterexample of conjecture 4.

4.1 Conjecture 2 holds for $\neg PHP_p^h$

From our pigeonhole axioms, inconsistency can be proven without the NAND-clauses growing larger than the size of the OR-clauses. That is to say, any $\neg PHP_p^h$ theory has a narrow refutation. Before proving this we give some useful definitions and prove some relevant lemmata.

Definition 4.1. *We define a $\neg PHP_p^h$ NAND-clause (3.7) to be 'hole-distinct' if it comprises of atoms for distinct holes.*

Below are some examples of hole-distinct NAND-clauses:

- $\overline{2_1 3_3 5_4 7_3}$
- $\overline{1_1 2_1 3_1}$

Observe once more the first and second lines of the unary NAND-clause derivations in proposition 2.17. The first NAND-clauses derived are of size 3 (size of the OR-clauses) and are hole-distinct. Further, given some choice of OR-clause as a side condition, there are several different choices of NAND-clauses for the premise that would result in a legal rule application. This is the observation that motivates definition 4.1 and the lemma below:

Lemma 4.2. *Using only $\neg PHP_p^h$ axioms (3.7), every hole-distinct NAND-clause of size h can be proven in a single rule application.*

Proof. Letting $P = \{1, ..., p\}$, all OR-axioms will have the form $1_\pi 2_\pi \dots h_\pi$, $\pi \in P$. Now letting $H = \{1, ..., h\}$ and $j_i \in P$, a generalised construction of a NAND-clause of size h from axioms will look exactly as in proposition 3.8 (given again for convenience with slightly different notation):

$$\frac{\overline{1_\pi 1_{j_1}} \quad \overline{2_\pi 2_{j_2}} \quad \dots \quad \overline{h_\pi h_{j_h}}}{\overline{1_{j_1} 2_{j_2} \dots h_{j_h}}} \, 1_\pi 2_\pi \dots h_\pi$$

To derive $\overline{1_{j_1} 2_{j_2} \dots h_{j_h}}$ for any $j_1, j_2, \dots, j_h \in P$, take any $\pi \notin \{j_1, j_2, \dots, j_h\}$, which exists since $h < p$. We have axiom $1_\pi 2_\pi \dots h_\pi \in OR$ and for each $1 \leq k \leq h$, $\overline{k_\pi k_{j_i}} \in NAND$. These axioms give the hole-distinct NAND-clause $\overline{1_{j_1} 2_{j_2} \dots h_{j_h}}$ by a single rule application of Rneg as shown above. Remark 3.4 assures that we get a NAND-clause of size h in the conclusion as all $\overline{k_\pi k_{j_i}}$ are mutually distinct.

□

Observe now the second and third lines of the unary NAND-clause derivations in proposition 2.17. Present among the NAND-clauses in the premises is a NAND-clause of size 3, while the conclusions are NAND-clauses of size 2 (one less) and are a proper subset of the size 3 NAND-clause. The premise NAND-clauses in each such derivation satisfy the conditions of remark 3.5. This observation gives rise to the next lemma:

Lemma 4.3. *Given our $\neg PHP_p^h$ axioms (3.7), if all hole-distinct NAND-clauses of size k ($k \leq h$) are provable (in Neg), then all hole-distinct NAND-clauses of size $k-1$ are provable without using any NAND-clauses longer than those used in the proofs of the k-length NAND-clauses.*

Proof. Consider an arbitrarily chosen hole-distinct NAND-clause of size (k-1), $\overline{\alpha^1_{j_1} \dots \alpha^{k-1}_{j_{k-1}}}$.[1] As in the proof of lemma 4.2, it is always possible to find an OR-clause $1_\pi 2_\pi \dots h_\pi$ such that $\pi \notin \{j_i \in P | 1 \leq i \leq k, i \in \mathbf{N}\}$ (since $k \leq h < p$). Given this and the fact that our desired NAND-clause is hole-distinct, it is possible to separate the OR-clause into two parts, $\alpha^1_\pi \dots \alpha^{k-1}_\pi \beta^1_\pi \dots \beta^{h-(k-1)}_\pi$. The α^i_π's are those atoms in the OR-clause for which the same hole occurs in the desired NAND-clause. The β^i_π's are the atoms in the OR-clause for holes not occurring in the concluding NAND-clause. With that we derive our chosen NAND-clause as follows:

$$\frac{\overline{\alpha^1_\pi \alpha^1_{j_1}} \dots \overline{\alpha^{k-1}_\pi \alpha^{k-1}_{j_{k-1}}} \quad \overline{\beta^1_\pi \alpha^1_{j_1} \dots \alpha^{k-1}_{j_{k-1}}} \dots \overline{\beta^{h-(k-1)}_\pi \alpha^1_{j_1} \dots \alpha^{k-1}_{j_{k-1}}}}{\overline{\alpha^1_{j_1} \dots \alpha^{k-1}_{j_{k-1}}}} \, \alpha^1_\pi \dots \alpha^{k-1}_\pi \beta^1_\pi \dots \beta^{h-(k-1)}_\pi$$

Each $\overline{\alpha^i_\pi \alpha^i_{j_i}}$ among the premises is axiomatic (and so is binary) while each $\overline{\beta^i_\pi \alpha^1_{j_1} \dots \alpha^{k-1}_{j_{k-1}}}$ is a hole-distinct NAND-clause of size k. We have the conditions of remark 3.5 so the concluding NAND-clause is of size $k-1$.

[1] The superscripts of these variables are simply to distinguish them from one another and are not to be mistaken for exponents.

□

Now we can prove that any hole-distinct NAND-clause is provable in Neg without NAND-clauses growing beyond the OR-clauses:

Proposition 4.4. *Given our $\neg PHP_p^h$ axioms (3.7), for each $1 \leq k \leq h$, every hole-distinct NAND-clause of length k is provable in Neg without use of any NAND-clause longer than h.*

Proof. The proof is by induction on k, with the basis $k = h$ given by lemma 4.2, and induction step by lemma 4.3.

□

Thus conjecture 2 holds for $\neg PHP_p^h$:

Corollary 4.5. *For $\neg PHP_p^h$ axioms from 3.7, there is always a Neg proof of inconsistency in which no NAND-clause grows beyond the size of the largest OR-clause.*

Proof. By proposition 4.4, all unary NAND-clauses are provable without use of any NAND-clause longer than h, since any unary NAND-clause is hole-distinct. We can now use equation 3.1 taking any OR-clause to perform the final step as in 3.2 selecting what unary NAND-clauses that are necessary.

□

Thus we have found a relative upper bound on the size of the greatest NAND-clauses necessary to refute a $\neg PHP_p^h$ theory (also h).

4.2 $\neg PHP_5^{2,2}$

The counterexample to conjecture 2 that will be shown here is on a NAND-OR theory representing a variation of the pigeonhole principle. In this variation the capacity, c, of the holes can be specified. We denote it as $\neg PHP_p^{h,c}$ where $p \geq hc + 1$. With that, in order to reasonably expect to be able to derive a contradiction, it will be required to for instance fit 5 pigeons into 2 holes. $\neg PHP_5^{2,2}$ axioms will be as follows:

$$\begin{aligned} OR: &= \left\{1_1 2_1, 1_2 2_2, 1_3 2_3, 1_4 2_4, 1_5 2_5\right\} \\ NAND: &= \left\{\begin{array}{l} \overline{1_1 1_2 1_3}, \overline{1_1 1_2 1_4}, \overline{1_1 1_2 1_5}, \overline{1_1 1_3 1_4}, \overline{1_1 1_3 1_5}, \\ \overline{1_1 1_4 1_5}, \overline{1_2 1_3 1_4}, \overline{1_2 1_3 1_5}, \overline{1_2 1_4 1_5}, \overline{1_3 1_4 1_5}, \\ \overline{2_1 2_2 2_3}, \overline{2_1 2_2 2_4}, \overline{2_1 2_2 2_5}, \overline{2_1 2_3 2_4}, \overline{2_1 2_3 2_5}, \\ \overline{2_1 2_4 2_5}, \overline{2_2 2_3 2_4}, \overline{2_2 2_3 2_5}, \overline{2_2 2_4 2_5}, \overline{2_3 2_4 2_5} \end{array}\right\} \end{aligned} \tag{4.6}$$

The NAND-clauses have increased in size compared to the $\neg PHP_p^h$ clauses, because now no 3 distinct pigeons can occupy the same hole as opposed to the previous cases where no 2 pigeons could occupy the same hole.

It is immediate that $\neg PHP_5^{2,2}$ will have a wide refutation given that NAND-axioms are greater than OR-axioms. We show its refutation as it well be useful later.

Proposition 4.7. *$\neg PHP_5^{2,2}$ is inconsistent in Neg.*

Proof.

$$\dfrac{\dfrac{\overline{2_2 2_4 2_5}\quad \dfrac{\overline{1_1 1_2 1_3}\quad \overline{2_1 2_4 2_5}}{\overline{1_2 1_3 2_4 2_5}}\,1_1 2_1}{\overline{1_3 2_4 2_5}}\,1_2 2_2 \qquad \overline{2_3 2_4 2_5}}{\overline{2_4 2_5}}\,1_3 2_3$$

$$\dfrac{\dfrac{\overline{1_1 1_3 1_4}\quad \overline{2_1 2_2 2_5}}{\overline{1_3 1_4 2_2 2_5}}\,1_1 2_1 \qquad \overline{1_2 1_3 1_4}}{\overline{1_3 1_4 2_5}}\,1_2 2_2 \qquad \dfrac{\dfrac{\overline{1_1 1_2 1_4}\quad \overline{2_1 2_3 2_5}}{\overline{1_2 1_4 2_3 2_5}}\,1_1 2_1 \qquad \overline{2_2 2_3 2_5}}{\overline{1_4 2_3 2_5}}\,1_2 2_2}{\overline{1_4 2_5}}\,1_3 2_3$$

$$\dfrac{\dfrac{\overline{2_1 2_2 2_3}\quad \dfrac{\overline{1_1 1_4 1_5}}{\overline{2_2 2_3 1_4 1_5}}\,1_1 2_1}{\overline{2_3 1_4 1_5}}\,1_2 2_2 \qquad \overline{1_3 1_4 1_5}}{\overline{1_4 1_5}}\,1_3 2_3$$

$$\dfrac{\dfrac{\overline{2_1 2_3 2_4}\quad \overline{1_1 1_2 1_5}}{\overline{2_3 2_4 1_2 1_5}}\,1_1 2_1 \qquad \overline{2_2 2_3 2_4}}{\overline{2_3 2_4 1_5}}\,1_2 2_2 \qquad \dfrac{\dfrac{\overline{2_1 2_2 2_4}\quad \overline{1_1 1_3 1_5}}{\overline{2_2 2_4 1_3 1_5}}\,1_1 2_1 \qquad \overline{1_2 1_3 1_5}}{\overline{2_4 1_3 1_5}}\,1_3 2_3}{\overline{2_4 1_5}}\,1_2 2_2$$

$$\dfrac{\dfrac{\overline{2_4 2_5}\quad \overline{1_4 2_5}}{\overline{2_5}}\,1_4 2_4 \qquad \dfrac{\overline{1_4 1_5}\quad \overline{1_5 2_4}}{\overline{1_5}}\,1_4 2_4}{\{\}}\,1_5 2_5$$

□

Not only does this NAND-OR theory have axioms where the NAND-clauses are greater than the OR-clauses, but the NAND-clauses must grow once more in order for an inconsistency to be derived.

Proposition 4.8. *In every refutation of $\neg PHP_5^{2,2}$ in Neg the NAND-clauses grow longer than the OR-clauses.*

Proof. Using an arbitrarily chosen OR-clause as the side condition, the first creation of a new NAND-clause will always create one of size 4.

$$\frac{\overline{1_i 1_{j_1} 1_{j_2}} \quad \overline{2_i 2_{j_3} 2_{j_4}}}{\overline{1_{j_1} 1_{j_2} 2_{j_3} 2_{j_4}}}\, 1_i 2_i$$

We have the conditions of remark 3.4 so we take the difference between the sum of the lengths of the premise NAND-clauses and the length of the OR-clause. The proof is similar to the one of proposition 3.8 and so is not heavily commented.

□

We have found that conjecture 2 does not hold. A theory being in NAND-OR form is not enough to guarantee it will have a narrow refutation.

4.3 NAND-binary NAND-OR theories

The counter-example of section 4.2 does not disprove conjecture 3 nor conjecture 4. In section 2.1 it was shown that NAND-OR theories derived from GNF theories will always be NAND-binary (all NAND-axioms in the theory are binary). Further, section 4.1 showed that conjecture 2 holds for $\neg PHP_p^h$ – a NAND-binary theory not derived directly from a GNF theory –. Notice also that in both $\neg PHP_p^h$ and $\neg PHP_5^{2,2}$ NAND-clauses grew only in the first step, after which we were able to derive smaller NAND-clauses until we eventually derived the empty clause. Remark 3.6 guarantees that Rneg applications using only binary NAND-clauses will produce NAND-clauses of the size of the premise OR-clause at greatest. If in addition theories need only grow once (in the first step for example) then all NAND-binary NAND-OR theories have narrow refutations. Our counter-example to conjecture 3 shows that this is not the case.

Before we move on to the next counter-example, we would like to know both that any instance of $\neg PHP_p^{h,c}$ for $c > 1$ can be refuted in Neg and that all refutations are wide. This result will be useful when presenting the counter-example to conjecture 3 which will be a modification to such a theory.

We make explicit in set notation the axioms of a general $\neg PHP_p^{h,c}$ theory as was done in 3.7 for $\neg PHP_p^h$:

$$\begin{aligned} OR: &= \left\{1_i 2_i \ldots h_i | i \leq p\right\} \\ NAND: &= \left\{\overline{x_{i_1} x_{i_2} \ldots x_{i_{c+1}}} | 0 < i_1 < i_2 < \ldots i_{c+1} \leq p\ , 0 < x \leq h\right\} \end{aligned} \tag{4.9}$$

The following definitions will be useful in proving that $\neg PHP_p^{h,c}$ theories have wide refutations:

Definition 4.10. *We define a $\neg PHP_p^{h,c}$ NAND-clause to be 'n-hole-distinct' if it comprises of atoms where each atom represents the same hole as at most $n-1$ other atoms in the NAND-clause (there can be at most n atoms representing the same hole).*

Example:

- $\overline{2_1 2_3 2_4 7_3}$ is 3-hole-distinct
- $\overline{1_1 2_1 3_2}$ is 1-hole distinct

In general if a NAND-clause is n-hole-distinct for a particular n then it is also m-hole-distinct for any $m > n$.

Definition 4.11. *We define a $\neg PHP_p^{h,c}$ NAND-clause to be 'hole-n-saturated' if it contains exactly c (the hole capacity of the theory in question) different atoms of the form n_i.*

Example:

- In $\neg PHP_5^{2,2}$, $\overline{1_2 1_3 2_4 2_5}$ is hole-1-saturated and hole-2-saturated.
- In $\neg PHP_4^{3,1}$ $\overline{1_1 2_2 3_2}$ is hole-1-saturated, hole-2-saturated and hole-3-saturated

Lemma 4.12. *Using only $\neg PHP_p^{h,c}$ axioms (4.9), every c-hole-distinct NAND-clause of size $h \cdot c$ can be proven in a single rule application.*

Proof. This proof is a generalisation of lemma 4.2 now also accommodating capacities for the holes. Let $\overline{1_{j_1} \ldots 1_{j_c} 2_{j_{c+1}} \ldots 2_{j_{2c}} \ \ldots \ h_{j_{(h\text{-}1)c+1}} \ldots h_{j_{hc}}}$ be an arbitrary c-hole-distinct NAND-clause of size $h \cdot c$. Letting $P = \{1, ..., p\}$, all axiomatic OR-clauses have the form $1_\pi 2_\pi \ldots h_\pi$, $\pi \in P$. Now letting $H = \{1, ..., h\}$ and $j_i \in P$, a generalised construction of a NAND-clause of size $h \cdot c$ from axioms will look as follows:

$$\frac{\overline{1_\pi 1_{j_1} \ldots 1_{j_c}} \quad \overline{2_\pi 2_{j_{c+1}} \ldots 2_{j_{2c}}} \quad \ldots \quad \overline{h_\pi h_{j_{(h\text{-}1)c+1}} \ldots h_{j_{hc}}}}{\overline{1_{j_1} \ldots 1_{j_c} 2_{j_{c+1}} \ldots 2_{j_{2c}} \ \ldots \ h_{j_{(h\text{-}1)c+1}} \ldots h_{j_{hc}}}} \, 1_\pi 2_\pi \ldots h_\pi$$

To derive $\overline{1_{j_1} \ldots 1_{j_c} 2_{j_{c+1}} \ldots 2_{j_{2c}} \ \ldots \ h_{j_{(h\text{-}1)c+1}} \ldots h_{j_{hc}}}$ for $j_i \in P$, take any $\pi \notin \{j_i, \ldots j_{hc}\}$, which exists since $h \cdot c < p$. We have axiom $1_\pi 2_\pi \ldots h_\pi \in OR$ and for each $1 \leq k \leq h$, $\overline{k_\pi k_{i_1} \ldots k_{i_c}} \in NAND$. These axioms give $\overline{1_{j_1} \ldots 1_{j_c} 2_{j_{c+1}} \ldots 2_{j_{2c}} \ \ldots \ h_{j_{(h\text{-}1)c+1}} \ldots h_{j_{hc}}}$ for $j_i \in P$ by a single rule application of Rneg. Remark 3.4 assures that we get a concluding NAND-clause of size $h \cdot c$.

□

Lemma 4.12 also shows that refutations must be wide for capacity greater than 1.

Corollary 4.13. *In every inconsistency proof of $\neg PHP_p^{h,c}$ ($c > 1$) in Neg, NAND-clauses grow longer than the OR-clauses.*

Proof. This follows immediately from lemma 4.12. Using an arbitrarily chosen OR-clause as the side condition, the first creation of a new NAND-clause will always create one of size $h \cdot c$. For $c > 1$ this will give a NAND-clause greater than the OR-clauses. The proof of this is a generalisation of the one in proposition 4.8. The illustrative rule application is next to identical to the one in lemma 4.12.

□

Lemma 4.14. *Given our $\neg PHP_p^{h,c}$ axioms (4.9), if all c-hole-distinct NAND-clauses of size k (for $k \leq h \cdot c$) are provable (in Neg), then all c-hole-distinct NAND-clauses of size $k-1$ are provable.*

Proof. This proof is a generalisation of lemma 4.3 without a size restriction on the NAND-clauses used in the derivation. Consider an arbitrarily chosen c-hole-distinct NAND-clause of size $k-1$, $\overline{\alpha^1_{j_1}\ldots\alpha^1_{j_c}\ \ldots\ \alpha^m_{j_{(m\text{-}1)c+1}}\ldots\alpha^m_{j_{mc}}\ldots\alpha^n_{j_{k-1}}}$.[2] We assume without loss of generality that the first $m \cdot c$ atoms in this desired NAND-clause are those that represent holes for which the NAND-clause is hole-α^i-saturated (m then represents the number of different such holes). As in the proof of lemma 4.12, it is always possible to find an OR-clause $1_\pi 2_\pi \ldots h_\pi$ with $\pi \notin \{j_i \in P | 1 \leq i \leq k, i \in \mathbf{N}\}$ (since $k \leq h \cdot c < p$). Given this and the fact that our desired NAND-clause is c-hole-distinct, it is possible to separate the OR-clause into two parts, $\alpha^1_\pi \ldots \alpha^m_\pi \beta^1_\pi \ldots \beta^{h-m}_\pi$. The α^i_π's, $i \leq m$, are those atoms in the OR-clause for which $\overline{\alpha^1_{j_1}\ldots\alpha^1_{j_c}\ \ldots\ \alpha^m_{j_{(m\text{-}1)c+1}}\ldots\alpha^m_{j_{mc}}\ldots\alpha^n_{j_{k-1}}}$ is hole-α^i-saturated. The β^i_π's are the atoms in the OR-clause for which the concluding NAND-clause is not hole-β-saturated. With that we derive our chosen NAND-clause as follows:

$$\frac{\overline{\alpha^1_\pi\alpha^1_{j_1}\ldots\alpha^1_{j_c}}\ldots\overline{\alpha^m_\pi\alpha^m_{j_{(m\text{-}1)c+1}}\ldots\alpha^m_{j_{mc}}}\ \ \overline{\beta^1_\pi\alpha^1_{j_1}\ldots\alpha^n_{j_{k-1}}}\ldots\overline{\beta^{h-m}_\pi\alpha^1_{j_1}\ldots\alpha^n_{j_{k-1}}}}{\overline{\alpha^1_{j_1}\ldots\alpha^1_{j_c}\ \ldots\ \alpha^m_{j_{(m\text{-}1)c+1}}\ldots\alpha^m_{j_{mc}}\ldots\alpha^n_{j_{k-1}}}}\ \alpha^1_\pi\ldots\alpha^m_\pi\beta^1_\pi\ldots\beta^{h-m}_\pi$$

Each $\overline{\alpha^i_\pi\alpha^i_{j_r}\ldots\alpha^i_{j_s}}$ among the premises is axiomatic while each $\overline{\beta^i_\pi\alpha^1_{j_1}\ldots\alpha^n_{j_{k-1}}}$ is a c-hole-distinct NAND-clause of size k, derivable by assumption. Remark 3.5 assures that we have a concluding NAND-clause of size $k-1$.

□

Now similar to the strategy in proving $\neg PHP_p^h$ theories to have narrow refutations, we show that $\neg PHP_p^{h,c}$ theories are refutable:

Proposition 4.15. *Given our $\neg PHP_p^{h,c}$ axioms (4.9), for each $1 \leq k \leq h \cdot c$, every c-hole-distinct NAND-clause of length k is provable in Neg.*

Proof. The proof is by induction on k, with the basis $k = h \cdot c$ given by lemma 4.12, and induction step by lemma 4.14.

□

Corollary 4.16. *$\neg PHP_p^{h,c}$ is refutable in Neg.*

Proof. This follows immediately from proposition 4.15. All unary NAND-clauses are provable since any unary NAND-clause is c-hole-distinct. We can now use equation 3.1 taking any OR-clause to perform the final step as in 3.2 selecting what unary NAND-clauses that are necessary.

[2] The superscripts of these variables are again simply to distinguish from one another.

□

Thus we have shown that any $\neg PHP_p^{h,c}$ theory has a refutation and every such refutation is wide (for $c > 1$), as opposed to $\neg PHP_p^h$ theories which were narrow.

We now present a NAND-binary NAND-OR theory which will be a modification of a $\neg PHP_p^{h,c}$ theory and will be a counter-example of conjecture 3.

4.3.1 $bin(\neg PHP_7^{3,2})$

Consider $\neg PHP_7^{3,2}$ shown below in abbreviated form:

$$\begin{aligned} OR &:= \left\{1_1 2_1 3_1, 1_2 2_2 3_2, \ \ldots\ 1_7 2_7 3_7\right\} \\ NAND &:= \left\{\begin{array}{l} \overline{1_1 1_2 1_3}, \overline{1_1 1_2 1_4}, \ \ldots\ , \overline{1_5 1_6 1_7}, \\ \overline{2_1 2_2 2_3}, \overline{2_1 2_2 2_4}, \ \ldots\ , \overline{2_5 2_6 2_7}, \\ \overline{3_1 3_2 3_3}, \overline{3_1 3_2 3_4}, \ \ldots\ , \overline{3_5 3_6 3_7} \end{array}\right\} \end{aligned} \tag{4.17}$$

By corollaries 4.13 and 4.16 we know that we can derive the empty clause from this theory and that every such derivation is wide. While adding extra clauses to this theory may affect whether NAND-clauses greater than the OR-clauses are required to derive the empty clause, it will not affect its derivability. With that, in similar fashion to the weakening rule of traditional resolution, we can add clauses to our theory without affecting its unsatisfiabilty. Consider now $bin(\neg PHP_7^{3,2})$, a transformation of $\neg PHP_7^{3,2}$ into a NAND-binary NAND-OR theory:

$$\begin{aligned} OR &:= \left\{\begin{array}{l} 1_1' 1_2' 1_3', 1_1' 1_2' 1_4', \ \ldots\ , 1_5' 1_6' 1_7', \\ 2_1' 2_2' 2_3', 2_1' 2_2' 2_4', \ \ldots\ , 2_5' 2_6' 2_7', \\ 3_1' 3_2' 3_3', 3_1' 3_2' 3_4', \ \ldots\ , 3_5' 3_6' 3_7', \\ 1_1 2_1 3_1, 1_2 2_2 3_2, \ \ldots\ 1_7 2_7 3_7 \end{array}\right\} \\ NAND &:= \left\{\begin{array}{l} \overline{1_1 1_1'}, \overline{1_2 1_2'}, \overline{1_3 1_3'}, \overline{1_4 1_4'}, \overline{1_5 1_5'}, \overline{1_6 1_6'}, \overline{1_7 1_7'}, \\ \overline{2_1 2_1'}, \overline{2_2 2_2'}, \overline{2_3 2_3'}, \overline{2_4 2_4'}, \overline{2_5 2_5'}, \overline{2_6 2_6'}, \overline{2_7 2_7'}, \\ \overline{3_1 3_1'}, \overline{3_2 3_2'}, \overline{3_3 3_3'}, \overline{3_4 3_4'}, \overline{3_5 3_5'}, \overline{3_6 3_6'}, \overline{3_7 3_7'} \end{array}\right\} \end{aligned} \tag{4.18}$$

All NAND-Clauses from 4.17 with their atoms relabelled are turned into OR-clauses while new, binary NAND-clauses are added. We also retain OR-clauses from 4.17. Observe that each $\neg PHP_7^{3,2}$ NAND-clause can be derived in one step. For example:

$$\frac{\overline{1_1 1_1'} \quad \overline{1_2 1_2'} \quad \overline{1_3 1_3'}}{\overline{1_1 1_2 1_3}}\ 1_1' 1_2' 1_3' \tag{4.19}$$

In this way we can derive every NAND-clause from $\neg PHP_7^{3,2}$ in our $bin(\neg PHP_7^{3,2})$ theory. At this point it is clear that $\neg PHP_7^{3,2}$ is a subtheory of $bin(\neg PHP_7^{3,2})$ as all $\neg PHP_7^{3,2}$ axioms are derivable from $bin(\neg PHP_7^{3,2})$. Letting $D[X]$ represent the deductive closure in Neg of a theory X – the set of all formulae derivable from X – we have the following equality:

$$D[bin(\neg PHP_7^{3,2})] = D[bin(\neg PHP_7^{3,2})] \cup D[\neg PHP_7^{3,2}] \tag{4.20}$$

Every axiom of $\neg PHP_7^{3,2}$ is either present among the $bin(\neg PHP_7^{3,2})$ axioms or is derivable from them. We are now ready to prove that $bin(\neg PHP_7^{3,2})$ does not have a narrow refutation in Neg.

Proposition 4.21. *There is no Neg refutation of $bin(\neg PHP_7^{3,2})$ where the NAND-clauses don't grow beyond the size of the OR-clauses.*

Proof. Every $\neg PHP_7^{3,2}$ OR-clause is already present in $bin(\neg PHP_7^{3,2})$ and every $\neg PHP_7^{3,2}$ NAND-clause can be derived in $bin(\neg PHP_7^{3,2})$ (as illustrated in figure 4.19) with the following general Rneg application ($\alpha \in \{1,2,3\}$):

$$\frac{\overline{\alpha_i\alpha_i'} \quad \overline{\alpha_j\alpha_j'} \quad \overline{\alpha_k\alpha_k'}}{\overline{\alpha_i\alpha_j\alpha_k}}\ \alpha_i'\alpha_j'\alpha_k'$$

This gives now the following collection of clauses:

$$OR := \left\{ \begin{array}{l} 1_1'1_2'1_3', 1_1'1_2'1_4', \ \ldots\ , 1_5'1_6'1_7', \\ 2_1'2_2'2_3', 2_1'2_2'2_4', \ \ldots\ , 2_5'2_6'2_7', \\ 3_1'3_2'3_3', 3_1'3_2'3_4', \ \ldots\ , 3_5'3_6'3_7', \\ 1_12_13_1, 1_22_23_2, \ \ldots\ 1_72_73_7 \end{array} \right\}$$

$$NAND := \left\{ \begin{array}{l} \overline{1_11_1'}, \overline{1_21_2'}, \overline{1_31_3'}, \overline{1_41_4'}, \overline{1_51_5'}, \overline{1_61_6'}, \overline{1_71_7'}, \\ \overline{2_12_1'}, \overline{2_22_2'}, \overline{2_32_3'}, \overline{2_42_4'}, \overline{2_52_5'}, \overline{2_62_6'}, \overline{2_72_7'}, \\ \overline{3_13_1'}, \overline{3_23_2'}, \overline{3_33_3'}, \overline{3_43_4'}, \overline{3_53_5'}, \overline{3_63_6'}, \overline{3_73_7'}, \\ \overline{1_11_21_3}, \overline{1_11_21_4}, \overline{1_11_21_5}, \ \ldots\ , \overline{1_51_61_7}, \\ \overline{2_12_22_3}, \overline{2_12_22_4}, \overline{2_12_22_5}, \ \ldots\ , \overline{2_52_62_7}, \\ \overline{3_13_23_3}, \overline{3_13_23_4}, \overline{3_13_23_5}, \ \ldots\ , \overline{3_53_63_7} \end{array} \right\} \tag{4.22}$$

We already know from corollary 4.13 that a refutation using only $\neg PHP_7^{3,2}$ clauses will require NAND-clauses larger than the OR-clauses. The only fact that remains to be shown is that the NAND-clauses must grow even when using some combination of axioms and clauses that include at least one clause unique to $bin(\neg PHP_7^{3,2})$.

We use the OR-clauses of the form $1_i2_i3_i$ and the binary axioms to create all clauses of the form $1_i'2_i'3_i'$:

We add these clauses to our theory:

$$\frac{\overline{1_i 1_i'} \quad \overline{2_i 2_i'} \quad \overline{3_i 3_i'}}{\overline{1_i' 2_i' 3_i'}}\, 1_i 2_i 3_i$$

$$OR := \left\{ \begin{array}{l} 1_1' 1_2' 1_3', 1_1' 1_2' 1_4', \ \ldots\ , 1_5' 1_6' 1_7', \\ 2_1' 2_2' 2_3', 2_1' 2_2' 2_4', \ \ldots\ , 2_5' 2_6' 2_7', \\ 3_1' 3_2' 3_3', 3_1' 3_2' 3_4', \ \ldots\ , 3_5' 3_6' 3_7', \\ 1_1 2_1 3_1, 1_2 2_2 3_2, \ \ldots\ 1_7 2_7 3_7 \end{array} \right\}$$

$$NAND := \left\{ \begin{array}{l} \overline{1_1 1_1'}, \overline{1_2 1_2'}, \overline{1_3 1_3'}, \overline{1_4 1_4'}, \overline{1_5 1_5'}, \overline{1_6 1_6'}, \overline{1_7 1_7'}, \\ \overline{2_1 2_1'}, \overline{2_2 2_2'}, \overline{2_3 2_3'}, \overline{2_4 2_4'}, \overline{2_5 2_5'}, \overline{2_6 2_6'}, \overline{2_7 2_7'}, \\ \overline{3_1 3_1'}, \overline{3_2 3_2'}, \overline{3_3 3_3'}, \overline{3_4 3_4'}, \overline{3_5 3_5'}, \overline{3_6 3_6'}, \overline{3_7 3_7'}, \\ \overline{1_1 1_2 1_3}, \overline{1_1 1_2 1_4}, \overline{1_1 1_2 1_5}, \ \ldots\ , \overline{1_5 1_6 1_7}, \\ \overline{2_1 2_2 2_3}, \overline{2_1 2_2 2_4}, \overline{2_1 2_2 2_5}, \ \ldots\ , \overline{2_5 2_6 2_7}, \\ \overline{3_1 3_2 3_3}, \overline{3_1 3_2 3_4}, \overline{3_1 3_2 3_5}, \ \ldots\ , \overline{3_5 3_6 3_7} \\ \overline{1_1' 2_1' 3_1'}, \overline{1_2' 2_2' 3_2'}, \overline{1_3' 2_3' 3_3'}, \ \ldots\ , \overline{1_7' 2_7' 3_7'} \end{array} \right\} \tag{4.23}$$

At this point we cannot create a new NAND-clause of size 3 or less. In all derivations up until this point we have used only binary NAND-clauses together with each type of OR-clause that we have among our axioms. This created NAND-clauses of the type $\overline{\alpha_i \alpha_j \alpha_k}$ and $\overline{1_i' 2_i' 3_i'}$ respectively, each of size 3. We show now that involving even a single ternary NAND-clause will produce a NAND-clause of size greater than 3. First using an OR-axiom unique to $bin(\neg PHP_7^{3,2})$ ($\beta, \gamma \in \{1,2,3\}$):

$$\frac{\overline{\alpha_i \alpha_i'} \quad \overline{\alpha_j \alpha_j'} \quad \overline{\alpha_k' \beta_k' \gamma_k'}}{\overline{\alpha_i \alpha_j \beta_k' \gamma_k'}}\, \alpha_i' \alpha_j' \alpha_k'$$

All premise NAND-clauses are disjoint so by remark 3.4 we take the difference of the sum of the lengths of the premie NAND-clauses and the OR-clause. This gives a conclusion of length 4. The same applies when using a $\neg PHP_7^{3,2}$ OR-axiom instead:

$$\frac{\overline{\alpha_i \alpha_i'} \quad \overline{\beta_i \beta_i'} \quad \overline{\gamma_i \gamma_j \gamma_k}}{\overline{\alpha_i' \beta_i' \gamma_j \gamma_k}}\, \alpha_i \beta_i \gamma_i$$

Involving more than one ternary NAND-clause would only increase the sum of the lengths of the premise NAND-clauses as it will still be the case that these clauses are all distinct. At this stage, any arbitrary Rneg application using two ternary and one binary NAND-clause will yield a NAND-clause of size 5. First using an OR-axiom unique to $bin(\neg PHP_7^{3,2})$:

$$\frac{\overline{\alpha_i\alpha_i'} \quad \overline{\alpha_j'\beta_j'\gamma_j'} \quad \overline{\alpha_k'\beta_k'\gamma_k'}}{\overline{\alpha_i\beta_j'\gamma_j'\beta_k'\gamma_k'}}\,\alpha_i'\alpha_j'\alpha_k'$$

Now using a $\neg PHP_7^{3,2}$ OR-axiom instead:

$$\frac{\overline{\alpha_i\alpha_i'} \quad \overline{\beta_i\beta_l\beta_m} \quad \overline{\gamma_i\gamma_j\gamma_k}}{\overline{\alpha_i'\beta_l\beta_m\gamma_j\gamma_k}}\,\alpha_i\beta_i\gamma_i$$

Using all ternary NAND-clauses will yield a NAND-clause of size 6. Again we first using an OR-axiom unique to $bin(\neg PHP_7^{3,2})$:

$$\frac{\overline{\alpha_i'\beta_i'\gamma_i'} \quad \overline{\alpha_j'\beta_j'\gamma_j'} \quad \overline{\alpha_k'\beta_k'\gamma_k'}}{\overline{\beta_i'\gamma_i'\beta_j'\gamma_j'\beta_k'\gamma_k'}}\,\alpha_i'\alpha_j'\alpha_k'$$

And now using a $\neg PHP_7^{3,2}$ OR-axiom instead:

$$\frac{\overline{\alpha_i\alpha_n\alpha_o} \quad \overline{\beta_i\beta_l\beta_m} \quad \overline{\gamma_i\gamma_j\gamma_k}}{\overline{\alpha_n\alpha_o\beta_l\beta_m\gamma_j\gamma_k}}\,\alpha_i\beta_i\gamma_i$$

Having exhausted all possibilities, we can conclude that any refutation of $bin(\neg PHP_7^{3,2})$ must use a NAND-clause of size 4.

□

Hence, having found a counterexample to conjecture 3 it is not the case that any NAND-binary NAND-OR theory has a narrow refutation. That is, a theory being in NAND-binary NAND-OR from is not a sufficient condition to guarantee a narrow proof. As a final comment we show that the above result would not have held for any smaller instance of $bin(\neg PHP_p^{h,c})$ $(c > 1)$.

Consider $bin(\neg PHP_5^{2,2})$:

$$OR := \left\{ \begin{array}{l} 1_1'1_2'1_3', 1_1'1_2'1_4', 1_1'1_2'1_5', 1_1'1_3'1_4', 1_1'1_3'1_5', \\ 1_1'1_4'1_5', 1_2'1_3'1_4', 1_2'1_3'1_5', 1_2'1_4'1_5', 1_3'1_4'1_5', \\ 2_1'2_2'2_3', 2_1'2_2'2_4', 2_1'2_2'2_5', 2_1'2_3'2_4', 2_1'2_3'2_5', \\ 2_1'2_4'2_5', 2_2'2_3'2_4', 2_2'2_3'2_5', 2_2'2_4'2_5', 2_3'2_4'2_5', \\ 1_11_2, 1_22_2, 1_32_3, 1_42_4, 1_52_5 \end{array} \right\} \tag{4.24}$$

$$NAND := \left\{ \begin{array}{l} \overline{1_11_1'}, \overline{1_21_2'}, \overline{1_31_3'}, \overline{1_41_4'}, \overline{1_51_5'}, \\ \overline{2_12_1'}, \overline{2_22_2'}, \overline{2_32_3'}, \overline{2_42_4'}, \overline{2_52_5'}, \\ \overline{3_13_1'}, \overline{3_23_2'}, \overline{3_33_3'}, \overline{3_43_4'}, \overline{3_53_5'} \end{array} \right\}$$

After recreating its $\neg PHP_5^{2,2}$ NAND-clauses we get:

$$OR := \left\{ \begin{array}{l} 1_1'1_2'1_3', 1_1'1_2'1_4', 1_1'1_2'1_5', 1_1'1_3'1_4', 1_1'1_3'1_5', \\ 1_1'1_4'1_5', 1_2'1_3'1_4', 1_2'1_3'1_5', 1_2'1_4'1_5', 1_3'1_4'1_5', \\ 2_1'2_2'2_3', 2_1'2_2'2_4', 2_1'2_2'2_5', 2_1'2_3'2_4', 2_1'2_3'2_5', \\ 2_1'2_4'2_5', 2_2'2_3'2_4', 2_2'2_3'2_5', 2_2'2_4'2_5', 2_3'2_4'2_5', \\ 1_11_2, 1_22_2, 1_32_3, 1_42_4, 1_52_5 \end{array} \right\}$$

$$NAND := \left\{ \begin{array}{l} \overline{1_11_21_3}, \overline{1_11_21_4}, \overline{1_11_21_5}, \overline{1_11_31_4}, \overline{1_11_31_5}, \\ \overline{1_11_41_5}, \overline{1_21_31_4}, \overline{1_21_31_5}, \overline{1_21_41_5}, \overline{1_31_41_5}, \\ \overline{2_12_22_3}, \overline{2_12_22_4}, \overline{2_12_22_5}, \overline{2_12_32_4}, \overline{2_12_32_5}, \\ \overline{2_12_42_5}, \overline{2_22_32_4}, \overline{2_22_32_5}, \overline{2_22_42_5}, \overline{2_32_42_5} \\ \overline{1_11_1'}, \overline{1_21_2'}, \overline{1_31_3'}, \overline{1_41_4'}, \overline{1_51_5'}, \overline{2_12_1'}, \overline{2_22_2'}, \\ \overline{2_32_3'}, \overline{2_42_4'}, \overline{2_52_5'}, \overline{3_13_1'}, \overline{3_23_2'}, \overline{3_33_3'}, \overline{3_43_4'}, \\ \overline{3_53_5'} \end{array} \right\} \tag{4.25}$$

We are now able to involve a ternary NAND-clause in a rule application without creating a NAND-clause of size 4 or greater. What is more, we can use this to the effect of creating a new binary NAND-clause:

$$\dfrac{\dfrac{\overline{1_11_21_3} \quad \overline{2_32_3'}}{\overline{1_11_22_3'}}\, 1_32_3 \quad \dfrac{\overline{1_11_21_4} \quad \overline{2_42_4'}}{\overline{1_11_22_4'}}\, 1_42_4 \quad \dfrac{\overline{1_11_21_5} \quad \overline{2_52_5'}}{\overline{1_11_22_5'}}\, 1_52_5}{\overline{1_11_2}}\, 2_3'2_4'2_5' \tag{4.26}$$

The derivation in 4.26 can be generalised such that any binary NAND-clause of the form, $\overline{x_ix_j}$ where $i, j \in \{1 \ldots 5\}$ and $i \neq j$, can be created.

With that, we can now prove that $bin(\neg PHP_5^{2,2})$ has a narrow refutation in Neg:

Proposition 4.27. *There exists a Neg refutation of $bin(\neg PHP_5^{2,2})$ where the NAND-clauses don't grow beyond the size of the OR-clauses.*

Proof. Take $\alpha, \beta \in \{1,2\}$ where $\alpha \neq \beta$ and take $i_1, \ldots, i_5 \in \{1, \ldots, 5\}$ where $i_j \neq i_k$. We create every binary NAND-clause of the form $\overline{x_ix_j}$ with the following generalised sequence of rule applications:

$$\dfrac{\dfrac{\overline{\alpha_{i_1}\alpha_{i_2}\alpha_{i_3}} \quad \overline{\beta_{i_3}\beta_{i_3}'}}{\overline{\alpha_{i_1}\alpha_{i_2}\beta_{i_3}'}}\, \alpha_{i_3}\beta_{i_3} \quad \dfrac{\overline{\alpha_{i_1}\alpha_{i_2}\alpha_{i_4}} \quad \overline{\beta_{i_4}\beta_{i_4}'}}{\overline{\alpha_{i_1}\alpha_{i_2}\beta_{i_4}'}}\, \alpha_{i_4}\beta_{i_4} \quad \dfrac{\overline{\alpha_{i_1}\alpha_{i_2}\alpha_{i_5}} \quad \overline{\beta_{i_5}\beta_{i_5}'}}{\overline{\alpha_{i_1}\alpha_{i_2}\beta_{i_5}'}}\, \alpha_{i_5}\beta_{i_5}}{\overline{\alpha_{i_1}\alpha_{i_2}}}\, \beta_{i_3}'\beta_{i_4}'\beta_{i_5}'$$

Every rule application in the above figure is valid and uses clauses from 4.25 so the entire sequence of rule applications is valid. Observe now that we have present among our $bin(\neg PHP_5^{2,2})$ OR-clauses, clauses of the form $1_j 2_j$ where $j \in \{1,\ldots,5\}$ and NAND-clauses of the form $\overline{x_i x_j}$. These are precisely the axioms of the theory $\neg PHP_5^2$ (see 3.7). By corollary 4.5 we can derive the empty clause from these axioms without having NAND-clauses grow passed the size of the OR-clauses. With that, we can simply continue with these clauses to derive the empty clause in $bin(\neg PHP_5^{2,2})$.

□

4.4 GNF theories

On a final note, we give some words on the search for a counterexample to conjecture 4. We saw in the previous section that translating $\neg PHP_5^{2,2}$, our counterexample of conjecture 2, into a NAND-Binary NAND-OR theory did not result in a counterexample of conjecture 3. Converting it to a GNF theory with the intention of finding a counterexample of conjecture 4 will not work either. We translate $\neg PHP_5^{2,2}$ into a GNF theory, $GNF(\neg PHP_5^{2,2})$ (see appendix A and equations 2.4 - 2.9 for translation):

$$OR := \left\{ \begin{array}{l} 1_1 1_1', 1_2 1_2', 1_3 1_3', 1_4 1_4', 1_5 1_5', \\ 2_1 2_1', 2_2 2_2', 2_3 2_3', 2_4 2_4', 2_5 2_5', \\ \alpha_1 1_1 2_1, \alpha_2 1_2 2_2, \ldots, \alpha_5 1_5 2_5, \\ \alpha_6 1_1' 1_2' 1_3', \alpha_7 1_1' 1_2' 1_4', \ldots, \alpha_{15} 1_3' 1_4' 1_5', \\ \alpha_{16} 2_1' 2_2' 2_3', \alpha_{17} 2_1' 2_2' 2_4', \ldots, \alpha_{25} 2_3' 2_4' 2_5' \end{array} \right\}$$

$$NAND := \left\{ \begin{array}{l} \overline{1_1 1_1'}, \overline{1_2 1_2'}, \overline{1_3 1_3'}, \overline{1_4 1_4'}, \overline{1_5 1_5'}, \\ \overline{2_1 2_1'}, \overline{2_2 2_2'}, \overline{2_3 2_3'}, \overline{2_4 2_4'}, \overline{2_5 2_5'}, \\ \overline{\alpha_1}, \overline{\alpha_2}, \overline{\alpha_3}, \overline{\alpha_4}, \overline{\alpha_5}, \overline{\alpha_6}, \ldots, \overline{\alpha_{25}}, \\ \overline{\alpha_1 1_1}, \overline{\alpha_2 1_2}, \overline{\alpha_3 1_3}, \overline{\alpha_4 1_4}, \overline{\alpha_5 1_5}, \\ \overline{\alpha_1 2_1}, \overline{\alpha_2 2_2}, \overline{\alpha_3 2_3}, \overline{\alpha_4 2_4}, \overline{\alpha_5 2_5}, \\ \overline{\alpha_6 1_1'}, \overline{\alpha_6 1_2'}, \overline{\alpha_6 1_3'}, \ldots, \overline{\alpha_{15} 1_5'}, \\ \overline{\alpha_{16} 2_1'}, \overline{\alpha_{16} 2_2'}, \overline{\alpha_{16} 2_3'}, \ldots, \overline{\alpha_{25} 2_5'} \end{array} \right\} \quad (4.28)$$

Notice that $GNF(\neg PHP_5^{2,2})$ contains $bin(\neg PHP_5^{2,2})$. $bin(\neg PHP_p^{h,c})$ was in fact discovered from converting $\neg PHP_p^{h,c}$ theories into GNF. It is immediate that $GNF(\neg PHP_5^{2,2})$ has a narrow refutation since it contains $\neg PHP_p^{h,c}$. The natural question now is, does this apply to $GNF(\neg PHP_7^{3,2})$ aswell? $GNF(\neg PHP_7^{3,2})$:

$$
\begin{aligned}
OR \;:=\; & \left\{ \begin{array}{l}
1_1 1_1', 1_2 1_2', 1_3 1_3', 1_4 1_4', 1_5 1_5', 1_6 1_6', 1_7 1_7', \\
2_1 2_1', 2_2 2_2', 2_3 2_3', 2_4 2_4', 2_5 2_5', 2_6 2_6', 2_7 2_7', \\
3_1 3_1', 3_2 3_2', 3_3 3_3', 3_4 3_4', 3_5 3_5', 3_6 3_6', 3_7 3_7', \\
\alpha_1 1_1 2_1 3_1, \alpha_2 1_2 2_2 3_2, \ldots, \alpha_7 1_7 2_7 3_7, \\
\alpha_8 1_1' 1_2' 1_3', \alpha_9 1_1' 1_2' 1_4', \ldots, \alpha_{42} 1_5' 1_6' 1_7', \\
\alpha_{43} 2_1' 2_2' 2_3', \alpha_{44} 2_1' 2_2' 2_4', \ldots, \alpha_{77} 2_5' 2_6' 2_7', \\
\alpha_{79} 3_1' 3_2' 3_4', \alpha_{80} 3_1' 3_2' 3_5', \ldots, \alpha_{112} 3_5' 3_6' 3_7'
\end{array} \right\} \\
NAND \;:=\; & \left\{ \begin{array}{l}
\overline{1_1 1_1'}, \overline{1_2 1_2'}, \overline{1_3 1_3'}, \overline{1_4 1_4'}, \overline{1_5 1_5'}, \overline{1_6 1_6'}, \overline{1_7 1_7'}, \\
\overline{2_1 2_1'}, \overline{2_2 2_2'}, \overline{2_3 2_3'}, \overline{2_4 2_4'}, \overline{2_5 2_5'}, \overline{2_6 2_6'}, \overline{2_7 2_7'}, \\
\overline{3_1 3_1'}, \overline{3_2 3_2'}, \overline{3_3 3_3'}, \overline{3_4 3_4'}, \overline{3_5 3_5'}, \overline{3_6 3_6'}, \overline{3_7 3_7'}, \\
\overline{\alpha_1}, \overline{\alpha_2}, \overline{\alpha_3}, \overline{\alpha_4}, \overline{\alpha_5}, \overline{\alpha_6}, \overline{\alpha_7}, \overline{\alpha_8}, \ldots, \overline{\alpha_{112}}, \\
\overline{\alpha_1 1_1}, \overline{\alpha_1 2_1}, \overline{\alpha_1 3_1}, \ldots, \overline{\alpha_7 1_7}, \overline{\alpha_7 2_7}, \overline{\alpha_7 3_7}, \\
\overline{\alpha_8 1_1'}, \overline{\alpha_8 1_2'}, \overline{\alpha_8 1_3'}, \overline{\alpha_9 1_1'}, \ldots, \overline{\alpha_{42} 1_7'}, \\
\overline{\alpha_{43} 2_1'}, \overline{\alpha_{43} 2_2'}, \overline{\alpha_{43} 2_3'}, \overline{\alpha_{44} 2_1'}, \ldots, \overline{\alpha_{77} 2_7'}, \\
\overline{\alpha_{78} 3_1'}, \overline{\alpha_{78} 3_2'}, \overline{\alpha_{78} 3_3'}, \overline{\alpha_{79} 3_1'}, \ldots, \overline{\alpha_{112} 3_7'}
\end{array} \right\}
\end{aligned}
\tag{4.29}
$$

The fact that $bin(\neg PHP_7^{3,2})$ is a subtheory of $GNF(\neg PHP_7^{3,2})$ does not in this case discount it from only having wide refutations. Translating to GNF has introduced several new types of OR and NAND-clauses. While using $bin(\neg PHP_7^{3,2})$ clauses only would of course give a wide refutation, the presence of these new clauses presents the potential to refute $GNF(\neg PHP_7^{3,2})$ by some other means. A proof by exhaustion has proven to complex to achieve by hand. Thus we have no counterexample of conjecture 4, though we strongly suspect it should not hold.

Chapter 5

Conclusions and Future Work

In this thesis we had the aim of analysing NAND-clause growth in the Neg proof system for arbitrary NAND-OR theories, NAND-binary NAND-OR theories and GNF theories. In that endeavour we proved conjecture 1 which stated that NAND-clauses had to grow to the size of the largest OR-clause in our interpretation of the pigeonhole principle as a NAND-OR theory ($\neg PHP_p^h$). Furthermore, we found counter examples disproving conjectures 2 3 stating that inconsistent NAND-OR and NAND-binary NAND-OR theories have refutations in Neg where NAND-clauses do not grow larger than the size of the greatest OR-clause ($\neg PHP_5^{2,2}$ and $bin(\neg PHP_7^{3,2})$). Conjecture 4, stating that the same is true for GNF theories, remains unsolved.

That being said, proving any of conjectures 2, 3 or 4 would have been a surprising result and was never expected to hold. While we did not obtain a counterexample of conjecture 4 (for a GNF theory), however the presented results, in particular that of the NAND-binary NAND-OR theory in 4.3.1, strengthen the expectation that conjecture 4 should also fail. Further, truth of any of these conjectures would suggest some strong limits on proof length that would be hard to believe. With the Neg system being a resolution-like system, one would expect it to have similar proof length lower bounds to other resolution-like systems. Of these, resolution and tree-like resolution are among those known to have superpolynomial complexity when it comes to proof length [9] [10].

Another interesting approach at finding lower bounds for Neg would be to explore just how similar the Neg system is to other known resolution style systems. It would be significant if we could for example find that Neg and resolution are p-equivalent – show that proofs in either system can be translated into proofs in the other in polynomial time –. This would settle the question of finding short proofs for tautologies in Neg as p-equivalence between two propositional proof systems implies they have the same proof complexity bounds [8]. Such a strategy would cover the aim of this thesis should it succeed. That being said, finding p-equivalences are typically very difficult which is why this strategy was not elected in this thesis.

Looking for a counterexample to conjecture 4 would be a natural continuation of this thesis. Finding such a theory may be difficult to do by hand however. An approach in the likes of creating all clauses possible to create in 1 step, and then 2 steps and so on until one is forced to create a NAND-clause that is undesirably large, has proven to quickly become quite complex. Resolution style proofs systems are known to be well suited for automated proofing systems. Creating an automated proof assistant

based on the Neg system would greatly increase efficiency of checking if GNF theories have refutations where NAND-clauses do not grow too large. Achieving this may only require modifying existing theorem provers to reason with the Rneg rule.

Lastly, having found a relative bound on the NAND-clause growth in refutations of $\neg PHP_p^h$ theories is a promising result. This should make finding bounds on proof lengths for this family of tautologies feasible. Work in this area may result in further evidence separating NP and coNP.

Appendix A

A.1 Translating clausal theories into GNF

Since any clausal theory can be translated into CNF, showing that any theory T in CNF is equisatisfiable with a GNF theory, $GNF(T)$, is enough to show that any clausal theory is equisatisfiable with a theory in GNF. Below we give a procedure that does precisely this. Another version of this translation procedure can be found in [5] as well as in [4] where it is described more formally.

We start with a theory in CNF so every formula in the theory is a conjunction of clauses, $C_1 \wedge \ldots \wedge C_n$, where each C_i is a disjunction of literals, $l_1 \vee \ldots \vee l_m$. For each formula in the theory follow the steps below.

step 1: For each variable l_i in the formula, introduce fresh variable $l_i^{'}$ and two new GNF formulae: $l_i^{'} \leftrightarrow \neg l_i$ and $l_i \leftrightarrow \neg l_i^{'}$ (given this hasn't already been done in translating a previous formula).

step 2: Replace each negative literal $\neg l_i$ in the formula with its corresponding $l_i^{'}$ from the previous step. Every clause in the formula will now only contain positive literals.

step 3: Replace each clause $(l_1 \vee \ldots \vee l_j \vee l_{j+1}^{'} \vee \ldots \vee l_m^{'})$, with the following GNF formula where c is fresh: $c \leftrightarrow (\neg l_1 \wedge \ldots \wedge \neg l_j \wedge \neg l_{j+1}^{'} \wedge \ldots \wedge \neg l_m^{'} \wedge \neg c)$.

Following these steps for each formula in our theory will result in an equisatisfiable GNF theory. To see this, observe that adding bi-implication formulae in step 1 does not affect satisfiabiliy/unsatisfiability as one variable is fresh in these bi-implications. Replacing negative literals by their corresponding positive literals in step 2, also does not affect satisfiability. In step 3 we replace clauses with equisatisfiable formula and so again satisfiability is not affected. Further, each new formula introduced is a GNF formula. With that, since satisfiability is not affected in any step and each formula is replaced with GNF formulae we can conclude that our new theory is in GNF and is equisatisfiable to our original CNF theory.

Example: We translate $\neg PHP_4^3$ (described in section 2.2.2) into a GNF theory, GNF($\neg PHP_4^3$).

First we rewrite $\neg PHP_4^3$ such that its in CNF:

$$\neg PHP_4^3 := \begin{Bmatrix} 1_1 2_1 3_1, 1_2 2_2 3_2, \\ 1_3 2_3 3_3, 1_4 2_4 3_4 \end{Bmatrix} \cup \begin{Bmatrix} \overline{1}_1\overline{1}_2, \overline{1}_1\overline{1}_3, \overline{1}_1\overline{1}_4, \overline{1}_2\overline{1}_3, \overline{1}_2\overline{1}_4, \overline{1}_3\overline{1}_4, \\ \overline{2}_1\overline{2}_2, \overline{2}_1\overline{2}_3, \overline{2}_1\overline{2}_4, \overline{2}_2\overline{2}_3, \overline{2}_2\overline{2}_4, \overline{2}_3\overline{2}_4, \\ \overline{3}_1\overline{3}_2, \overline{3}_1\overline{3}_3, \overline{3}_1\overline{3}_4, \overline{3}_2\overline{3}_3, \overline{3}_2\overline{3}_4, \overline{3}_3\overline{3}_4 \end{Bmatrix} \tag{A.1}$$

The OR-clauses were already in GNF so we simply seperate the NAND-clauses into disjunctions of negated atoms.

Next, after applying step 1 of the procedure to all formulas we add F_1 (below) to our theory:

$$F_1 := \begin{Bmatrix} 1_1 \leftrightarrow \overline{1'}_1, 1'_1 \leftrightarrow \overline{1}_1, 1_2 \leftrightarrow \overline{1'}_2, 1'_2 \leftrightarrow \overline{1}_2, \\ 1_3 \leftrightarrow \overline{1'}_3, 1'_3 \leftrightarrow \overline{1}_3, 1_4 \leftrightarrow \overline{1'}_4, 1'_4 \leftrightarrow \overline{1}_4, \\ 2_1 \leftrightarrow \overline{2'}_1, 2'_1 \leftrightarrow \overline{2}_1, 2_2 \leftrightarrow \overline{2'}_2, 2'_2 \leftrightarrow \overline{2}_2, \\ 2_3 \leftrightarrow \overline{2'}_3, 2'_3 \leftrightarrow \overline{2}_3, 2_4 \leftrightarrow \overline{2'}_4, 2'_4 \leftrightarrow \overline{2}_4, \\ 3_1 \leftrightarrow \overline{3'}_1, 3'_1 \leftrightarrow \overline{3}_1, 3_2 \leftrightarrow \overline{3'}_2, 3'_2 \leftrightarrow \overline{3}_2, \\ 3_3 \leftrightarrow \overline{3'}_3, 3'_3 \leftrightarrow \overline{3}_3, 3_4 \leftrightarrow \overline{3'}_4, 3'_4 \leftrightarrow \overline{3}_4 \end{Bmatrix} \tag{A.2}$$

Next we apply steps 2 and 3 to all the OR-clauses:

$$F_2 := \begin{Bmatrix} \alpha_1 \leftrightarrow \overline{1}_1 \wedge \overline{2}_1 \wedge \overline{3}_1 \wedge \overline{\alpha}_1, \\ \alpha_2 \leftrightarrow \overline{1}_2 \wedge \overline{2}_2 \wedge \overline{3}_2 \wedge \overline{\alpha}_2, \\ \alpha_3 \leftrightarrow \overline{1}_3 \wedge \overline{2}_3 \wedge \overline{3}_3 \wedge \overline{\alpha}_3, \\ \alpha_4 \leftrightarrow \overline{1}_4 \wedge \overline{2}_4 \wedge \overline{3}_4 \wedge \overline{\alpha}_4 \end{Bmatrix} \tag{A.3}$$

And then the NAND-clauses:

$$F_3 := \begin{Bmatrix} \alpha_5 \leftrightarrow \overline{1'}_1 \wedge \overline{1'}_2 \wedge \overline{\alpha}_5, \alpha_6 \leftrightarrow \overline{1'}_1 \wedge \overline{1'}_3 \wedge \overline{\alpha}_6, \\ \alpha_7 \leftrightarrow \overline{1'}_1 \wedge \overline{1'}_4 \wedge \overline{\alpha}_7, \alpha_8 \leftrightarrow \overline{1'}_2 \wedge \overline{1'}_3 \wedge \overline{\alpha}_8, \\ \alpha_9 \leftrightarrow \overline{1'}_2 \wedge \overline{1'}_4 \wedge \overline{\alpha}_9, \alpha_{10} \leftrightarrow \overline{1'}_3 \wedge \overline{1'}_4 \wedge \overline{\alpha}_{10}, \\ \alpha_{11} \leftrightarrow \overline{2'}_1 \wedge \overline{2'}_2 \wedge \overline{\alpha}_{11}, \alpha_{12} \leftrightarrow \overline{2'}_1 \wedge \overline{2'}_3 \wedge \overline{\alpha}_{12}, \\ \alpha_{13} \leftrightarrow \overline{2'}_1 \wedge \overline{2'}_4 \wedge \overline{\alpha}_{13}, \alpha_{14} \leftrightarrow \overline{2'}_2 \wedge \overline{2'}_3 \wedge \overline{\alpha}_{14}, \\ \alpha_{15} \leftrightarrow \overline{2'}_2 \wedge \overline{2'}_4 \wedge \overline{\alpha}_{15}, \alpha_{16} \leftrightarrow \overline{2'}_3 \wedge \overline{2'}_4 \wedge \overline{\alpha}_{16}, \\ \alpha_{17} \leftrightarrow \overline{3'}_1 \wedge \overline{3'}_2 \wedge \overline{\alpha}_{17}, \alpha_{18} \leftrightarrow \overline{3'}_1 \wedge \overline{3'}_3 \wedge \overline{\alpha}_{18}, \\ \alpha_{19} \leftrightarrow \overline{3'}_1 \wedge \overline{3'}_4 \wedge \overline{\alpha}_{19}, \alpha_{20} \leftrightarrow \overline{3'}_2 \wedge \overline{3'}_3 \wedge \overline{\alpha}_{20}, \\ \alpha_{21} \leftrightarrow \overline{3'}_2 \wedge \overline{3'}_4 \wedge \overline{\alpha}_{21}, \alpha_{22} \leftrightarrow \overline{3'}_3 \wedge \overline{3'}_4 \wedge \overline{\alpha}_{22} \end{Bmatrix} \tag{A.4}$$

Taking unions we get, GNF($\neg PHP_4^3$) = $F_1 \cup F_2 \cup F_3$.

Bibliography

[1] Stephen A. Cook and R. A. Reckhow. On the lengths of proof in propositional calculus. *Proceedings of the Sixth Annual ACM Symposium on the Theory of Computing*, pages 135–148, 1974. 1.1

[2] Stephen A. Cook. The complexity of theorem proving procedures. *Proceedings of the Third Annual ACM Symposium on the Theory of Computing*, pages 151–158, 1971. 1.1

[3] Michał Walicki. Resolving infinitary paradoxes. *Journal of Symbolic Logic*, 82(2):709–723, 2017. 1.1, 2.1, 2.2, 2.2.1, 3.1

[4] Michał Walicki. *Introduction to Mathematical Logic*. World Scientific Publishing Co. Pte. Ltd., 5 Toh Tuck Link, Singapore 596224, extended edition, 2017. 2.1, A.1

[5] Kjetil Golid. Incompleteness of the Inference System BNeg. Master's thesis. University of Bergen (UiB), Bergen, Norway. 2016. 2.1, 2.2.2, 3.1, A.1

[6] Michał Walicki Marc Bezem, Clemens Grabmeyer. Expressive power of digraph solvability. *Annals of Pure and Applied Logic*, 163:200–213, 2012. 2.1

[7] Sjur Dyrkolbotn and Michał Walicki. Propositional discourse logic. *Synthese*, 191(5):863–899, 2014. 2.1

[8] P. Beame and T. Pitassi. Propositional proof complexity: Past, present and future. *Electron. Colloquium Comput. Complex.*, 5, 1998. 2.2.2, 5

[9] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985. 5

[10] S.R. Buss. Towards np-p via proof complexity and proof search. *Annals of Pure Applied Logic*, 163:1163–1182, 2012. 5